NGL 44-005-084

N71 74823
(ACCESSION NUMBER) (THRU)
116 None
(PAGES) (CODE)
CR-119702
(NASA CR OR TMX OR AD NUMBER) (CATEGORY)
FACILITY FORM 602

UNIVERSITY OF HOUSTON

SEP 1971
RECEIVED
NASA STI FACILITY
INPUT BRANCH

STIL SYSTEMS MANUAL

Charles E. Donaghey
and
Osman S. Ozkul

RE 12-69
September, 1969

Reproduction in whole or in part is permitted for any purpose of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

PROJECT THEMIS
Information Processing Systems

ONR Contract: N00014-68-A-0151

STIL SYSTEMS MANUAL

by

Charles E. Donaghey and Osman S. Ozkul

Department of Industrial Engineering
Cullen College of Engineering
University of Houston

RE 12-69

September, 1969

© Copyright 1970

Reproduction in whole or in part is permitted for any purpose of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

ACKNOWLEDGEMENTS

This work was supported by Project THEMIS, ONR Contract N00014-68-0151.

TABLE OF CONTENTS

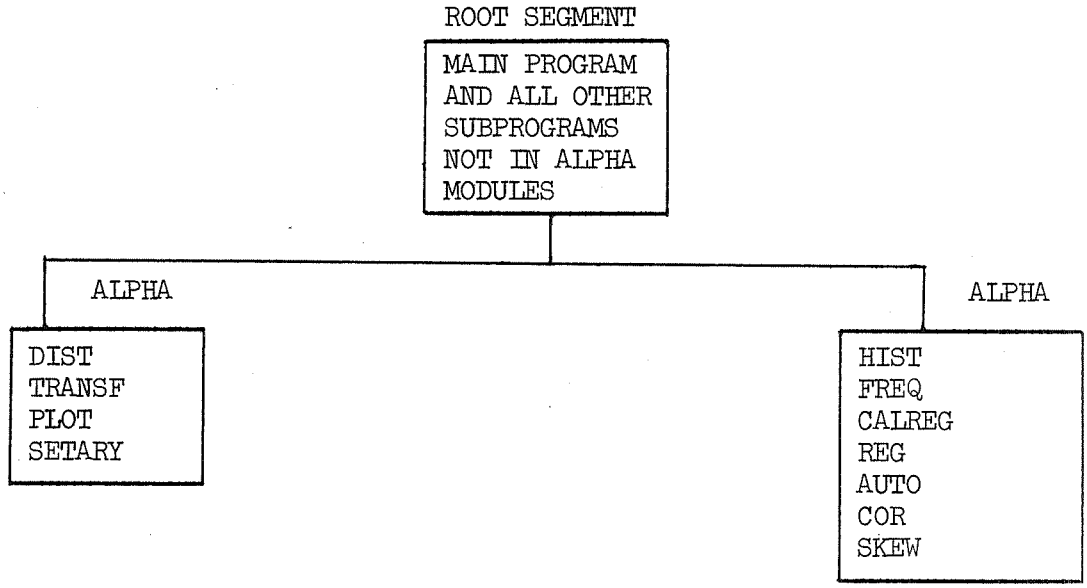
	Page
Introduction	1
Main	2
Input	4
Indata	8
Read	14
Interp	20
Output	23
Freall	25
Freear	27
Dump	29
Ranent	32
Plot	34
Avg	38
Var	40
Range	42
Regr	44
Cor	52
Auto	55
Freq	59
Hist	62
Skew	66
Calreg	69
Setary	74
Transf	77
Dist	83
Bad	91
Block Data	93
Decode	95
Integ	97
Frac	99
Ran	100
Destry	103
Ckname	105
Asname	107
Ascell	109

INTRODUCTION

This manual is intended for use by those concerned with the design of the STIL language. For those interested in just the use of STIL, they are referred to the STIL USERS MANUAL.

The STIL interpreter has been entirely written in FORTRAN IV and has been compiled and utilized on a number of computer systems. The interpreter consists of a main program and 33 sub-programs. This manual describes and gives a listing of each section of the interpreter.

The entire program requires about 25K words of memory. This includes the space for the users STIL program. About 5K of memory can be saved by using an overlay structure as shown below



OVERLAY STRUCTURE

MAIN PROGRAM

The main program performs a number of initialization tasks, prints the word "STIL" and then calls subroutine INPUT to begin reading the user's STIL program.

```
C     MAIN PROGRAM
C
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/LSEM,LWC,JLPRC,JRPRC
C
C     INITIALIZE THE VARIABLES AND ARRAYS, AND CALL SUBROUTINE INPUT TO
C     PROCESS THE STIL PROGRAM
C
      DO 10 I=1,MASZ
      NAME(I)=0
      KST(I)=0
      LNGT(I)=0
      AVRG(I)=JPERC
      STND(I)=JPERC
   10 CONTINUE
      DO 15 I=1,MPSZ
      K(I)=JBLKC
   15 CONTINUE
      DO 20 I=1,MDSZ
      DATA(I)=JBLKC
      NX(I)=JPERC
   20 CONTINUE
      K(JPOS)=JSEMC
      JPOS=JPOS+1
      WRITE (6,22)
   22 FORMAT(  '  STIL ',// )
      CALL INPUT
      END
```

SUBROUTINE INPUT

This routine exercises control over reading the user's STIL program. Each card in the program is read into a vector named L under an 80A1 format specification. The non-blank characters are then placed in consecutive order into a vector named K. K is capable of holding 4000 characters. This reading process is stopped under the following conditions.

(a) When an asterisk '*' is encountered. It is then assumed that the STIL instructions which are read in so far need to be executed. A pointer (JPOS) is set to 1 and subroutine INTERP is called to begin executing the STIL instructions.

(b) When a formatted read statement is found. It is then necessary to execute the portion of the STIL program up to this point. A flag (KFLAG) is set to 22, JPOS is set to 1 and subroutine INTERP is called.

(c) When the K array is filled. KFLAG is set to 1, JPOS is set to 1 and subroutine INTERP is called to begin executing the instructions so far stored in the K array.

Control is returned to INPUT after executing the program segment previously read. INPUT will examine KFLAG and if it is equal to 22 it will call subroutine READ. If KFLAG is 1 some of the characters at the end of K may have to be placed in the beginning of the K vector. For example assume the locations 3997-4000 in K contain the characters ';FIN'. The characters in K_1 - K_{3996} have just been executed. Characters in K_{3997} - K_{4000} will be transferred to K_1 - K_4 and the non-blank characters of the remainder of this instruction and the remaining portion of the program will be stored in K starting with location 5.

If the reading is stopped due to the presence of an asterisk, an attempt will be made to read more data cards and the cycle will be repeated. If there are no cards to read, it is assumed that the program is finished and an asterisk is placed in K(2). When subroutine INTERP is called this will bring the execution to a normal stop.

```
      SUBROUTINE INPUT
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION LP(80)
C     IF THE PROGRAM IS BEING EXECUTED IN SEGMENTS GO TO STATEMENT 55
      IF( KFLAG .EQ. 1 .AND. JPOS .GE. LSEM) GO TO 55
      IF( KFLAG .EQ. 22 .AND. JPOS .GE. LSEM) GO TO 55
C     READ AND PRINT A CARD
   10 READ (5,12,END=25 ) (L(I),I=1,80)
   12 FORMAT(80A1)
      WRITE(6,13) L
  13   FORMAT(1X,80A1)
      J=1
C     SUPPRESS BLANKS
      DO 15 I=1,80
      IF( L(I) .EQ. JBLKC ) GO TO 15
      LP(J)=L(I)
      J=J+1
   15 CONTINUE
      IF( J .EQ. 1 ) GO TO 10
C     IF THE INSTRUCTION IS A FORMATTED READ GO TO STATEMENT 135
      IF( LP(6) .EQ. JRC .AND. LP(7) .EQ. JMC .AND. LP(10).EQ. JLPRC )
     1 GO TO 135
      N=J-1
C     STORE THE CONTENTS OF THE CARD INTO K ARRAY
      DO 20 I=1,N
      IF( LP(I) .EQ. JASTC ) GO TO 25
      K(JPOS)=LP(I)
      IF( JPOS .EQ. MPSZ ) GO TO 35
      JPOS=JPOS+1
   20 CONTINUE
      GO TO 10
C     PUT AN ASTERISK INTO THE LAST POSITION TO MARK THE END OF THE STIL PROGRAM
   25 K(JPOS)=JASTC
      KEND=JPOS
      WRITE(6,27)
   27 FORMAT(////)
C     SET JPOS TO THE BEGINNING OF THE K ARRAY AND CALL SUBROUTINE INTERP
C     TO START THE EXECUTION OF STILL INSTRUCTIONS
      JPOS=1
      CALL INTERP
C PROGRAM TO BE RUN IN SECTIONS
C     SET KFLAG TO SEGMENT THE PROGRAM
   35 KFLAG=1
      LCLM=I+1
      GO TO 140
  135 KFLAG=22
C     LOCATE THE LAST COMPLETE INSTRUCTION BY SEARCHING A SEMICOLON FROM THE
C     END OF THE SEGMENT
```

```
  140 DO 40 I=1,MPSZ
      IF( K(JPOS) .EQ. JSEMC ) GO TO 50
      JPOS=JPOS-1
   40 CONTINUE
      WRITE(6,42)
   42 FORMAT(  '  NO SEMI-COLON IN PROGRAM SECTION'  )
      STOP
   50 LSEM=JPOS
C     SET JPOS TO THE BEGINNING OF K ARRAY AND CALL SUBROUTINE INTERP TO
C     BEGIN EXECUTION
      KEND=MPSZ
      JPOS=1
      WRITE(6,27)
      CALL INTERP
C END OF INPUT AND EDITING FOR A SECTION
   55 K(1)=JSEMC
C     ERASE ALL THE EXECUTED INSTRUCTIONS FROM K ARRAY
      JPOS=2
      DO 60 I=2,LSEM
      K(I)=JBLKC
   60 CONTINUE
      IF( KFLAG .EQ. 22 ) GO TO 85
      IF( LSEM .EQ. MPSZ ) GO TO 70
C     TRANSFER THE INFORMATION BEYOND THE LAST EXECUTED INSTRUCTION TO THE
C     BEGINNING OF THE K ARRAY
      N=MPSZ-LSEM
      J=LSEM+1
      DO 65 I=J,MPSZ
      K(JPOS)=K(I)
      JPOS=JPOS+1
   65 CONTINUE
   70 IF( LCLM .GT. 80 ) GO TO 80
C     TRANSFER THE INFORMATION WHICH WAS LEFT ON THE LAST CARD TO K ARRAY
      DO 75 I=LCLM,80
      IF( L(I) .EQ. JASTC ) GO TO 25
      IF( L(I) .EQ. JBLKC ) GO TO 75
      K(JPOS)=L(I)
      JPOS=JPOS+1
   75 CONTINUE
C     RESET KFLAG AND GO BACK TO 10 TO READ THE REMAINING CARDS
   80 KFLAG=0
      LSEM=MPSZ
      GO TO 10
   85 CALL READ
      GO TO 80
C END OF HOUSE-CLEANING FOR NEW SECTION
      END
```

SUBROUTINE INDATA

Format free data input is read in by subroutine INDATA. Common ways of writing numbers are recognized as the normal procedure and the scientific notation for representing numbers (i.e., 34.6×10^3) is not recognized. The maximum number of decimal digits allowed in a number is ten. The values can be written with or without a decimal point and with or without a sign. If a number does not have a sign preceding it, it is assumed to be a positive number, and all the numbers are treated as real (floating point) values. The numbers must be separated by commas and the end of an array of numbers must be indicated by a semicolon just like any other STIL instruction.

