SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

066003

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

CONTRIBUTED PROGRAM LIBRARY SUBMITTAL FORM
(for IBM S/360, 1130 and 1800)

IBM Corporation
Program Information Department (PID)
40 Saw Mill River Road
Hawthorne, New York 10532, U.S.A.
Attention: Program Control Desk

This form should be completed and submitted with the program package to PID at the address shown above. Standards and instructions for submitting programs are in your *User Group Reference Manual* or the *Contributed Program Submittal Standards Manual* available from PID.

(1) Program Order Number *(to be filled in by PID)* 360D-06.6.003

(2) System Type *(machine)* . . . . . S / 360

(3) Search Key . . . . . FORTRAN CHARACTER STRING PACKAGE

(4) Programming Language . . . . . 360 OS ALF

(5) Author's Name and Address . . . . Harold P. Sieglaff
3610 W Northview
Phoenix
Arizona 85021

(6) Direct Inquiries to Name and Address *(if different than Author)*

(7) Title of Program . . . . FORTRAN CHARACTER STRING PACKAGE

DISCLAIMER

Triangle Universities Computation Center (TUCC) serves solely as the distribution agent for contributed programs and does not test or maintain them. They are distributed essentially in the original form submitted by the author. Neither TUCC nor SHARE, INC. makes any warranty, expressed or implied, as to the documentation, function, or performance of the contributed programs.

(8) Submitter's User Group Affiliation Code and Installation Code . . . . . N

(9) Submitter's Own Program Identification and Suffix *(optional)* . . . . . F C S P

(10) Primary Subject Code . . . . . 0 6 . 6

(11) Secondary Subject Codes . . . . . 0 6 . 4 | 0 6 . 3 | . | .

(12) Operating or Monitor System Required . . . . . SEE ABSTRACT

(13) New or Revision Code *(if revision, show prior Program Order Number in item 1)* . . . . . N

(14) Year Completed 68

(15) Date of Submittal 12 17 68

(16) Documentation *(number of original pages submitted)* 25

(17) Abstract *(should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.*

CONTRIBUTED PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide

a. Purpose
b. Programming Language used
c. Version and modification level or release number of IBM Programming System used, or program order number for non-IBM authored program used
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements
g. Engineering Changes (EC) level of equipment (if pertinent)

ABSTRACT The package enables the Fortran programmer to process character strings.

The package consists of subroutines written in OS Assembly Language F and was tested using OS Fortran Language 4 G Level and OS Version 13 on a S/360 model 50.

The Fortran programmer can compare, filter, insert, move, scan, and binary search character strings by calling one or more of the six subroutines.

Maximum core requirement is 1K.

The package should work on any S/360 machine which has Fortran IV G and OS.

(Please attach additional pages if necessary) . . . . . **Total pages attached** _____

Permission to Publish

"I hereby give anyone permission to reprint, reproduce, and distribute this program to anyone else."

(18) Signature of Submitter and Date Harold P. Sieglaff 17 DEC 68

(19) Signature of Installation Addressee _____

T4SF

TABLE OF CONTENTS

DECK KEY

Deck #1 Source deck of BINSER, sequence 00 thru 560 in cc 73-80; 59 cards.

Deck #2 Source deck of CMPARE, sequence 00 thru 620 in cc 73-80; 64 cards.

Deck #3 Source deck of INSERT, sequence 00 thru 170 in cc 73-80; 18 cards.

Deck #4 Source deck of LEG, sequence 00 thru 250 in cc 73-80; 28 cards.

Deck #5 Source deck of MOVEIT, sequence 00 thru 300 in cc 73-80; 32 cards.

Deck #6 Source deck of SETSKP, sequence 00 thru 1080 in cc 73-80; 112 cards.

