ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

CESTI ~~201-2~~

H.C. \$3.00 MM. 65

MACRO-FORTRAN,
A Facility for Programmer-defined
Macro-Instructions in FORTRAN Programs

by

George A. Robinson*

Applied Mathematics Division

*Now at Computer Sciences Department,
The University of Wisconsin, Madison

June 1967

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION.	5
II. THE MACRO-FORTRAN LANGUAGE.	7
A. Psuedo-operations	8
B. Macro Definitions and Macro Calls	10
C. Examples.	11
D. Permitted Contexts.	13
III. DECK STRUCTURE FOR 3600 MACRO-FORTRAN	14
IV. DIAGNOSTIC MESSAGES	14
V. IMPLEMENTATION.	15

MACRO-FORTRAN,
A FACILITY FOR PROGRAMMER-DEFINED
MACRO-INSTRUCTIONS IN FORTRAN PROGRAMS

by

George A. Robinson

I. INTRODUCTION

A computer program is an algorithm specifying how to carry out the computations required to solve a certain class of problems. The exact sequence of computations carried out by a given program may vary, depending on choices of the input parameters specified when the program is to be executed. A (closed) subroutine is (usually) an algorithm for carrying out a task required more than one place in a program. A number of reasons commonly exist for splitting off the program for a closed subroutine as an independent entity from the main program, rather than incorporating a copy of the subroutine each place it is used. Two important reasons are:

1. Saving of machine memory space by having only one copy in memory.
2. Reduction of tedious clerical work by the programmer, since only one copy must be supplied.

When the sequence of computations to be performed depends only on the value of a small number of arguments that all vary dynamically at execution time, use of the closed subroutine is usually the most efficient technique. However, when either

1. Some of the parameters depend only on the point in the main program at which the task must be performed (and therefore remain constant during execution time), or
2. Several parameters with no simple interrelations must be communicated to the subroutine,

then the use of a macro-instruction may substantially enhance efficiency.

A macro-instruction is essentially an algorithm for *generating* an algorithm (i.e., for generating a sequence of statements in some algorithmic language). For example, suppose we desire to write an algorithm to generate a FORTRAN algorithm to form

$$y = \sum_{i=j}^k a_i x^i,$$

where j , k , x , and the a_i ($k \geq i \geq j$) are to be specified at compile time and the a_i are either nonnegative constants or variables. To make the example interesting, we require that the terms $a_i x^i$ actually be generated and summed rather than using some alternate technique, and we assume that FORTRAN

can handle exponents only ≥ 2 . Suppose that $k = 3$, $j = -2$, $x = AB$, $a_3 = 7.4$, $a_2 = 0$, $a_1 = 152.43$, $a_0 = 79.0$, $a_{-1} = 46.5$, and $a_{-2} = 54.7$ are the values supplied at compile time. We should like the macro-algorithm to generate

$$Y = 7.4*AB**3+152.43*AB+79.0+46.5/AB+54.7/AB**2.$$

Two properties are particularly worth noting here: the form of the i th term depends on the value of i , and the presence or absence of the i th term depends on whether a_i is nonzero.

A possible way of writing the macro-algorithm (although not precisely the way to be used in the system we are to describe) might be

Define $PQR(x,k,j,a_k,\dots,a_j)$ as

Generate $Y =$

For $i = k,\dots,j$ do

If $a_i \neq 0$ do

If $i < k$ generate +

Generate a_i

If $i > 1$ generate $*x**i$

If $i = 1$ generate $*x$

If $i = -1$ generate $/x$

If $i < -1$ generate $/x**(-i)$.

Then $PQR(AB,3,-2,7.4,0,152.43,79.0,46.5,54.7)$ would generate the algorithm for the case described above.

By contrast, $PQR(Q2,2,0,5.7,0,47.9)$ would call for the generation of

$$Y = 5.7*Q2**2+47.9$$

The example was chosen to be easy to follow, not to illustrate a situation in which a macro-algorithm would be particularly useful. (Since the effort required to write down the expression $PQR(Q2,2,0,5.7,47.9)$ or $PQR(AB,3,-2,7.4,0,152.43,79.0,46.5,54.7)$ is as much as or more than that required to write the desired statements themselves, it is unlikely that one would employ a macro-algorithm in this case.)

The language MACRO-FORTRAN to be described in this report enables the programmer to write specifications for generating sequences of FORTRAN statements and to incorporate these sequences into his FORTRAN program in a natural fashion. (The specialization to FORTRAN is very slight. Minor changes would produce a MACRO-ALGOL, etc.)

A translator for MACRO-FORTRAN has been implemented at Argonne on the Control Data 3600. Although in principle such a translator could be incorporated directly into the FORTRAN compiler itself, the present implementation has been made in the form of a preprocessor that generates input for the

FORTRAN compiler. The MACRO-FORTTRAN translator is itself written entirely in 3600 FORTRAN. The FORTRAN language generated by MACRO-FORTTRAN need not be 3600 FORTRAN, but can be any language using the same continuation statement format as 3600 FORTRAN, using the 026-H or 026-F keypunch character codes, and not using the 4-8 punch in source code.

Note that a facility similar in intent to MACRO-FORTTRAN is included in the specifications for PL/I.*

II. THE MACRO-FORTTRAN LANGUAGE

Suppose that a line of source code contains no break characters and is not part of a macro definition and is not within the scope of a SKIP or SKIF pseudo-operation. Then that line will be passed on unaltered by MACRO-FORTTRAN to the FORTRAN compiler. If, however, a line of source code contains a value occurrence of a name (i.e., a name preceded by a break character #) and the MACRO-FORTTRAN translator has in its tables a value for that name, the break character and name are replaced by the value (which may be a string or an integer). For example, suppose that the translator has in its tables the integer value 57 for the name AB and the string value XY52Z for the name PQ. Then if the line

$U\#AB = W + \#PQ$

is encountered in the programmer's source code, it will be transmitted to FORTRAN as

$U57 = W + XY52Z.$

In this report, the *break* character (4-8 punch) will be indicated by the inequality sign, the symbol it prints as on the 501 printer. It can be keypunched with a single keystroke on the standard 026 keypunch (apostrophe in the H set, redundant minus in the F set) as the alphabetic character on the equals key. The blank character will be indicated by a lower-case letter b. The period will be indicated by o when confusion with periods in the text of this report might otherwise occur.

A *name* consists of one through eight alphanumeric characters, the first of which is alphabetic. The period will be counted as an alphabetic character for this purpose, except that it may not occur as the first character of a name. A *value occurrence* of a name is an occurrence of that name immediately preceded by a break character.

A *formal parameter* consists of the break character followed by a single nonzero decimal digit.

*The following IBM publications describing the language PL/I and its implementations on the IBM System/360 were available at the time of this report:

C28-6571-3 PL/I Language Specifications

C28-6594-1 PL/I (F) Programmer's Guide.

A *line terminator* is either of the following:

- # b Terminates the text on a source card. Next card is a continuation.
- # \$ Terminates the text on a source card. Next card begins a new statement (as far as MACRO-FORTRAN is concerned)--this does not affect the continuation convention of FORTRAN.

When something of the form

name b

is encountered, the blank immediately following the *name* is deleted.

A text element is a sequence of characters not including a break character.

An *sfield* is a sequence of characters composed of the following:

- Text elements
- Value occurrences of integer-valued names (magnitude $< 10^8$)
- Value occurrences of string-valued names
- Formal parameters (must have an *sfield* as actual parameter).