If a problem is encountered during the process, such as finding a character other than a decimal digit within the field, or locating a number which has more than 10 decimal digits, or running out of empty cells in the DATA array, an appropriate message is printed, the array name and the information gathered about it so far is erased from the memory and subroutine BAD is called to locate the next instruction in the program.

In general the procedure of conversion goes as follows. First, the field of a number is determined - the field starts with a comma and ends with another comma (except the first and last elements of an array). If there are no decimal digits in between the commas the value of that element is assumed to be zero. If there is a decimal point in the field the number is evaluated in two parts and added together. If there isn't a decimal point it is assumed to be a whole number. After storing the calculated value into the DATA array the next field is determined and the procedure

is repeated until a semicolon is found at the end of a field. This last number is considered to be the last element in the array and an end of array indicator is set in the NX vector for the array in question. Since a semicolon indicates the end of an instruction subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE INDATA
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     IF THE VARIABLE IS ALREADY IN MEMORY, ALL DATA PERTAINING TO IT IS ERASED
      KF=KFLAG
      KFLAG=2
      JPOS=JPOS-4
      CALL FREEAR
      JPOS=JPOS+4
      KFLAG=KF
      KFLG=99
      LLWC=LWC
C     AN EMPTY CELL IN NAME ARRAY IS LOCATED. IF NONE IS AVAILABLE, A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THE INSTRUCTION
      DO 10 I=1,MASZ
      IF( NAME(I) .EQ. 0 ) GO TO 15
   10 CONTINUE
      WRITE(6,12) K(JPOS)
   12 FORMAT(  '  TOO MANY ARRAYS   ',A1 )
      CALL BAD
C     VARIABLE NAME AND THE BEGINNING LOCATION ARE STORED IN NAME AND KST
C     ARRAYS RESPECTIVELY
   15 KARY=I
      NAME(KARY)=K(JPOS)
      KNTR=0
  20  IF(LWC .EQ. MDSZ) GO TO 25
      JR=LWC+1
      GO TO 30
  25  JR=1
 30   IF(NX(JR) .EQ. JPERC) GO TO 40
      KATP=1
  32  CALL RANENT
      IF(NX(JR) .EQ. JPERC) GO TO 40
      IF( KATP .GT. MDSZ ) GO TO 35
      KATP=KATP+1
       GO TO 32
  35   WRITE(6,37) NAME(KARY)
  37  FORMAT(' DATA ARRAY FILLED ', A1)
      LWC=LLWC
      NAME(KARY)=0
      CALL BAD
   40 KST(KARY)=JR
      KATP=1
   45 SUM=0.0
C     THE FIELD WIDTH OF AN ELEMENT IS DETERMINED
      KK=2
      IF(K(JPOS+2) .EQ. JPLSC .OR. K(JPOS+2) .EQ. JMINC) KK=3
      NUMST=JPOS+KK
      JFIN=NUMST+10
      DO 50 I=NUMST,JFIN
```

```
      IF(K(I) .EQ. JPERC .OR. K(I) .EQ. JCOMC .OR. K(I) .EQ. JSEMC)
     1GO TO 55
   50 CONTINUE
C     IF THE FIELD IS GREATER THAN 10 DIGITS, A MESSAGE IS PRINTED, ARRAY IS
C     ERASED AND SUBROUTINE BAD IS CALLED TO SKIP THE INSTRUCTION
      WRITE(6,52) (K(J),J=NUMST,JFIN)
   52 FORMAT(' DATA FIELD TOO WIDE ',10A1)
  510 NAME(KARY)=0
      KNTR=KNTR-1
      KP=KST(KARY)
      KST(KARY)=0
      IF( KNTR .LE. 1 ) GO TO 516
      DO 515 I=1,KNTR
      KN=NX(KP)
      NX(KP)=JPERC
      KP=KN
  515 CONTINUE
      NX(KP)=JPERC
  516 LWC=LLWC
      CALL BAD
C     THE VALUE OF AN ELEMENT IS CALCULATED AND STORED IN DATA ARRAY
   55 NUMEN=I
      NPOS=NUMEN-NUMST
      IF(NPOS .EQ. 0) GO TO 105
      DO 100 I=1,NPOS
      J=NUMEN-I
      IF(K(J) .EQ. J0C) GO TO 60
      IF(K(J) .EQ. J1C) GO TO 61
      IF(K(J) .EQ. J2C) GO TO 62
      IF(K(J) .EQ. J3C) GO TO 63
      IF(K(J) .EQ. J4C) GO TO 64
      IF(K(J) .EQ. J5C) GO TO 65
      IF(K(J) .EQ. J6C) GO TO 66
      IF(K(J) .EQ. J7C) GO TO 67
      IF(K(J) .EQ. J8C) GO TO 68
      IF(K(J) .EQ. J9C) GO TO 69
      WRITE(6,57) (K(J), J=NUMST,NUMEN)
   57 FORMAT( ' BAD DATA  ',20A1)
      GO TO 510
   60 D=0.0
      GO TO 70
   61 D=1.0
      GO TO 70
   62 D=2.0
      GO TO 70
   63 D=3.0
      GO TO 70
   64 D=4.0
      GO TO 70
   65 D=5.0
      GO TO 70
   66 D=6.0
      GO TO 70
   67 D=7.0
      GO TO 70
   68 D=8.0
```

```
      GO TO 70
 69   D=9.0
 70   SUM=SUM+D*10**(I-1)
100   CONTINUE
105   IF(K(NUMEN) .EQ. JCOMC .OR. K(NUMEN) .EQ. JSEMC) GO TO 200
      IF(K(NUMEN+1) .EQ. JCOMC .OR. K(NUMEN+1) .EQ. JSEMC)GO TO 200
      DO 150 I=1,6
      J=NUMEN+I
      IF(K(J) .EQ. JCOMC .OR. K(J) .EQ. JSEMC) GO TO 190
      IF(K(J) .EQ. J0C) GO TO 130
      IF(K(J) .EQ. J1C) GO TO 131
      IF(K(J) .EQ. J2C) GO TO 132
      IF(K(J) .EQ. J3C) GO TO 133
      IF(K(J) .EQ. J4C) GO TO 134
      IF(K(J) .EQ. J5C) GO TO 135
      IF(K(J) .EQ. J6C) GO TO 136
      IF(K(J) .EQ. J7C) GO TO 137
      IF(K(J) .EQ. J8C) GO TO 138
      IF(K(J) .EQ. J9C) GO TO 139
      KK=NUMEN+6
      WRITE(6,57) (K(J),J=NUMST,KK)
      GO TO 510
130   D=0.0
      GO TO 140
131   D=1.0
      GO TO 140
132   D=2.0
      GO TO 140
 133   D=3.0
      GO TO 140
134   D=4.0
      GO TO 140
135   D=5.0
      GO TO 140
 136  D=6.0
      GO TO 140
137   D=7.0
      GO TO 140
138   D=8.0
      GO TO 140
139   D=9.0
140   SUM=SUM+D/(10.0**I)
150   CONTINUE
 190   NUMEN=J
200   IF(K(JPOS+2) .EQ. JMINC)SUM=-SUM
      DATA(JR)=SUM
      KNTR=KNTR+1
      IF( KFLG .EQ. 0 ) NX(LWC)=JR
      KFLG=0
      LWC=JR
       IF(K(NUMEN) .EQ. JASTC .OR. K(NUMEN+1) .EQ. JASTC) GO TO 600
      IF(K(NUMEN) .EQ. JSEMC .OR. K(NUMEN+1) .EQ. JSEMC) GO TO 205
      IF(K(NUMEN) .EQ. JPERC .AND. K(NUMEN+1) .EQ. JCOMC) NUMEN=NUMEN+1
      JPOS=NUMEN-1
      JR=LWC+1
 520  IF( JR .GT. MDSZ ) JR=1
```

```
      IF( NX(JR) .EQ. JPERC ) GO TO 45
      JR=JR+1
      KATP=KATP+1
      IF( KATP .LT. MDSZ ) GO TO 520
C     IF DATA ARRAY IS FILLED BEFORE ALL THE ELEMENTS ARE STORED, A MESSAGE
C     IS PRINTED, ARRAY IS ERASED AND SUBROUTINE BAD IS CALLED
      WRITE(6,37) NAME(KARY)
C     THE LENGTH OF THE ARRAY IS SET, JPOS IS UPDATED AND SUBROUTINE INTERP
C     IS CALLED TO WORK ON THE NEXT INSTRUCTION
      GO TO 510
 205  NX(LWC)=JASTC
      LNGT(KARY)= KNTR
C     A SEMICOLON INDICATES THE END OF AN ARRAY
      IF( K(NUMEN) .EQ. JSEMC ) GO TO 525
      JPOS=NUMEN+1
      CALL INTERP
  525 JPOS=NUMEN
  530 CALL INTERP
  600 JPOS=NUMEN
      CALL BAD
      END
```

SUBROUTINE READ

Data input to a STIL program is normally made in a format free form. However, in order to enable some users who may have their data already punched in a formatted form, to use the STIL language without repunching their data a limited kind of formatted input is provided. Subroutine READ handles simple formats, and reads and stores the data in the DATA array.

Since the STIL package treats all the variables as single precision floating point variables, double precision and integer formats cannot be used. Only one to four variables can be read by one read statement. If one variable is being read a format statement can specify any number of fields on a card. If two, three or four variables are being read, each card can contain only one set of values. If needed, certain columns on the card can be skipped by using the "X" specification.

Before processing, the interpreter checks the validity of the READ and FORMAT instructions. If their syntax are correct the number of variables and the number of values to be read are determined. If the variable names are already in memory it erases all the information about these arrays from the lists. Then it checks the DATA and the NAME arrays to see if there are enough empty cells for the values to be stored. If this check is positive, it allocates the required number of cells and completes the housekeeping jobs to set up the arrays. Different read statements are provided depending upon the number of variables being read. However the format statement furnished by the user is utilized as it is with all the read statements. An image of the card read is printed and the values are stored in the previously reserved cells. After reading the required number of values control is returned to the calling routine which is subroutine INPUT.

