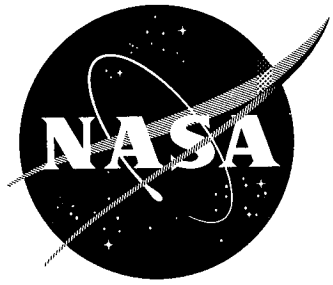
NASA TN D-1092



TECHNICAL NOTE

D-1092

AN INPUT ROUTINE USING ARITHMETIC STATEMENTS

FOR THE IBM 704 DIGITAL COMPUTER

By Don N. Turner and Vearl N. Huff

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

WASHINGTON

September 1961

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TECHNICAL NOTE D-1092

AN INPUT ROUTINE USING ARITHMETIC STATEMENTS FOR THE IBM 704 DIGITAL COMPUTER

By Don N. Turner and Vearl N. Huff

SUMMARY

An input routine has been designed for use with FORTRAN or SAP coded programs which are to be executed on an IBM 704 digital computer. All input to be processed by the routine is punched on IBM cards as declarative statements of the arithmetic type resembling the FORTRAN language.

The routine is 850 words in length. It is capable of loading fixed- or floating-point numbers, octal numbers, and alphabetic words, and of performing simple arithmetic as indicated on input cards. Provisions have been made for rapid loading of arrays of numbers in consecutive memory locations.

INTRODUCTION

The need for a method of reading input data different from that made available by compilers such as FORTRAN has been indicated by many problems coded at the Lewis Research Center. Most compilers now in operation require that the programmer itemize each input parameter in a read-type statement somewhere in the coding of the problem. While this procedure is often adequate, additional flexibility is desirable, especially concerning the order and quantity of values that may be read as well as the language in which they are presented - such as decimal, octal, or alphabetic data.

From a study of the features deemed desirable for a flexible input routine, it was decided that such a routine should have the following capabilities:

(1) To load decimal, octal, or alphabetic data into the machine memory where both the data and the location or destination are taken from statements on the input medium.

(2) To provide a means of operating on data through simple arithmetic, such as conversion of units.

(3) To accept an algebraic form of input that is independent of card format and easily understood.

(4) To provide an effective diagnosis of any errors detected by the machine in the input statements.

The INPUT routine described herein was subsequently designed for use on an IBM 704 digital computer that utilizes cards as the input medium. The logical structure of this routine is adaptable to other machines whether they utilize cards, paper tape, magnetic tape, or typewriter input.

GENERAL DESCRIPTION

The INPUT routine is 850 words in length and is coded in the SAP language. It is constructed as a subroutine of the type described in the IBM FORTRAN II manual (International Business Machines form (28-6000)); that is, it uses the standard subroutine linkage and is loaded by the Binary Symbolic Subroutine (BSS) loader associated with FORTRAN.

The INPUT routine, as presented, has been designed for use with the MONITOR system in use on the 704 computer at the Lewis Research Center. This MONITOR is used to facilitate the handling of program decks, to keep accounting records, and to decrease the idle machine time between jobs. In the MONITOR system of operation, all input is read from logical tape 7 (a card-to-tape converter is assumed) and all output is written on logical tape 6 (an auxiliary tape-to-printer is assumed). The INPUT routine may be easily adapted for use with different monitors or other systems.

The object of the INPUT routine is to make possible the transmission of data to the computer in the form of algebraic statements, such as

TEMPERATURE = 201.7

This form resembles that of the FORTRAN language and is readily understood and easily checked for errors. The function of the INPUT routine, then, is to relate the name TEMPERATURE to a specific memory address and to place the value 201.7 in that location. To accomplish this, the following steps are performed:

(1) Transfer control to the INPUT routine.

(2) Check to see if data should be read at this time.

(3) Relate the input data name to a specific memory address.

CONTENTS

(4) Convert the data value to the required form.

(5) Store the data value in its assigned location.

(6) Repeat steps (3), (4), and (5) until the data of that data group are exhausted.

(7) Return control to the calling program.

The details of the logical structure of the routine to accomplish these steps are presented in appendix A in the form of definitions. Appendixes B to E give, respectively, an illustration of the variable names in the table, error diagnostics, storage allocation, and the listing of the input routine.

ENTERING THE ROUTINE

The following discussion explains how to call (i.e., code the entry to) the INPUT routine.

CALL Statement

The standard CALL statement of FORTRAN II is used to transfer control to the INPUT routine. Three pieces of information are required as arguments in the calling sequence.

FORTRAN	SAP
CALL INPUT (589, X, TABLE)	TSX INPUT, 4 HTR CON589 HTR X HTR TABLE

These are examples of calling sequences in FORTRAN II and SAP. In either system the arguments 589, X, and TABLE are necessary for proper reading of data. After execution of the INPUT routine, control is returned to the statement following the calling sequence. A careful examination of each of the three arguments of the calling sequence will explain their function in the reading of data.

Argument 1. - The number 589 serves as identification of a data group. This value is compared with an identification number occurring on an input card. If these values agree, the processing of data is initiated and continues until another data identification card is encountered. If these values do not agree, no data are processed and control is returned to the calling program where normal program execution continues until the next input calling sequence is encountered. Argument 1 may be either a fixed-point number or a fixed-point variable.

Argument 2. - The location of X serves as a reference point for the storing of input data. Inasmuch as all data to be processed will

be stored relative to the memory location of X, the programmer is expected to provide fixed relations between the location of X and other locations to be loaded. In the FORTRAN language, these relative assignments are accomplished by use of COMMON or EQUIVALENCE statements or by relying on the order given automatically by FORTRAN. In any case, the INPUT routine assumes that serial memory locations will be arranged from larger to smaller absolute addresses; this is the way FORTRAN stores an array.

Argument 3. - The address of TABLE locates the first member of an array to be constructed and used by the INPUT routine. This array will contain the names of variables that are to be used on the cards and their memory location relative to X. Sufficient space in memory must be reserved for storing the table of names. If the program is written in FORTRAN, this is done by means of a DIMENSION statement. The names assigned by the user are stored six alphabetic characters per word. The last word may contain less than six characters and is filled out with zeros. The length of the array can be determined precisely by adding the locations required to store each name plus one location per name for a code number plus one location for a final zero entry that terminates the table. The information for the construction of the table is supplied via input cards.

TYPES OF INPUT STATEMENTS

The remainder of the report is devoted to a discussion of the input statements to be punched on cards. There are four basic types of statements, namely; (1) data group identifiers, (2) table statements, (3) loading statements, and (4) comments statements which this input routine is capable of processing. In the following paragraphs a discussion of each of these types is given along with examples of their use. In general, the statement is punched on an IBM card using the card from left to right and ending in column 72, assuming the card reader board is wired to read columns 1 to 72. Blanks are always ignored (except for ALF type data to be mentioned later).

$DATA Statement

The $DATA statement is a data group identifier. It specifies the group identification number associated with a particular group of input data. It always precedes a data group and must be the first statement on a card. For example,

```
$DATA = 589                                                    card 1
```

on the first card of a data group will cause the value 589 to be compared with the first argument of the calling sequence. If they are unequal, tape 7 will backspace one record (the equivalent of one card), and control will be returned to the calling or object program. It is

assumed that a calling sequence with this identification number will be forthcoming. If the value on the $DATA card equals the value of the first argument of the CALL statement, the routine will load data until another $DATA statement is encountered. Tape 7 is then backspaced one record to leave the new $DATA statement ready to be read, and control is returned to the calling program. The $DATA statement, then, serves to initiate the processing of data and to transfer control to the calling program at the end of the data group.

Whether data are loaded or not, the value of the identification on the $DATA card present when the routine is called is stored in the decrement of absolute location 77463. This is the last erasable store and is accessible to FORTRAN programs as I(0) if I is the first array in COMMON. This number may be used by the programmer for logical control of the INPUT subroutine.

$TABLE Statement

The object of $TABLE statements is to make a table of names. In typical use, this table of names is initially empty, and the names to be used on subsequent cards are entered by processing $TABLE statements. The statement begins with the word $TABLE; this is followed by a comma, then the table information, and is terminated with the slash character.

Consider, for example, that the variable names VELOCITY, MASS, and RADIUS are to be assigned to memory locations in the array designated by X, the second argument of the calling sequence, as X(1), X(2), and X(3), respectively. The card would be punched

```
$TABLE, 1 = VELOCITY, 2 = MASS, 3 = RADIUS/                    card 2
```

After card 2 is processed, the names VELOCITY, MASS, and RADIUS are in the table and can be used in subsequent loading statements. Their locations as the first, second, and third members of the array X are also in the table. No limit is placed on the length of a name, and it may begin with any alphabetic letter. If the name is to be associated with fixed-point data, which are considered to be less frequent, a decimal point is included with the location. For example, the statement

```
$TABLE, 20. = INDEX, 21. = SUBSCRIPT, 22. = I /                card 3
```

will place these names in the table and label them as fixed point. Any value subsequently loaded into INDEX, SUBSCRIPT, and I will be placed in the decrement part as an integer. The numbers will be stored in the 20th, 21st, and 22nd subscript positions of the array X. Appendix B contains a sample of the array TABLE to illustrate the contents of the table after cards 2 and 3 have been processed.

Note that all the values on cards 4 and 5 were placed in memory in accordance with the table assignments made on cards 2 and 3. The statement INDEX = 3 was continued between cards 4 and 5.

A variable name may be singly subscripted. A subscript must be an unsigned integer. For example:

```
RADIUS(2) = 6, 8, -10, , +24                                    card 6
```

will arrange data in the array X as

```
X(4) = 6.0
X(5) = 8.0
X(6) = -10.0
X(7) = no change
X(8) = 24.0
```

} floating point

Thus, the subscript (2) on the variable name RADIUS advances the storage counter to the second position of the array RADIUS before storing the values. Because the name RADIUS was made equivalent to X(3) by card 2, the value for RADIUS(2) is placed in X(4). The commas appearing between the numerical values on card 6 are an example of loading into consecutive memory locations. A series of numbers separated by commas are loaded in successive memory locations. Two commas in succession with no number included cause a memory location to be skipped without change, as for X(7).

Octal values. - A data value preceded by (OCT) will be loaded without conversion. The rightmost twelve digits, or less, if there are less, immediately following the) character, blanks omitted, are loaded as an octal number. In storage the octal number will be right-adjusted. The loading statement

```
I(2) = (OCT) 175 326                                            card 7
```

will store directly, without conversion, the octal number 000000175326 into the second member of the I array. The digits 8 and 9 will be stored modulo 8.

Alphabetic values. - A data value preceded by (ALF n) will be stored in binary coded decimal. This form, n being an integer, will cause the next n columns immediately following the) character, blanks included, to be read in BCD mode and stored consecutively six characters per location. If the last word is less than six characters in length, it will be filled out with blanks. All alphabetic, numeric, and special characters may be loaded with this form of loading statement. Symbols such as $DATA will be interpreted as alphabetic data and will not be interpreted as control statements within the n columns of an (ALF n) form. The length of the alphabetic loading statement has

practical limitations only; but, if it exceeds 924 characters (or columns), machine memory locations smaller than octal 77462 will be overwritten.

An example of an alphabetic-type loading statement would be

```
I(3) = (ALF 23) MAKE IT UNDERSTANDABLE.                    card 8
```

where the 23 columns of characters will be placed in four consecutive words of memory.

Internally addressed values. - An internally addressed value is one that refers to the contents of memory by name. In the example

```
RADIUS(7) = RADIUS(3)                                      card 9
```

it is demonstrated that a name may refer to a piece of data currently in storage. The loading of this statement results in the replacing of the contents of the storage equivalent to RADIUS(7) with the value found in the location RADIUS(3). Any name that is used as an internally addressed value must have appeared previously in $TABLE statement.

Arithmetic expressions. - Provisions have been made to allow simple arithmetic to be performed on data at execution time. Any number may be altered by addition (+), subtraction (-), multiplication (*), or division (/) with other numbers or names, provided all names used in statements appear previously in $TABLE statement. Operations are performed in sequence from left to right on the card. Consider the statement

```
RADIUS(2) = 0.5 * RADIUS(3), RADIUS(3) = 1.06E+2 - ()/I    card 10
```

as being executed subsequent to card 6. The result of loading this statement would be to replace the contents of the location RADIUS(2) with the product of 0.5 and RADIUS(3) or, using the value loaded from card 6, RADIUS(2) would contain 4.0 in floating-point form. Execution of the second statement on the card will then change RADIUS(3). The value of the arithmetic statement following RADIUS(3) will be computed as follows: First the difference between 106 and the current value of RADIUS(3) (which is denoted by an empty set of parentheses) will be computed. Then the difference will be divided by the contents of the location equivalent to I. The value, therefore, placed in the location RADIUS(3) would be 2.0 in floating-point form. No regard need be given the modes of the numbers in a loading statement, since all operations are performed in floating-point form with the proper conversion provided. No provisions have been made for subgrouping or nesting operations. If more than one operator appears in sequence (such as A/-3), an error diagnostic will result.

practical limitations only; but, if it exceeds 924 characters (or columns), machine memory locations smaller than octal 77462 will be overwritten.

An example of an alphabetic-type loading statement would be

```
I(3) = (ALF 23) MAKE IT UNDERSTANDABLE.                    card 8
```

where the 23 columns of characters will be placed in four consecutive words of memory.

Internally addressed values. - An internally addressed value is one that refers to the contents of memory by name. In the example

```
RADIUS(7) = RADIUS(3)                                      card 9
```

it is demonstrated that a name may refer to a piece of data currently in storage. The loading of this statement results in the replacing of the contents of the storage equivalent to RADIUS(7) with the value found in the location RADIUS(3). Any name that is used as an internally addressed value must have appeared previously in $TABLE statement.