A character string package for the FORTRAN programmer

written by Harold P. Sieglaff

BINSER	makes a binary search of an alphanumerically ordered table for a specified string of characters
CMPARE	scans AREA2 searching for AREA1
INSERT	inserts characters from one array into another array
LEG	makes an alphanumeric comparison of A and B and indicates whether or not A is less than, equal to, or greater than B
MOVEIT	moves N bytes from AREA1 to AREA2
SETSKP	provides a means to skip or seek specified characters while scanning a string of characters

To provide character strings in FORTRAN one can use a statement of the form:

LOGICAL * 1 DATA(xxx) , STRING(xx,xxx)

One can read/write into/from any part of the strings by using format A1 . The examples in the subroutine write-ups use this approach.

Definition of a binary search:

search, binary, a search in which the series of items is divided into two parts, one of which is rejected, and the process repeated on the unrejected part until the item with the desired property is found. This process usually depends upon the presence of a known sequence in the series. Synonymous with (dichotomizing search).

Automatic data processing glossary, reprinted by

Datamation Magazine, 35 Mason St., Greenwich, Conn. 06830, pp. 47-48.

Because we do not have mathematical symbols on our typewriter the the write-up uses FORTRAN symbols for:

less than:	.LT.
less than or equal to:	.LE.
greater than:	.GT.

For increased clarity these symbols have been underlined wherever they appear.

NAME: What is the name of the subroutine ?

ENTRY POINT(S): If different from the name

DESCRIPTION: What does the subroutine do ?

AUTHOR: Who wrote it ?

LANGUAGE: What language was it written in ? (Fortran, Cobol, Bal, etc.)

MEMORY REQUIREMENTS: How much memory does it use ?

AVAILABILITY: Where can a copy be obtained ? What library is it in ?

CALLING SEQUENCE: How do you call the subroutine ?

PARAMETER DESCRIPTIONS: Describe each parameter.
What is the function of each of the parameters ?

RESTRICTIONS: What limitations does the subroutine have ?

ERROR INDICATIONS: What error indications (if any) are returned to the calling program ?

OTHER SUBROUTINES REQUIRED: What other subroutines (if any) are required to use this one ?

EXAMPLES OF ITS USE: Give concrete examples with comments as to what is being done in each example.

FLOWCHART: Include a flowchart of the subroutine.

NAME: BINSER

ENTRY POINT: BINSER

DESCRIPTION: BINSER makes a binary search of an alphanumerically ordered table for a specified string of characters.

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY REQUIREMENTS: 216 bytes

AVAILABILITY: BINSER is available as a source deck from the author.

CALLING SEQUENCE: CALL BINSER (TABLE(1) , LSTART, LEND, NBYTES, X, NCHARX, MP, IND)

PARAMETER DESCRIPTIONS:

- TABLE(1) = first position in the table (TABLE can have one or more dimensions.)
- LSTART = no. of the position where the search is to start
- LEND = no. of the position where the search is to end
- NBYTES = no. of bytes used for each entry in the table
- X = name of the string of characters to be searched for (X can be a dimensioned array.)
- NCHARX = no. of characters of X which are to be searched for

upon return to the calling program:

MP = no. of the position of the table where the search ended

If IND = 0 then TABLE(MP) = X and the search was successful.

If IND = -1 then TABLE(MP) .LT. X .LT. TABLE(MP+1) and the search was unsuccessful.

(note: -1 + LSTART .LE. MP .LE. LEND)

* RESTRICTIONS: The same no. of bytes must be used for each entry in the table.

NCHARX determines the no. of characters of each table entry which are compared with X and the no. of characters of X used in the comparisons.