```
      SUBROUTINE RFAD
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION LR(160),TEMP(80),KMTR(70),KMTW(70)
C     INITIALIZE AND SUPPRESS BLANKS
      DO 10 I=1,160
   10 LR(I)=JBLKC
      J=1
      DO 15 I=1,80
      IF( L(I) .EQ. JBLKC ) GO TO 15
      LR(J)=L(I)
      J=J+1
   15 CONTINUE
C     IF THF INSTRUCTION IS NOT COMPLETE READ THE NEXT CARD
      N=0
      DO 20 I=1,J
      IF( LR(I) .EQ. JSEMC ) GO TO 19
      GO TO 20
   19 N=N+1
      IF ( N .EQ. 1 ) J1=I
   20 CONTINUE
      IF( N .GT. 2 .OR. N .LT. 1 ) GO TO 70
      IF( N .EQ. 1 ) GO TO 50
      JN=J-1
C     IDENTIFY THE READ STATEMENT
   25 IF( LR(J1+1) .EQ. JRC .AND. LR(J1+2) .EQ. JEC ) GO TO 30
      GO TO 70
   30 DO 35 I=J1,JN
      IF( LR(I) .EQ. JRPRC ) GO TO 40
   35 CONTINUE
      GO TO 70
C     COUNT THE NUMBER OF VARIABLES TO BE READ
   40 J2=I
      N=0
      DO 45 I=J2,JN
      IF( LR(I) .EQ. JCOMC ) N=N+1
   45 CONTINUE
      IF( N .GT. 3 ) GO TO 70
      NVRBL=N+1
      JJ=-1
C     IF THE VARIABLE NAMES ARE ALREADY IN MEMORY, ERASE THE RELATED INFORMATION
      DO 51 I=1,NVRBL
      JJ=JJ+2
      DO 46 II=1,MASZ
      IF( LR(J2+JJ) .EQ. NAME(II))GO TO 47
   46 CONTINUE
      GO TO 51
   47 IX=KST(II)
      KST(II)=0
      NAME(II)=0
```

```
      LLL=LNGT(II)-1
      LNGT(II)=0
      AVRG(II)=JPERC
      STND(II)=JPERC
      IF( LLL .LT. 1 ) GO TO 49
      DO 48 M=1,LLL
      IXN=NX(IX)
      NX(IX)=JPERC
      IX=IXN
   48 CONTINUE
   49 NX(IX)=JPERC
   51 CONTINUE
      GO TO 75
C     READ THE NEXT CARD, SUPPRESS BLANKS AND CHECK THE VALIDITY OF INSTRUCTION
   50 READ (5,52) L
      WRITE (6,54) L
   52 FORMAT(80A1)
   54 FORMAT(1X, 80A1)
      DO 65 I=1,80
      IF( L(I) .EQ. JBLKC ) GO TO 65
      LR(J)=L(I)
      J=J+1
   65 CONTINUE
      JN=J-1
      IF( LR(JN) .EQ. JSEMC ) GO TO 25
   70 WRITE (6,72)
   72 FORMAT( '  ILLEGAL INSTRUCTION  ' /)
      GO TO 130
C     CALCULATE THE NUMBER OF ELEMENTS TO BE READ
   75 NUM=0
      JB=J1+6
      JE=J2-1
      IF( JE .GE. JB ) GO TO 80
      GO TO 70
   80 N=JE-JB+1
      IF( N .GT. 4 ) GO TO 70
      J=JE
      DO 85 I=1,N
      M=LR(J)
      CALL DECODE(M,M1)
      IF( M1 .GT. 9 ) GO TO 70
      NUM=NUM+M1*10**(I-1)
      J=J-1
   85 CONTINUE
   90 N=NVRBL*NUM
C     CHECK TO SEE IF THERE ARE ENOUGH EMPTY CELLS IN NAME AND DATA ARRAYS
C     FOR THE VARIABLES TO BE READ IN
      NSUM=0
      DO 105 I=1,MASZ
  105 NSUM=NSUM+LNGT(I)
      IF( (NSUM+N) .GE. MDSZ ) GO TO 120
      N=0
      DO 110 I=1,MASZ
      IF( NAME(I) .EQ. 0 ) N=N+1
  110 CONTINUE
      IF( N .GE. NVRBL )  GO TO 140
```

```
  120 WRITE (6,122)
  122 FORMAT( '  TOO MANY ARRAYS OR DATA SIZE EXCEEDED  ' )
  130 STOP
  140 J=1
      DO 170 I=1,NVRBL
  155 IF( NAME(J) .NE. 0 ) GO TO 160
      IF( I .EQ. 1 ) KARYX=J
      IF( I .EQ. 2 ) KARYY=J
      IF( I .EQ. 3 ) KARYZ=J
      IF( I .EQ. 4 ) KARYW=J
      GO TO 165
  160 J=J+1
      IF( J .GT. MASZ ) GO TO 120
      GO TO 155
  165 J=J+1
      IF( J .GT. MASZ ) GO TO 120
  170 CONTINUE
C     LOCATE EMPTY CELLS IN DATA ARRAY
      DO 200 I=1,NVRBL
      IX=LWC+1
  175 IF( NX(IX) .EQ. JPERC ) GO TO 180
      IX=IX+1
      GO TO 175
  180 IF( I .EQ. 1 ) KST(KARYX)=IX
      IF( I .EQ. 2 ) KST(KARYY)=IX
      IF( I .EQ. 3 ) KST(KARYZ)=IX
      IF( I .EQ. 4 ) KST(KARYW)=IX
      LWC=IX
      IX=IX+1
      IF( NUM .LT. 2 ) GO TO 196
      DO 195 J=2,NUM
  185 IF( NX(IX) .EQ. JPERC ) GO TO 190
      IX=IX+1
      GO TO 185
  190 NX(LWC)=IX
      LWC=IX
      IX=IX+1
  195 CONTINUE
  196 NX(LWC)=JASTC
  200 CONTINUE
C     FORM THE WRITE FORMAT FROM THE READ FORMAT STATEMENT
      DO 205 I=1,70
      KMTR(I)=JBLKC
  205 KMTW(I)=JBLKC
      N=J1-1
      KMTR(1)=JLPRC
      KMTW(1)=JLPRC
      KMTW(2)=J1C
      KMTW(3)=JXC
      KMTW(4)=JCOMC
      J=2
      DO 210 I=11,N
      KMTR(J)=LR(I)
      KMTW(J+3)=LR(I)
  210 J=J+1
C     STORE THE VARIABLE NAMES AND LENGTHS IN NAME AND LENGTH ARRAYS
```

```
      IF( NVRBL .EQ. 1 ) GO TO 250
      IF( NVRBL .EQ. 2 ) GO TO 220
      IF( NVRBL .EQ. 3 ) GO TO 215
      NAME(KARYW)=LR(J2+7)
      LNGT(KARYW)=NUM
      IW=KST(KARYW)
  215 NAME(KARYZ)=LR(J2+5)
      LNGT(KARYZ)=NUM
      IZ=KST(KARYZ)
  220 NAME(KARYY)=LR(J2+3)
      LNGT(KARYY)=NUM
      IY=KST(KARYY)
      NAME(KARYX)=LR(J2+1)
      LNGT(KARYX)=NUM
      IX=KST(KARYX)
C     READ AND STORE THE VALUES FOR TWO, THREE OR FOUR  VARIABLE READ COMMANDS
      DO 245 I=1,NUM
      IF( NVRBL .EQ. 2 ) GO TO 240
      IF( NVRBL .EQ. 3 ) GO TO 235
      READ (5,KMTR) X,Y,Z,W
      WRITE (6,KMTW) X,Y,Z,W
      DATA(IW)=W
      IW=NX(IW)
  225 DATA(IZ)=Z
      IZ=NX(IZ)
  230 DATA(IY)=Y
      IY=NX(IY)
      DATA(IX)=X
      IX=NX(IX)
      GO TO 245
  235 READ (5,KMTR) X,Y,Z
      WRITE (6,KMTW) X,Y,Z
      GO TO 225
  240 READ (5,KMTR) X,Y
      WRITE (6,KMTW) X,Y
      GO TO 230
  245 CONTINUE
      GO TO 300
C     CALCULATE THE NUMBER OF ELEMENTS AND THE NUMBER OF CARDS TO BE READ IN
C     FOR ONE VARIABLE READ COMMANDS
  250 JB=11
      JE=J1-2
      N=0
      DO 270 I=JB,JE
      M=LR(I)
      IF( M .EQ. JFC .OR. M .EQ. JEC .OR. M .EQ. JIC ) GO TO 255
      GO TO 270
  255 J=1
  260 M=LR(I-J)
      CALL DECODE(M,M1)
      IF( M1 .GT. 9 .AND. J .EQ. 1 ) GO TO 265
      IF( M1 .GT. 9 ) GO TO 270
      N=N+M1*10**(J-1)
      J=J+1
      IF( J .GT. 2 ) GO TO 270
      GO TO 260
```

```
  265 N=N+1
  270 CONTINUE
      IF( N .GT. 80 ) GO TO 70
      M=NUM/N
      IF( M*N .LT. NUM )  GO TO 280
      NCRDS=M
      GO TO 285
  280 NCRDS=M+1
C     READ AND STORE THE VALUES
  285 NAME(KARYX)=LR(J2+1)
      LNGT(KARYX)=NUM
      IX=KST(KARYX)
      DO 295 I=1,NCRDS
      READ (5,KMTR) (TEMP(J),J=1,N)
      WRITE(6,KMTW) (TEMP(J),J=1,N)
      DO 290 J=1,N
      DATA(IX)=TEMP(J)
      IX=NX(IX)
      IF( IX .EQ. JASTC ) GO TO 295
  290 CONTINUE
  295 CONTINUE
C     RETURN TO SUBROUTINE INPUT
  300 RETURN
      END
```

SUBROUTINE INTERP

Subroutine INTERP identifies the instructions, calls the appropriate routines and controls the general flow of execution.

If all the instructions in one segment of the program have been executed, subroutine INPUT is called to read another portion of the program. If the whole program is processed execution is terminated, otherwise the next instruction in the program is decoded. If the instruction cannot be decoded, it is assumed to be a comment and subroutine BAD is called to locate the next instruction in the program. This feature also enables the program to skip over an illegal instruction and execute all the remaining instructions in the program which are not affected by this instruction.

The order of checking is important since all the characters in an instruction are not checked for efficiency purposes. Theoretically it is possible that a comment statement will match one of the tests and a command subroutine will be called. This usually results in the printing of some error message. This message may surprise the user, but it does not harm the program in any way and has not caused any problems with the package in actual use since the probability of this happening is very low. It may prove necessary to add some convention for comments at a later date, and this can easily be done.