Arithmetic expressions. - Provisions have been made to allow simple arithmetic to be performed on data at execution time. Any number may be altered by addition (+), subtraction (-), multiplication (*), or division (/) with other numbers or names, provided all names used in statements appear previously in $TABLE statement. Operations are performed in sequence from left to right on the card. Consider the statement

```
RADIUS(2) = 0.5 * RADIUS(3), RADIUS(3) = 1.06E+2 - ()/I    card 10
```

as being executed subsequent to card 6. The result of loading this statement would be to replace the contents of the location RADIUS(2) with the product of 0.5 and RADIUS(3) or, using the value loaded from card 6, RADIUS(2) would contain 4.0 in floating-point form. Execution of the second statement on the card will then change RADIUS(3). The value of the arithmetic statement following RADIUS(3) will be computed as follows: First the difference between 106 and the current value of RADIUS(3) (which is denoted by an empty set of parentheses) will be computed. Then the difference will be divided by the contents of the location equivalent to I. The value, therefore, placed in the location RADIUS(3) would be 2.0 in floating-point form. No regard need be given the modes of the numbers in a loading statement, since all operations are performed in floating-point form with the proper conversion provided. No provisions have been made for subgrouping or nesting operations. If more than one operator appears in sequence (such as A/-3), an error diagnostic will result.

The array X would finally appear as:

X(1) = 3.4 X(2) = 32.0 X(3) = 4.0×10^{21} X(4) = 4.0 X(5) = 2.0 X(6) = -10.0	floating point
X(7) = whatever was present originally	
X(8) = 24.0 X(9) = 8.0	floating point
X(20) = 3 X(21) = 47 X(22) = 49	fixed point
X(23) = 000000175326	OCTAL
X(24) = MAKE I	BCD
X(25) = T UNDE	BCD
X(26) = RSTAND	BCD
X(27) = ABLE.	BCD

Continuation of loading statement. - To continue a right side of a loading statement from one card to the next, simply continue key punching on the next card. Continuation of the right side never fails.

The name on the left of the equal sign, however, cannot generally be continued. It will be found that continuation of the left side does work unless the part of the statement appearing on the second card appears (by itself) to be a left side, that is, a name followed by an equal sign. This restriction on left-side continuation was adopted to permit the omission of the final comma on the right side of loading statements. In actual use, this left-side restriction has not been inconvenient; in fact, the legibility is better when the name is entirely on one card. An example of correct continuation of a loading statement was given on cards 4 and 5.

$$ Comment Statement

This type statement causes the entire card on which the $$ pair appears to be written in BCD on tape 6 for listing off-line. The $$ characters may appear anywhere on any type card and even ahead of a $DATA type card. The effect of the $$ symbol pair is to move the end

of a card from column 72 forward to the column ahead of the symbol pair. Anything appearing to the right of the $$ symbols will be listed but will not be treated as a loading statement. For example, two cards containing

```
$$ THIS IS THE FOURTH CASE.                                  card 11

I = 4, VELOCITY(9) = 3.762E7, $$ INPUT CARD                  card 12
```

will cause these cards to be written on tape 6 along with all other program output. Furthermore, if these cards are processed along with the previous cards, the values for I and VELOCITY(9) will be changed to a fixed-point 4 and a floating-point 37620000.0, respectively.

DIAGNOSIS OF ERRORS ON INPUT CARDS

The greatest source of errors introduced during preparation of input statements arises from misspelling names, improper use of the numeric-alphabetic shift on the key punch, and misuse of operators, commas, and equal signs. Considerable care has been taken within the program to detect errors that lead to illegitimate situations in the processing of the data. However, a few illegitimate statements are not detected by the routine and will process improperly or, as likely, be omitted.

When an error is detected on a card, the type of error and the complete offending card image are written on tape 6. An asterisk is recorded beneath the last character processed and is usually within a few characters of the one causing the error. If the asterisk is beneath the first character of a card, the error was probably on the preceding card. Generally, a cursory examination of the printed card and asterisk is sufficient for detection and correction of the card. A complete list of error types and associated discussions is given in appendix C for a more complete analysis.

As an illustration of a machine-detected error, assume that the cards

```
$DATA = 589, $TABLE, 1 = TX/                                 $$ card 13

TY = 35.7                                                    $$ card 14

$DATA = 589                                                  $$ card 15
```

are to be processed by the input routine. An error would be detected when the variable name TY is processed because it is not in the table. Tape 6 would subsequently contain the following BCD information for listing off-line:

```
(T) NO ENTRY IN TABLE TY = 35.7                              $$ card 14
                        *
```

Examination of the error type and discussion in appendix C reveals that a type T error means "There is no table entry for a variable name."

EXAMPLE

The following FORTRAN program is presented as an example of a calling sequence for the INPUT subroutine and may be used to check the functioning of the routine.


```
C       THIS IS A SAMPLE PROGRAM TO CALL THE INPUT SUBROUTINE.
C       IT READS A SET OF DATA AND PRINTS OUT AN AREA OF STORAGE.
        COMMON C
        DIMENSION X(30), TABLE(15),C(65),L(65)
        EQUIVALENCE(X,C),(TABLE,C(50)),(L,C)
C       TRANSFER TO THE INPUT ROUTINE
      1 CALL INPUT (589, X, TABLE)
C       PRINT OUT THE STORE.
        WRITE OUTPUT TAPE 6,101,(J,X(J),J=1,9),(J,L(J),J=20,27)
C       TRANSFER TO READ MORE DATA
        GO TO 1
  101 FORMAT(9(3H X(I2,1H)1PE18.6/),  3(3H X(I2,1H),I7/)3H X(,I2,1H),
     1015/,4(3H X(I2,1H)A8/))
C       END OF FORTRAN STATEMENTS.
```

The data cards listed in the text are used. For clarity, all cards have been terminated with $$ so that they will be listed on the output, and two extra $DATA cards are inserted, making four groups of data. The data cards are as follows:

```
$DATA = 589                                                    $$ CARD  1
$TABLE, 1 = VELOCITY, 2 = MASS, 3 = RADIUS/                    $$ CARD  2
$TABLE,20. = INDEX, 21. = SUBSCRIPT, 22. = I    /              $$ CARD  3
VELOCITY = 3.4, MASS = 32, RADIUS = 4E+21, INDEX               $$ CARD  4
= 3, SUBSCRIPT = 47, I = 49                                    $$ CARD  5
RADIUS(2) = 6,8,-10,,+24                                       $$ CARD  6
$DATA = 589,                                                   $$INSERTED
I(2) = (OCT) 175 326                                           $$ CARD  7
I(3) = (ALF23)MAKE IT UNDERSTANDABLE.                          $$ CARD  8
RADIUS(7) = RADIUS(3)                                          $$ CARD  9
RADIUS(2) = 0.5 *RADIUS(3) ,  RADIUS(3) =1.06E+2-()/I          $$ CARD 10
      $DATA = 589                                              $$INSERTED
$$THIS IS THE FOURTH CASE                                         CARD 11
I = 4, VELOCITY(9) = 3.762E7, $$INPUT CARD                        CARD 12
$DATA = 589, $TABLE, 1 = TX  /                                 $$ CARD 13
TY = 35.7                                                      $$ CARD 14
$DATA = 589                                                    $$ CARD 15
```

In this example the storage will be printed out after the data cards have been processed for each group. (A group is all cards between $DATA cards.) The listing made from tape 6 is as follows.

REMARKS

The INPUT routine as described removes the necessity for the programmer to supply detail concerning data at the time the problem is coded. At most, he must decide the appropriate time and number of times during execution of the program that input is to be read. The routine also provides increased flexibility at execution time to meet the requirement of specific problems.

It is felt that the form of the statements for loading data presented in this report is considerably more legible and, consequently, is easier to check for accuracy and completeness than the form previously in general use.

A binary deck can be obtained for this program from the Lewis Research Center (Att'n Mr. Vearl N. Huff). A similar deck for the 7090 is also available.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, June 28, 1961

APPENDIX A

LOGICAL STRUCTURE OF INPUT ROUTINE

The following definitions and specifications define the external characteristic of the INPUT routine.

(1) There are two classes of statements: loading statements, and control statements.

(2) A loading statement consists of a left side, an equal sign, and one or more right sides.

(a) A loading statement loads the machine memory location corresponding to the left side with the value obtained from the right side.

(b) Successive machine memory locations are loaded with the values obtained from successive right sides.

(3) A left side must consist of a name and must be followed by an equal sign. It may follow previous loading statements. A left side is normally restricted to one card.

(4) Any name:

(a) Must begin with an alphabetic character.

(b) May contain any number of alpha-numeric characters (practical limits only).

(c) Must be defined as to corresponding location and as either fixed or floating point in the table of names (failure to enter a name in TABLE will cause an error diagnostic).

(d) Will have zero characters translated to the letter O in any position after the first.

(e) May be singly subscripted.

(f) Is ended by an operator or the end of a side.

(g) Will have the value of the contents of the location to which it corresponds. Normally, the value is treated as a floating-point number. However, if the name is designated as fixed-point, then the value will be fixed before storing and floated when obtained from storage. Fixed-point numbers are stored in the prefix and decrement as is done with FORTRAN.

APPENDIX A

LOGICAL STRUCTURE OF INPUT ROUTINE

The following definitions and specifications define the external characteristic of the INPUT routine.

(1) There are two classes of statements: loading statements, and control statements.

(2) A loading statement consists of a left side, an equal sign, and one or more right sides.

(a) A loading statement loads the machine memory location corresponding to the left side with the value obtained from the right side.

(b) Successive machine memory locations are loaded with the values obtained from successive right sides.

(3) A left side must consist of a name and must be followed by an equal sign. It may follow previous loading statements. A left side is normally restricted to one card.

(4) Any name:

(a) Must begin with an alphabetic character.

(b) May contain any number of alpha-numeric characters (practical limits only).

(c) Must be defined as to corresponding location and as either fixed or floating point in the table of names (failure to enter a name in TABLE will cause an error diagnostic).

(d) Will have zero characters translated to the letter O in any position after the first.

(e) May be singly subscripted.

(f) Is ended by an operator or the end of a side.

(g) Will have the value of the contents of the location to which it corresponds. Normally, the value is treated as a floating-point number. However, if the name is designated as fixed-point, then the value will be fixed before storing and floated when obtained from storage. Fixed-point numbers are stored in the prefix and decrement as is done with FORTRAN.

(5) A subscript (if any) follows and is part of a name; it begins with a left parenthesis followed by digits, followed by a right parenthesis. The subscript refers to members of arrays:

(a) A variable (or named) subscript is illegal.

(b) A subscript may be used on any name.

(6) A right side:

(a) Must follow an equal sign or another right side.

(b) May be a name.

(c) May be a number.

(d) May be an octal number.

(e) May be an alphabetic field.

(f) May be an arithmetic expression.

(g) May be blank. A blank is two commas in succession (a blank will not change the present value of the memory).

(h) Must be terminated by a comma or the end of a card, but the end of a card terminates a right side only if it is determined to be the end of the loading statement (see (7)).

(i) May be continued from one card to another.

(7) The end of a card terminates a loading statement if:

(a) The next card is $DATA or $TABLE.

(b) The next card begins with a left side (i.e., begins with a name followed by an equal sign on the same card).

(8) A number is an unsigned string of digits that may contain a decimal and may be followed by a base 10 exponent. If there are more than ten digits in the string exclusive of the exponent, only the ten most significant digits will be used in conversion to find its value. Decimal places are counted for indefinitely long strings. A number is terminated by an operator or the end of the right side.

An exponent begins with the letter E and may be followed by a sign and/or a string of digits whose value is used as the exponent. The exponent, when present, is used in conversion to compute the final value of the number. Overflow is possible. A signed number is treated as an arithmetic statement.

(9) An octal number begins with a left parenthesis followed by the letter O, which may be followed by any number of letters followed by a right parenthesis, followed by any number of digits, normally followed by a comma (see (6h)). The rightmost twelve digits, or less if there are less, are used as the value right-adjusted. Neither floating- nor fixed-point designations have any effect for storing octal numbers.

(10) An alphabetic field begins with a left parenthesis followed by the letter A, and may be followed by any number of letters, and must be followed by any number of digits (which will designate the number of card columns in the alphabetic field) followed by a right parenthesis, followed by the alphabetic field in successive columns (to col. 72), and continuing if necessary on successive cards until the designated number of columns has been stored. Any BCD characters can be punched in the field. A comma normally follows the alphabetic field to terminate the right side (see (6h)). Neither floating- nor fixed-point designation has any effect when storing ALF data.

(11) An arithmetic expression contains one or more operators. The following rules must be observed in arithmetic statements:

(a) The operators are plus, minus, asterisk, and slash.

(b) An arithmetic expression also contains at least one operand.

(c) The operands may be names, numbers, or empty parentheses.

(d) Empty parentheses are used to designate the current value of the current left side.

(e) Two operators must not appear together.

(f) Operations are performed from left to right on the values of the operands in the sequence given, and the result is the value of the arithmetic statements.

(g) Parentheses to indicate order of operations are illegal.

(h) Storage in named memory locations may be used to accumulate factors for subsequent statements.

(12) A $TABLE control statement stores names and equivalent machine locations relative to the second argument of the calling sequence in a table located relative to the third argument of the calling sequence. The $TABLE control statement begins with a dollar sign followed by a T, followed by any number of alphabetic characters, followed by a comma. Subsequent substatements are interpreted as $TABLE substatements.

(a) $TABLE substatement begins with a numeric string that contains a decimal point only if a fixed-point variable is being defined, and is followed by an equal sign followed by a name, followed by a comma.

(b) Any number of $TABLE substatements may be used and may be unconditionally continued over any number of cards. The last statement is followed by a slash that ends the $TABLE control statement.