Alphanumeric ordering is as follows:

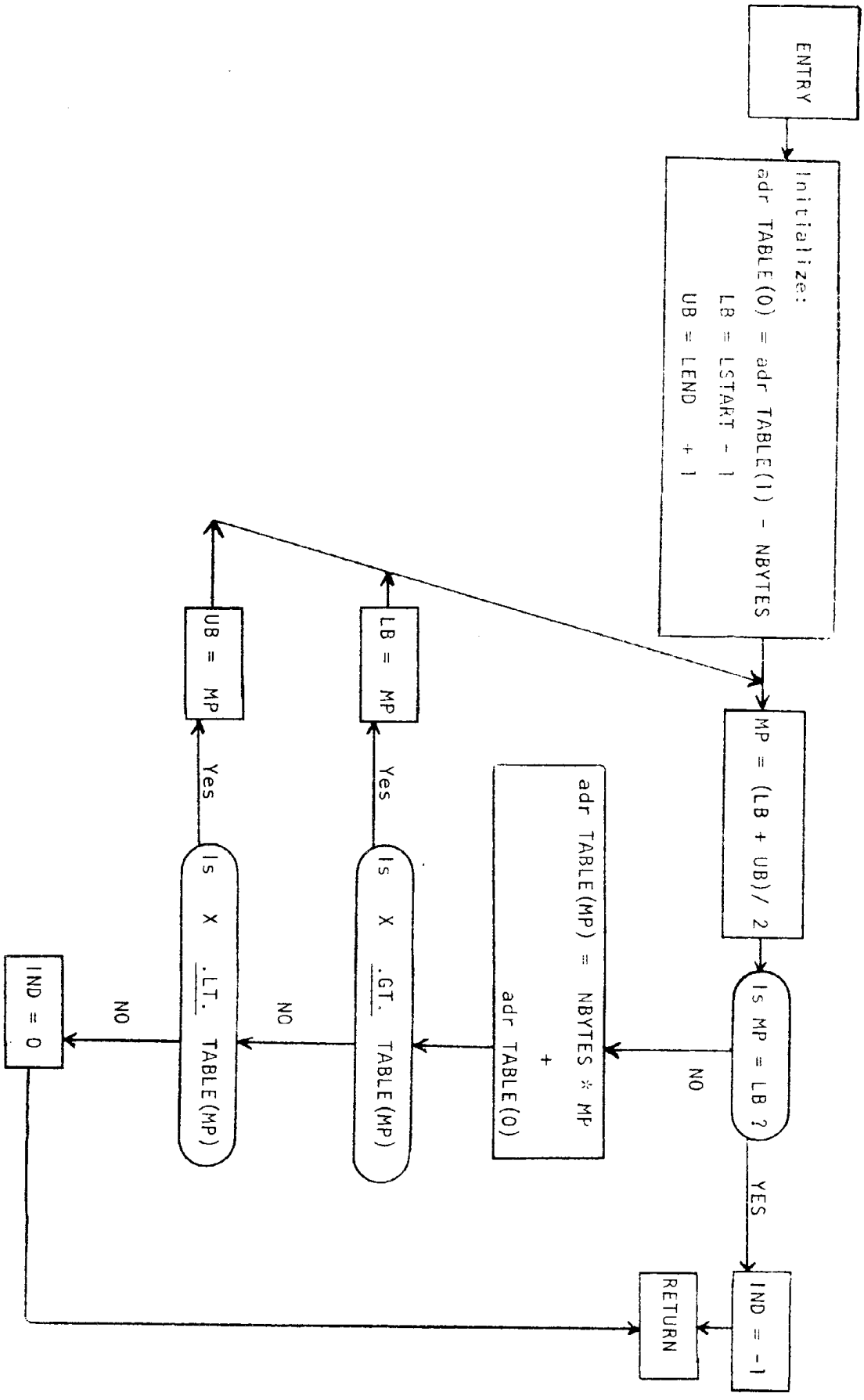
blank .LT. A .LT. B .LT. C .LT. D ... .LT. Z .LT. 0 .LT. 1 ... .LT. 9

(There are 256 possible characters; they are ordered by the collating sequence.)