```
      SUBROUTINE INTERP
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ JOC,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JFQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C  IF THE POINTER IS GONE BEYOND THE END OF THE PROGRAM STOP THE EXECUTION
      IF( JPOS .GT. KEND ) GO TO 20
      IF( K(2) .EQ. JASTC ) GO TO 20
C  IF THE PROGRAM IS SEGMENTED AND THE CURRENT SEGMENT IS ALREADY EXECUTED CALL
C  SUB. INPUT TO READ THE NEXT PORTION OF THE PROGRAM
      IF(JPOS.GE.LSEM.AND.(KFLAG.EQ.1.OR.KFLAG.EQ.22))CALL INPUT
C  CALL SUB. BAD TO LOCATE THE NEXT SEMICOLON IF THE POINTER IS NOT AT AN
C  INSTRUCTION
      IF( K(JPOS) .NE. JSEMC ) CALL BAD
C  SHIFT THE POINTER TO THE BEGINNING OF AN INSTRUCTION
      JPOS=JPOS+1
      M0=K(JPOS)
      M1=K(JPOS+1)
      M2=K(JPOS+2)
      M3=K(JPOS+3)
      M4=K(JPOS+4)
      M5=K(JPOS+5)
      M6=K(JPOS+6)
      M7=K(JPOS+7)
C  DECODE THE INSTRUCTION AND CALL THE APPROPRIATE SUBROUTINE
C  THE ORDER OF CHECKING IS IMPORTANT
      IF( M1 .EQ. JFQLC ) CALL INDATA
      IF( M0 .EQ. JWC .AND. M1 .EQ. JRC ) CALL OUTPUT
      IF(M0.EQ.JFC .AND. M1.EQ.JRC .AND. M2.EQ.JEC .AND. M4.EQ.JAC
     1.AND. M5.EQ.JLC .AND. M6.EQ.JLC ) CALL FREALL
      IF(M0.EQ.JFC.AND.M1.EQ.JRC.AND.M2.EQ.JEC.AND.M3.EQ.JFC)CALL FREEAR
      IF( M0 .EQ. JDC .AND. M1 .EQ. JUC ) CALL DUMP
      IF( M0 .EQ. JSC .AND. M4 .EQ. JSC ) CALL RANENT
      IF( M0 .EQ. JPC .AND. M1 .EQ. JLC ) CALL PLOT
      IF( M0 .EQ. JSC .AND. M4 .EQ. JEQLC ) GO TO 10
      IF( M4 .EQ. JMC .AND. M5 .EQ. JEC ) CALL AVG
      IF( M4 .EQ. JVC .AND. M5 .EQ. JAC ) CALL VAR
      IF( M4 .EQ. JSC .AND. M5 .EQ. JTC ) CALL VAR
      IF( M4 .EQ. JRC .AND. M5 .EQ. JAC ) CALL RANGE
      IF( M4 .EQ. JRC .AND. M5 .EQ. JEC ) CALL REGR
      IF( M4 .EQ. JCC .AND. M5 .EQ. JOC ) CALL COR
      IF( M4 .EQ. JAC .AND. M5 .EQ. JUC ) CALL AUTO
      IF( M4 .EQ. JDC .AND. M5 .EQ. JIC ) CALL FREQ
      IF( M4 .EQ. JHC .AND. M5 .EQ. JIC ) CALL HIST
      IF( M4 .EQ. JSC .AND. M5 .EQ. JKC ) CALL SKEW
      IF( M4 .EQ. JKC .AND. M5 .EQ. JUC ) CALL SKEW
   10 IF(M5.EQ. JRC .AND. M6.EQ.JEC ) CALL CALREG
      IF(M6.EQ.JPLSC .OR. M6.EQ.JMINC .OR. M6.EQ.JDIVC .OR. M6.EQ.JPERC)
     1CALL SETARY
      IF( M5 .EQ. JEC .AND. M6 .EQ. JXC ) CALL TRANSF
      IF( M5 .EQ. JLC .AND. M7 .EQ. JLPRC ) CALL TRANSF
      IF( M5 .EQ. JSC .AND. M6 .EQ. JQC ) CALL TRANSF
```

```
      IF( M5 .EQ. JIC .AND. M6 .EQ. JNC ) CALL TRANSF
      CALL DECODE( M5,MM)
      IF( MM .LE. 9 ) CALL DIST
      IF( M6 .EQ. JEC ) CALL TRANSF
C  IF THE INSTRUCTION CAN NOT BE IDENTIFIED CALL SUB. BAD TO SKIP THE
C  INSTRUCTION
      CALL BAD
   20 STOP 222
      END
```

SUBROUTINE OUTPUT

Subroutine OUTPUT displays the values of the elements of an array upon encountering a WRITE instruction. Five values are printed on each line according to a 5G12.4 format following a heading. The list does not exceed the width of an $8\frac{1}{2}$" x 11" standard paper. If the variable cannot be found in the name list a message "ARRAY NOT IN MEMORY" is printed along with the variable name and subroutine BAD is called to locate the next instruction in the program. If the array is successfully displayed, subroutine INTERP is called to decode the following STIL instruction.

```
      SUBROUTINE OUTPUT
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION TDATA(10)
C     VARIABLE IS LOCATED IN NAME ARRAY. IF IT IS NOT IN MEMORY A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THIS INSTRUCTION
      DO 20 I=1,MASZ
      IF(K(JPOS+5) .EQ. NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+5)
   22 FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
C     PRINT ARRAY NAME AND SET INDEX
C     AN ASTERISK IN NX ARRAY INDICATES THE END OF THE ARRAY
   25 WRITE(6,27) K(JPOS+5)
   27 FORMAT(' ARRAY ',A1)
      IX=KST(I)
C     TRANSFER DATA TO TEMPORARY ARRAY (TDATA) TO BE PRINTED
   30 DO 40 J=1,10
      TDATA(J)=DATA(IX)
      IF(NX(IX) .EQ. JASTC) GO TO 50
      IX=NX(IX)
   40 CONTINUE
      WRITE(6,42) (TDATA(J),J=1,10)
   42 FORMAT(5G12.4)
      GO TO 30
C     PRINT THE LAST LINE, RESET JPOS FOR NEXT INSTRUCTION AND CALL INTERP
   50 WRITE(6,42) (TDATA(I),I=1,J)
      WRITE(6,52)
   52 FORMAT(//)
      JPOS=JPOS+6
      CALL INTERP
      END
```

SUBROUTINE FREALL

Subroutine FREALL is executed upon a FREE ALL instruction and it destroys all the arrays currently present in the memory. If two types of problems are being solved in one loading and the data for each one is independent from the other, it provides a shortcut to clear the memory for the second problem. Actually a series of 'FREE X' instructions for all the variables in memory can produce the same result.

When all the variables are destroyed the pointer (JPOS) is updated and subroutine INTERP is called to decode the next instruction in the STIL program.

```
      SUBROUTINE FREALL
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
C  SUBROUTINE FREALL DESTROYS ALL THE ARRAYS THAT ARE CURRENTLY IN MEMORY
C  AND MAKES THE CELLS AVAILABLE FOR FURTHER USE
      DO 10 I=1,MASZ
      IF( NAME(I) .EQ. 0) GO TO 10
      CALL DESTRY(I)
   10 CONTINUE
      JPOS=JPOS+7
      CALL INTERP
      END
```

SUBROUTINE FREEAR

Subroutine FREEAR destroys the elements of an array and makes the cells available for future use. All the related information about the array is also erased.

If the variable cannot be located in memory a message 'ARRAY NOT IN MEMORY' is printed along with the variable name. If the routine is called by a FREE (variable name) instruction subroutine INTERP is called after the execution. Otherwise, control is returned to the calling routine.

```
      SUBROUTINE FREEAR
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     LOCATE THE VARIABLE IN NAME ARRAY
      DO 10 I=1,MASZ
      IF( K(JPOS+4) .EQ. NAME(I)) GO TO 15
   10 CONTINUE
      IF( KFLAG .EQ. 2 ) RETURN
      WRITE (6,12) K(JPOS+4)
   12 FORMAT(  '  ARRAY NOT IN MEMORY   ', A1)
      CALL BAD
C     ERASE ALL THE INFORMATION ABOUT THE ARRAY
   15 IX=KST(I)
      KST(I)=0
      NAME(I)=0
      LL=LNGT(I)-1
      LNGT(I)=0
      AVRG(I)=JPERC
      STND(I)=JPERC
      IF( LL .LT. 1 ) GO TO 21
      DO 20 I=1,LL
      IXT=NX(IX)
      NX(IX)=JPERC
      IX=IXT
   20 CONTINUE
   21 NX(IX)=JPERC
C  IF THE SUBROUTINE IS CALLED BY A ROUTINE OTHER THAN INTERP RETURN TO IT
C     OTHERWISE SET JPOS TO THE NEXT INSTRUCTION AND CALL SUBROUTINE INTERP
      IF( KFLAG .EQ. 2 ) RETURN
      JPOS=JPOS+5
      CALL INTERP
      END
```

SUBROUTINE DUMP

Subroutine DUMP is provided as a debugging aid to users who might want to add other routines to the existing STIL package, and it has been utilized during the development of the package. Instructions 'DUMP K' and 'DUMP DATA' calls this routine, but these instructions are not mentioned to normal STIL users since they will have no use for them.

'DUMP K' instruction displays the K array where the program is stored under alphanumeric format. 'DUMP DATA' displays the names, lengths, beginning locations, averages and the standard deviations of all arrays, along with the contents of DATA array where all the values are stored. If the average or the standard deviation of an array has not been calculated at the time of the display, a floating point representation of alphanumeric '.' (period) appears on the printout.

After the execution the pointer (JPOS) is updated and subroutine INTERP is called to decode the next instruction.

```
      SUBROUTINE DUMP
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION KNT(15),KCON(15),DT(5)
C     IDENTIFY THE TYPE OF REQUEST
      IF( K(JPOS+4) .EQ. JDC ) GO TO 40
C     PRINT HEADING
      WRITE(6,7)
    7 FORMAT( 40X, ' USERS PROGRAM ( K ARRAY ) ' /)
      J=1
C     FORM THE PRINTING LINES FOR PRINTING K ARRAY
      DO 20 I=1,MPSZ
      IF( K(I) .EQ. JBLKC ) GO TO 20
      KNT(J)=I
      KCON(J)=K(I)
      J=J+1
      IF( J .LE. 15 ) GO TO 20
      WRITE(6,12) (KNT(M),KCON(M),M=1,15 )
   12 FORMAT( 1X,14( I4,1X,A1,2X ),I4,1X,A1 )
      J=1
   20 CONTINUE
      M=J-1
      IF( M .LE. 0 ) GO TO 25
      WRITE(6,12) ( KNT(I),KCON(I),I=1,M )
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
   25 JPOS=JPOS+5
      CALL INTERP
C     PRINT HEADING
   40 WRITE(6,42)
   42 FORMAT(/2X,5HINDEX,1X,4HNAME, 2X,6HLENGTH,2X,8HLOCATION,5X,
     17HAVERAGE, 6X,18HSTANDARD DEVIATION ,/ )
C     PRINT THE VALUES OF NAME,LNGT,KST,AVRG AND STND ARRAYS
      DO 50 I=1,MASZ
      IF( NAME(I) .EQ. 0 ) GO TO 50
      WRITE(6,47) I,NAME(I),LNGT(I),KST(I),AVRG(I),STND(I)
   47 FORMAT( 1X,I4,5X,A1,4X,I4,5X,I4,4X,G12.4,5X, G12.4 )
   50 CONTINUE
C     PRINT THE HEADING
      WRITE(6,52)
   52 FORMAT(// 1X,4HINDX,2X,2HNX,6X,4HDATA,6X,4HINDX,2X,2HNX,6X,4HDATA,
     16X,4HINDX,2X,2HNX,6X,4HDATA,6X,4HINDX,2X,2HNX,6X,4HDATA,5X,
     24HINDX,2X,2HNX,7X,4HDATA / )
      J=1
C     FORM THE PRINTING LINES FOR PRINTING DATA ARRAY
      DO 70 I=1,MDSZ
      IF( NX(I) .EQ. JPERC ) GO TO 70
      KNT(J)=I
      KCON(J)=NX(I)
      DT(J)=DATA(I)
      J=J+1
```

```
      IF( J .LE. 5 ) GO TO 70
      WRITE(6,57) ( KNT(N),KCON(N),DT(N),N=1,5 )
   57 FORMAT( 1X,4( I4,1X,I4,1X,G12.4,2X ),I4,1X,I4,1X,G12.4 )
      J=1
   70 CONTINUE
      N=J-1
      IF( N .LE. 0 ) GO TO 75
      WRITE(6,57) ( KNT(I),KCON(I),DT(I),I=1,N )
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
   75 JPOS=JPOS+8
      CALL INTERP
      END
```

SUBROUTINE RANENT

Subroutine RANENT accomplishes two types of tasks. If a "SET NSEED" instruction is being executed, RANENT evaluates the constant written under alphanumeric code, stores its value into NSEED and calls subroutine INTERP to decode the next instruction in the program. Only integer constants are allowed and the number of decimal digits cannot exceed 10. If there is an illegal character within the field or if one of the above restrictions are violated an error message is printed and subroutine BAD is called to locate the next instruction in the program.