(13) A $DATA control statement is used to establish correspondence between a group of data and a particular calling sequence and to signal an end of a group of data. It begins with a dollar sign in the first nonblank column on a card, and is followed by the letter D followed by any number of letters, followed by an equal sign, followed by a string of digits, followed by a comma or the end of the card. The value of the string of digits is used to compare with the first argument of the calling sequence. The entire statement must be on one card.

(14) The symbol pair $$ occurring on a card has the effect of ending the processing of the card at the column ahead of the $$ symbol pair and causes the entire card (72 columns) to be written on output tape 6 along with any other output. The next character processed will be read from the next card. The $$ symbol pair serves to insert comments in the output and may be used to list the input cards with the output. It may be used anywhere but will not be interpreted within an alphabetic field.

APPENDIX B

ILLUSTRATION OF VARIABLE NAMES IN TABLE

Assuming that the following $TABLE cards have been processed, the array TABLE would appear as indicated.

```
$TABLE, 1 = VELOCITY, 2 = MASS, 3 = RADIUS/

$TABLE, 20. = INDEX, 21. = SUBSCRIPT, 22. = I/
```

TABLE LOCATION	OCTAL VALUE	BCD CONTENTS*
1	000003000001	(code word)
2	652543462331	VELOCI
3	637000000000	TY
4	000002000002	(code word)
5	442162620000	MASS
6	000002000003	(code word)
7	512124316462	RADIUS
8	400002000024	(code word)
9	314524256700	INDEX
10	400003000025	(code word)
11	626422622351	SUBSCR
12	314763000000	IPT
13	400002000026	(code word)
14	310000000000	I
15	000000000000	(zero code)

*Except for code word. The decrement of each code word contains a number equal to 1 plus the number of stores used for the name. The address of each code word contains an octal number corresponding to the table assignment of the name. The sign of a code word is negative if the variable is fixed point. A negative sign in this listing appears as a leading 4.

APPENDIX C

ERROR DIAGNOSTICS

The following is a list of errors referred to by the off-line listing.

Error type	Reason
(A)	A nonnumeric character appears in the numeric field of a $TABLE type card.
(B)	A comma was not used to terminate an ALF field.
(C)	An illegitimate character appears in the subscript of a name.
(D)	The equal sign of a $DATA type statement is missing or preceded by a nonalphabetic character.
(E)	An illegitimate character appears in the numeric field of a number.
(F)	An illegitimate character appears in the exponent field of an E format number.
(G)	A special character was used in a name of a table entry.
(J)	An (OCT) field contains a nonnumeric character.
(K)	A special character appears in character count of an (ALF) type field, or the character count was zero.
(L)	A special character other than +, -, * , or / was interpreted as an arithmetic operator.
(M)	A nonnumeric character appears in the identification field of a $DATA type card.
(N)	More than one decimal point occurs in a number.
(RTT)	A REDUNDANCY tape test failed five times.
(S)	At least one numeric did not precede the E of an E format number.

(T) There is no table entry for a variable name.

(U) A $DATA type card was not found as the first card of a set of data.

(V) The exponent of a floating-point number was out of range.

APPENDIX D

STORAGE ALLOCATION

Nearly all intermediate data generated by the INPUT routine are stored in an area that is generally inaccessible to programmers coding in the FORTRAN language. This area is the last 205 core positions of the computer. It is temporary storage utilized by many of the library routines written for the 704 computer as well as an area from which the BSS loader operates in loading program decks. Location 77463 (octal), while in the erasable area, is readily accessible to FORTRAN coders. The location will contain the identification number from the $DATA card present when the INPUT routine is called, whether data are loaded or not. When this number is to be used later, it should be moved to a reserved location. The assignments in the common area are as follows:

OCT	
77776	RECORD(1)
77775	(2)
77774	(3)
77773	(4)
77772	(5)
77771	(6)
77770	(7)
77767	(8)
77766	(9)
77765	(10)
77764	(11)
77763	(12)
77762	(13)
77761	(14)
77760	I
77757	KK
77756	Q
77755	WORD
77754	OPER
77753	Temporary Index B in SUB TABLE. Also NEXP in SUB NUMBER.
77752	J
77751	MSHIFT
77750	ILOC
77747	TEMP
77746	KNT1
77745	KNT2
77744	KNT3
77743	SIGN in SUB CHRCTR
77742	ALF in SUB CHRCTR
77741	TAG in SUB CHRCTR
77740	Temporary MQ in SUB STORE
77737	JK
77736	Index A storage in SUB CHRCTR
77735	Index B storage in SUB CHRCTR
77734	
77733	Index C storage in SUB LOOK
77732	Index A storage in SUB STORE
77731	Index B storage in SUB STORE
77730	Index C storage in SUB TABLE
77727	Table entry index in SUB TABLE
77726	Index C storage in SUB CHRCTR
77725	Temporary MQ storage in SUB TEST
77724	Temporary ACC storage in SUB TEST
77723	Index C storage in SUB DATA
77722	Third argument to SUB INPUT
77721	JK1
77720	Index C storage in SUB NUMBER
77717	Pseudo ACC
77716	ILOC1
77715	
77714	VAR(1)
. . .	. . .
77462	VAR(153)
77463	IDENT, $DATA identification number