In the evaluation of an *sfield*, formal parameters are replaced by their corresponding actual parameters, value occurrences of integer-valued names are replaced by the decimal expansions of their absolute values (0 for zero, leading zeros omitted elsewhere), value occurrences of string-valued names are replaced by their values, and the results are all concatenated. The result must be a string of no more than eight characters.

An *afield* is a sequence of characters composed of the following:

- Value occurrences of integer-valued names
- Decimal integer constants
- + - * / (no grouping by parentheses allowed)
- Formal parameters (must have an *afield* as actual parameter).

In the evaluation of an *afield*, formal parameters are replaced by their corresponding actual parameters, integer constants and value occurrences of integer-valued names are evaluated, and the indicated arithmetic is carried out.

A. Pseudo-operations

#SI(*sfield*,*afield*)

The integer value of *afield* is computed and assigned as the value of the name formed by the string value of *sfield*.

#SST(sfield1,sfield2)

The string value of *sfield2* is formed and assigned as the value of the name formed by the string value of *sfield1*.

#PVS(sfield1,sfield2)

The string value of *sfield2* is formed. That value is treated as a name *name2*. Then the *name1* formed by the string value of *sfield1* is defined to be synonymous with *name2*. That is, as the value of *name2* changes, so will the value of *name1*. Another way of looking at this is that *name1* points to the value of *name2*.

#SKIP(sfield)

Processing of subsequent text is suppressed until we encounter an occurrence of the string value of *sfield* preceded by a break and a period. The value of *sfield* must be no more than seven characters long.

#SKIF(sfield1,afield1,sfield2,afield2)

Parameters *afield1* and *afield2* are evaluated. If the values satisfy the criterion specified by *sfield2*, then the result is the same as for *#SKIP(sfield1)*. Otherwise the pseudo-operation is ignored. Among the possible criteria are:

EQ first integer value = second integer value.
 NE first integer value $\neq$ second integer value.
 GT first integer value $>$ second integer value.
 GE first integer value $\geq$ second integer value.
 LT first integer value $<$ second integer value.
 LE first integer value $\leq$ second integer value.

#SKIF(sfield1,afield,sfield2)

sfield2 specifies criterion:

ZR integer value = 0.
 NZ integer value $\neq$ 0.
 NN integer value $\geq$ 0.
 NP integer value $\leq$ 0.
 PL integer value $>$ 0.
 MI integer value $<$ 0.

#SKIF(sfield1,sfield2,sfield3,sfield4)

sfield3 specifies criterion:

CG string values congruent (identical).
 NC string values not congruent.

`#SKIF(sfield1,sfield2,sfield3)`

sfield3 specifies criterion:

NULL value of *sfield2* is null string (no characters at all).

NNULL value of *sfield2* is not null string.

B. Macro Definitions and Macro Calls

A macro definition begins with

`#DEF#name(n)` ,

where *name* is to be the macro name and *n* is an integer ≤ 9 specifying the number of formal parameters. It ends with `#ENDD`. It may contain any sequence of FORTRAN statements or statement-generating lines. It may contain any pseudo-operations except `#DEF` and `#ENDD`. It may contain calls of other macros or of itself.

When encountered in the source code, a macro definition is filed away awaiting the occurrence of a call to that macro. A macro call is of the form

`#name(actual parameter list)` ,

or just

`#name` .

The *n* items on the actual parameter list are to be substituted for the corresponding formal parameters in the macro definition. The actual parameters are separated by commas. The entire list must be balanced with respect to parentheses in the following sense: If we start with the parenthesis on the left of the list and scans to the right, the number of left parentheses must exceed the number of right parentheses until the parenthesis on the right end of the list is reached. Then the number of left parentheses must equal the number of right parentheses. This restriction applies even though some of the parentheses may fall inside Hollerith literals. Unused actual parameters may be omitted. That is, `#name(P1,,P3)`, `#name(P1)`, and `#name` are all valid calls of a three-parameter macro.

When a macro call is encountered in the programmer's source code or during the expansion of another macro call, it is replaced by its expansion as follows: Each occurrence of a formal parameter is replaced by the corresponding actual parameter. For example, `#3` is replaced by the third parameter in the actual parameter list, supplied as part of the macro call. Any value occurrences of names are also replaced by the current value of the names. Pseudo-operations are carried out as they are encountered. Calls of other macros or of the same macro are expanded as they are encountered while expanding the given macro. The principal difference between the expansion of a macro and the treatment of ordinary lines of the programmer's source code is that the latter do not involve occurrences of formal parameters.

A word about defining names before using them: Every name must be *defined* before it is *used*. A name is defined by a DEF, SI, SST, or PVS pseudo-operation. A name is used when a value occurrence of it appears in some code (outside of a macro definition) or is encountered in expanding a macro. A name occurring within the text of a macro definition is not considered to be used until the macro is expanded. Similarly, an SI, SST, or PVS occurring within a macro definition does not define a symbol until the macro is expanded.

C. Examples (In a few cases, the actual spacing in the generated code will differ from that shown here.)

<u>Generated Code</u>	<u>Source Code</u>	<u>Explanation</u>
	<code>#DEF#ABC(2)X=#1*#2+7.5</code> <code>#2=4*X+32.95#ENDD#</code> <code>#ABC((Z7+15.2),Z28)</code>	
<code>X=(Z7+15.2)*Z28+7.5</code> <code>Z28=4*Z+32.95</code>		
	<code>#SI(Q,25)#</code> <code>X#Q=Z(#Q)</code>	Sets integer 25 as value of Q.
<code>X25=Z(25)</code>		
	<code>#SI(Q,#Q+4)#</code> <code>X#Q=Z(#Q)</code>	Increases value of Q to integer 29.
<code>X29=Z(29)</code>		
	<code>#SST(R,AB#QbCD)#</code> <code>#R=Z</code>	Integer value of Q converted to decimal equivalent and then concatenated with AB and CD to form string value AB29CD, which is assigned to R.
<code>AB29CD=Z</code>		
	<code>#SST(TX,#Q+4)#</code> <code>I=#TX</code>	Integer value of Q converted to decimal equivalent and then concatenated (not added) to +4 to form string value 29 + 4, which is assigned to TX.
<code>I=29+4</code>		
	<code>#PVS(T5,Q)#</code> <code>#SI(W,#Q)#</code> <code>X#T5=Z#W</code>	Defines T5 as synonymous with Q. Assigns current value of Q (integer 29) to W.
<code>X29=Z29</code>		
	<code>#SI(Q,#Q+5)#</code> <code>X#T5=Z#W</code>	

<u>Generated Code</u>	<u>Source Code</u>	<u>Explanation</u>
X34=Z29	<pre> #DEF#A(2)X=#1 Y=#2+Z#ENDD# #DEF#B(1)#1=Z #A(#1Q,C)#ENDD# #DEF#D(1)U=2 #B(#1=GH) Y=3#ENDD# #D(ABC) </pre>	T5 changes with change in Q, W does not.
U=2		
ABC=GH=Z		
X=ABC=GHQ		
Y=C+Z		
Y=3		
	<pre> #DEF#GH(1)X=#SKIF(A,#1,GT,5)B#1+C#SKIP(D)# #.AbBX+C#1#.D#ENDD# #GH(4) </pre>	Since 4 $\nless$ 5, the SKIF is in- effective. The SKIP is not itself skipped, so it is effective.
X=B4+C		
	<pre> #GH(7) </pre>	Since 7 > 5, the SKIF is effective. The SKIP is then itself skipped.
X=BX+C7		
	<pre> #DEF#BX(1)X#1=Y#1 #SKIF(A,#1,GT,900000)# #BX(#12#1)#.A# Z=#1 #ENDD# #BX(5) </pre>	
X5=Y5		
X525=Y525		
X5252525=Y5252525		
Z=5252525		
Z=525		
Z=5		