```
      SUBROUTINE RANENT
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C  IF A 'SET NSEED' INSTRUCTION IS BEING EXECUTED  GO TO STATEMENT 15
      IF( K(JPOS+4) .EQ. JSC .AND. K(JPOS+5) .EQ. JEC ) GO TO 15
C  OBTAIN AND RETURN A RANDOM NUMBER BETWEEN 1 AND MDSZ
      JR=RAN(NSEED)*MDSZ
      RETURN
C  CALCULATE THE VALUE TO BE STORED INTO NSEED
   15 KB=JPOS+8
      N=0
      DO 20 I=1,10
      IF( K(KB+I) .EQ. JSEMC ) GO TO 30
   20 N=N+1
C  IF THERE IS ANY ILLEGAL CHARACTER WITHIN THE FIELD CALL SUBROUTINE BAD
C  AND SKIP THE INSTRUCTION
   25 WRITE(6,27) K(JPOS)
   27 FORMAT(1X, ' ILLEGAL INSTRUCTION  ',A1 /)
      CALL BAD
   30 IF( N .EQ. 0 ) GO TO 25
C  STORE THE NUMBER INTO NSEED AND CALL SUB. INTERP TO DECODE THE NEXT
C  INSTRUCTION IN THE PROGRAM
      NSEED=INTEG(KB,N)
      JPOS=KB+N+1
      CALL INTERP
      END
```

SUBROUTINE PLOT

This subroutine is utilized to plot one or more variables against a base variable. The maximum number of variables which can be plotted is five, and the first variable is always taken as the base variable.

The execution of the routine starts by searching the variable names in the NAME array. If the variable names cannot be located in memory, or if the lengths of the variables do not agree with one another, or if more than five variables are being plotted an appropriate error message is printed and subroutine BAD is called to skip this instruction. Otherwise the base and the cross variables are scaled such that the graph can fit on $8\frac{1}{2}$" x 11" standard paper.

The names of the dependent variables are used as the symbols for plotting. If at least two curves go through the same plot point an asterisks is printed instead of either symbol. After a successful execution of the routine, subroutine BAD is again utilized to locate the beginning of the next instruction in the program.

```
      SUBROUTINE PLOT
C
      INTEGER OUT
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION A(1000),JPCAR(5),XPR(11),OUT(101)
 1    FORMAT(1H1 '            GRAPH OF ',A1,' VS ',4(A1,1X))
    2 FORMAT(1H ,G11.4,5X,101A1)
    3 FORMAT(1H )                                                          PLOT 041
7     FORMAT(1H ,15X,'.        .        .        .        .        .        .
     1       .        .        .        .')
  8   FORMAT(1H0,14X,11G8.2/)
 9    FORMAT(1H ,40X,A1/////)
      KEX=4
      NPVAR=1
      KJ=1
   5  DO 20 I=1,MASZ
      IF(K(JPOS+KEX) .EQ. NAME(I)) GO TO 25
  20  CONTINUE
      WRITE(6,22) K(JPOS+KEX)
  22  FORMAT( ' ARRAY NOT IN MEMORY ' ,A1)
      CALL BAD
  25  IF(NPVAR .NE. 1) JPCAR(NPVAR-1)=NAME(I)
      IF(NPVAR .EQ. 1) JBCAR=NAME(I)
      IX=KST(I)
      DO 40 J=KJ,1000
      A(J)=DATA(IX)
      IF(NX(IX) .EQ. JASTC) GO TO 50
      IX=NX(IX)
  40  CONTINUE
 50   IF(NPVAR .EQ. 1)N=LNGT(I)
      IF(N .EQ. LNGT(I)) GO TO 54
      WRITE(6,52) NAME(I)
  52  FORMAT( ' ARRAY NOT OF SAME LENGTH AS BASE VARIABLE ',A1)
      CALL BAD
  54  KEX=KEX+1
      IF(K(JPOS+KEX) .EQ. JSEMC .OR. K(JPOS+KEX ) .EQ. JASTC)GO TO 100
      KEX=KEX+1
      NPVAR=NPVAR+1
      KJ=KJ+N
      IF(NPVAR .LE. 5) GO TO 5
      WRITE(6,56)
  56  FORMAT( ' ATTEMPTING TO PLOT TOO MANY VARIABLES')
      CALL BAD
 100  IF(NPVAR .GT. 1) GO TO 110
      WRITE(6,105)
 105  FORMAT( ' PLOTTING ONLY ONE VARIABLE')
      CALL BAD
110   DO 115 I=1,N
      DO 114 J=I,N
```

```
      IF(A(I)-A(J))114,114,111
 111  IL=I-N
      LL=J-N
      DO 112IK=1,NPVAR
      IL=IL+N
      LL=LL+N
      F=A(IL)
      A(IL)=A(LL)
 112    A(LL)=F
 114   CONTINUE
 115   CONTINUE
      MY=NPVAR-1
 120  WRITE(6,1) JBCAR,(JPCAR(IZ),IZ=1,MY)
C
C FIND SCALE FOR BASE VARIABLE
C
      XSCAL=(A(N)-A(1))/80.0
C
C FIND SCALE FOR CROSS-VARIABLES
C
      M1=N+1
      YMIN=A(M1)
      YMAX=YMIN
      M2=NPVAR*N
      DO 140 J=M1,M2
      IF(A(J)-YMIN)128,126,126
 126   IF(A(J)-YMAX)140,140,130
 128  YMIN=A(J)
      GO TO 140
 130  YMAX=A(J)
 140  CONTINUE
      YSCAL=(YMAX-YMIN)/39.0
C
C FIND BASE VARIABLE PRINT POSITION
C
      YB=YMAX
      IL=1
 145  F=IL-1
      YPR=YB-F*YSCAL
 146  DO 147 IZ=1,81
 147   OUT(IZ)=JBLKC
      DO 155 IZ=1,MY
      LL=IZ*N+1
      LN=LL+N-1
      DO 152 IN=LL,LN
      IF(A(IN)-YPR)152,148,148
 148  KK=IN-LL+1
      JP=(A(KK)-A(1))/XSCAL+1.0
      IF(OUT(JP)-JBLKC)149,150,149
 149  OUT(JP)=JASTC
      GO TO 151
 150  OUT(JP)=JPCAR(IZ)
 151  A(IN)=YMIN-10.0
 152  CONTINUE
 155  CONTINUE
      WRITE(6,2) YPR,(OUT(IZ),IZ=1,81)
```

```
      IL=IL+1
      IF(IL-40)145,184,186
184    YPR=YMIN
      GO TO 146
186     XPR(1)=A(1)
      DO 190 KN=1,9
190    XPR(KN+1)=XPR(KN)+XSCAL*8.0
      XPR(11)=A(N)
      WRITE(6,7)
      WRITE(6,8) (XPR(IP),IP=1,11)
      WRITE(6,9)JBCAR
      JPOS=JPOS+1
      CALL BAD
      END
```

SUBROUTINE AVG

Subroutine AVG calculates the mean of an array and stores it into the AVRG array for future uses. If the user's program has asked for the value of the mean, it is printed along with the variable name. Subroutine INTERP is then called to decode the next instruction in the program. If the subroutine AVG is called by another routine (like VAR) control is returned to the calling routine after the calculations are completed.

If the variable cannot be found in the name list a message 'ARRAY NOT IN MEMORY' is printed along with the variable name and subroutine BAD is called to locate the next instruction in the program.

```
      SUBROUTINE AVG
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAMF(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     VARIABLE IS LOCATED IN NAME ARRAY. IF IT IS NOT IN MEMORY A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THIS INSTRUCTION
      DO 20 I=1,MASZ
      IF(K(JPOS+8) .EQ. NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+8)
   22 FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
   25 SUM=0.0
C     SET INDEX AND LENGTH
      JARY=I
      IX=KST(JARY)
      LL=LNGT(JARY)
C     CALCULATE THE MEAN
      DO 30 J=1,LL
      SUM=SUM+DATA(IX)
      IX=NX(IX)
   30 CONTINUE
      AV=SUM/LL
      AVRG(JARY)=AV
C     RETURN TO THE CALLING ROUTINE IF IT IS NOT SUBROUTINE INTERP
      IF( KWAN .EQ. 1 ) RETURN
C     PRINT THE RESULTS
      WRITE(6,55) K(JPOS+8),AV
   55 FORMAT( ' THE MEAN OF ',A1,' IS ',F14.5//)
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+9
      CALL INTERP
      END
```

SUBROUTINE VAR

Subroutine VAR calculates the standard deviation and the variance of an array and stores the value for the standard deviation into the STND array for future uses. Variance is calculated by

$$\frac{1}{n} \left(\sum_{i=1}^{n} X_i^2 \right) - \overline{X}^2$$

and no attempt is made to estimate the universe standard deviation. If the user's program has asked for either the standard deviation or the variance the required value is printed and then subroutine INTERP is called to decode the next instruction in the program. Otherwise the control is returned to the calling routine.

If the variable cannot be located in the name list a message 'ARRAY NOT IN MEMORY' is printed along with the variable name and subroutine BAD is called to locate the next instruction in the program.

```
      SUBROUTINE VAR
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      KW=KWAN
C     VARIABLE IS LOCATED IN NAME ARRAY. IF IT IS NOT IN MEMORY A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THIS INSTRUCTION
      DO 20 I=1,MASZ
      IF(K(JPOS+7) .EQ. NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+7)
   22 FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
   25 SUMSQ=0.0
C     SET INDEX AND LENGTH
      JARY=I
      IX=KST(I)
      LL=LNGT(I)
C     RETRIEVE MEAN
      IF( AVRG(I) .NE. JPERC ) GO TO 30
      KWAN=1
      JPOS=JPOS-1
      CALL AVG
      KWAN=KW
      JPOS=JPOS+1
C     CALCULATE THE STANDARD DEVIATION
   30 DO 40 I=1,LL
      D=DATA(IX)
      SUMSQ=SUMSQ+ D*D
      IX=NX(IX)
   40 CONTINUE
      A=AVRG(JARY)
      AVAR=SQRT( SUMSQ/LL -A*A)
      STND(JARY)= AVAR
C     RETURN TO THE CALLING ROUTINE IF IT IS NOT SUBROUTINE INTERP
      IF(KWAN .EQ. 1) RETURN
C     PRINT THE RESULTS
      IF(K(JPOS+4) .EQ. JVC) GO TO 60
      WRITE(6,52) K(JPOS+7),AVAR
   52 FORMAT( ' THE STANDARD DEVIATION OF ',A1,' IS ',F14.5//)
      GO TO 70
  60  AVAR=AVAR*AVAR
      WRITE(6,63) K(JPOS+7),AVAR
   63 FORMAT( ' THE VARIANCE OF ',A1,' IS ',F14.5//)
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
   70 JPOS=JPOS+8
      CALL INTERP
      END
```

SUBROUTINE RANGE

Subroutine RANGE locates the largest and the smallest elements of an array, and calculates the difference between the two. The range, and the extreme points are printed along with the variable name. Subroutine INTERP is then called to work on the next instruction in the program.

If the variable cannot be found in the name list a message 'ARRAY NOT IN MEMORY' is printed along with the variable name and subroutine BAD is called to locate the next instruction in the program.