APPENDIX E

LISTING OF INPUT ROUTINE

```
                                        THIS IS SUBROUTINE INPUT.  ITS CALLING SEQUENCE
                                        CONTAINS THREE ARGUMENTS---AN IDENTIFICATION
                                        CODE NUMBER, THE FIRST LOCATION RELATIVE TO WHICH
                                        ALL DATA IS TO BE LOADED, AND THE FIRST LOCATION
                                        OF A TABLE TO BE USED BY THE ROUTINE.


                                        INCLUDED IN THIS ASSEMBLY ARE SUBROUTINES
                                           1  INPUT
                                           2  CHRCTR
                                           3  CLEAR
                                           4  COMPAR
                                           5  ERROR
                                           6  LOOK
                                           7  NAME
                                           8  NUMBR
                                           9  STORE
                                          10  TABLE
                                          11  TEST
                                          12  ACCUM, FIX, FLT, BINARY

                                        CODING FOR THE PROGRAM CARD.
                    00000           ORG 0
                                    PGM
00000   0 00000 0 01521             PZE EXP+1,0,0
00001   0 00000 0 77462             PZE 32562
00002   314547646360                BCD 1INPUT
00003   0 00000 0 00004             PZE INPUT

                                        THESE TEMPORARY VARIABLES ARE LOCATED IN
                                        ERASABLE UPPER MEMORY.
                                    REL
                    00000           ORG 0
                    77776 RECORD    SYN 32766       CARD IMAGE.
                    77760 I         SYN 32752       WORD POINTER FOR CARD
                    77757 KK        SYN 32751       CHARACTER POINTER FOR CARD.
                    77756 Q         SYN 32750       UNTESTED CHARACTERS IN COMPAR.
                    77755 WORD      SYN 32749       CURRENT CHARACTER IN ADDRESS PART.
                    77754 OPER      SYN 32748       DIGIT IN DECREMENT REPRESENTS OPERATOR.
                    77753 B         SYN 32747       TEMP INDEX IN SUB TABLE.
                    77753 NEXP      SYN 32747       NUMERIC VALUE OF EXPONENT.
                    77752 J         SYN 32746       COUNTER IN SUB STORE.
                    77751 MSHIFT    SYN 32745       COUNTER IN SUB STORE.
                    77750 ILOC      SYN 32744       DATA BROUGHT FROM TABLE.
                    77747 TEMP      SYN 32743       TEMPORARY STORAGES.
                    77746 KNT1      SYN 32742       COUNTER TOTAL DIGITS.
                    77745 KNT2      SYN 32741       NONZERO UNTIL DECIMAL IN NUMBR.
                    77744 KNT3      SYN 32740       ZERO UNTIL DIGIT IN EXPONENT.
                    77714 VAR       SYN 32716       SPACE FOR NAMES ETC.
                    77463 IDENT     SYN 32563       DECREMENT HAS IDENT FROM CARD.
                    77743 SIGN      EQU TEMP-4      MINUS IF SUB CHRCTR READS CARD.
                    77742 TAG       EQU TEMP-5      SAVES $ IN SUB CHRCTR.
                    77741 ALF       EQU TEMP-6      NONZERO MEANS ALF MODE IN CHRCTR.
                    77737 JK        EQU TEMP-8      SUBSCRIPT CORRESPONDS TO NAME.
                    77734 RTT       EQU TEMP-11     COUNTS TAPE READ FAILURES.
                    77721 JK1       EQU TEMP-22     CURRENT SUBSCRIPT OF LEFT SIDE.
                    77717 ACC       EQU TEMP-24     PSEUDO ACCUM.
                    77716 ILOC1     EQU TEMP-25     ILOC FOR LEFT SIDE.
                    77715 KNT4      EQU TEMP-26     NONZERO AFTER EXPONENT SIGN.
```

```
00000  0 00000 0 00000 INDX   HTR 0              STORAGE FOR INDEX A.
00001  0 00000 0 00000        HTR 0              STORAGE FOR INDEX B.
00002  0 00000 0 00000        HTR 0              STORAGE FOR INDEX C.
00003  314547646360           BCD 1INPUT
00004 -0 63400 1 00000 INPUT  SXD INDX,1         SAVE INDEX REGISTER A.
00005 -0 63400 2 00001        SXD INDX+1,2       SAVE INDEX REGISTER B.
00006 -0 63400 4 00002        SXD INDX+2,4       SAVE INDEX REGISTER C.
00007  0 50000 0 00351        CLA ONEA
00010  0 40000 4 00002        ADD 2,4            2,4 IS THE BASE LOCATION.
00011  0 62100 0 00127        STA SET
00012  0 62100 0 00153        STA LOC1
00013  0 62100 0 00311        STA LOC4
00014  0 50000 4 00001        CLA 1,4            1,4 IS THE IDENTIFICATION NUMBER.
00015  0 62100 0 00102        STA NREG1
00016  0 53400 1 00304        LXA LOCBB-2,1      INITIALIZE 36
00017  0 60000 1 77761        STZ I+1,1            LOCATIONS
00020  2 00001 1 00017        TIX *-1,1,1          TO ZERO.
00021  0 60100 0 77716        STO ILOC1          MAKE NON-ZERO.
00022  0 50000 4 00003        CLA 3,4            3,4 IS THE LOCATION OF THE TABLE.
00023  0 62100 0 00677        STA LOCFC          PREPARE
00024  0 40000 0 00351        ADD ONEA            THE
00025  0 62100 0 00664        STA LOCFA            ARGUMENT STORAGES
00026  0 62100 0 00707        STA LOCFF
00027  0 07400 4 00517        TSX CLEAR,4        CLEAR THE VAR REGION.

                                  LOOK AT  THE FIRST CHARACTER ON THE FIRST CARD
                                  IN SEARCH OF A $ SIGN.
00030  0 07400 4 00352 LOCAA  TSX CHRCTR,4
00031  0 76000 0 00003        SSP
00032  0 40200 0 00350        SUB DOLLAR         CHECK FOR A $ SIGN.
00033  0 60100 0 77755        STO WORD
00034  0 07400 4 00524        TSX COMPAR,4
00035  246300000000           BCD 1DT0000
00036  0 02000 2 00042        TRA *+4,2
00037  0 02000 0 00345        TRA ERRU           JUNK
00040  0 02000 0 00111        TRA LOCAH          T
00041  0 53400 2 00102 LOCAB  LXA NREG1,2        D
00042  3 00000 2 00050        TXH LOCAD,2,0      NREG1 IS ZERO IF A $DATA CARD HAS
                                  ALREADY BEEN READ.
00043  0 76400 0 00207 LOCAC  BST 7
                                  THIS IS THE PROGRAM RETURN.
00044 -0 53400 1 00000 RTN    LXD INDX,1         RESET INDEX A.
00045 -0 53400 2 00001        LXD INDX+1,2       RESET INDEX B.
00046 -0 53400 4 00002        LXD INDX+2,4       RESET INDEX C.
00047  0 02000 4 00004        TRA 4,4            RETURN TO CALLING PROGRAM.
                                  HUNT FOR THE = SIGN OF THE $ DATA CARD.
00050  0 07400 4 00352 LOCAD  TSX CHRCTR,4
00051  0 07400 4 00524        TSX COMPAR,4
00052  130000000000           BCD 1=00000
00053  0 02002 2 00060        TRA *+5,2,2
00054  0 02000 0 00335        TRA ERRD           JUNK
00055  0 02000 0 00050        TRA LOCAD          ALPHABETIC
00056  0 02000 0 00335        TRA ERRD           NUMERIC
00057 -0 63400 4 77741        SXD ALF,4          = SIGN
                                  USE ALF MODE TO TEST ALL CHARACTERS.
00060  0 02000 0 00062        TRA LOCAF

00061  0 07400 4 01453 LOCAE  TSX BINARY,4       FORM BINARY WORD IN VAR.
```

```
                                        COMES HERE WHEN = SIGN HAS BEEN FOUND.  GET THE
                                        IDENTIFICATION NUMBER FROM THE CARD.
00062   0 07400 4 00352 LOCAF   TSX CHRCTR,4
00063   0 12000 0 00067         TPL **+4            IF NEGATIVE, NO COMMA WAS ON THE
00064   0 76400 0 00207         BST 7               $DATA CARD SO BACKSPACE TAPE.
00065   0 60000 0 77760         STZ I               INITIALIZE I TO READ A CARD.
00066   0 02000 0 00077         TRA LOCAG
00067   0 07400 4 00524         TSX COMPAR,4
00070   735360000000            BCD 1,$ 000
00071   0 02002 2 00100         TRA **+7,2,2
00072   0 02000 0 00343         TRA ERRM            JUNK
00073   0 02000 0 00343         TRA ERRM            ALPHABETIC
00074   0 02000 0 00061         TRA LOCAE           NUMERIC
00075   0 02000 0 00062         TRA LOCAF           BLANK
00076   0 60100 0 77743         STO SIGN            DOLLARS
                                        COMES HERE TO CHECK THE REGION CODE AND THE
                                        VALUE APPEARING ON THE $DATA CARD.
00077   0 50000 0 77714 LOCAG   CLA VAR            COMMA
00100   0 76700 0 00022         ALS 18
00101   0 60100 0 77463         STO IDENT           SAVE IDENT AT 77463.
00102   0 40200 0 00000 NREG1   SUB **              PLACE FIRST ARG IN THIS ADDRESS.
00103  -0 10000 0 00043         TNZ LOCAC           ZERO IF CALL CODE = $DATA CODE.
00104   0 60000 0 77741         STZ ALF             ALF = 0 MEANS NO ALF INFO.
00105   0 62100 0 00102         STA NREG1           TO INDICATE THAT A $DATA CARD HAS
00106  -0 53400 2 77741         LXD ALF,2           BEEN READ.
00107  -0 63400 2 00121         SXD TESTJK,2        ZERO THE DECREMENT.
00110   0 02000 0 00155         TRA LOCAN1

                                        COMES HERE IF IT WAS A $ TABLE CARD.
00111   0 53400 2 00102 LOCAH   LXA NREG1,2         CHECK TO SEE IF A $ DATA CARD HAS
00112  -2 00000 2 00114         TNX LOCAJ,2,0       BEEN READ.  NREG1 = 0 IF SO.
00113   0 02000 0 00345         TRA ERRU
00114   0 07400 4 01242 LOCAJ   TSX TABLE,4
00115   0 02000 0 00157         TRA LOCAN3

                                        COMES HERE IF AN ALPHABETIC CHARACTER WAS FOUND.
00116   0 07400 4 00714 LOCAK   TSX NAME,4
00117  -0 10000 0 00126         TNZ SET-1           ZERO MEANS ON LEFT OF = SIGN.
00120  -0 53400 1 77721         LXD JK1,1           IF JK1 DIDNOT INCREASE THEN
00121  -3 00000 1 00161 TESTJK  TXL ERRL,1,**       AN = SIGN WAS NOT USED.
00122  -0 63400 2 00121         SXD TESTJK,2        SAVE JK1 FOR NEXT TEST.
00123   0 50000 0 77750         CLA ILOC            SAVE SIGN OF TABLE ENTRY.
00124   0 60100 0 77716         STO ILOC1
00125   0 02000 0 00156         TRA LOCAN2

00126  -0 53400 2 77737         LXD JK,2            PREPARE TO ACCUMULATE THE NUMBERS
00127   0 50000 2 00000 SET     CLA **,2            IN THE PSEUDO ACCUMULATOR.
00130   0 60100 0 77747         STO TEMP
00131   0 50000 0 77750         CLA ILOC
00132   0 12000 0 00137         TPL LOCAM           MINUS MEANS FLOAT THE NUMBER.
00133   0 07400 4 01462         TSX FLT,4
00134   0 02000 0 00137         TRA LOCAM

                                        COMES HERE IF NUMERIC FIELD.
00135   0 07400 4 00777 LOCAL   TSX NUMBER,4
00136   0 60100 0 77747         STO TEMP
00137   0 07400 4 01476 LOCAM   TSX ACCUM,4         ACCUMULATE RESULTS IN ACC.
00140   0 07400 4 00517         TSX CLEAR,4
00141   0 07400 4 00524         TSX COMPAR,4
00142   730000000000            BCD 1,00000         LOOK FOR COMMA
```

```
00143   0 02000 2 00146          TRA *+3,2,0
00144   0 02000 0 00200          TRA LOCAR          OTHER THAN COMMA.
00145  -0 53400 2 77721          LXD JK1,2          COMMA
00146   0 50000 0 77717          CLA ACC
00147   0 60000 0 77717          STZ ACC            INITIALIZE
00150   0 56000 0 77716          LDQ ILOC1          IS THIS VARIABLE FIXED POINT.
00151   0 16200 0 00153          TQP LOC1           NEGATIVE IS FIXED POINT.
00152   0 07400 4 01470          TSX FIX,4
00153   0 60100 2 00000 LOC1     STO **,2           STORE THE NUMBER RELATIVE TO BASE.
00154  -0 53400 2 77721 LOCAN    LXD JK1,2
00155   1 00001 2 00156 LOCAN1   TXI *+1,2,1        RAISE STORING INDEX BY ONE.
00156  -0 63400 2 77721 LOCAN2   SXD JK1,2          SAVE IT.
00157  -0 53400 1 77754 LOCAN3   LXD OPER,1         ANY OPERATORS LEFT OVER.
00160  -3 00000 1 00163          TXL *+3,1,0
00161   0 07400 4 00564 ERRL     TSX ERROR,4
00162   607443346060             BCD 1 (L)
00163   0 50000 0 77717          CLA ACC            ANY DATA LEFT OVER.
00164  -0 10000 0 00161          TNZ ERRL

                                     CALL THIS THE SWITCH HOUSE.
00165   0 07400 4 00517 LOCA0    TSX CLEAR,4
00166   0 07400 4 00352 LOCAP    TSX CHRCTR,4
00167   0 07400 4 00524 LOCAQ    TSX COMPAR,4
00170   336174000000             BCD 1./(000
00171   0 02002 2 00200          TRA *+7,2,2
00172   0 02000 0 00200          TRA LOCAR          $D, $T, OR OPERATORS.
00173   0 02000 0 00116          TRA LOCAK          ALPHABETIC
00174   0 02000 0 00135          TRA LOCAL          NUMERIC
00175   0 02000 0 00212          TRA LOCAT          ( SIGN
00176   0 02000 0 00200          TRA LOCAR          / SIGN
00177   0 02000 0 00135          TRA LOCAL          DECIMAL
00200  -0 53400 1 77754 LOCAR    LXD OPER,1         ANY OPERATORS LEFT OVER.
00201   3 00000 1 00161          TXH ERRL,1,0       HIGH MEANS ALREADY HAS OPERATOR.
00202   0 40200 0 00350          SUB DOLLAR         SPLIT OFF $ CHARACTER FROM OTHERS.
00203   0 12000 0 00033          TPL LOCAA+3        IF PLUS, PROCESS THE $ TYPE CHARACTER.
                                     WHAT KIND OF OPERATOR IS THIS.
00204   0 07400 4 00524          TSX COMPAR,4
00205   204061547300             BCD 1+-/*,0
00206   3 00005 2 00161          TXH ERRL,2,5       REMOVE THE JUNK.
00207   3 00004 2 00154          TXH LOCAN,2,4      COMMA
00210  -0 63400 2 77754          SXD OPER,2         SAVE REST, WILL BRANCH IN SUB ACCUM
00211   0 02000 0 00166          TRA LOCAP          AFTER BOTH OPERANDS HAVE BEEN FOUND.

                                     COMES HERE IF THE OCT OR ALF MODE.
00212   0 07400 4 00352 LOCAT    TSX CHRCTR,4
00213   0 07400 4 00524          TSX COMPAR,4
00214   344621000000             BCD ))0A000
00215   0 02000 2 00222          TRA *+5,2
00216   0 02000 0 00161          TRA ERRL           JUNK
00217   0 02000 0 00262          TRA LOCAZ          A CHARACTER
00220   0 02000 0 00230          TRA LOCAU          O CHARACTER

                                     COMES HERE IF EMPTY PARENTHESIS WERE FOUND.
00221   0 07400 4 00352          TSX CHRCTR,4       )SIGN,  GET NEXT CHARACTER.
00222   0 12000 0 00224          TPL *+2.           MINUS MEANS NEW CARD.
00223   0 07400 4 01377          TSX TEST,4         INSERT COMMA IF NEEDED.
00224   0 50000 0 77716          CLA ILOC1
00225   0 60100 0 77750          STO ILOC           PREPARE TO GET VALUE OF
00226  -0 53400 2 77721          LXD JK1,2          CURRENT LEFT SIDE.
00227   0 02000 0 00127          TRA SET
```

```
                                        COMES HERE IF OCTAL MODE.