<u>Generated Code</u>	<u>Source Code</u>	<u>Explanation</u>
	#BX(498)	
X498=Y498		
X4982498=Y4982498		
Z=4982498		
Z=498		
	#SI(Q,54)	Assigns integer 54 as value of Q.
	#SST(R,Q)	Assigns string Q as value of R.
	#SST(S,#Q)	Takes value of name Q (i.e., 54) and converts it to a decimal string, then assigns that string as value of S.
	#SST(#R,AB)	Assigns the string AB as the value of the name Q formed by taking the value of R and treating it as a name.
	QRS=#Q#R#S	
QRS=ABQ54		

D. Permitted Contexts

	<u>Outside of Macro Definition</u>	<u>Within Text of Macro Definition</u>	<u>Within Actual Parameter</u>	<u>Within afield</u>	<u>Within sfield</u>
Value occurrence of string-valued name	Yes	Yes	Yes	No	Yes
Value occurrence of integer-valued name	Yes	Yes	Yes	Yes	Yes
Formal parameter	No	Yes	Yes	Yes ¹	Yes ²
Pseudo-operation	Yes	Yes	No	No	No
Macro definition	Yes	No	No	No	No
Macro call	Yes	Yes	No	No	No
Integer constant	Yes ³	Yes ³	Yes ³	Yes	Yes ³
String constant	Yes	Yes	Yes	No	Yes
Floating-point constant	Yes ³	Yes ³	Yes ³	No	Yes ³

¹Corresponding actual parameter must be valid in afield.

²Corresponding actual parameter must be valid in sfield.

³It will be treated as a string, not as a number.

III. DECK STRUCTURE FOR 3600 MACRO-FORTRAN

Job Card
Time Card (for use at Argonne only)
Optional Equip Card if you wish to punch up FORTRAN deck from LUN 20
Binary deck of MACRO, FETCH, and DISPATCH
⁷ ⁹ RUN
Your source deck
bbbbbbbbbbSCOPE
End-of-file card
⁷ ⁹ FTN,I=20 plus your choice of parameters
End-of-file card

IV. DIAGNOSTIC MESSAGES

SCAN ERROR

This diagnostic is printed by the main scan routine when it encounters an error in its bookkeeping induced by an earlier source-code error. If this occurs while processing an actual parameter, the remainder of that parameter is skipped. Otherwise, if it occurs while expanding a macro, the remainder of the macro text is skipped. If it occurs while processing source code, the remainder of the source statement may be skipped, depending on the severity of the error.

UNDIAGNOSED SFIELD ERROR. SFIELD SET TO SINGLE BLANK

This diagnostic is printed by the sfield processing routine when it encounters an error in its bookkeeping, probably induced by an earlier source-code error.

UNDIAGNOSED AFIELD ERROR. AFIELD SET TO SINGLE BLANK

This diagnostic is similar to the preceding diagnostic.

ILLEGAL VALUE FOR SFIELD SYMBOL. SFIELD SET TO SINGLE BLANK

A name appearing in an sfield is either undefined or fails to have a string, an integer, or a synonym value.

ILLEGAL VALUE FOR AFIELD SYMBOL. AFIELD SET TO ZERO

A name appearing in an afield is either undefined or fails to have an integer or synonym value.

UNDEFINED NAME. NAME AND PAR LIST SKIPPED

An undefined name has been encountered. This differs from the preceding two diagnostics in that the name has not occurred in an sfield or an afield.

ILLEGAL SKIP TARGET. TREATED AS NO SKIP

An error was encountered while processing the field specifying the target for a skip (or that field consisted of a single blank).

MORE THAN 24 LEVELS FORMAL PARAMETER NESTING. ABANDON

This can presumably occur only within at least 24 levels of macro call nesting (counting pseudo-operations as macro calls). It claims that a chain of references to actual parameters of higher-level cells exists spanning more than 24 levels of nesting; hence a table has overflowed.

ACTUAL PARAMETER STACK OVERFLOW. ABANDON

The 5000 words of temporary storage allocated for the actual parameter stack have been overflowed. To try again, reduce the length or number of actual parameters and/or eliminate some nesting of macro calls.

MACRO-TEXT STORAGE OVERFLOW. ABANDON

The 9500 words of storage allocated to the filing of macro definitions have been overflowed. To try again, segment the program to reduce the amount of macro text storage required.

FETCH ERROR

In the source code, a break character has been immediately followed by some character not permitted after a break.

V. IMPLEMENTATION

In the current implementation of MACRO-FORTRAN, the entire translator is written in 3600 FORTRAN, including the input and output routines FETCH and DISPATCH. This was done to facilitate checkout and modification, but has the disadvantage of running much slower than would be the case if it were rewritten in assembly language and optimized.

The listings that follow are not of the original Argonne version but of an improved 1604-compatible version developed at The University of Wisconsin with the support of The University of Wisconsin Computing Center.

```

PROGRAM MACRO
COMMON K1604
EQUIVALENCE(KSP,KSW)
DIMENSION NAMETAB(1000)
DIMENSION T(15000), LIN(200), S(50)
DIMENSION M(8),N(8)
DIMENSION KT(64)
TYPE INTEGER S,T
DATA(N = 1000000000000000B, 1000000000000000B, 1000000000000000B,
X 1000000000B, 1000000B, 10000B, 100B, 1)
DATA(M = 77000000000000000B, 77000000000000000B, 770000000000000B,
X 77000000000B, 77000000B, 770000B, 7700B, 77B)
DATA(KT = 8,8,8,8, 8,8,8,8, 8,8,0,6, 5,0,0,0, 1,9,9,9, 9,9,9,9,
X 9,9,0,6, 6,0,0,0, 2,9,9,9, 9,9,9,9, 9,9,0,6, 3,0,0,0,
X 7,4,9,9, 9,9,9,9, 9,9,0,6, 6,0,0,0)
DATA(MCAV1=5500)
DATA(MAV1=500)
DATA(KSB=1)
DATA(NAMETAB=2425261400000000B,-54,0,6231140000000000B,-71,2,
X 6242312614000000B,-72,4,6242314714000000B,-73,1,
X 4765621400000000B,-74,2,6262631400000000B,-75,2,
X 2647140000000000B,-76,1,2550140000000000B,-99,1,
X 4525140000000000B,-99,2,2763140000000000B,-99,3,
X 4363140000000000B,-99,4,2725140000000000B,-99,5,
X 4325140000000000B,-99,6,7151140000000000B,-99,21,
X 4571140000000000B,-99,22,
X 4743140000000000B,-99,23,4431140000000000B,-99,24,
X 2327140000000000B,-99,31,4523140000000000B,-99,32,
X 4564434314000000B,-99,41,4545644343140000B,-99,42)
DATA (NTABMAX=69)
TYPE INTEGER BREAK $ DATA(BREAK = 14B)
TYPE INTEGER ZEBR $ DATA(ZEBR=0014000000000000B)
599 CONTINUE
MA = MCAV1 - 50
MCA = 24950
MS = 49
MCAV = MCAV1
MAV = MAV1
JA = 0
JS = -1
JLIN = 1
KSKFL = 0
MLEV = 0
KNULL = MCAV1-1
T(KNULL) = -104
GO TO 450
C LOOK AT NEXTTABLE ENTRY
503 CONTINUE
K = T(KSP)
KSP = KSP + 1
IF(K-100)500,500,98
500 CONTINUE
IF(K)401,402,403
C POSITIVE -- N WORDS TEXT