ERROR INDICATIONS: none

* NCHARX must be less than or equal to 256

FLOWCHART OF BINSER



```
      LOGICAL * 1    TABLE(80,500) , CARD(80)
      EQUIVALENCE  (ITEST , CARD(1))
      WRITE (6,16)
   16 FORMAT (1H1)
C***  read end of data indicator
      READ (5,3) IENDAT
    3 FORMAT (A4)
C***  read cards which make up the table
      DO  1  NCARD = 1,500
      READ (5,2) CARD
    2 FORMAT (80 A1)
      IF (IENDAT .EQ. ITEST) GO TO 5
      NENTRY = NCARD
C***  write the card & the no. of its position in the table
      WRITE(6,15)  NENTRY , CARD
   15 FORMAT (1HO, I5, 5X, 80A1)
C***  move the card into the table
      DO  10  N = 1,80
   10 TABLE(N,NCARD)  =  CARD(N)
    1 CONTINUE
C***  read  X  card  (contains words to be searched for)
    5 READ (5,11)  (CARD(N) , N = 1,40) , NUMCHR
   11 FORMAT (40A1 , I2)
C***  is this the end of data card ?
      IF (IENDAT  .EQ. ITEST) GO TO 12
      IF (NUMCHR  .LT. 0)     GO TO 12
C***  search the  TABLE  for  X
      CALL BINSER (TABLE(1,1) , 1,  NENTRY, 80, CARD(1), NUMCHR, MP, IND)
C***  write out the  X  card and the indicator telling whether or not the
C
C     search was successful
      WRITE (6,13)  (CARD(N) , N = 1,40) , MP, IND
   13 FORMAT (1HO, 40A1, I9, I6)
C***  if the search was successful, write out the table entry X is equal to
      IF (IND .EQ. 0)  WRITE (6,14) (TABLE(N,MP) , N = 1,80)
   14 FORMAT (1HO, 80A1)
      GO TO 5
   12 STOP
      END
```

NAME: CMPARE

ENTRY POINT: CMPARE

DESCRIPTION: CMPARE scans AREA2 searching for AREA1

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY REQUIREMENTS: 460 bytes

AVAILABILITY: CMPARE is available as a source deck from the author

CALLING SEQUENCE: CALL CMPARE (AREA1(NS1),AREA1(NE1), AREA2(NS2),AREA2(NE2), IND)

PARAMETER DESCRIPTIONS: Scan AREA2(NS2) thru AREA2(NE2) for AREA1(NS1) thru AREA1(NE1)

Upon return if:

AREA1 ≠ a portion of AREA2 IND = 0

AREA1 = a portion of AREA2 then IND = the no. of the AREA2 position where equality started

IND = 1 for NS2
= 2 for NS2 + 1
= 3 for NS2 + 2 etc.

RESTRICTIONS: (NE1 - NS1) and (NE2 - NS2) must both be less than or equal to 255

ERROR INDICATIONS: none

EXAMPLE OF ITS USE: CALL CMPARE (AREA1(1),AREA1(10),AREA2(1),AREA2(256), IND)

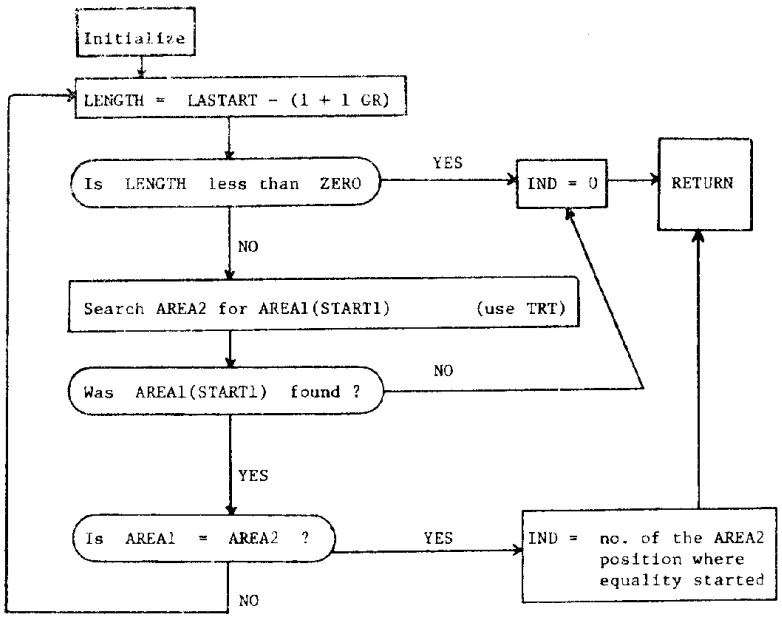
Scan AREA2(1) thru AREA2(256) for AREA1(1) thru AREA1(10)