00230   0 07400 4 00352 LOCAU   TSX CHRCTR,4
00231   0 07400 4 00524         TSX COMPAR,4
00232   340000000000            BCD 1)00000
00233   0 02000 2 00236         TRA *+3,2
00234   0 02000 0 00230         TRA LOCAU          OTHERS
00235   0 02000 0 00245         TRA LOCAW          ) SIGN
00236  -0 32000 0 00064 LOCAV   ANA LOCAF+2         SAVE ONLY THE OCTAL NUMBER.
00237   0 60100 0 77755         STO WORD
00240   0 56000 0 77714         LDQ VAR
00241  -0 76300 0 00003         LGL 3              USE THE MQ TO ELIMINATE OVERFLOWS.
00242  -0 60000 0 77714         STQ VAR
00243  -0 50000 0 77755         CAL WORD           ALL NUMBERS MODULO 8.
00244  -0 60200 0 77714         ORS VAR
                                        COMES HERE WHEN ) IS FOUND.
00245   0 07400 4 00352 LOCAW   TSX CHRCTR,4
00246   0 12000 0 00250         TPL *+2
00247   0 07400 4 01377         TSX TEST,4
00250   0 07400 4 00524         TSX COMPAR,4
00251   730000000000            BCD 1,00000
00252   0 02002 2 00257         TRA *+5,2,2
00253   0 02000 0 00337         TRA ERRJ           JUNK
00254   0 02000 0 00337         TRA ERRJ           ALPHABETIC
00255   0 02000 0 00236         TRA LOCAV          NUMERIC
00256  -0 53400 2 77721 LOCAX   LXD JK1,2          COMMA
00257   0 50000 0 77714         CLA VAR
00260   0 02000 0 00153         TRA LOC1
                                                   CONVERT THE NUMBER TO BINARY.
00261   0 07400 4 01453 LOCAY   TSX BINARY,4

                                        COMES HERE IF ALF MODE.
00262   0 07400 4 00352 LOCAZ   TSX CHRCTR,4
00263   0 07400 4 00524         TSX COMPAR,4
00264   340000000000            BCD 1)00000
00265   0 02002 2 00272         TRA *+5,2,2
00266   0 02000 0 00341         TRA ERRK           JUNK
00267   0 02000 0 00262         TRA LOCAZ          ALPHABETIC
00270   0 02000 0 00261         TRA LOCAY          NUMERIC
                                        COMES HERE WHEN ) IS FOUND
00271   0 53400 1 77714 LOCBA   LXA VAR,1           ) SIGN
00272  -2 00000 1 00341         TNX ERRK,1,0       ALF COUNT WAS ZERO.
00273  -0 63400 1 77741         SXD ALF,1
00274   0 07400 4 00517         TSX CLEAR,4
00275   0 07400 4 00352         TSX CHRCTR,4       PULL THROUGH CHARACTERS AND STORE
00276   0 07400 4 01216         TSX STORE,4        THEM ONE AT A TIME.
00277   2 00001 1 00275         TIX *-2,1,1        GO BACK TILL NCHAR IS ONE.
00300  -0 53400 1 77752         LXD J,1
00301  -0 53400 4 77751         LXD MSHIFT,4
00302   0 56000 0 00636         LDQ BLANK
00303  -0 75400 0 00000         PXD 0,0
00304  -0 76300 4 00044         LGL 36,4
00305  -0 60200 1 77715         ORS VAR+1,1        FILL IN PARTIAL WORD WITH BLANKS.
00306   0 53400 4 00303 LOCBB   LXA *-3,4          SET INDEX C TO ZERO.
00307  -0 53400 2 77721         LXD JK1,2
```

```
00310   0 50000 4 77714 LOCBC   CLA VAR,4         PREPARE TO STORE ALPHABETIC WORDS.
00311   0 60100 2 00000 LOC4    STO **,2
00312   1 00001 4 00313         TXI **+1,4,1      J = J + 1
00313  -0 75400 4 00000         PXD 0,4
00314   0 40200 0 77752         SUB J
00315   0 10000 0 00321         TZE LOCBD
00316   1 00001 2 00317         TXI **+1,2,1      JK1 = JK1 + 1
00317  -0 63400 2 77721         SXD JK1,2
00320   0 02000 0 00310         TRA LOCBC
00321   0 60000 0 77741 LOCBD   STZ ALF
00322   0 07400 4 00517         TSX CLEAR,4
00323   0 07400 4 00352         TSX CHRCTR,4      LOOK AT NEXT CHARACTER.
00324   0 12000 0 00326         TPL **+2
00325   0 07400 4 01377         TSX TEST,4        PUT IN COMMA IF NEEDED.
00326   0 07400 4 00524         TSX COMPAR,4
00327   730000000000            BCD 1,00000
00330   0 02000 2 00333         TRA **+3,2,0
00331   0 02000 0 00333         TRA ERRB
00332   0 02000 0 00154         TRA LOCAN         GO RAISE AND STORE JK1.

                                    THESE ARE ERROR CALLS
00333   0 07400 4 00564 ERRB    TSX ERROR,4
00334   607422346060            BCD 1 (B)
00335   0 07400 4 00564 ERRD    TSX ERROR,4
00336   607424346060            BCD 1 (D)
00337   0 07400 4 00564 ERRJ    TSX ERROR,4
00340   607441346060            BCD 1 (J)
00341   0 07400 4 00564 ERRK    TSX ERROR,4
00342   607442346060            BCD 1 (K)
00343   0 07400 4 00564 ERRM    TSX ERROR,4
00344   607444346060            BCD 1 (M)
00345   0 07400 4 00564 ERRU    TSX ERROR,4
00346   607464346060            BCD 1 (U)

00347  +000001000000    ONED    OCT 1000000       PROGRAM
00350   000000005300    DOLLAR  BCD 10000$0         CONSTANTS
00351  +000000000001    ONEA    OCT 1

                                    END OF THE SAP MAIN SEGMENT TO INPUT.
```

```
                                THIS IS SUBROUTINE CHRCTR.  IT STORES SUCCESSIVE
                                CHARACTERS FROM THE CARD AT LOCATION WORD, READS
                                SUCCESSIVE CARDS INTO THE ARRAY RECORD, AND PRINTS
                                $$ TYPE CARDS. THE FIRST CHARACTER FROM A NEW CARD
                                IS STORED IN WORD WITH A MINUS SIGN.

00352 -0 63400 1 77736 CHRCTR SXD TEMP-9,1
00353 -0 63400 2 77735        SXD TEMP-10,2
00354 -0 63400 4 77726        SXD TEMP-17,4
00355  0 60000 0 77742        STZ TAG
00356  0 60000 0 77760        CLA I
00357  0 10000 0 00455        TZE LOCCG          I = 0 MEANS READ A CARD
00360 -0 73400 2 00000        PDX 0,2            I = I IN INDEX B.
00361  0 56000 0 77756        LDQ Q              HAS UNUSED CHARACTERS FROM BEFORE.
00362 -0 53400 1 77757 LOCCA  LXD KK,1           PICK UP CHARACTER COUNTER OF WORD.
00363  0 53400 4 77743        LXA SIGN,4         ZERO UNLESS ON $DATA CARD.
00364  0 60000 0 77741        CLA ALF
00365 -0 10000 0 00367        TNZ *+2            ZERO MEANS NO ALPHABETIC.
00366  1 00001 4 00367        TXI *+1,4,1
00367  1 00001 4 00370        TXI *+1,4,1        C = 2 IF NOT IN ALF MODE.
00370 -0 75400 0 00000 LOCCB  PXD 0,0
00371 -0 76300 0 00006        LGL 6              ONE CHARACTER FROM MQ TO AC.
00372  1 00001 1 00373        TXI *+1,1,1        KK = KK + 1
00373 -3 00005 1 00401        TXL LOCCC,1,5      DONE WITH WORD IF KK OVER 5.
00374 -0 73400 1 00000        PDX 0,1            RESET KK TO ZERO.
00375  0 56000 2 77776        LDQ RECORD,2
00376  1 00001 2 00377        TXI *+1,2,1        RAISE I BY ONE.
00377 -3 00014 2 00401        TXL *+2,2,12       DONE WITH CARD IF I OVER 12.
00400 -0 73400 2 00000        PDX 0,2            RESET I TO ZERO.
00401 -3 00001 4 00417 LOCCC  TXL LOCCF,4,1      GO STORE CHARACTER IF IN ALF MODE.
00402  0 34000 0 00515        CAS BLDG           HUNT FOR BLANK.
00403  0 02000 0 00405        TRA *+2            IS NOT BLANK.
00404  0 02000 0 00415        TRA LOCCE          IS BLANK.
00405  0 34000 0 00516        CAS DOLDG          HUNT FOR $ CHARACTER.
00406  0 02000 0 00410        TRA *+2            IS NOT $ CHARACTER.
00407  0 02000 0 00411        TRA LOCCD          IS $ CHARACTER.
00410  0 02000 0 00417        TRA LOCCF          GO STORE THE CHARACTER.
00411  3 00002 4 00446 LOCCD  TXH PRINT,4,2      TWO $ SIGNS MEANS PRINT CARD.
00412  0 76700 0 00006        ALS 6              SAVE THE 1ST $ CHARACTER AT TAG.
00413  0 60100 0 77742        STO TAG
00414  1 00001 4 00415        TXI LOCCE,4,1      C = C + 1 FOR $ CHARACTER.
00415  3 00000 2 00370 LOCCE  TXH LOCCB,2,0      CONTINUE PROCESSING IF NOT THE
00416  0 02000 0 00455        TRA LOCCG          END OF THE CARD.  OTHERWISE READ
00417 -0 63400 2 77760 LOCCF  SXD I,2            A NEW CARD.
00420 -0 63400 1 77757        SXD KK,1
00421 -0 60000 0 77756        STQ Q              SAVE CHARACTERS LEFT IN MQ.
00422  0 56000 0 77743          RETURN
00423  0 40000 0 77742             TO THE
00424  0 76300 0 00000                CALLING
00425  0 60100 0 77755                  PROGRAM.
00426  0 60000 0 77743        STZ SIGN
00427 -0 53400 4 77726        LXD TEMP-17,4
00430 -0 53400 1 77736        LXD TEMP-9,1
00431 -0 53400 2 77735        LXD TEMP-10,2
00432 -0 76000 0 00012        RTT                IS TAPE CHECK ON.
00433 -0 12000 0 00436        TMI *+3            IF MINUS THIS IS A NEW CARD.
00434  0 60000 0 77734        STZ RTT            RESET ERROR COUNT.
00435  0 02000 4 00001        TRA 1,4            RETURN TO THE CALLING PROGRAM.
```

```
00436 -0 53400 2 77734          LXD RTT,2          COUNT REREADS.
00437 -3 00004 2 00442          TXL *+3,2,4
00440  0 07400 4 00564          TSX ERROR,4        OVER 5 REREADS.
00441  607451636334       R     BCD 1 (RTT)
00442  0 76400 0 00207          BST 7              REREAD THE LAST RECORD (CARD).
00443  1 00001 2 00444          TXI *+1,2,1
00444 -0 63400 2 77734          SXD RTT,2
00445  0 02000 0 00454          TRA LOCCG-1


                                    COMES HERE IF IT IS A $$ CARD
00446  0 76600 0 00206 PRINT    WTD 6
00447 -0 73400 1 00000           PDX 0,1
00450  0 70000 0 00636           CPY BLANK
00451  0 70000 1 77776           CPY RECORD,1
00452  1 00001 1 00453           TXI *+1,1,1
00453 -3 00013 1 00451           TXL *-2,1,11
00454  0 60000 0 77742           STZ TAG            INITIALIZE TAG.
00455  0 60000 0 77757 LOCCG     STZ KK             INITIALIZE KK.
00456  0 53400 2 00447           LXA PRINT+1,2      SET INDEX TO COPY LOOP AT ZERO.
00457  0 53400 4 00447           LXA PRINT+1,4
00460  0 76200 0 00207           RTD 7              READ A NEW CARD.
00461  0 70000 2 77776 READ      CPY RECORD,2
00462  0 02000 0 00470           TRA LOCCH          NORMAL
00463  0 02000 0 00500           TRA LOCCK          END OF FILE
00464  0 50000 0 00636           CLA BLANK          END OF RECORD
00465  3 00013 2 00472           TXH LOCCJ,2,11     FILL IN SHORT RECORDS WITH BLANK
00466  0 60100 2 77776           STO RECORD,2       WORDS.
00467  1 00001 2 00465           TXI *-2,2,1
00470  1 00001 2 00471 LOCCH     TXI *+1,2,1
00471 -3 00013 2 00461           TXL READ,2,11      FINISHED IF INDEX B IS GREATER THAN
00472 -0 53400 2 00401 LOCCJ     LXD LOCCC,2        SET I = 1.                    11.
00473  0 50200 0 77757           CLS KK             PICK UP MINUS ZERO.
00474  0 60100 0 77743           STO SIGN
00475  0 76600 0 00333           IOD
00476  0 56000 0 77776           LDQ RECORD
00477  0 02000 0 00362           TRA LOCCA
00500  0 76600 0 00206 LOCCK     WTD 6              PRINT END OF FILE AND CALL MONITOR.
00501 -0 53400 1 00576           LXD FOUR,1
00502  0 70000 1 00515           CPY OUT+4,1
00503  2 00001 1 00502           TIX *-1,1,1

00504  0 76200 0 00221 LOCOUT    RTB 1              STANDARD
00505  0 76000 0 00140           SLF                  LEWIS RESEARCH CENTER
00506  0 70000 0 00000           CPY 0                  MONITOR
00507  0 70000 0 00001           CPY 1                    CALLING
00510  0 02000 0 00000           TRA 0                      SEQUENCE.

00511  602545246046      OUT     BCD 4 END OF FILE ON TAPE 7.
00512  266026314325
00513  604645606321
00514  472560073360
00515  000000000060      BLDG    BCD 100000
00516  000000000053      DOLDG   BCD 100000$

                                    END OF THE SAP SUBROUTINE CHRCTR.
```

```
                                        THIS IS SUBROUTINE CLEAR.  IT INITIALIZES
                                        NECESSARY PARAMETERS FOR SUBROUTINE STORE.

00517   0 50000 0 00347 CLEAR   CLA ONED
00520   0 60100 0 77752         STO J           SET J EQUAL TO ONE.
00521   0 60000 0 77714         STZ VAR         CLEAR VAR(1).
00522   0 60000 0 77751         STZ MSHIFT      CLEAR MSHIFT.
00523   0 02000 4 00001         TRA 1,4         RETURN TO CALLING PROGRAM

                                        END OF THE SAP SUBROUTINE CLEAR.



                                        THIS IS FUNCTION COMPAR.  IT EXAMINES THE CURRENT
                                        CHARACTER AND TESTS IT AGAINST THE CHARACTERS
                                        FOUND IN THE ARGUMENT. ALPHABETIC AND NUMERIC
                                        SPLITS ARE MADE IF THE CHARACTER IS NOT FOUND
                                        IN THE ARGUMENT. THESE TESTS ARE COUNTED AND
                                        THE NUMBER LEFT IN INDEX 2 CORRESPONDS TO THE
                                        SUCCESSFUL TEST. IF NO TEST IS SUCCESSFUL
                                        THEN INDEX 2 CORRESPONDS TO THE TOTAL TESTS +1.

00524   0 56000 4 00001 COMPAR  LDQ 1,4         USE FIRST ARGUMENT IN CALLING
00525  -0 50000 0 77755         CAL WORD        SEQUENCE AS THE TEST WORD.
00526   0 60100 0 77755         STO WORD        SIGN OF WORD NOW PLUS.
00527   0 53400 2 00351         LXA ONEA,2
00530  -0 75400 0 00000 LOCDA   PXD 0,0
00531  -0 76300 0 00006         LGL 6           PULL IN 1ST TEST CHARACTER.
00532   0 10000 0 00541         TZE LOCDD       DONE IF ZERO.
00533   0 34000 0 77755         CAS WORD        CHECK TEST WORD AGAINST CARD
00534   1 00001 2 00530         TXI LOCDA,2,1   CHARACTER.
00535   0 02000 0 00537         TRA LOCDC       EQUAL.
00536   1 00001 2 00530 LOCDB   TXI LOCDA,2,1   NOT EQUAL.  GET NEXT TEST
00537   0 50000 0 77755 LOCDC   CLA WORD        CHARACTER.
00540   0 02000 4 00002         TRA 2,4         PROGRAM RETURN.
00541   0 50000 4 00002 LOCDD   CLA 2,4         USE SECOND ARGUMENT IN THE CALLING
00542  -0 73400 1 00000         PDX 0,1         SEQUENCE (DECREMENT) AS THE TEST
00543  -2 02000 1 00537         TNX LOCDC,1,1024 FOR ALPHABETIC-NUMERIC SPLIT.
00544   0 50000 0 77755         CLA WORD        LOOK FOR NUMERIC.
00545   0 40200 0 00562         SUB TENA
00546  -0 12000 0 00537         TMI LOCDC       NEGATIVE MEANS NUMERIC.
00547   0 50000 0 77755 LOCDE   CLA WORD        LOOK FOR ALPHABETIC.
00550   0 40200 0 00351         SUB ONEA
00551  -0 32000 0 00563         ANA MASK1
00552   0 40200 0 00561         SUB NINE
00553   0 02000 1 00556         TRA *+3,1
00554   0 02000 0 00557         TRA LOCDF
00555  -0 12000 0 00537         TMI LOCDC
00556   1 00001 2 00537         TXI LOCDC,2,1   ADJUST INDEX B ACCORDING
00557  -0 12000 0 00556 LOCDF   TMI *-1         TO JUNK OR ALPHABETIC.
00560   1 00002 2 00537         TXI LOCDC,2,2

00561 +000000000011     NINE    OCT 11
00562 +000000000012     TENA    OCT 12
00563 +000000007717     MASK1   OCT 7717

                                        END OF THE SAP SUBROUTINE COMPAR.
```