```



```

403 CONTINUE
    IF(KSKFL)521,520,520
521 CONTINUE
    KSP = KSP + K
    GO TO 503
520 CONTINUE
504 CONTINUE
    LIN(JLIN) = T(KSP)
    KSP = KSP + 1
    JLIN = JLIN + 1
    K = K - 1
    IF(K)503,503,504
C    ZERO -- UNUSED
402 CONTINUE
    GO TO 98
C    NEGATIVE
401 CONTINUE
    IF(K+1000)452,452,400
400 CONTINUE
    IF(K+100)406,407,408
C    LESS THAN -100
406 CONTINUE
    KK = K + 110
    GO TO ( 98,598,451,450,503,460,411, 98,412) KK
C*****
C    S STACK OVERFLOW
90 CONTINUE
    PRINT 9912
    RETURN
C    ERROR
98 CONTINUE
    PRINT 9902
466 CONTINUE
    IF(JS)464,464,460
464 CONTINUE
    IF(JA)463,463,407
463 CONTINUE
    IF(KSP-2)450,450,503
C    FORMAL PARAMETER (-1 TO -99)
408 CONTINUE
    IF(KSKFL)503,522,522
522 CONTINUE
    JS = JS + 2
    IF(JS-MS)820,90,90
820 CONTINUE
    S(JS) = KSP
    S(JS+1) = JA
    KSP = JA - K + 1
    JA = T(JA)
    GO TO 503
C    END ACTUAL PARAMETER (-104)
460 CONTINUE
    IF(KSKFL)503,465,465
465 CONTINUE

```

```

      KSP = S(JS)
      JA = S(JS+1)
      JS = JS - 2
      IF(JS+1)554,555,555
554 CONTINUE
      JS = -1
555 CONTINUE
      GO TO 503
C      END MACRO TEXT (-100)
407 CONTINUE
      MAV = JA
      KSP = T(JA+1)
      JA = T(JA)
      IF(MAV)501,501,503
501 CONTINUE
      GO TO 98
C      END OF STATEMENT (-101)
412 CONTINUE
      IF(KSKFL)503,519,519
519 CONTINUE
      CALL DISPATCH(LIN,JLIN-1)
      JLIN = 1
      GO TO 503
C      END SOURCE BUFFER (-106)
450 CONTINUE
      CALL FETCH(T(KSB),NIW)
      KSB1 = KSB + NIW
      KSP = KSB
      GO TO 503
C      END PROGRAM (-107)
451 CONTINUE
      CALL DISPATCH(LIN,JLIN-1)
      REWIND 20
      RETURN
C      RE-INITIALIZE (-108)
598 CONTINUE
      CALL DISPATCH(LIN,JLIN-1)
      GO TO 599
C      INDIRECT (LT - 999)
452 CONTINUE
      KSP = K + 30000
      GO TO 503
C      MACRO NAME (-103)
411 CONTINUE
      IF(KSKFL)517,514,514
517 CONTINUE
      KSP = KSP + 1
      IF(T(KSP-1)-KSKSY)503,518,503
518 CONTINUE
      KSKFL = 0
      KSP = KSP + 1
      GO TO 503
514 CONTINUE
      K1 = T(KSP)

```

```
KSP = KSP + 1
J = K1 .AND. 770000000000000000B
IF(J-330000000000000000B)510,503,510
510 CONTINUE
DO 420 KK = 1,NTABMAX,3
IF(NAMETAB(KK) - K1)420,422,420
420 CONTINUE
GO TO 467
422 CONTINUE
KM = KK
NP = NAMETAB(KK+1)
NQ = NP
IF(NP)516,423,423
516 CONTINUE
NP = NAMETAB(KM+2)
JT = -49-NQ
GO TO (467,511,512,513,424,423) JT
467 CONTINUE
PRINT 9911
GO TO 407
C ONE WORD STRING
511 CONTINUE
LIN(JLIN) = NAMETAB(KM+2)
508 CONTINUE
JLIN = JLIN + 1
GO TO 503
C INTEGER
512 CONTINUE
I4 = NAMETAB(KM+2)
ASSIGN 509 TO K786
GO TO 786
509 CONTINUE
LIN(JLIN) = I2
GO TO 508
C VALUE OF SYMBOL
513 CONTINUE
K1 = NAMETAB(KM+2)
GO TO 510
C*****
C MAV OVERFLOW
91 CONTINUE
PRINT 9913
RETURN
423 CONTINUE
T(MAV)=JA
JA = MAV
MAV = JA + NP + 2
C PRESET ALL ACTUAL PARAMETERS EMPTY
DO 425 KK=1,NP
T(JA+KK+1) = -104
425 CONTINUE
MLEV = 1
C MOVE ALL ACT PARS
DO 426 KK = 1,NP
```

```

      IF(T(KSP)+105)432,470,432
470  CONTINUE
      KSP = KSP + 1
      GO TO 431
432  CONTINUE
      T(JA+KK+1) = MAV - 30000
427  CONTINUE
      IF(MAV-MA)824,91,91
824  CONTINUE
      K = T(KSP)
      IF(K)433,433,428
428  CONTINUE
      N6 = K + 1
C    MOVE K + 1 WORDS
      DO 429 JJ = 1,N6
      T(MAV+JJ-1) = T(KSP+JJ-1)
429  CONTINUE
      KSP = KSP + N6
      MAV = MAV + N6
      GO TO 427
433  CONTINUE
C    MOVE ONE WORD
      T(MAV) = K
      KSP = KSP + 1
      MAV = MAV + 1
      IF(K+103)462,461,462
461  CONTINUE
      T(MAV) = T(KSP)
      KSP = KSP + 1
      MAV = MAV + 1
      MLEV = MLEV + 1
      GO TO 427
462  CONTINUE
      IF(K+105)430,468,430
468  CONTINUE
      MLEV = MLEV - 1
      IF(MLEV)431,431,427
430  CONTINUE
      IF(K+106)474,475,474
475  CONTINUE
      MAV = MAV - 1
      CALL FETCH(T(KSB),NIW)
      KSB1 = KSB + NIW
      KSP = KSB
      GO TO 427
474  CONTINUE
      IF(K+104)427,469,427
469  CONTINUE
      IF(MLEV-1)426,426,427
426  CONTINUE
      IF(T(KSP)+105)431,471,431
471  CONTINUE
      KSP = KSP + 1
C    END ACT PAR LIST (-105)

```

```

431 CONTINUE
    T(JA+1) = KSP
    IF(NQ)435,434,434
434 CONTINUE
    KSP = NAMETAB(KM+2)
    GO TO 503
435 CONTINUE
    JT = -69-NQ
    GO TO (407,534,530,682,436,437,800,407) JT
C    POINT TO VALUE OF SYMBOL
436 CONTINUE
    KVT = - 53
    GO TO 438
C    SET STRING
437 CONTINUE
    KVT = - 51
438 CONTINUE
    KSW = T(JA+3) + 30000
    ASSIGN 439 TO K752
    GO TO 752
439 CONTINUE
    KFLD2 = KFLD
    GO TO 553
C    MCAV OVERFLOW
92 CONTINUE
    PRINT 9914
    RETURN
C    MACRO DEF
424 CONTINUE
    KSP = KSP + 2
    NAMETAB(NTABMAX+1) = T(KSP)
    K2 = T(KSP+2) .AND. 7700000000000000B
    NAMETAB(NTABMAX+2) = K2/1000000000000000B
    NAMETAB(NTABMAX+3) = MCAV
    NTABMAX = NTABMAX + 3
    KSP = KSP + 5
C    MOVE MACRO TEXT
440 CONTINUE
    IF(MCAV-MCA)825,92,92
825 CONTINUE
    K = T(KSP)
    IF(K)441,441,442
C    MOVE K+1 WORDS OF TEXT
442 CONTINUE
    N6 = K + 1
    DO 443 JJ = 1,N6
    T(MCAV+JJ-1) = T(KSP+JJ-1)
443 CONTINUE
    KSP = KSP + N6
    MCAV = MCAV + N6
    GO TO 440
C    MOVE ONE WORD
441 CONTINUE
    T(MCAV) = K