if not found set IND = 0

if found set IND = the no. of the AREA2 position where equality started


```
      LOGICAL * 1   AREA1(80) , AREA2(320)
      WRITE (6,4)
    4 FORMAT (1H1)
C***  READ DATA INTO AREA1  and  AREA2
      READ (5,1)  AREA1 , AREA2
    1 FORMAT (80 A 1)
      DO  2  NCALL = 1,12
C***  Scan AREA2  for  AREA1
      CALL CMPARE (AREA1(NCALL) , AREA(12) , AREA2(1) , AREA2(256) , IND)
C***  Write the result of the scan
      WRITE (6,3)  NCALL , IND
    3 FORMAT (1HO, 'NCALL = ' , I 2  ,  '  IND = ' , I 3 )
    2 CONTINUE
      STOP
      END
```

Flowchart of CMPARE



1 GR = AREA2 adr where AREA1(START1) was last found

adr for start of search = 1 + 1 GR

adr for end of search = END2 - (END1 - START1) = LASTART

= highest possible starting AREA2 adr if equality is to be found

LENGTH = LASTART - (1 + 1 GR) = -1 + no. of arguments in AREA2

In FORTRAN provide core storage for AREA1 and AREA2 as follows:

LOGICAL * 1 AREA1( ) , AREA2( )

Read into AREA1 and AREA2 using A1 format.

NAME: INSERT

ENTRY POINT: INSERT

DESCRIPTION: INSERT inserts characters from one array into another array.

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY REQUIREMENTS: 120 bytes

AVAILABILITY: INSERT is available as a source deck from the author.

CALLING SEQUENCE: CALL INSERT (A(LSTART) , A(LEND) , B(NSTART))

PARAMETER DESCRIPTIONS: Insert characters A(LSTART) thru A(LEND) into B starting at NSTART

RESTRICTIONS: LEND - LSTART must be equal to or less than 255

ERROR INDICATIONS: none

EXAMPLES OF ITS USE: CALL INSERT (A(109) , A(359) , B(175))

CALL INSERT (S(L) , S(K) , TERM(J))

```
      LOGICAL * 1  A(320) , B(400)
C***  Read in  A   4 cards of  80  characters
      READ (5,1)  A
    1 FORMAT ( 80 A 1 )
C***  Read starting & ending positions of  A  &  starting for  B
   10 READ (5,2)  LSTART , LEND , NSTART
    2 FORMAT ( 3 I 5 )
      IF  (LSTART .LT. 0) GO TO 3
      NEND = NSTART  +  (LEND - LSTART)
C***  Insert the specified characters where specified
      CALL INSERT (A(LSTART) , A(LEND) , B(NSTART) )
C***  Dump the result of the insertion
      CALL PDUMP (LSTART, LSTART, 4, LEND, LEND, 4, NSTART, NSTART, 4,
     X            NEND, NEND, 4, B(NSTART), B(NEND), 9 )
      GO TO 10
    3 STOP
      END
```


NAME: LEG

ENTRY POINT: LEG

DESCRIPTION: LEG makes an alphanumeric comparison of A and B and indicates whether A is less than, equal to, or greater than B

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY REQUIREMENTS: 160 bytes

AVAILABILITY: LEG is available as a source deck from the author.