E-1088

THIS IS SUBROUTINE ERROR. IT IS CALLED IF AN
ERROR WAS DETECTED ON ANY OF THE INPUT CARDS.
IT EXITS VIA THE LEWIS RESEARCH CENTER MONITOR.

```
00564   0 50000 4 00001 ERROR  CLA 1,4          GET ARGUMENT FOR PRINTOUT.
00565   0 76600 0 00206        WTD 6
00566   0 70000 4 00001        CPY 1,4          PRINT TYPE OF ERROR.
00567  -0 53400 1 00655        LXD THREE,1      PREPARE TO PRINT SERIES OF BLANKS.
00570   0 34000 0 00441        CAS R            TEST FOR TYPE SPLIT.
00571   1 00001 1 00601        TXI LOCEA,1,1
00572   1 00004 1 00604        TXI LOCEA+3,1,4  (RTT) USES 7 BLANKS.
00573  -0 32000 0 00654        ANA BLOT         ERROR TYPE WAS GREATER THAN R.
00574  -0 73400 2 00000 LOCEC  PDX 0,2          SET INDEX FOR =LONG= PRINT.
00575   0 70000 2 00655        CPY LONG+2,2     COPY NON-STANDARD PRINTOUT.
00576   1 00004 2 00577 FOUR   TXI *+1,2,4      INCREMENT BY FOUR.
00577  -3 00021 2 00575        TXL *-2,2,17     FINISHED IF OVER 17.
00600   0 02000 0 00604        TRA LOCEB
00601   0 70000 0 00631 LOCEA  CPY LOCED        ERROR TYPE WAS LESS THAN R.
00602   0 70000 0 00632        CPY LOCED+1
00603   0 70000 0 00633        CPY LOCED+2
00604   0 70000 0 00636 LOCEB  CPY LONG-13      COPY THREE MORE BLANKS.
00605   2 00001 1 00604        TIX *-1,1,1      LOOP.
00606   0 70000 1 77777        CPY RECORD+1,1   COPY THE CARD IMAGE.
00607   1 00001 1 00610        TXI *+1,1,1
00610  -3 00014 1 00606        TXL *-2,1,12
00611   0 76600 0 00333        IOD
00612   0 56000 0 77757        LDQ KK           PREPARE AN INDEX FOR SHIFTING THE
00613   0 20000 0 00655        MPY THREE        ASTERISK.  MULTIPLY BY SIX.
00614   0 73400 1 00000        PAX 0,1
00615  -0 53400 2 77760        LXD I,2
00616   3 00000 1 00620        TXH *+2,1,0
00617   1 77777 2 00620        TXI *+1,2,-1     REDUCE I BY ONE IF KK IS ZERO.
00620   1 00007 2 00621        TXI *+1,2,7      SET I FOR LOOPING.
00621   0 56000 0 00656        LDQ ASTR
00622  -0 77300 1 00044        RQL 36,1         SHIFT ASTERISK.
00623  -0 60000 0 77747        STQ TEMP
00624   0 76600 0 00206        WTD 6
00625   0 70000 0 00636        CPY LONG-13      COPY A SERIES OF BLANKS.
00626   2 00001 2 00625        TIX *-1,2,1      LOOP.
00627   0 70000 0 77747        CPY TEMP         COPY * SIGN IN PROPER LOCATION.
00630   0 02000 0 00504        TRA LOCOUT

                                      PROGRAM DATA CONSTANTS.

00631  314343252721    LOCED  BCD 1ILLEGA
00632  436023302151           BCD 1L CHAR
00633  212363255133           BCD 1ACTER.
00634  253360606060           BCD 1E.
00635  665146452733           BCD 1WRONG.
00636  606060606060    BLANK  BCD 1
00637  602533606060           BCD 1 E.
00640  266051214527           BCD 1F RANG
00641  452760465160           BCD 1NG OR
00642  632122432533           BCD 1TABLE.
00643  222526465125           BCD 1BEFORE
00644  604664636046           BCD 1 OUT O
00645  604431626231           BCD 1 MISSI
00646  517060314560           BCD 1RY IN
00647  633162622160           BCD 1TISSA
00650  256747464533           BCD 1EXPON.
00651  536063704725           BCD 1$ TYPE
00652  454660254563           BCD 1NO ENT
```

```
00653  454660442145     LONG   BCD 1NO MAN
00654 +000007000000     BLOT   OCT 7000000
00655 +000003000001     THREE  OCT 3000001
00656  606060606054     ASTR   BCD 1     *

                                END OF THE SAP SUBROUTINE ERROR.
```

```
                                THIS IS SUBROUTINE LOOK.  IT SEARCHES THE TABLE
                                FOR THE NAME STORED AT LOCATION VAR.  IF FOUND,
                                THE ACC IS NON-ZERO AT THE RETURN.

00657 -0 63400 4 77733 LOOK    SXD TEMP-12,4     SAVE INDEX REGISTER C.
00660  0 50000 0 77752         CLA J             SUBROUTINE.
00661  0 62200 0 00705         STD LOCFE
00662  0 53400 2 00351         LXA ONEA,2        JK = 1 IN INDEX B.
00663  0 53400 1 00351         LXA ONEA,1        J1 = 1 IN INDEX A.
00664 -0 50000 2 00000 LOCFA   CAL **,2          CAL TABV(JK).
00665 -0 10000 0 00667         TNZ *+2           IF ZERO, NO ENTRY WAS FOUND FOR
00666  0 02000 0 00711         TRA LOCFG         THIS VARIABLE.  EXIT WITH ZERO IN
00667  0 62200 0 00702         STD LOCFD         INDEX B.  DECREMENT HAS NEXT
00670 -0 32000 0 00713         ANA MASK2         ENTRY LOCATION.  SAVE THE DECREMENT
00671  0 40200 0 77752         SUB J             ONLY.   CHECK ENTRY LENGTH.
00672  0 40200 0 00347         SUB ONED
00673 -0 10000 0 00702         TNZ LOCFD         IF NOT THE SAME, LOOK AT NEXT ENTRY.
00674 -0 75400 2 00000         PXD 0,2
00675 -0 73400 4 00000         PDX 0,4           JM = JK IN INDEX C.
00676  0 50000 1 77715 LOCFB   CLA VAR+1,1       SEE IF VAR AND THIS
00677  0 34000 4 00000 LOCFC   CAS **,4          ENTRY AGREE.
00700  0 02000 0 00702         TRA LOCFD         IF NOT SO, GO TO NEXT ENTRY.
00701  0 02000 0 00703         TRA *+2           IF SO, CHECK REST OF NAME.
00702  1 00000 2 00663 LOCFD   TXI LOCFA-1,2,**    IF NOT SO, GO TO NEXT ENTRY.
00703  1 00001 4 00704         TXI *+1,4,1       RAISE JM BY ONE.
00704  1 00001 1 00705         TXI *+1,1,1       RAISE J1 BY ONE.
00705 -3 00000 1 00676 LOCFE   TXL LOCFB,1,**    FINISHED IF J1 IS GREATER THAN J.
00706  0 07400 4 00517         TSX CLEAR,4       CLEAR IF THE ENTRY AGREES.
00707  0 50000 2 00000 LOCFF   CLA **,2
00710  0 60100 0 77750         STO ILOC          SAVE COMMON INDEX AT ILOC.
00711 -0 53400 4 77733 LOCFG   LXD TEMP-12,4     PREPARE TO RETURN.
00712  0 02000 4 00001         TRA 1,4           RETURN TO THE CALLING PROGRAM.

00713 +377777000000     MASK2  OCT 377777000000

                                END OF THE SAP SUBROUTINE LOOK.
```

```
                                    THIS IS SUBROUTINE NAME.  IT IS USED TO
                                    CORRELATE NAMES FROM INPUT CARDS WITH INTERNAL
                                    MEMORY LOCATIONS BY REFERRING TO THE TABLE.

00714 -0 63400 4 77723 NAME    SXD TEMP-20,4     SAVE INDEX C.
                                    GET THE REST OF THE VARIABLE NAME. STOP AT ANY
                                    NON ALPHANUMERIC CHARACTER.
00715  0 07400 4 01216 LOCGB   TSX STORE,4
00716  0 07400 4 00352 LOCGC   TSX CHRCTR,4
00717  0 12000 0 00721         TPL *+2
00720  0 07400 4 01377         TSX TEST,4        COMMA MAY BE NEEDED.
00721 -0 10000 0 00724         TNZ *+3           LOOK FOR ZERO. IF ZERO, MAKE IT
00722  0 36100 0 00776         ACL OH            A LETTER O.
00723  0 60100 0 77755         STO WORD
00724  0 07400 4 00524 LOCGE   TSX COMPAR,4
00725  611374000000            BCD 1/=(000
00726  0 02001 2 00734         TRA *+6,2,1
00727  0 02000 0 00733         TRA LOCGF         JUNK OR OPERATORS
00730  0 02000 0 00715         TRA LOCGB         NUMERIC OR ALPHABETIC
00731  0 02000 0 00737         TRA LOCGG         ( SIGN
00732  0 60000 0 77716         STZ ILOC1         = SIGN
                                    GO TO THE TABLE LOOKUP ROUTINE IF AN = SIGN
                                      OR AN OPERATOR WAS FOUND.
00733  0 07400 4 00657 LOCGF   TSX LOOK,4        FIND THE NAME IN TABLE.
00734  0 10000 0 00741         TZE ERRT          NAME WAS FOUND IN TABLE IF NON-ZERO.
00735  0 53400 2 77750         LXA ILOC,2
00736  0 02000 0 00770         TRA LOCGL

                                    GO TO THE TABLE VARIABLE LOOKUP ROUTINE IF A
                                    ( SIGN WAS FOUND.
00737  0 07400 4 00657 LOCGG   TSX LOOK,4
00740 -0 10000 0 00744         TNZ LOCGJ
00741  0 07400 4 00564 ERRT    TSX ERROR,4
00742  607463346060            BCD 1 (T)
                                    CONVERT THE INDEX TO BINARY.
00743  0 07400 4 01453 LOCGH   TSX BINARY,4
                                    GET THE NUMERICS FOR THE INDEX TO THE VARIABLE.
00744  0 07400 4 00352 LOCGJ   TSX CHRCTR,4
00745  0 07400 4 00524         TSX COMPAR,4
00746  340000000000            BCD 1)00000
00747  0 02002 2 00754         TRA *+5,2,2
00750  0 02000 0 00774         TRA ERRC          JUNK
00751  0 02000 0 00774         TRA ERRC          ALPHABETIC
00752  0 02000 0 00743         TRA LOCGH         NUMERIC
00753  0 07400 4 00352         TSX CHRCTR,4      ) SIGN. GET NEXT CHARACTER.
00754  0 12000 0 00756         TPL *+2           MINUS MEANS FROM NEW CARD.
00755  0 07400 4 01377         TSX TEST,4        COMMA MAYBE NEEDED.
00756  0 07400 4 00524         TSX COMPAR,4
00757  611300000000            BCD 1/=0000

00760  0 02001 2 00765         TRA *+5,2,1
00761  0 02000 0 00764         TRA LOCGK         OPERATORS
00762  0 02000 0 00161         TRA ERRL          ALPHABETIC AND NUMERIC
00763  0 60000 0 77716         STZ ILOC1         = SIGN

00764  0 50000 0 77714 LOCGK   CLA VAR           COMPUTE STORING INDEX.
00765  0 40200 0 00351         SUB ONEA
00766  0 40100 0 77750         ADM ILOC
```

```
00767   0 73400 2 00000          PAX 0,2          STORE ADDRESS AT DECREMENT WITHOUT
00770  -0 63400 2 77737 LOCGL    SXD JK,2         ACCUMULATOR OVERFLOW.
00771   0 50000 0 77716          CLA ILOC1
00772  -0 53400 4 77723          LXD TEMP-20,4    RESTORE INDEX C.
00773   0 02000 4 00001          TRA 1,4          RETURN TO CALLING PROGRAM.
                                      CONSTANTS AND ERROR CALL.
00774   0 07400 4 00564 ERRC     TSX ERROR,4
00775   607423346060             BCD 1 (C)
00776   000000000046     0H      BCD 1000000      5 ZEROS AND ONE 0

                                 END OF THE SAP SUBROUTINE NAME.
```

```
                                 THIS IS SUBROUTINE NUMBER.  IT IS USED TO
                                 ASSEMBLE NUMERIC DATA FROM CARDS.  ALL VALUES ARE
                                 TREATED AS FLOATING POINT NUMBERS IN THIS ROUTINE.

00777  -0 63400 4 77720 NUMBER SXD TEMP-23,4    SAVE INDEX C.
01000  -0 63400 4 77745         SXD KNT2,4       INITIALIZE
01001   0 60000 0 77744         STZ KNT3           THE SUBROUTINE
01002   0 60000 0 77746         STZ KNT1             BRANCH PARAMETERS.
01003   0 60000 0 77715         STZ KNT4
01004   0 60000 0 77747         STZ TEMP
01005   0 50000 0 01214         CLA LOCHR        INITIALIZE THE TRANSFER AT LOCHD TO
01006   0 62100 0 01035         STA LOCHD        LOCHD1.
01007   0 02000 0 01013         TRA LOCHB

01010   0 07400 4 00352 LOCHA   TSX CHRCTR,4
01011   0 12000 0 01013         TPL LOCHB
01012   0 07400 4 01377         TSX TEST,4
01013   0 07400 4 00524 LOCHB   TSX COMPAR,4
01014   332561000000            BCD 1.E/000
01015   0 02002 2 01024         TRA *+7,2,2
01016   0 02000 0 01113         TRA LOCHK        JUNK OR AN OPERATOR
01017   0 02000 0 01174         TRA ERRE         ALPHABETIC
01020   0 02000 0 01032         TRA LOCHC        NUMERIC
01021   0 02000 0 01113         TRA LOCHK        SLASH
01022   0 02000 0 01050         TRA LOCHE        E
01023   0 50000 0 77745         CLA KNT2         DECIMAL POINT.
01024  -0 10000 0 01027         TNZ *+3          ZERO MEANS THIS IS THE SECOND POINT.
01025   0 07400 4 00564         TSX ERROR,4
01026   607445346060     N      BCD 1 (N)
01027   0 60000 0 77745         STZ KNT2
01030   0 60000 0 77753         STZ NEXP
01031   0 02000 0 01010         TRA LOCHA
01032   0 50000 0 77753 LOCHC   CLA NEXP         COUNT THE NUMBER OF DIGITS BEHIND
01033   0 40000 0 00351         ADD ONEA         THE DECIMAL POINT IF THERE IS ONE.
01034   0 60100 0 77753         STO NEXP
01035   0 02000 0 00000 LOCHD   TRA **           EITHER LOCHD1 OR LOCHD2.
01036   0 07400 4 01453 LOCHD1  TSX BINARY,4     CONVERT THE DIGIT TO BINARY.
01037   0 10000 0 01010         TZE LOCHA        DO NOT COUNT LEADING ZEROS.
01040   0 50000 0 77746 LOCHD2  CLA KNT1         COUNT THE TOTAL NUMBER OF DIGITS.
01041   0 40000 0 00351         ADD ONEA
01042   0 60100 0 77746         STO KNT1
```

```
01043   0 40200 0 00562          SUB TENA
01044  -0 10000 0 01010          TNZ LOCHA
01045   0 50000 0 01215          CLA LOCHS        PULL THROUGH REMAINING DIGITS.
01046   0 62100 0 01035          STA LOCHD        TURN OFF ACCUMALATION OF DIGITS.
01047   0 02000 0 01010          TRA LOCHA
                                     COMES HERE WHEN THE EXPONENT FIELD IS
01050   0 50000 0 77746 LOCHE    CLA KNT1         ENCOUNTERED.
01051  -0 10000 0 01064          TNZ LOCHH        THERE MUST BE AT LEAST ONE DIGIT
01052   0 07400 4 00564          TSX ERROR,4      BEFORE THE E OF AN E FORMAT NUMBER.
01053   607462346060             BCD 1 (S)
01054   0 50000 0 77744 LOCHF    CLA KNT3         SEE IF EXPONENT DIGITS HAVE ARRIVED.
01055   0 02000 0 01057          TRA *+2
01056   0 50200 0 77744 LOCHG    CLS KNT3         SEE IF EXPONENT DIGITS HAVE ARRIVED.
01057  -0 10000 0 01111          TNZ LOCHK-2      NON ZERO MEANS SIGN IS OPERATOR.