```
      SUBROUTINE RANGE
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ JOC,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSFED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     LOCATE THE VARIABLE IN NAME ARRAY
      DO 20 I=1,MASZ
      IF(K(JPOS+9) .EQ. NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+9)
   22 FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
C     SET INDEX AND LENGTH
   25 IX=KST(I)
      LL=LNGT(I)
      KARY=I
C     DETERMINE THE LARGEST AND SMALLEST VALUES
      VHIGH=DATA(IX)
      VLOW=DATA(IX)
      DO 40 I=1,LL
      IF(DATA(IX) .GT. VHIGH) VHIGH=DATA(IX)
      IF(DATA(IX) .LT. VLOW) VLOW=DATA(IX)
      IX=NX(IX)
   40 CONTINUE
      R=VHIGH-VLOW
C     PRINT THE RESULTS
      WRITE(6,55) K(JPOS+9),R,VHIGH,VLOW
   55 FORMAT( ' THE RANGE OF ',A1,' IS ',F14.5/' LARGEST VALUE ',F14.5/
     1'        LOWEST VALUE ',F14.5//)
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+10
       CALL INTERP
      END
```

SUBROUTINE REGR

Subroutine REGR can solve up to four variable linear regression problems. The solution is obtained in a stepwise fashion in the following manner.

First, the following quantities are calculated for the first two variables.

$$SUMX1 = \sum^{n} X1 \quad ; \quad SUMX2 = \sum^{n} X2$$

$$SUM1S = \sum^{n} X1^2 \quad ; \quad SMX2S = \sum^{n} X2^2$$

$$SX1X2 = \sum^{n} (X1)(X2);$$

If there are no other variables listed in the instruction the coefficients A and B (for the equation Y = A + BX), the coefficient of multiple correlation (R) and the standard error of estimate (STER) are calculated by the following formulas.

$$B = \frac{SX1X2 - (1/n)\ (SUMX1)\ (SUMX2)}{SMX2S - n\overline{X2}^2}$$

$$A = \overline{X1} - B\ (\overline{X2})$$

where n = number of elements

$$\overline{X}_1 = SUMX1/n \quad \text{and} \quad \overline{X}_2 = SUMX_2/n$$

$$R = \frac{SX1X2 - (SUMX1)\ (SUMX2)\ (1/n)}{\sqrt{(SMX1S - n\ \overline{X1}^2)\ (SMX2S - n\ \overline{X2}^2)}}$$

$$STER^2 = \frac{(SMX1S - n\ \overline{X1}^2)\ (1. - R^2)}{n - 2}$$

After printing the results the value of KFLAG is checked. If it is equal to 10 control is returned to subroutine CALREG, otherwise, subroutine INTERP is called to decode the next instruction in the program.

If there is a third variable the following values are calculated.

$$SUMX3 = \overset{n}{\Sigma} X3 \quad ; \quad SMX3S = \overset{n}{\Sigma} X3^2$$

$$SX1X3 = \overset{n}{\Sigma}(X1)\ (X3) \quad ; \quad SX2X3 = \overset{n}{\Sigma}(X2)\ (X3)$$

If there isn't another variable in the instruction the coefficients A, B and C (for the equation $Y = A + BX + CZ$), the coefficient of multiple correlation (R) and the standard error of estimate (STER) are calculated.

$$B = \frac{(SMX3S-n\overline{X3}^2)(SX1X2-n(\overline{X1})(\overline{X2}))-(SX2X3-n(\overline{X2})(\overline{X3}))(SX1X3-n(\overline{X1})(\overline{X3}))}{(SMX3S-n\overline{X3}^2)(SMX2S-n\overline{X2}^2)-(SX2X3-n(\overline{X2})(\overline{X3}))^2}$$

$$C = \frac{(SX1X2-n(\overline{X1})(\overline{X2}))-B(SMX2S-n\overline{X2}^2)}{SX2X3-n(X2)(X3)}$$

$$A = \overline{X1} - B(\overline{X2}) - C(\overline{X3})$$

$$R^2 = \frac{B(SX1X2-n(\overline{X1})(\overline{X2}))+C(SX1X3-n(\overline{X1})(\overline{X3})}{SMX1S-n\overline{X1}^2}$$

$$STER^2 = \frac{(SMX1S-n\overline{X1}^2)-B(SX1X2-n(\overline{X1})(\overline{X2}))-C(SX1X3-n(\overline{X1})(\overline{X3}))}{(n-3)}$$

After printing these values the same procedure for returning control is followed as in two variable regression problems.

If the problem is a four variable regression problem the coefficients A, B, C and D (for $Y = A + BX + CZ + DT$), the coefficient of

multiple correlation (R) and the standard error of estimate (STER) are calculated by similar formulas and printed.

The first variable is always taken to be the dependent variable and X1, X2, X3 and X4 are equivalent to Y, X, Z and T in equation $Y = A + BX + CZ + DT$.

If there are more than four variables, or if the lengths of the variables do not agree with one another, or if one of the variables cannot be found in the NAME array, an appropriate error message is printed and subroutine BAD is called to locate the next instruction in the program. Otherwise subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE REGR
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      COMMON/BLK9/ KORDER,CNST1,CNST2,CNST3,CNST4
C     LOCATE THE FIRST VARIABLE IN NAME ARRAY
      DO 20 I=1,MASZ
      IF(K(JPOS+7) .EQ. NAME(I)) GO TO 25
 20   CONTINUE
      WRITE(6,22) K(JPOS+7)
  22  FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
C     LOCATE THE SECOND VARIABLE IN NAME ARRAY
   25 DO 30 J=1,MASZ
      IF(K(JPOS+9) .EQ. NAME(J)) GO TO 35
  30  CONTINUE
      WRITE(6,22) K(JPOS+9)
      CALL BAD
C     SET INDEX AND LENGTH
  35  IX=KST(I)
      LL=LNGT(I)
      IY=KST(J)
C     COMPARE THE LENGTHS
      IF( LL .EQ. LNGT(J) ) GO TO 40
       WRITE (6,42) K(JPOS+7), K(JPOS+9),LL,LNGT(J)
   42 FORMAT(  '  ARRAYS NOT OF EQUAL LENGTH  ',A1, 5X, A1,2I8 )
      CALL BAD
C     CALCULATE INTERMEDIATE VALUES FOR TWO VARIABLES
   40 KOUNT=LL
      SUMX1=0.0
      SUMX2=0.0
      SMX1S=0.0
      SMX2S=0.0
      SX1X2=0.0
      DO 50 IZ=1,LL
      DX=DATA(IX)
      DY=DATA(IY)
      SUMX1=SUMX1+DX
      SUMX2=SUMX2+DY
      SMX1S=SMX1S+ DX*DX
      SMX2S=SMX2S+ DY*DY
      SX1X2=SX1X2+DX*DY
      IX=NX(IX)
      IY=NX(IY)
  50  CONTINUE
C     IF THERE IS A THIRD VARIABLE GO TO STATEMENT 80
      IF( K(JPOS+10) .EQ. JCOMC ) GO TO 80
C     CALCULATE REGRESSION COEFFICIENTS AND THE STANDARD ERROR OF ESTIMATE
C     FOR TWO VAARIABLES
      C1=(SX1X2-SUMX1*SUMX2/KOUNT)/(SMX2S-KOUNT*(SUMX2/KOUNT)**2)
      A=SUMX1/KOUNT-C1*SUMX2/KOUNT
```

```
      R=(SX1X2-SUMX1*SUMX2/KOUNT)/(SQRT((SMX1S-KOUNT*(SUMX1/KOUNT)**2)
     1*(SMX2S-KOUNT*(SUMX2/KOUNT)**2)))
      STER=SQRT((SMX1S-KOUNT*(SUMX1/KOUNT)**2)*(1.0-R*R)/(KOUNT-2.0))
C     PRINT THE RESULTS
      WRITE(6,76) K(JPOS+7),K(JPOS+9)
   76 FORMAT( ' FOR THE EQUATION ',A1,'=A+B*',A1/)
      WRITE(6,77) A,C1,R
   77  FORMAT(5X,' A=',E16.6/5X,' B=',E16.6/5X,' CORRELATION=',F8.3)
      WRITE(6,78) STER
   78 FORMAT(5X,' STANDARD ERROR OF ESTIMATE=',E16.6//)
C     RETURN TO THE CALLING ROUTINE IF IT IS NOT SUBROUTINE INTERP
      IF( KFLAG .EQ. 10 ) GO TO 500
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+10
      CALL INTERP
  500 CNST1=A
      CNST2=C1
       RETURN
C     LOCATE THE THIRD VARIABLE IN NAME ARRAY
   80 DO 90 I3=1,MASZ
      IF(K(JPOS+11) .EQ. NAME(I3)) GO TO 95
   90 CONTINUE
      WRITE(6,22) K(JPOS+11)
      CALL BAD
   95 SUMX3=0.0
      SX1X3=0.0
      SX2X3=0.0
      SMX3S=0.0
C     COMPARE THE LENGTHS
      IF( LL .EQ. LNGT(I3) ) GO TO 98
      WRITE (6,42) K(JPOS+7),K(JPOS+11),LL,LNGT(I3)
      CALL BAD
C     SET INDEXES
   98 IX=KST(I)
      IY=KST(J)
      IK=KST(I3)
C     CALCULATE INTERMEDIATE VALUES
      DO 100 IZ=1,LL
      DK=DATA(IK)
      DX=DATA(IX)
      DY=DATA(IY)
      SUMX3= SUMX3+DK
      SX1X3=SX1X3+DX*DK
      SX2X3=SX2X3+DY*DK
      SMX3S=SMX3S+ DK*DK
      IX=NX(IX)
      IY=NX(IY)
      IK=NX(IK)
  100 CONTINUE
      X1B= SUMX1/KOUNT
      X2B=SUMX2/KOUNT
      X3B=SUMX3/KOUNT
      S2SA=SMX2S-KOUNT*X2B*X2B
      S23A=SX2X3-KOUNT*X2B*X3B
      S1SA=SMX1S-KOUNT*X1B*X1B
      S12A=SX1X2-KOUNT*X1B*X2B
```

```
      S3SA=SMX3S-KOUNT*X3B*X3B
      S13A=SX1X3-KOUNT*X1B*X3B
C     IF THERE IS A FOURTH VARIABLE GO TO STATEMENT 180
      IF(K(JPOS+12) .EQ. JCOMC) GO TO 180
C     CALCULATE REGRESSION COEFFICIENTS AND THE STANDARD ERROR OF ESTIMATE
C     FOR THREE VARIABLES
      B=(S3SA*S12A-S23A*S13A)/(S3SA*S2SA-S23A*S23A)
      C=(S12A-B*S2SA)/S23A
      A=X1B-B*X2B-C*X3B
      STER=(S1SA-B*S12A-C*S13A)/(KOUNT-3.0)
      STER=SQRT(STER)
      R=(B*S12A+C*S13A)/S1SA
      R=ABS(R)
      R=SQRT(R)
C     PRINT THE RESULTS
      WRITE(6,126) K(JPOS+7),K(JPOS+9),K(JPOS+11)
 126  FORMAT( ' FOR THE EQUATION ',A1,'=A+B*',A1,'+C*',A1/)
      WRITE(6,128) A,B,C,R
 128  FORMAT(5X,'A=',E16.6/5X,'B=',E16.6/5X,'C=',E16.6/5X,'CORRELATION='
     1,F8.3)
      WRITE(6,129) STER
 129  FORMAT(5X,'STANDARD ERROR OF ESTIMATE=',E16.6//)
C     RETURN TO THE CALLING ROUTINE IF IT IS NOT SUBROUTINE INTERP
      IF( KFLAG .EQ. 10 ) GO TO 510
      JPOS=JPOS+12
      CALL INTERP
  510 CNST1=A
      CNST2=B
      CNST3=C
      RETURN
C     LOCATE THE FOURTH VARIABLE IN NAME ARRAY
  180 DO 190 I4=1,MASZ
      IF(K(JPOS+13) .EQ. NAME(I4)) GO TO 195
 190   CONTINUE
      WRITE(6,22) K(JPOS+13)
      CALL BAD
 195  SUMX4=0.0
      SX1X4=0.0
       SX2X4=0.0
      SX3X4=0.0
      SMX4S=0.0
C     COMPARE THE LENGTHS
      IF( LL .EQ. LNGT(I4) ) GO TO 198
      WRITE (6,42) K(JPOS+7),K(JPOS+13),LL,LNGT(I4)
      CALL BAD
C     SET INDEX AND LENGTH
  198 KOUNT=LL
      IX=KST(I)
      IY=KST(J)
      IK=KST(I3)
      IL=KST(I4)
C     CALCULATE INTERMEDIATE VALUES
      DO 200 IZ=1,LL
      DL=DATA(IL)
      DX=DATA(IX)
      DY=DATA(IY)
```

```
      DK=DATA(IK)
      SUMX4=SUMX4+DL
      SX1X4=SX1X4+DX*DL
      SX2X4=SX2X4+DY*DL
      SX3X4=SX3X4+DK*DL
      SMX4S=SMX4S+DL*DL
      IX=NX(IX)
      IY=NX(IY)
      IK=NX(IK)
      IL=NX(IL)
 200  CONTINUE
      X4B= SUMX4/KOUNT
      S14A=SX1X4-KOUNT*X1B*X4B
      S24A=SX2X4-KOUNT*X2B*X4B
      S34A=SX3X4-KOUNT*X3B*X4B
      S4SA=SMX4S-KOUNT*X4B*X4B
C IF THERE IS ANOTHER VARIABLE GO TO STATEMENT 280
      IF(K(JPOS+14) .EQ. JCOMC) GO TO 280
C     CALCULATE REGRESSION COEFFICIENTS AND THE STANDARD ERROR OF ESTIMATE
C     FOR FOUR VARIABLES
      F1=S2SA
      F2=S23A
      F3=S24A
      F4=S12A
      F5=S3SA
      F6=S34A
      F7=S13A
      F8=S4SA
      F9=S14A
      D=F1*(F5*F8-F6*F6)-F2*(F2*F8-F6*F3)+F3*(F2*F6-F5*F3)
      DB=F4*(F5*F8-F6*F6)-F7*(F2*F8-F6*F3)+F9*(F2*F6-F5*F3)
      DC=F1*(F7*F8-F9*F6)-F2*(F4*F8-F9*F3)+F3*(F4*F6-F7*F3)
      DD=F1*(F5*F9-F6*F7)-F2*(F2*F9-F6*F4)+F3*(F2*F7-F5*F4)
      B=DB/D
      C=DC/D
      D=DD/D
      A=X1B-B*X2B-C*X3B-D*X4B
      R=(B*S12A+C*S13A+D*S14A)/S1SA
      R=ABS(R)
      R=SQRT(R)
      STER=(S1SA-B*S12A-C*S13A-D*S14A)/(KOUNT-4.0)
      STER=SQRT(STER)
C     PRINT THE RESULTS
      WRITE(6,226) K(JPOS+7),K(JPOS+9),K(JPOS+11),K(JPOS+13)
 226  FORMAT( ' FOR THE EQUATION ',A1,'=A+B*',A1,'+C*',A1,'+D*',A1/)
      WRITE(6,228)A,B,C,D,R,STER
 228  FORMAT(5X,'A=',E16.6/5X,'B=',E16.6/5X,'C=',E16.6/5X,'D=',E16.6/
     15X,'CORRELATION=',F8.3/5X,'STANDARD ERROR OF ESTIMATE=',E16.6//)
C     RETURN TO THE CALLING ROUTINE IF IT IS NOT SUBROUTINE INTERP
      IF( KFLAG .EQ. 10 ) GO TO 520
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+14
      CALL INTERP
  520 CNST1=A
      CNST2=B
      CNST3=C
```

```
      CNST4=D
      RETURN
280   JPOS=JPOS+1
      CALL  BAD
      END
```