CALLING SEQUENCE: CALL LEG (A(LSTART) , A(LEND) , B(NSTART) , IND)

PARAMETER DESCRIPTIONS: Compare A(LSTART) thru A(LEND) with B starting at NSTART

upon return to the calling program:

if A is less than B IND = -1

is equal to B IND = 0

is greater than B IND = +1

* RESTRICTIONS: LEND - LSTART must be equal to or less than 255

ERROR INDICATIONS: none

EXAMPLES OF ITS USE:

```
CALL LEG (A(LSTART) , A(LEND) , B(NSTART) , IND)
CALL LEG (A(1) , A(256) , B(1) , IND)
CALL LEG (A(3) , A(89) , B(1) , IND)
CALL LEG (A(41), A(48) , B(60), IND)
```

* Alphanumeric ordering is as follows:

A .LT. B .LT. C .LT. D ... .LT. Z .LT. 0 .LT. 1 ... .LT. 9

(There are 256 possible characters; they are ordered by the collating sequence.)

```
      LOGICAL * 1  A(320) , B(400)
C***  read in  A  &  B     (4 & 5  cards of 80 characters each)
      READ (5,1)  A , B
    1 FORMAT (80 A1)
C***  read starting & ending positions of A  &  starting for B
   10 READ (5,2) LSTART , LEND , NSTART
    2 FORMAT (3 I 5)
      IF (LSTART .LT. 0) GO TO 3
      NEND = NSTART + (LEND - LSTART)
      CALL LEG (A(LSTART) , A(LEND) , B(NSTART) , IND)
C***  dump the result of the comparison
      CALL PDUMP (LSTART,LSTART,4,LEND,LEND,4,NSTART,NSTART,4,
     X      NEND,NEND,4,A(LSTART),A(LEND),9,B(NSTART),B(NEND),9,IND,IND,4)
      GO TO 10
    3 STOP
      END
```


NAME: MOVEIT

ENTRY POINT: MOVEIT

DESCRIPTION: MOVEIT moves N bytes from AREA1 to AREA2

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY REQUIREMENTS: 180 bytes

AVAILABILITY: MOVEIT is available as a source deck from the author.

CALLING SEQUENCE: CALL MOVEIT (AREA1(LSTART) , AREA2(NSTART) , NBYTES)

PARAMETER DESCRIPTIONS: Move NBYTES from AREA1 to AREA2
starting at AREA1(LSTART) and AREA2(NSTART)

RESTRICTIONS: none

ERROR INDICATIONS: none

OTHER SUBROUTINES REQUIRED : none

EXAMPLES OF ITS USE:

```
CALL MOVEIT (DATA(L) , DATAB(N) , NBYTES)
CALL MOVEIT (BUFFER(1) , AREA(1) , 1000)
CALL MOVEIT (TEMP(1,1,1) , CMPUTE(1,1,1) , 4000)
```

```
      DIMENSION  DATA1(1000) , DATA2(1000)
C***  Initialize  DATA1
      DO  1  N = 1,1000
      DATA1(N) = N
    1 CONTINUE
C***  Move more than 256 bytes
      CALL MOVEIT (DATA1(1) , DATA2(1) , 4000)
      CALL PDUMP  (DATA2(1) , DATA2(1000) , 5)
C***  Move 256 bytes
      CALL MOVEIT  (DATA1(1) , DATA2(7) , 256)
      CALL PDUMP (DATA2(7) , DATA2(70) , 5)
C***  Move fewer than 256 bytes
      CALL MOVEIT (DATA1(1) , DATA2(100) , 100)
      CALL PDUMP (DATA2(100) , DATA2(124) , 5)
      STOP
      END
```


NAME: SETSKP

ENTRY
POINTS:

SETSKP	supplies a string of characters which are to be ignored (skipped) while scanning a string supplied by the CALL SKIP statement
SKIP	supplies a string of characters which is to be scanned while skipping the characters supplied by the CALL SETSKP statement
SETSEK	supplies a string of characters which are to be searched for while scanning a string supplied by the CALL SEEK statement
SEEK	supplies a string of characters which is to be scanned in a search for the characters supplied by the CALL SETSEK statement

DESCRIPTION: SETSKP provides a means to skip or seek specified characters while scanning a string of characters

AUTHOR: Harold P. Sieglaff

LANGUAGE: BAL

MEMORY
REQUIREMENTS: 916 bytes (2 tables of 256 bytes each are used)

AVAILABILITY: SETSKP is available as a source deck from the author.