01060   0 60100 0 77747          STO TEMP         STORE SIGN OF EXPONENT.
01061   0 50000 0 77715          CLA KNT4
01062  -0 10000 0 01176          TNZ ERRF         NONZERO MEANS MORE THAN 1 EXP SIGN.
01063  -0 63400 2 77715          SXD KNT4,2       MAKE NOZERO.
01064   0 07400 4 00352 LOCHH    TSX CHRCTR,4
01065   0 12000 0 01067          TPL *+2
01066   0 07400 4 01377          TSX TEST,4
01067   0 07400 4 00524          TSX COMPAR,4
01070   204061330000             BCD 1+-/.00
01071   0 02002 2 01101          TRA *+8,2,2
01072   0 02000 0 01111          TRA LOCHK-2      OTHERS
01073   0 02000 0 01176          TRA ERRF         ALPHABETIC
01074   0 02000 0 01101          TRA LOCHJ        NUMERIC
01075   0 02000 0 01176          TRA ERRF         DECIMAL
01076   0 02000 0 01111          TRA LOCHK-2      SLASH
01077   0 02000 0 01056          TRA LOCHG        MINUS
01100   0 02000 0 01054          TRA LOCHF        PLUS
                                     CONVERT THE EXPONENT TO BINARY.
01101   0 50000 0 77747 LOCHJ    CLA TEMP
01102   0 76700 0 00002          ALS 2
01103   0 40000 0 77747          ADD TEMP
01104   0 76700 0 00001          ALS 1
01105   0 36100 0 77755          ACL WORD
01106   0 60100 0 77747          STO TEMP
01107  -0 63400 2 77744          SXD KNT3,2       RECORD FACT FOR SECOND SIGN.
01110   0 02000 0 01064          TRA LOCHH
                                     COMES HERE WHEN AN OPERATOR WAS FOUND.
01111   0 50000 0 77744          CLA KNT3        TEST FOR THE  PRESENCE OF EXPONENT.
01112   0 10000 0 01176          TZE ERRF          ZERO MEANS NO EXPONENT CAME.
01113   0 50000 0 77745 LOCHK    CLA KNT2
01114   0 10000 0 01116          TZE *+2
01115   0 60000 0 77753          STZ NEXP
01116   0 50000 0 77746          CLA KNT1         SEE IF MORE THAN TEN NUMBERS HAVE
01117   0 40200 0 00562          SUB TENA         BEEN CONVERTED.
01120   0 12000 0 01122          TPL *+2          IF SO, USE THE DIFFERENCE IN THE
01121  -0 75400 0 00000          PXD 0,0          COMPUTATION OF THE EXPONENT.
01122   0 40200 0 77753          SUB NEXP
01123   0 40000 0 77747          ADD TEMP
01124   0 60100 0 77753          STO NEXP
                                     MANTISSA IN VAR AND THE EXPONENT IS IN NEXP.
01125   0 50000 0 77714          CLA VAR
01126   0 10000 0 01172          TZE LOCHQ        SHORT CUT IF ZERO.
01127   0 62100 0 01203          STA K1
01130   0 77100 0 00017          ARS 15
01131  -0 50100 0 01204          ORA K2
01132   0 30000 0 01204          FAD K2
```

```
01133   0 60100 0 77714          STO VAR
01134   0 76000 0 00000          CLM
01135  -0 50100 0 01203          ORA K1
01136   0 30000 0 77714          FAD VAR
01137  -0 77300 0 00010          RQL 8
01140   0 76000 0 00010          RND
01141  -0 50100 0 01205          ORA K3
01142   0 60100 0 77714          STO VAR
01143   0 50000 0 77753          CLA NEXP
01144   0 10000 0 01171          TZE LOCHP          IF ZERO, NO EXPONENT COMPUTATION
01145  -0 53400 2 00347          LXD ONED,2         NECESSARY.
01146   0 56000 0 01202          LDQ FLOAT1         PUT A ONE IN THE MQ.
01147   0 76000 0 00001 LOCHL    LBT                EXPONENT IS IN ACCUMULATOR.
01150   0 02000 0 01157          TRA LOCHM
01151   3 00006 2 01200          TXH ERRV,2,6       EXPONENT GREATER THAN 64
01152   0 60100 0 77711          STO VAR-3
01153   0 26000 2 01214          FMP TAB+1,2        COMPUTE POWERS OF TEN.
01154   0 60100 0 77712          STO VAR-2
01155   0 56000 0 77712          LDQ VAR-2
01156   0 50000 0 77711          CLA VAR-3
01157   0 77100 0 00001 LOCHM    ARS 1              CHECK NEXT BIT OF EXPONENT.
01160   0 10000 0 01162          TZE LOCHN
01161   1 00001 2 01147          TXI LOCHL,2,1
01162  -0 12000 0 01165 LOCHN    TMI LOCHO          IF NEGATIVE, PERFORM DIVISION.
01163   0 26000 0 77714          FMP VAR            IF POSITIVE, PERFORM MULTIPLICATION.
01164   0 02000 0 01172          TRA LOCHQ
01165  -0 60000 0 77712 LOCHO    STQ VAR-2
01166   0 50000 0 77714          CLA VAR
01167   0 24100 0 77712          FDP VAR-2
01170  -0 60000 0 77714          STQ VAR
01171   0 50000 0 77714 LOCHP    CLA VAR
01172  -0 53400 4 77720 LOCHQ    LXD TEMP-23,4      RESTORE INDEX C.
01173   0 02000 4 00001          TRA 1,4            RETURN TO CALLING PROGRAM.
                                  THESE ARE THE ERROR CALLS FOR SUB NUMBR.
01174   0 07400 4 00564 ERRE     TSX ERROR,4
01175   607425346060             BCD 1 (E)
01176   0 07400 4 00564 ERRF     TSX ERROR,4
01177   607426346060             BCD 1 (F)


01200   0 07400 4 00564 ERRV     TSX ERROR,4
01201   607465346060             BCD 1 (V)
01202 +201400000000     FLOAT1   DEC 1.
                                  THESE ARE THE OCTAL CONSTANTS TO BE USED WITH
01203 +233000000000     K1       OCT 233000000000          THE DBC ROUTINE.
01204 +252000000000     K2       OCT 252000000000
01205 +000400000000     K3       OCT 400000000
                                  THIS IS THE FLOATING PT. TABLE USED IN DBC
01206 +353473426555              DEC 1E+32                 CONVERSION.
01207 +266434157116              DEC 1E+16
01210 +233575360400              DEC 1E+08
01211 +216470400000              DEC 1E+04
01212 +207620000000              DEC 1E+02
01213 +204500000000     TAB      DEC 10.
01214   0 00000 0 01036 LOCHR    HTR LOCHD1
01215   0 00000 0 01040 LOCHS    HTR LOCHD2

                                  END OF THE SAP SUBROUTINE NUMBER.
```

```
                                    THIS IS SUBROUTINE STORE.  IT STORES CHARACTERS
                                    AT THE ARRAY VAR.

01216 -0 63400 1 77732 STORE  SXD TEMP-13,1     SAVE INDEX A.
01217 -0 63400 2 77731        SXD TEMP-14,2     SAVE INDEX B.
01220 -0 53400 1 77752        LXD J,1           PUT J INTO INDEX REGISTER A.
01221 -0 53400 2 77751 LOCJA  LXD MSHIFT,2      LOAD INDEX B WITH MSHIFT.
01222 -3 00036 2 01227        TXL LOCJB,2,30    RESET MSHIFT IF IT IS OVER 30.
01223  0 60000 0 77751        STZ MSHIFT
01224  1 00001 1 01225        TXI *+1,1,1       RAISE J BY ONE IF MSHIFT IS OVER
01225  0 60000 1 77715        STZ VAR+1,1       MAXIMUM.
01226  0 02000 0 01221        TRA LOCJA
01227  0 56000 0 77755 LOCJB  LDQ WORD          PUT WORD IN MQ TO BE SHIFTED.
01230 -0 77300 2 00036        RQL 30,2          SHIFT WORD THE CORRECT NUMBER OF
01231 -0 60000 0 77740        STQ TEMP-7        PLACES TO THE LEFT.
01232 -0 50000 0 77740        CAL TEMP-7
01233 -0 60200 1 77715        ORS VAR+1,1       STORE THE CHARACTER AT VAR.
01234  1 00006 2 01235        TXI *+1,2,6       RAISE MSHIFT BY SIX.
01235 -0 63400 2 77751        SXD MSHIFT,2      SAVE MSHIFT.
01236 -0 63400 1 77752        SXD J,1           SAVE J.
01237 -0 53400 1 77732        LXD TEMP-13,1     RESTORE INDEX A.
01240 -0 53400 2 77731        LXD TEMP-14,2     RESTORE INDEX B.
01241  0 02000 4 00001        TRA 1,4           RETURN TO CALLING PROGRAM.

                                    END OF THE SAP SUBROUTINE STORE.
```

```
                                    THIS IS SUBROUTINE TABLE.  IT IS USED TO
                                    CONSTRUCT A TABLE OF NAMES TO BE USED ON CARDS
                                    AND THEIR MEMORY LOCATIONS RELATIVE TO ARG 2 OF
                                    THE CALLING SEQUENCE.

01242 -0 63400 4 77730 TABLE  SXD TEMP-15,4     SAVE INDEX C.
01243  0 50000 0 00707        CLA LOCFF         INITIALIZE ADDRESSES TO 3RD ARG + 1.
01244  0 62100 0 01343        STA LOCKL
01245  0 62100 0 01352        STA LOCKN
01246  0 62100 0 01370        STA LOCKS-1
01247  0 62100 0 01371        STA LOCKS
01250  0 62100 0 01355        STA LOCKO
01251  0 62100 0 01347        STA LOCKM+1
01252  0 50000 0 00347        CLA ONED          INITIALIZE J FOR TABLE LOOKUP.
01253  0 60100 0 77752        STO J
01254  0 50000 0 01263        CLA LOCKA+2       GIVE IT AN IMPOSSIBLE WORD.
01255  0 60100 0 77714        STO VAR
01256  0 07400 4 00657        TSX LOOK,4        NO MATCH FOR IMPOSSIBLE WORD IN
01257 -0 63400 2 77727        SXD TEMP-16,2     THE TABLE GIVES NEXT FREE LOCATION.
01260  0 60000 0 77747        STZ TEMP
01261  0 07400 4 00352 LOCKA  TSX CHRCTR,4
01262  0 07400 4 00524        TSX COMPAR,4
01263  730000000000           BCD 1,00000
01264  0 02002 2 01271        TRA *+5,2,2
01265  0 02000 0 01301        TRA LOCKD+1       JUNK
01266  0 02000 0 01261        TRA LOCKA         ALPHABETIC
01267  0 02000 0 01301        TRA LOCKD+1       NUMERIC
01270  0 60000 0 77747 LOCKB  STZ TEMP          COMMA
01271  0 02000 0 01300        TRA LOCKD
```

```
                                        COMES HERE TO CONVERT THE ADDRESS TO OCTAL FOR
01272   0 50000 0 77747 LOCKC   CLA TEMP          THE TABLE.
01273   0 76700 0 00002         ALS 2
01274   0 40000 0 77747         ADD TEMP
01275   0 76700 0 00001         ALS 1
01276   0 40000 0 77755         ADD WORD
01277   0 60100 0 77747         STO TEMP

                                        COMES HERE TO GET NUMERICS.
01300   0 07400 4 00352 LOCKD   TSX CHRCTR,4
01301   0 07400 4 00524         TSX COMPAR,4
01302   331361000000            BCD 1.=/000
01303   0 02002 2 01312         TRA **7,2,2
01304   0 02000 0 01373         TRA ERRA            JUNK
01305   0 02000 0 01373         TRA ERRA            ALPHABETIC
01306   0 02000 0 01272         TRA LOCKC           NUMERIC
01307   0 02000 0 01365         TRA LOCKQ           / CHARACTER
01310   0 02000 0 01315         TRA LOCKF           = SIGN

                                        COMES HERE IF A DECIMAL PT WAS FOUND.
01311   0 50000 0 77747 LOCKE   CLA TEMP            DECIMAL PT
01312  -0 76000 0 00003         SSM
01313   0 60100 0 77747         STO TEMP
01314   0 02000 0 01300         TRA LOCKD

                                        COMES HERE IF AN = SIGN WAS FOUND.
01315   0 07400 4 00517 LOCKF   TSX CLEAR,4
01316   0 07400 4 00352 LOCKG   TSX CHRCTR,4

01317   0 12000 0 01321         TPL **2
01320   0 07400 4 01377         TSX TEST,4
01321   0 10000 0 01332         TZE LOCKH
01322   0 07400 4 00524         TSX COMPAR,4
01323   617300000000            BCD 1/,0000
01324   0 02001 2 01331         TRA **5,2,1
01325   0 02000 0 01375         TRA ERRG            JUNK
01326   0 02000 0 01334         TRA LOCKJ           ALPHABETIC OR NUMERIC
01327  -0 63400 2 77753         SXD B,2             COMMA
01330  -0 63400 2 77753         SXD B,2             SLASH
01331   0 02000 0 01336         TRA LOCKK

                                        COMES HERE TO STORE CHARACTER.
01332   0 36100 0 00776 LOCKH   ACL OH              REPLACE ZERO BY CHARACTER O.
01333   0 60100 0 77755         STO WORD
01334   0 07400 4 01216 LOCKJ   TSX STORE,4
01335   0 02000 0 01316         TRA LOCKG
                                        COMES HERE AT END OF NAME.
01336   0 07400 4 00657 LOCKK   TSX LOOK,4
01337  -0 10000 0 01367         TNZ LOCKR           GOES TO LOCKR IF THERE IS AN ENTRY
01340  -0 53400 1 77727         LXD TEMP-16,1       EQUAL.
01341  -0 53400 4 77727         LXD TEMP-16,4
01342   0 50000 0 77747         CLA TEMP
01343   0 60100 1 00000 LOCKL   STO **,1            STORE THE NAME IN THE TABLE.
01344   1 00001 1 01345         TXI **1,1,1
01345  -0 53400 2 00347         LXD ONED,2
01346   0 50000 2 77715 LOCKM   CLA VAR+1,2
01347   0 60100 1 00000         STO **,1
01350   1 00001 1 01351         TXI **1,1,1
01351  -0 63400 1 77727         SXD TEMP-16,1
01352   0 60000 1 00000 LOCKN   STZ **,1
```

```
01353   1 00001 2 01354          TXI **+1,2,1
01354  -0 75400 2 00000          PXD 0,2
01355   0 62200 4 00000 LOCKO    STD **,4
01356   0 40200 0 77752          SUB J
01357   0 40200 0 00347          SUB ONED
01360   0 10000 0 01362          TZE LOCKP
01361  -0 12000 0 01346          TMI LOCKM

                                     REEXAMINE THE CUT OFF CHARACTER.
01362  -0 53400 2 77753 LOCKP    LXD 8,2
01363   0 02000 2 01366          TRA **+3,2
01364   0 02000 0 01270          TRA LOCKB          COMMA
01365  -0 53400 4 77730 LOCKQ    LXD TEMP-15,4      / CHARACTER
01366   0 02000 4 00001          TRA 1,4            RETURN.

                                     COMES HERE TO REPLACE NAME
01367  -0 50000 0 77747 LOCKR    CAL TEMP
01370   0 63000 2 00000          STP **,2
01371   0 62100 2 00000 LOCKS    STA **,2
01372   0 02000 0 01362          TRA LOCKP

                                     THESE ARE THE ERROR CALLS.
01373   0 07400 4 00564 ERRA     TSX ERROR,4
01374   607421346060             BCD 1 (A)
01375   0 07400 4 00564 ERRG     TSX ERROR,4
01376   607427346060             BCD 1 (G)

                                     END OF THE SAP SUBROUTINE TABLE
```