```

```

      KSP = KSP + 1
      MCAV = MCAV + 1
      IF(K+103)472,444,472
444  CONTINUE
C    TEST FOR ENDD
      IF(T(KSP)-25452424140000008) 445,446,445
446  CONTINUE
      T(MCAV-1) = - 100
      KSP = KSP + 2
      GO TO 503
445  CONTINUE
      T(MCAV) = T(KSP)
      KSP = KSP + 1
      MCAV = MCAV + 1
      GO TO 440
472  CONTINUE
      IF(K+106)440,473,440
473  CONTINUE
      MCAV = MCAV - 1
      CALL FETCH(T(KSB),NIW)
      KSB1 = KSB + NIW
      KSP = KSB
      GO TO 440
C*****
C    ARITHMETIC
661  CONTINUE
      J = T(JA+2)
      LSYMB = T(J+30001)
      KSW = T(JA+3)+30000
662  CONTINUE
      KS = 0
      KPREV = 1
      KNEXT = 1
      NW = 0
      JAR = JA
      KQ = 0
C    NEXT WORD
612  CONTINUE
      K = T(KSW)
      KSW = KSW+1
      IF(NW)613,613,614
613  CONTINUE
      IF(K)615,97,616
616  CONTINUE
      IF(K-100 )620,620,97
620  CONTINUE
      NW = K
      GO TO 612
615  CONTINUE
      IF(K+1000)617,617,618
617  CONTINUE
      KSW = K+30000
      GO TO 612
618  CONTINUE

```

```

        IF(K+100)543,97,619
619  CONTINUE
        JS = JS+2
        IF(JS-MS)822,90,90
822  CONTINUE
        S(JS) = KSW
        S(JS+1) = JA
        JA = T(JA)
        KSW = JA-K+1
        GO TO 612
543  CONTINUE
        JT = -100-K
        GO TO (612, 97,623,624,612, 97, 97) JT
623  CONTINUE
        K = T(KSW)
        KSW = KSW+2
540  CONTINUE
        DO 799 KK=1,NTABMAX,3
        IF(NAMETAB(KK)-K)799,798,799
799  CONTINUE
        GO TO 646
798  CONTINUE
        KVAL = NAMETAB(KK+2)
        KTOP = NAMETAB(KK+1)
        JT = -50-KTOP
        GO TO (646,542,544,646) JT
646  CONTINUE
        PRINT 9909
        GO TO 647
544  CONTINUE
        K = KVAL
        GO TO 540
542  CONTINUE
        KA = KVAL
        KQ = KVAL
        GO TO 612
624  CONTINUE
        IF(JA-JAR)645,655,655
655  CONTINUE
        KA = KQ
        KQ = 0
        KPREV = KNEXT
        GO TO 601
645  CONTINUE
        KSW = S(JS)
        JA = S(JS+1)
        JS = JS-2
        IF(JS+1)556,557,557
556  CONTINUE
        JS = -1
557  CONTINUE
        GO TO 612
614  CONTINUE
        NW = NW-1

```

```

      KSC = 1
600  CONTINUE
      IF(KSC-8)651,651,650
650  CONTINUE
      KSC = 1
      GO TO 612
651  CONTINUE
      J = K.AND.M(KSC)
      IC = J/N(KSC)
      IC = IC.AND.77B
      IT = KT(IC+1)
      KSC = KSC+1
      IF(IT-4)610,610,611
610  CONTINUE
      KPREV = KNEXT
      KNEXT = IT
      KA = KQ
      KQ = 0
611  CONTINUE
      GO TO (601,602,603,604,605, 97,600,608, 97) IT
C    NEXT + OR -
601  CONTINUE
602  CONTINUE
      GO TO (631,632,633,634) KPREV
C    +..+-
631  CONTINUE
      KS = KS+KA
652  CONTINUE
      IF(K+104)600,656,600
656  CONTINUE
      IF(JA-JAR)600,545,545
545  CONTINUE
      GO TO KH
C    -..+-
632  CONTINUE
      KS = KS-KA
      GO TO 652
C    *..+-
633  CONTINUE
      KS = KS+KP*KA
      GO TO 652
C    /..+-
634  CONTINUE
      KS = KS+KP/KA
      GO TO 652
C    NEXT * OR /
603  CONTINUE
604  CONTINUE
      GO TO (641,642,643,644) KPREV
C    +..*/
641  CONTINUE
      KP = KA
      GO TO 600
C    -..*/

```



```

642 CONTINUE
    KP = -KA
    GO TO 600
C    *..*/
643 CONTINUE
    KP = KP*KA
    GO TO 600
C    /..*/
644 CONTINUE
    KP = KP/KA
    GO TO 600
C    NEXT IS BREAK
605 CONTINUE
    GO TO 612
C    NEXT IS DIGIT
608 CONTINUE
    KQ = KQ*10+IC
    GO TO 600
C*****
C    SKIF
530 CONTINUE
    KSW = T(JA+4) + 30000
    ASSIGN 566 TO K752
    GO TO 752
566 CONTINUE
    KRIT = KFLD
531 CONTINUE
    DO 797 KK=1,NTABMAX,3
    IF(NAMETAB(KK)-KRIT)797,796,797
797 CONTINUE
    GO TO 95
796 CONTINUE
    KRIT = NAMETAB(KK+2)
    IF(NAMETAB(KK+1)+53)532,531,532
532 CONTINUE
    IF(NAMETAB(KK+1)+99)95,567,95
95 CONTINUE
    PRINT 9906
    GO TO 683
567 CONTINUE
    IF(KRIT-31)546,538,538
538 CONTINUE
    IF(KRIT-51)539,95,95
539 CONTINUE
    KSW = T(JA+3)+30000
    ASSIGN 541 TO K752
    GO TO 752
541 CONTINUE
    KFLD1 = KFLD
    IF(KRIT-41)547,548,548
548 CONTINUE
    KRIT = KRIT - 10
    KFLD1 = 14000000000000000000B
    GO TO 549