SUBROUTINE COR

Subroutine COR calculates and prints the correlation coefficient between two arrays. If any one of the array names are not present in memory, or if their lengths are not equal to each other, an error message is printed and the execution of the routine is skipped by calling subroutine BAD.

The correlation coefficient is calculated by $\frac{\sum_{i=1}^{n} X_i Y_i - n \bar{X} \bar{Y}}{n \sigma_x \sigma_y}$

where $\bar{X}$, $\bar{Y}$ are the means and σ_x and σ_y are the standard deviations of the arrays. The means and the standard deviations are retrieved from the AVRG and STND arrays and after printing the results the pointer (JPOS) is updated and subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE COR
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     LOCATE THE FIRST VARIABLE IN NAME ARRAY
      DO 10 I=1,MASZ
      IF( K(JPOS+7) .EQ. NAME(I)) GO TO 20
   10 CONTINUE
      WRITE(6,12) K(JPOS+7)
   12 FORMAT(  '  ARRAY NOT IN MEMORY   ',A1 )
      CALL BAD
   20 MX=I
      LL=LNGT(MX)
C     LOCATE THE SECOND VARIABLE IN NAME ARRAY
      DO 25 I=1,MASZ
      IF( K(JPOS+9) .EQ. NAME(I)) GO TO 30
   25 CONTINUE
      WRITE(6,12) K(JPOS+9)
      CALL BAD
   30 MY=I
C     COMPARE THE LENGTHS
      IF( LL .EQ. LNGT(MY) ) GO TO 50
      WRITE(6,32) K(JPOS+7), LL, K(JPOS+9) ,LNGT(MY)
   32 FORMAT(  '  ARRAYS NOT OF EQUAL LENGTH   ',A1,I6,5X,A1,I6 )
      CALL BAD
C     RETRIEVE THE STANDARD DEVIATION OF THE FIRST VARIABLE
   50 IF( STND(MX) .NE. JPERC ) GO TO 55
      KWAN=1
      CALL VAR
      KWAN=0
C     RETRIEVE THE STANDARD DEVIATION OF THE SECOND VARIABLE
   55 IF( STND(MY) .NE. JPERC ) GO TO 60
      KWAN=1
      JPOS=JPOS+2
      CALL VAR
      KWAN=0
      JPOS=JPOS-2
C     SET INDEXES
   60 IX=KST(MX)
      IY=KST(MY)
      SUM=0.0
C     CALCULATE INTERMEDIATE VALUES
      DO 65 I=1,LL
      SUM=SUM+DATA(IX)*DATA(IY)
      IX=NX(IX)
      IY=NX(IY)
   65 CONTINUE
C     CALCULATE CORROLATION COEFFICIENT
      C=( SUM-LL*AVRG(MX)*AVRG(MY))/( LL*STND(MX)*STND(MY))
C     PRINT THE RESULTS
      WRITE(6,77) K(JPOS+7) ,K(JPOS+9),C
```

```
   77 FORMAT(  '  CORRELATION BETWEEN   ',A1, ' AND ',A1,' IS ',F8.5//)
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+10
      CALL INTERP
      END
```

SUBROUTINE AUTO

Subroutine AUTO calculates, the autocorrelations for an array for a specified number of lag periods. The number of periods the lag can vary is from 1 to 99, however if the number of elements in the array is less than the number of periods in the lag, a message regarding the insufficiency of the length of the array is printed and the execution of the routine is ignored by calling subroutine BAD.

If there is enough data for the specified lag time, autocorrelation is calculated according to the following formula

$$\frac{SXY - k\bar{X}\bar{Y}}{\sqrt{(SXSQ - k\bar{X}^2)\ (SYSQ - k\bar{Y}^2)}}$$

where k = n - lag time

n = number of elements in the array

$$\bar{X} = \frac{1}{k} \sum_{i=n-k}^{n} X_i$$

$$\bar{Y} = \frac{1}{k} \sum_{i=n-k}^{n} Y_i$$

$$SXY = \sum_{i=n-k}^{n} X_i Y_i$$

$$SXSQ = \sum_{i=n-k}^{n} X_i^2$$

$$SYSQ = \sum_{i=n-k}^{n} Y_i^2$$

When all the autocorrelation factors are calculated for the specified lag times subroutine INTERP is called to decode the next instruction in the program. There is no upper limit for the number of lag times that may be specified.

If an asterisk is found at the end of the current instruction subroutine BAD is called. This enables the package to read another segment of the program.

```
      SUBROUTINE AUTO
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     LOCATE THE VARIABLE IN NAME ARRAY
      DO 20 I=1,MASZ
      IF(K(JPOS+11) .EQ. NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+11)
   22 FORMAT(  '  ARRAY NOT IN MEMORY  ',A1 )
      CALL BAD
   25 KNAME=NAME(I)
      KARY=I
      LLL=LNGT(I)
C     DETERMINE THE NUMBER OF PERIODS IN LAG TIME
      IF(K(JPOS+13) .EQ. J0C) N1=0
      IF(K(JPOS+13) .EQ. J1C) N1=1
      IF(K(JPOS+13) .EQ. J2C) N1=2
      IF(K(JPOS+13) .EQ. J3C) N1=3
      IF(K(JPOS+13) .EQ. J4C) N1=4
      IF(K(JPOS+13) .EQ. J5C) N1=5
      IF(K(JPOS+13) .EQ. J6C) N1=6
      IF(K(JPOS+13) .EQ. J7C) N1=7
      IF(K(JPOS+13) .EQ. J8C) N1=8
      IF(K(JPOS+13) .EQ. J9C) N1=9
        IF(K(JPOS+14).EQ. JSEMC .OR. K(JPOS+14) .EQ. JASTC .OR. K(JPOS+
     114) .EQ. JCOMC) GO TO 30
      IF(K(JPOS+14) .EQ. J0C) N2=0
      IF(K(JPOS+14) .EQ. J1C) N2=1
      IF(K(JPOS+14) .EQ. J2C) N2=2
      IF(K(JPOS+14) .EQ. J3C) N2=3
      IF(K(JPOS+14) .EQ. J4C) N2=4
      IF(K(JPOS+14) .EQ. J5C) N2=5
      IF(K(JPOS+14) .EQ. J6C) N2=6
      IF(K(JPOS+14) .EQ. J7C) N2=7
      IF(K(JPOS+14) .EQ. J8C) N2=8
      IF(K(JPOS+14) .EQ. J9C) N2=9
      N=N2+10*N1
      KXX=2
      GO TO 35
   30 N=N1
      KXX=1
C     INITIALIZE AND SET INDEXES
   35 SX=0.0
      SY=0.0
      SXSQ=0.0
      SYSQ=0.0
      SXY=0.0
      KOUNT=0
      IX=KST(I)
      IY=KST(I)
```