THIS IS SUBROUTINE TEST. IT LOOKS AHEAD TO CLASSIFY
A NEW CARD. ACOMMA WILL BE PUT INTO THE CURRENT
CHARACTER POSITION ONLY IF EITHER (1) THE NEXT
CARD BEGINS WITH A $ SIGN FOLLOWED BY SOME OTHER
CHARACTER OR (2) THE NEXT CARD BEGINS WITH AN
ALPHABETIC AND AN = SIGN IS FOUND AND IT PRECEEDS
ALL , AND $ SIGNS ON THAT CARD.

```
01377 -0 63400 4 77733 TEST   SXD TEMP-12,4     SAVE INDEX FOR RETURN.
01400  0 40000 0 00350        ADD DOLLAR         TEST FOR A $ SIGN.
01401 -0 12000 0 01440        TMI LOCLA          POSITIVE MEANS NOT A $ SIGN.
01402  0 07400 4 00524        TSX COMPAR,4       IS THIS CHARACTER ALPHABETIC.
01403  000000000000           BCD 1000000
01404  0 02002 2 01410        TRA *+4,2,2
01405  0 02000 0 01447        TRA LOCLB          OTHERS
01406  0 02000 0 01410        TRA LOCLC          ALPHABETIC AND /
01407  0 02000 0 01447        TRA LOCLB          NUMERIC
01410  0 53400 1 01414 LOCLC  LXA LOCLD,1        THE CHARACTER WAS NOT A $ SIGN.
01411  3 00013 1 01447        TXH LOCLB,1,11     DONE IF WHOLE CARD SCANNED.
01412  0 56000 1 77776        LDQ RECORD,1
01413  0 53400 2 01414        LXA LOCLD,2
01414 -0 75400 0 00000 LOCLD  PXD 0,0
01415 -0 76300 0 00006        LGL 6
01416 -0 60000 0 77725        STQ TEMP-18
01417 -0 10000 0 01424        TNZ LOCLE
01420 -0 50000 0 77725        CAL TEMP-18        SEE IF REST OF MQ IS ZERO.
01421 -0 10000 0 01414        TNZ LOCLD          GET NEXT CHARACTER IF MQ IS NOT
01422  0 50000 0 77755        CLA WORD                                    ZERO.
01423  1 00001 1 01411        TXI LOCLC+1,1,1
01424  0 56000 0 01452 LOCLE  LDQ TST1
01425  0 60100 0 77724        STO TEMP-19
01426 -0 75400 0 00000 LOCLF  PXD 0,0
01427 -0 76300 0 00006        LGL 6
01430 -0 10000 0 01433        TNZ LOCLG
01431  0 56000 0 77725        LDQ TEMP-18
01432  0 02000 0 01413        TRA LOCLD-1
01433  0 34000 0 77724 LOCLG  CAS TEMP-19
01434  0 02000 0 01436        TRA LOCLH
01435  0 02000 0 01437        TRA LOCLJ
01436  1 00001 2 01426 LOCLH  TXI LOCLF,2,1
01437 -3 00001 2 01447 LOCLJ  TXL LOCLB,2,1
01440  0 60000 0 77757 LOCLA  STZ KK
01441 -0 53400 1 01423        LXD LOCLE-1,1
01442 -0 63400 1 77760        SXD I,1
01443  0 50000 0 77776        CLA RECORD
01444  0 60100 0 77756        STO Q
01445  0 50000 0 01451        CLA COMMA          SUBSTITUTE A COMMA IF NEEDED.
01446  0 60100 0 77755        STO WORD
01447 -0 53400 4 77733 LOCLB  LXD TEMP-12,4
01450  0 02000 4 00001        TRA 1,4            RETURN TO THE CALLING PROGRAM.

01451  000000000073    COMMA  BCD 100000,
01452  735313000000    TST1   BCD 1,$=000
```

END OF THE SAP SUBROUTINE TEST.

THE FOLLOWING FOUR SUBROUTINES ARE USED TO CONVERT DECIMAL DIGITS TO BINARY IN VAR, FIX FLOATING POINT NUMBERS, FLOAT FIXED POINT NUMBERS, AND FORM ARITHMETIC RESULTS IN THE PSEUDO ACCUMULATOR (ACC) FOR EACH OPERATION ON A CARD.

```
01453   0 50000 0 77714 BINARY CLA VAR          ACCUMULATE A SERIES OF BASE 10
01454   0 76700 0 00002        ALS 2            DIGITS IN BINARY IN VAR.
01455   0 40000 0 77714        ADD VAR
01456   0 76700 0 00001        ALS 1
01457   0 36100 0 77755        ACL WORD
01460   0 60100 0 77714        STO VAR
01461   0 02000 4 00001        TRA 1,4

01462   0 50000 0 77747 FLT    CLA TEMP         CONVERT TO FLOATING POINT THE
01463   0 76500 0 00022        LRS 18           CONTENTS OF THE STORAGE CALLED
01464  -0 50100 0 01520        ORA EXP          TEMP.
01465   0 30000 0 01520        FAD EXP
01466   0 60100 0 77747        STO TEMP         LEAVE THE ANSWER IN TEMP.
01467   0 02000 4 00001        TRA 1,4

01470  -0 30000 0 01520 FIX    UFA EXP          CONVERT TO FIXED POINT THE CONTENTS
01471   0 76500 0 00000        LRS 0            OF THE ACCUMULATOR.
01472  -0 32000 0 01517        ANA FIXED
01473   0 76300 0 00000        LLS 0
01474   0 76700 0 00022        ALS 18           LEAVE THE FIXED POINT NUMBER IN
01475   0 02000 4 00001        TRA 1,4          THE ACCUMULATOR.

01476  -0 53400 2 77754 ACCUM  LXD OPER,2       BRANCH FOR OPERATOR
01477   0 60000 0 77754        STZ OPER         PREPARE FOR NEXT OPERATOR.
01500   0 50000 0 77747        CLA TEMP
01501   0 02000 2 01506        TRA *+5,2
01502   0 02000 0 01514        TRA LOCMB        *
01503   0 02000 0 01510        TRA LOCMA        /
01504   0 76000 0 00002        CHS              MINUS
01505   0 30000 0 77717        FAD ACC          PLUS
01506   0 60100 0 77717        STO ACC          NONE
01507   0 02000 4 00001        TRA 1,4

01510   0 50000 0 77717 LOCMA  CLA ACC          DIVIDE.
01511   0 24100 0 77747        FDP TEMP
01512  -0 60000 0 77717        STQ ACC
01513   0 02000 4 00001        TRA 1,4

01514   0 56000 0 77717 LOCMB  LDQ ACC          MULTIPLY.
01515   0 26000 0 77747        FMP TEMP
01516   0 02000 0 01506        TRA LOCMA-2

01517 +000000077777    FIXED  OCT 77777
01520 +233000000000    EXP    OCT 233000000000
```

END OF THE SAP SUBROUTINES ACCUM, FIX, FLOAT.

NASA TN D-1092
National Aeronautics and Space Administration.
AN INPUT ROUTINE USING ARITHMETIC STATEMENTS FOR THE IBM 704 DIGITAL COMPUTER.
Don N. Turner and Vearl N. Huff. September 1961.
47p. OTS price, $1.25.
(NASA TECHNICAL NOTE D-1092)

An input routine has been designed for use with FORTRAN or SAP coded programs which are to be executed on an IBM 704 digital computer. All input to be processed by the routine is punched on IBM cards as declarative statements of the arithmetic type resembling the FORTRAN language. The routine is 850 words in length. It is capable of loading fixed- or floating-point numbers, octal numbers, and alphabetic words, and of performing simple arithmetic as indicated on input cards. Provisions have been made for rapid loading of arrays of numbers in consecutive memory locations.

Copies obtainable from NASA, Washington

I. Turner, Don N.
II. Huff, Vearl N.
III. NASA TN D-1092

(Initial NASA distribution: 49, Simulators and computers.

NASA

NASA TN D-1092
National Aeronautics and Space Administration.
AN INPUT ROUTINE USING ARITHMETIC STATEMENTS FOR THE IBM 704 DIGITAL COMPUTER.
Don N. Turner and Vearl N. Huff. September 1961.
47p. OTS price, $1.25.
(NASA TECHNICAL NOTE D-1092)

An input routine has been designed for use with FORTRAN or SAP coded programs which are to be executed on an IBM 704 digital computer. All input to be processed by the routine is punched on IBM cards as declarative statements of the arithmetic type resembling the FORTRAN language. The routine is 850 words in length. It is capable of loading fixed- or floating-point numbers, octal numbers, and alphabetic words, and of performing simple arithmetic as indicated on input cards. Provisions have been made for rapid loading of arrays of numbers in consecutive memory locations.

Copies obtainable from NASA, Washington

I. Turner, Don N.
II. Huff, Vearl N.
III. NASA TN D-1092

(Initial NASA distribution: 49, Simulators and computers.

NASA

NASA TN D-1092
National Aeronautics and Space Administration.
AN INPUT ROUTINE USING ARITHMETIC STATEMENTS FOR THE IBM 704 DIGITAL COMPUTER.
Don N. Turner and Vearl N. Huff. September 1961.
47p. OTS price, $1.25.
(NASA TECHNICAL NOTE D-1092)

An input routine has been designed for use with FORTRAN or SAP coded programs which are to be executed on an IBM 704 digital computer. All input to be processed by the routine is punched on IBM cards as declarative statements of the arithmetic type resembling the FORTRAN language. The routine is 850 words in length. It is capable of loading fixed- or floating-point numbers, octal numbers, and alphabetic words, and of performing simple arithmetic as indicated on input cards. Provisions have been made for rapid loading of arrays of numbers in consecutive memory locations.

Copies obtainable from NASA, Washington

I. Turner, Don N.
II. Huff, Vearl N.
III. NASA TN D-1092

(Initial NASA distribution: 49, Simulators and computers.

NASA

NASA TN D-1092
National Aeronautics and Space Administration.
AN INPUT ROUTINE USING ARITHMETIC STATEMENTS FOR THE IBM 704 DIGITAL COMPUTER.
Don N. Turner and Vearl N. Huff. September 1961.
47p. OTS price, $1.25.
(NASA TECHNICAL NOTE D-1092)

An input routine has been designed for use with FORTRAN or SAP coded programs which are to be executed on an IBM 704 digital computer. All input to be processed by the routine is punched on IBM cards as declarative statements of the arithmetic type resembling the FORTRAN language. The routine is 850 words in length. It is capable of loading fixed- or floating-point numbers, octal numbers, and alphabetic words, and of performing simple arithmetic as indicated on input cards. Provisions have been made for rapid loading of arrays of numbers in consecutive memory locations.

Copies obtainable from NASA, Washington

I. Turner, Don N.
II. Huff, Vearl N.
III. NASA TN D-1092

(Initial NASA distribution: 49, Simulators and computers.

NASA