```

```
547 CONTINUE
    KSW = T(JA+5) + 30000
    ASSIGN 549 TO K752
    GO TO 752
549 CONTINUE
    KS = KFLD1 - KFLD
    KRIT = KRIT - 10
    GO TO 684
546 CONTINUE
    ASSIGN 680 TO KH
    GO TO 661
680 CONTINUE
    IF(KRIT-21)533,684,684
533 CONTINUE
    Kfv = KS
    KSW = T(JA+5)+30000
    ASSIGN 681 TO KH
    GO TO 662
681 CONTINUE
    KS = Kfv-KS
    KRIT = KRIT+20
684 CONTINUE
    JT = KRIT - 20
    GO TO (721,722,723,724,725,726) JT
C    EQ(ZR)
721 CONTINUE
    IF(KS)683,682,683
C    NE(NZ)
722 CONTINUE
    IF(KS)682,683,682
C    GT(PL)
723 CONTINUE
    IF(KS)683,683,682
C    LT(MI)
724 CONTINUE
    IF(KS)682,683,683
C    GE(NN)
725 CONTINUE
    IF(KS)683,682,682
C    LE(NP)
726 CONTINUE
    IF(KS)682,682,683
C    SKIP
682 CONTINUE
    KSW = T(JA+2) + 30000
    ASSIGN 551 TO K752
    GO TO 752
551 CONTINUE
    IF(KFLD-601400000000000000B)552,564,552
564 CONTINUE
    PRINT 9907
    GO TO 683
552 CONTINUE
    J = KFLD
```

```

      J = J.AND.7777777777777700B
      J = J/100B
      J = J.AND.77777777777777B
      KSKSY = J.OR.3300000000000000B
      KSKFL = -1
      GO TO 407
C     NO SKIP
683  CONTINUE
      KSKFL = 0
      GO TO 407
C     SI
534  CONTINUE
      ASSIGN 535 TO KH
      GO TO 661
535  CONTINUE
      KFLD2 = KS
      KVT = -52
553  CONTINUE
      KSW = T(JA+2) + 30000
      ASSIGN 550 TO K752
      GO TO 752
550  CONTINUE
      DO 795 KK=1,NTABMAX,3
      IF(NAMETAB(KK)-KFLD)795,794,795
795  CONTINUE
536  CONTINUE
      LSYM = NTABMAX+1
      NTABMAX = NTABMAX+3
      NAMETAB(LSYM) = KFLD
      GO TO 537
794  CONTINUE
      LSYM = KK
537  CONTINUE
      NAMETAB(LSYM+2) = KFLD2
      NAMETAB(LSYM+1) = KVT
      GO TO 407
      97  CONTINUE
      PRINT 9905
647  CONTINUE
      KS = 0
654  CONTINUE
      IF(JA-JAR)653,545,545
653  CONTINUE
      KSP = S(JS)
      JA = S(JS+1)
      JS = JS - 2
      IF(JS+1)558,559,559
558  CONTINUE
      JS = -1
559  CONTINUE
      GO TO 654
C     FP PSEUDO
800  CONTINUE
      KSW = T(JA+2) + 30000

```

```

      ASSIGN 801 TO KH
      GO TO 662
801  CONTINUE
      J = T(JA)
      K = T(J+KS+1)
      IF(T(J+2)+104)802,806,802
802  CONTINUE
      IF(T(J+2)+30000-J-2+KS)806,803,803
803  CONTINUE
      IF(K+104)804,806,804
804  CONTINUE
      IF(T(K+30000)+104)805,806,805
805  CONTINUE
      K = -KS
      MAV = JA
      KSP = T(JA+1)
      J = T(JA)
      GO TO 522
C    ACTUAL PARAMETER ABSENT OR NULL
806  CONTINUE
      IF(T(JA+3)+104)808,807,808
C    SKIP TARGET ABSENT
807  CONTINUE
      GO TO 407
808  CONTINUE
      KSW = T(JA+3)+30000
      ASSIGN 551 TO K752
      GO TO 752
C *****
C    CONVERT INTEGER
786  CONTINUE
      I3 = I2 = 0
      IF(I4)789,788,787
788  CONTINUE
      I2 = ZEBR
      GO TO 792
789  CONTINUE
      I4 = -I4
787  CONTINUE
      IX = 10*I4+5
      DO 793 IJ = 1,8
      I1 = IX/100000000
      IX = 10*(IX-I1*100000000)
      I2 = 64*I2+I1
      IF(I2)790,793,790
790  CONTINUE
      I3 = I3 + 1
793  CONTINUE
      IF(I3-8)791,792,792
791  CONTINUE
      I2 = I2*N(I3)
      J = BREAK*N(I3+1)
      I2 = I2.OR.J
792  CONTINUE

```

GO TO K786

C*****

C FIELD EVALUATOR/COMPRESSOR

752 CONTINUE

JA1 = JA

KCC = 1

KFLD = 0

750 CONTINUE

K = T(KSW)

KSW = KSW + 1

IF(K-100)772,772,96

772 CONTINUE

IF(K)758,96,753

C TEXT

753 CONTINUE

NW = K

754 CONTINUE

K = T(KSW)

KSW = KSW+1

751 CONTINUE

KTC = 1

755 CONTINUE

J = K.AND.M(KTC)

J = J/N(KTC)

J = J.AND.77B

IF(J-BREAK)756,757,756

757 CONTINUE

NW = NW-1

IF(NW)750,750,754

756 CONTINUE

J = J*N(KCC)

KFLD = KFLD.OR.J

KTC = KTC + 1

KCC = KCC + 1

IF(KTC-8)755,755,757

758 CONTINUE

IF(K+100)760,760,759

759 CONTINUE

JS = JS+2

IF(JS-MS)821,90,90

821 CONTINUE

S(JS) = KSW

S(JS+1) = JA

JA = T(JA)

KSW = JA - K + 1

GO TO 750

760 CONTINUE

IF(K+1000)777,778,778

777 CONTINUE

KSW = K + 30000

GO TO 750

778 CONTINUE

JT = -101-K

GO TO (750,761,762,750) JT

```

C      -103
761 CONTINUE
   K = T(KSW)
   KSW = KSW+2
763 CONTINUE
   DO 785 KK=1,NTABMAX,3
   IF(NAMETAB(KK)-K)785,784,785
785 CONTINUE
   GO TO 94
784 CONTINUE
   K9 = K
   K = NAMETAB(KK+2)
   K8 = NAMETAB(KK+1)
   JT = K8+55
   GO TO ( 94,763,764,765, 94) JT
94 CONTINUE
   PRINT 9908
   GO TO 773
C      STRING
765 CONTINUE
   NW = 1
   GO TO 751
C      INTEGER
764 CONTINUE
   IF(K)783,782,781
782 CONTINUE
   KCC = KCC+1
   GO TO 750
783 CONTINUE
   K = -K
781 CONTINUE
   I2=0
   IX = 10*K+5
   DO 780 IJ=1,8
   I1 = IX/1000000000
   IX = 10*(IX-I1*1000000000)
   I2 = I2+I1
   IF(I2)779,780,779
779 CONTINUE
   J = I1*N(KCC)
   KFLD = KFLD.OR.J
   KCC = KCC+1
780 CONTINUE
   GO TO 750
C      -104
762 CONTINUE
   IF(JA-JA1)771,766,766
771 CONTINUE
   KSW = S(JS)
   JA = S(JS+1)
   JS = JS-2
   IF(JS+1)562,563,563
562 CONTINUE
   JS = -1