CALLING SEQUENCES:

```
CALL SETSKP (IGNORE(1), NBYTES)
CALL SKIP   (STRING(1), NCHARS, NUMPOS)
CALL SETSEK (STOPIF(1), NBYTES)
CALL SEEK   (STRING(1), NCHARS, NUMPOS)
```

PARAMETER DESCRIPTIONS:

IGNORE(1) = first position of a string of characters which are to be ignored (skipped) when scanning a string supplied by the CALL SKIP statement

NBYTES = no. of bytes (characters) to be used from string IGNORE

STRING(1) = first position of a string of characters which is to be scanned while skipping the NBYTES characters of the string IGNORE

NCHARS = no. of bytes (characters) in the string STRING which are to be scanned

upon return to the calling program:

NUMPOS = no. of the first position in STRING not equal to any character in string IGNORE

If NUMPOS = 0 then all of the characters of STRING are equal to some character in IGNORE

CALL SKIP may be executed any number of times after CALL SETBKP has been executed.

CALL SEEK may be executed any number of times after CALL SETSEK has been executed.

STOPIF(1) = first position of a string of characters which are to be searched for while scanning a string supplied by the CALL SEEK statement

NBYTES = no. of bytes (characters) to be used from string STOPIF

STRING(1) = first position of a string of characters which is to be scanned in a search for the NBYTES characters of string STOPIF

NCHARS = no. of bytes (characters) in the string STRING which are to be searched

upon return to the calling program:

NUMPOS = no. of the first position in STRING equal to any of the characters in string STOPIF

If NUMPOS = 0 then no character in STRING is equal to any of the characters in string STOPIF

RESTRICTIONS:

NCHARS must be less than or equal to 256

NBYTES must be less than or equal to 255

ERROR INDICATIONS: none

EXAMPLES OF ITS USE:

```
CALL SETSKP (IGNORE(I), NBYTES)
CALL SKIP   (STRING(N), NCHARS, NPOS)
CALL SKIP   (STRING(K), NCHARS, NPOS)
CALL SETSEK (STOPIF(1), NUMSEK)
CALL SEEK   (ALPHNU(J), KSYMS, NUMCHR)
CALL SEEK   (ALPHNU(L), KSYMS, NUMCHR)
```

EXAMPLE OF ITS USE:

```
      LOGICAL * 1  STRING(80) , IGNORE(40) , STOPIF(40)

C***  read in  STRING

      READ (5,1)  STRING
    1 FORMAT (80 A1)
C***  read in string IGNORE and no. of chars in it
    4 READ (5,2)  IGNORE , NUMSKP
    2 FORMAT (40 A1, I 2)
C***  is this the end of data card ?

      IF (NUMSKP  .LT.  0)  GO TO 3
C***  read in string STOPIF  and no. of chars in it

      READ (5,2)  STOPIF , NUMSEK
C***  prepare to skip the characters in string  IGNORE

      CALL SETSKP (IGNORE(1) , NUMSKP)
C***  scan string STRING  skipping chars in string  IGNORE

      CALL SKIP (STRING(1) , 80, NUM)
C***  prepare to seek the characters in string  STOPIF

      CALL SETSEK (STOPIF(1) , NUMSEK)
C***  scan string STRING  searching for any character in string STOPIF

      CALL SEEK (STRING(1) , 80, KUM)
C***  dump the results of the skipping & seeking

      CALL PDUMP (STRING(1),STRING(80),9,IGNORE(1),IGNORE(40),9,
     X   NUMSKP,NUMSKP,4,STOPIF(1),STOPIF(40),9,NUMSEK,NUMSEK,4,
     X   NUM,NUM,4,KUM,KUM,4)
      GO TO 4
    3 STOP
      END
```