```

```

563 CONTINUE
    GO TO 750
766 CONTINUE
    IF(KCC-8)767,767,768
767 CONTINUE
    J = BREAK*N(KCC)
    KFLD = KFLD.OR.J
768 CONTINUE
    GO TO K752
  96 CONTINUE
    PRINT 9903
773 CONTINUE
    KFLD = 601400000000000000B
769 CONTINUE
    IF(JA-JA1)770,768,768
770 CONTINUE
    KSP = S(JS)
    JA = S(JS+1)
    JS = JS - 2
    IF(JS+1)560,561,561
560 CONTINUE
    JS = -1
561 CONTINUE
    GO TO 769
9902 FORMAT(12H SCAN ERROR      )
9903 FORMAT(57H UNDIAGNOSED SFIELD ERROR.  SFIELD SET TO SINGLE BLANK.
X      )
9905 FORMAT(49H UNDIAGNOSED AFIELD ERROR.  AFIELD SET TO ZERO.      )
9906 FORMAT(55H UNRECOGNIZED SKIP CRITERION.  TREATED AS NO SKIP.
X      )
9907 FORMAT(45H ILLEGAL SKIP TARGET.  TREATED AS NO SKIP.      )
9908 FORMAT(65H IMPROPER VALUE FOR SFIELD SYMBOL.  SFIELD SET TO SINGLE
X BLANK.      )
9909 FORMAT(57H ILLEGAL VALUE FOR AFIELD SYMBOL.  AFIELD SET TO ZERO.
X      )
9911 FORMAT(47H UNDEFINED NAME.  NAME AND PAR LIST SKIPPED.      )
9912 FORMAT(59H MORE THAN 24 LEVELS FORMAL PARAMETER NESTING.  ABANDON.
X      )
9913 FORMAT(46H ACTUAL PARAMETER STACK OVERFLOW.  ABANDON.      )
9914 FORMAT(43H MACRO TEXT STORAGE OVERFLOW.  ABANDON.      )
END
SUBROUTINE FETCH(KOW,NOW)
COMMON K1604
DATA(KT = 8,8,8,8, 8,8,8,8, 8,8,0,6, 1,0,0,0, 6,9,9,9, 9,9,9,9,
X 9,9,0,9, 2,0,0,0, 6,9,9,9, 9,9,9,9, 9,9,0,5, 6,0,0,0,
X 7,6,9,9, 9,9,9,9, 9,9,0,4, 3,0,0,0)
DIMENSION N(8),M(8),KIC(80),KOW(200),KT(64),LPC(50)
DATA(N = 1000000000000000B, 100000000000000B, 1000000000000B,
X 100000000B, 1000000B, 10000B, 100B, 1)
TYPE INTEGER BREAK
DATA(BREAK=14B),(LPAREN=74B),(NIC=80)
DIMENSION J1636(64)
DATA(J1636 = 12B,1,2,3,4,5,6,7,10B,11B,0,13B,14B,15B,16B,17B,
X 60B,61B,62B,63B,64B,65B,66B,67B,70B,71B,72B,73B,74B,75B,76B,77B,

```

```

X 40B,41B,42B,43B,44B,45B,46B,47B,50B,51B,52B,53B,54B,55B,56B,57B,
X 20B,21B,22B,23B,24B,25B,26B,27B,30B,31B,32B,33B,34B,35B,36B,37B)
TYPE INTEGER SLASH
DATA(SLASH = 61B)
DATA(INP=0)
IF(INP)20,21,20
21 CONTINUE
IF(1RA-21B)22,23,22
C 3600
23 CONTINUE
K1604 = 0
INP = 60
GO TO 20
C 1604
22 CONTINUE
K1604 = -1
INP = 50
20 CONTINUE
KPC = 0
JPC = 0
IOC = 1
IOW = 1
ISW = 0
MO = 2
KBF = 0
GO TO 27
C NEXT INP CHAR
1 CONTINUE
IOW1= IOW - 1
IIC = IIC + 1
IF(IIC-NIC)31,31,35
35 CONTINUE
IF(ISW)65,65,66
66 CONTINUE
IF(IOC-1)67,68,67
67 CONTINUE
J = BREAK * N(IOC)
KOW(IOW) = KOW(IOW) .OR. J
IOW = IOW + 1
ISW = ISW + 1
68 CONTINUE
IF(MO-8)69,65,69
69 CONTINUE
KOW(IOW-ISW) = ISW-1
65 CONTINUE
KOW(IOW) = -101
KOW(IOW+1) = -106
NOW = IOW+1
24 CONTINUE
IF(KBF)25,26,25
25 CONTINUE
PRINT 9904,(KIC(KK),KK=1,NIC)
26 CONTINUE
RETURN

```



```

30 CONTINUE
   IF(IOW-150)87,88,88
88 CONTINUE
   KOW(IOW) = -106
92 CONTINUE
   NOW = IOW
   GO TO 24
87 CONTINUE
   IF(KBF)28,27,28
28 CONTINUE
   PRINT 9904,(KIC(KK),KK=1,NIC)
27 CONTINUE
   READ 9901,(KIC(KK),KK=1,NIC)
   IF(EOF,INP)82,83
82 CONTINUE
   KOW(IOW) = -101
   KOW(IOW+1) = -107
   NOW = IOW + 1
   GO TO 24
83 CONTINUE
   IIC = 1
31 CONTINUE
   IC = KIC(IIC)
   IF(K1604)16,17,16
16 CONTINUE
   IC = J1636(IC+1)
17 CONTINUE
   IT = KT(IC+1)
   4 CONTINUE
   IF(IT-MO)33,32,32
33 CONTINUE
   KBF = -1
   IF(MO-5)73,70,73
70 CONTINUE
   GO TO (73,71,2,71) IT
71 CONTINUE
   IF(KPC-1)2,73,2
73 CONTINUE
   IF(ISW)2,2,34
34 CONTINUE
   IF(IOC-1)61,62,61
61 CONTINUE
   J = BREAK * N(IOC)
   KOW(IOW) = KOW(IOW) .OR. J
   IOW = IOW + 1
   ISW = ISW + 1
62 CONTINUE
   IF(MO-8)74,75,74
74 CONTINUE
   KOW(IOW-ISW) = ISW - 1
75 CONTINUE
   ISW = 0
   KOW(IOW) = 0
   IOC = 1

```

```

      GO TO 2
C     MOVE CHARACTER
32  CONTINUE
      IF(ISW)3,3,36
      3  CONTINUE
          IOW = IOW + 1
          ISW = 1
          IOC = 1
          KOW(IOW) = 0
36  CONTINUE
          J = IC * N(IOC)
          KOW(IOW) = KOW(IOW) .OR. J
          IOC = IOC + 1
          IF(IOC - 8) 1,1,37
37  CONTINUE
          IOC = 1
          IOW = IOW + 1
          ISW = ISW + 1
          KOW(IOW) = 0
          GO TO 1
      2  CONTINUE
          IF(MO-8)5,41,40
C     PACKING NAME
41  CONTINUE
          IF(IC-LPAREN)43,42,43
C     LPAREN FOLLOWSNAME
42  CONTINUE
          JPC = JPC + 1
          LPC(JPC) = KPC
          KPC = 1
          MO = 5
          GO TO 1
C     NO LPAREN AFTER NAME
43  CONTINUE
          KOW(IOW) = -105
          IOW = IOW + 1
          IF(IT-7)76,7,76
76  CONTINUE
          IF(KPC)44,44,45
44  CONTINUE
          IF(IT-1)64,63,64
63  CONTINUE
          MO = 10
          GO TO 1
64  CONTINUE
          MO = 2
          GO TO 3
45  CONTINUE
          MO = 5
          GO TO 4
C     IMM AFTER BREAK
40  CONTINUE
          IF(IT-8)81,47,46
81  CONTINUE

```

```

      IF(IT-7)80,79,80
C     BLANK
      79 CONTINUE
      IF(KPC)86,86,85
      86 CONTINUE
      MO = 2
      GO TO 30
      85 CONTINUE
      MO = 5
      GO TO 30
      80 CONTINUE
      IF(IT-5)89,84,89
      89 CONTINUE
      IF(IT-3)91,90,91
C     LPAREN
      90 CONTINUE
      KOW(IOW) = -108
      GO TO 92
      91 CONTINUE
      IF(IC-SLASH)93,65,93
      93 CONTINUE
      IF(IC-1R*)10,94,10
C     ASTERISK
      94 CONTINUE
      KOW(IOW) = -101
      IOW = IOW + 1
      GO TO 7
C     RIGHT PAREN
      10 CONTINUE
      IF(IT-2)98,82,98
      84 CONTINUE
C     DOLLAR
      KOW(IOW) = -101
      IOW = IOW + 1
      GO TO 79
C     DIGIT
      47 CONTINUE
      KOW(IOW) = - IC
      IOW = IOW + 1
      7 CONTINUE
      IF(KPC)49,49,48
      49 CONTINUE
      MO = 2
      GO TO 1
      48 CONTINUE
      MO = 5
      GO TO 1
C     LETTER
      46 CONTINUE
      MO = 8
      KOW(IOW) = -103
      GO TO 3
C     ERROR
      98 CONTINUE

```

```

      PRINT 9902
      GO TO 1
5     CONTINUE
      IF(MO-5)50,6,6
C     NORMAL MODE
C     MUST BE BREAK. TO TO IMM MODE.
50    CONTINUE
      MO = 10
      GO TO 1
C     ACT PAR LIST MODE
6     CONTINUE
      GO TO (52,54,53,51) IT
C     BREAK
52    CONTINUE
      MO = 10
      GO TO 1
C     RPAREN
54    CONTINUE
      KPC = KPC - 1
      IF(KPC)55,55,32
55    CONTINUE
      JPC = JPC - 1
      IF(JPC)98,56,56
56    CONTINUE
      KOW(IOW) = -104
      KOW(IOW+1) = -105
      IOW = IOW + 2
      GO TO 7
C     LPAREN
53    CONTINUE
      KPC = KPC + 1
      GO TO 32
C     COMMA
51    CONTINUE
      IF(KPC-1)32,57,32
57    CONTINUE
      KOW(IOW) = -104
      IOW = IOW + 1
      GO TO 1
9901  FORMAT(80R1)
9902  FORMAT(13H FETCH ERROR      )
9903  FORMAT(1H ,6017)
9904  FORMAT(20X,80R1)
9801  FORMAT(* SN 4 IIC =*           ,I1,* IC=*,R1,* IT=*,I1,* MO=*,I1,
      X * ISW=*,I2)
9802  FORMAT(* SN 36+2 IOW =*,I3,* IOC =*,I1,* IC=*,R1,* J=*,O16,* KOW=*
      X ,O16)
9803  FORMAT(* SN 34+2 IOW=*,I3,* IOC=*,I1,* ISW=*,I2,* J =*,O16,* KOW=*
      X ,O16)
9804  FORMAT(* SN2+1 IC=*,R1,* MO=*,I3,* KPC=*,I3,* JPC=*,I3)
9805  FORMAT(* SN1+1 IOW-1=*,I3,* KOW=*,O16)
      END
      SUBROUTINE DISPATCH(KIW,NIW)
      COMMON K1604

```

```

C      DIMENSION N(8),M(8),KBL(8),KIW(200),KOW(20)

      TYPE INTEGER OW,BREAK
      DATA(N = 1000000000000000B, 1000000000000000B, 1000000000000B,
X 1000000000B, 1000000B, 10000B, 10CB, 1)
      DATA(M = 77000000000000000B, 7700000000000000B, 7700000000000B,
X 77000000000B, 770000000B, 770000B, 7700B, 77B)
      DATA(KBL = 8R      ,7R      ,6R      ,5R      ,
X 4R      ,3R      ,2R      ,1R      )
      DATA(KABL=8R      )
      DATA(NOW = 9),(BREAK=14B)
      DIMENSION J1636(64)
      DATA(J1636 = 12B,1,2,3,4,5,6,7,10B,11B,0,13B,14B,15B,16B,17B,
X 60B,61B,62B,63B,64B,65B,66B,67B,70B,71B,72B,73B,74B,75B,76B,77B,
X 40B,41B,42B,43B,44B,45B,46B,47B,50B,51B,52B,53B,54B,55B,56B,57B,
X 20B,21B,22B,23B,24B,25B,26B,27B,30B,31B,32B,33B,34B,35B,36B,37B)
C      INITIALIZE
      ISW = 0
      IIW = IOW = IIC = IOC = 1
      IW = KIW(IIW)
      OW = 0
C      EXTRACT CHARACTER
1  CONTINUE
      K = IW .AND. M(IIC)
      K = K/N(IIC)
      K = K .AND. 77B
C      TEST FOR BREAK CHAR
      IF(K-BREAK)2,3,2
C      INSERT CHAR
2  CONTINUE
      ISW = 1
      IF(K1604)16,17,16
16 CONTINUE
      K = J1636(K+1)
17 CONTINUE
      J = K * N(IOC)
      OW = OW .OR. J
C      NEXT OUT CHAR
      IOC = IOC + 1
C      TEST END OF OUT WORD
      IF(IOC-9)5,4,4
C      NEXT OUT WORD
4  CONTINUE
      KOW(IOW) = OW
      OW = 0
      IOW = IOW + 1
      IOC = 1
C      NEXT INP CHAR
5  CONTINUE
      IIC = IIC + 1
C      TEST END OF INP WORD
      IF(IIC-9)8,3,3
C      NEXT INP WORD
3  CONTINUE

```

```

      IIW = IIW + 1
      IW = KIW(IIW)
      IIC = 1
C     TEST MORE INP WORDS
      7 CONTINUE
      IF(IIW-NIW)8,8,9
C     TEST MORE OUT WORDS
      8 CONTINUE
      IF(IOW-NOW)1,1,10
C     OUTPUT LINE
      10 CONTINUE
      IF(KOW(1)-8H      X  )31,33,31
      31 CONTINUE
      IF(KOW(1)-8H      )32,33,32
      32 CONTINUE
      PRINT 9100,(KOW(KK),KK=1,NOW)
      WRITE(20,9101),(KOW(KK),KK=1,NOW)
C     SET UP CONTINUATION
      35 CONTINUE
      KOW(1) = 6H      X
      IOW = 2
      ISW = 0
      IOC = 1
      GO TO 1
      33 CONTINUE
      DO 34 KK = 1,NOW
      IF(KOW(KK)-8H      )32,34,32
      34 CONTINUE
      GO TO 35
C     TEST ANY OUT WORDS
      9 CONTINUE
      IF(ISW)11,99,11
C     INSERT BLANK CHARS
      11 CONTINUE
      OW = OW .OR. KBL(IOC)
      KOW(IOW) = OW
C     NEXT OUT WORD
      12 CONTINUE
      IOW = IOW + 1
C     TEST MORE OUT WORDS
      IF(IOW-NOW)13,13,14
C     INSERT BLANK WORD
      13 CONTINUE
      KOW(IOW) = KABL
      GO TO 12
C     OUTPUT LINE
      14 CONTINUE
      IF(KOW(1)-8H      X  )21,23,21
      21 CONTINUE
      IF(KOW(1)-8H      )22,23,22
      22 CONTINUE
      PRINT9100,(KOW(KK),KK=1,NOW)
      WRITE(20,9101),(KOW(KK),KK=1,NOW)
      99 CONTINUE

```

```
      RETURN
23  CONTINUE
      DO 24 KK = 2,NOW
      IF(KOW(KK)-8H      )22,24,22
24  CONTINUE
      GO TO 99
9100 FORMAT(1H , 10A8)
9101 FORMAT(9A8,8H      )
      END
```