```
C       CHECK TO SEE IF THERE IS SUFFICIENT AMOUNT OF DATA
        DO 40 J=1,N
        IF( NX(IY) .EQ. JASTC ) GO TO 80
        IY=NX(IY)
 40     CONTINUE
C       CALCULATE INTERMEDIATE VALUES
        DO 45 J=1,LLL
        SX=SX+DATA(IX)
        SXSQ=SXSQ+DATA(IX)*DATA(IX)
        SY=SY+DATA(IY)
        SYSQ=SYSQ+DATA(IY)*DATA(IY)
        SXY=SXY+DATA(IX)*DATA(IY)
        KOUNT=KOUNT+1
        IF( NX(IY) .EQ. JASTC ) GO TO 50
        IX=NX(IX)
        IY=NX(IY)
  45    CONTINUE
   50   AVX=SX/KOUNT
        AVY=SY/KOUNT
C       CALCULATE THE AUTOCORROLATION
        R=(SXY-KOUNT*AVX*AVY)/SQRT((SXSQ-KOUNT*AVX*AVX)*(SYSQ-KOUNT*
       1AVY*AVY))
C       PRINT THE RESULTS
        WRITE(6,55) KNAME,R,N
  55    FORMAT( ' FOR ARRAY 'A1,3X,'  AUTOCORRELATION=',F7.3,3X,'  FOR N='
       1I5//)
        LL=JPOS+13+KXX
        IF(K(LL) .EQ. JASTC) GO TO 95
CHECK   TO SEE IF THERE IS ANOTHER REQUEST
        IF(K(LL) .EQ. JCOMC) GO TO 70
C       ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
        JPOS=LL
        CALL INTERP
  70    JPOS=JPOS+KXX+1
        GO TO 25
   80   WRITE (6,82) KNAME,N
   82   FORMAT(   '  NOT ENOUGH DATA  ',A1,I5 )
   90   CALL BAD
   95   JPOS=LL
        GO TO 90
        END
```

SUBROUTINE FREQ

Subroutine FREQ calculates the frequency, cumulative frequency and the cumulative relative frequency for an array. The number of intervals is determined by the following table.

Number of elements	Number of cells
2-4	2
5-9	3
greater than 10	8

Even though the determination of the number of cells seems to be quite arbitrary, it has not caused any problems in applications so far. And if needed, a different method of determination can be incorporated into the routine very easily.

Once the number of intervals is defined the range and the interval widths are calculated. From this information upper and lower limits of all the cells are determined. Then, the number of elements falling in each interval is counted to get the frequency distribution. From the frequency distribution, the cumulative frequency, and the cumulative relative frequency distributions are calculated. Upper and lower limits of the intervals and the calculated distributions are printed in a tabular form beginning with the first cell. The mean and the standard deviation of the array are printed at the end of this table and subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE FREQ
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION ULIM(8),BLIM(8),JFREQ(8),CUFR(8),CURFR(8)
C     LOCATE THE VARIABLE IN NAME ARRAY
      DO 20 I=1,MASZ
      IF(K(JPOS+8) .EQ. NAME(I)) GO TO 25
 20   CONTINUE
      WRITE(6,22) K(JPOS+8)
  22  FORMAT( ' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
C     SET INDEX AND LENGTH
  25  IX=KST(I)
      LL=LNGT(I)
      KARY=I
      KOUNT=LL
C     DETERMINE THE LARGEST AND SMALLEST VALUES
      VHIGH=DATA(IX)
      VLOW=DATA(IX)
      DO 40 I=1,LL
      D= DATA(IX)
      IF( D .GT. VHIGH ) VHIGH=D
      IF( D .LT. VLOW ) VLOW=D
      IX=NX(IX)
  40  CONTINUE
C     DETERMINE THE NUMBER OF CELLS
      NCEL=8
      IF(KOUNT .LT. 10) NCEL=3
      IF(KOUNT .LT. 5) NCEL=2
      IF(KOUNT .LT. 2) CALL BAD
C     DETERMINE THE RANGE AND THE CELL INCREMENTS
      R=VHIGH-VLOW
      AINC=R/NCEL
      BLIM(1)=VLOW
      ULIM(1)=VLOW+AINC
C     SET THE UPPER AND LOWER LIMITS FOR EACH CELL
      DO 60 J=2,NCEL
      BLIM(J)=ULIM(J-1)
  60  ULIM(J)=BLIM(J)+AINC
      ULIM(NCEL)=VHIGH
      DO 65 J=1,NCEL
  65  JFREQ(J)=0
      IX=KST(KARY)
C     DETERMINE THE NUMBER OF ELEMENTS IN EACH CELL
      DO 80 J=1,KOUNT
      DO 70 JX=1,NCEL
      IF(DATA(IX) .GT. ULIM(JX)) GO TO 70
      JFREQ(JX) = JFREQ(JX)+1
      IX=NX(IX)
      GO TO 80
```

```
   70  CONTINUE
   80  CONTINUE
C      PRINT THE HEADINGS
       WRITE(6,82) K(JPOS+8)
   82  FORMAT(21X,'DISTRIBUTION FOR ARRAY ',A1)
C      DETERMINE CUMULUTIVE AND RELATIVE FREQUENCIES
       CUFR(1)=JFREQ(1)
       CURFR(1)=CUFR(1)/KOUNT
       DO 90 J=2,NCEL
       CUFR(J)=CUFR(J-1)+JFREQ(J)
   90  CURFR(J)=CUFR(J)/KOUNT
C      PRINT THE HEADINGS
       WRITE(6,92)
   92  FORMAT(57X,'CUMULATIVE'/45X,'CUMULATIVE  RELATIVE'/11X,'CELL LIMIT
      1S',10X,'FREQUENCY',4X,'FREQUENCY',3X,'FREQUENCY')
C      PRINT RESULTS IN A TABULAR FORM
       WRITE(6,96)(BLIM(J),ULIM(J),JFREQ(J),CUFR(J),CURFR(J),J=1,NCEL)
   96  FORMAT(2F14.5,5X,I6,F13.0,F10.3)
C      RETRIEVE STANDARD DEVIATION FOR THE VARIABLE
       IF( STND(KARY) .NE. JPERC ) GO TO 95
       KWAN=1
       JPOS=JPOS+1
       CALL VAR
       KWAN=0
       JPOS=JPOS-1
    95 VR=STND(KARY)
       VR=VR*VR
C      PRINT THE MEAN AND THE STANDARD DEVIATION
       WRITE (6,99) AVRG(KARY), VR, LL
  99   FORMAT( ' MEAN=',F14.5/' VARIANCE=',F14.5/' N=',I6//)
C      ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
       JPOS=JPOS+9
       CALL INTERP
       END
```

SUBROUTINE HIST

Subroutine HIST prints the histogram of a given array. First, the number of intervals, the cell width, the interval limits and the frequency in each interval are determined in the same manner as in subroutine FREQ. In order to put the whole chart on one page, a scaling is done if necessary. The maximum number of elements in any one cell is found and divided by 50, the whole portion of the result plus one gives the number of elements each asterisk should represent. For example, if the maximum number of elements is 35 each asterisk represents a point, if it is 75 each asterisk represents two points.

The printing of the histogram starts on a new page each time it is called. The scaling factor and the actual frequencies are printed at the top of the diagram and a table of values showing interval mid-points is placed below the diagram. The histogram is completed by printing the asterisks wherever they may be needed line by line from top to bottom.

The dimension statement allows 20 intervals, in case the number intervals is required to be increased. If the array name cannot be located in memory, subroutine BAD is called and the current instruction is skipped. When the job is completed successfully the pointer is updated and subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE HIST
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION CMARK(20),ULIM(20),BLIM(20),JFREQ(20),JOUT(20)
C     LOCATE THE VARIABLE IN NAME ARRAY
      DO 20 I=1,MASZ
      IF(K(JPOS+8).EQ.NAME(I)) GO TO 25
   20 CONTINUE
      WRITE(6,22) K(JPOS+8)
   22 FORMAT(' ARRAY NOT IN MEMORY ',A1)
      CALL BAD
C     PRINT THE HEADINGS
   25 WRITE(6,23) K(JPOS+8)
  23  FORMAT(1H1,' HISTOGRAM FOR THE ARRAY ',A1)
C     SET INDEX AND LENGTH
      IX=KST(I)
      LL=LNGT(I)
      KARY=I
      KOUNT=LL
C     DETERMINE THE LARGEST AND SMALLEST VALUES
      VHIGH=DATA(IX)
      VLOW=DATA(IX)
      DO 40 J=1,LL
      IF(DATA(IX).GT.VHIGH) VHIGH=DATA(IX)
      IF(DATA(IX).LT.VLOW) VLOW=DATA(IX)
      IX=NX(IX)
   40 CONTINUE
C     DETERMINE THE NUMBER OF CELLS
      NCEL=8
      IF(KOUNT.LT.10) NCEL=3
      IF(KOUNT.LT.5) NCEL=2
      IF(KOUNT.LT.2) CALL BAD
      R=VHIGH-VLOW
      AINC=R/NCEL
C     SET THE UPPER AND LOWER LIMITS FOR EACH CELL
      BLIM(1)=VLOW
      ULIM(1)=VLOW+AINC
      CMARK(1)=(BLIM(1)+ULIM(1))/2
      DO 60 J=2,NCEL
      BLIM(J)=ULIM(J-1)
      ULIM(J)=BLIM(J)+AINC
   60 CMARK(J)=(ULIM(J)+BLIM(J))/2
      ULIM(NCEL)=VHIGH
      CMARK(NCEL)=(VHIGH+BLIM(NCEL))/2
      DO 65 J=1,NCEL
   65 JFREQ(J)=0
      IX=KST(I)
C     DETERMINE THE NUMBER OF ELEMENTS IN EACH CELL
```

```
      DO 80 J=1,KOUNT
      DO 70 JX=1,NCEL
      IF(DATA(IX).GT.ULIM(JX)) GO TO 70
      JFREQ(JX)=JFREQ(JX)+1
      IX=NX(IX)
      GO TO 80
   70 CONTINUE
   80 CONTINUE
C     PRINT THE HEADINGS
      IF(NCEL.EQ.2) WRITE(6,1) (JFREQ(I),I=1,NCEL)
      IF(NCEL.EQ.3) WRITE(6,2) (JFREQ(I),I=1,NCEL)
      IF(NCEL.EQ.8) WRITE(6,3) (JFREQ(I),I=1,NCEL)
    1 FORMAT(/' FREQUENCY',2I5)
    2 FORMAT(/' FREQUENCY',3I5)
    3 FORMAT(/' FREQUENCY',8I5)
      WRITE(6,4)
    4 FORMAT(2X,'-------------------------------------------------------')
C     SCALE THE NUMBER OF POINTS TO BE PRINTED SUCH THAT THE HISTOGRAM FITS
C     ON ONE PAGE
      JSCAL=1
      FMAX=JFREQ(1)
      DO 51 I=2,NCEL
      IF(JFREQ(I).GT.FMAX) FMAX=JFREQ(I)
   51 CONTINUE
      IF(FMAX.GT.50) JSCAL=(FMAX+49)/50
C     PRINT THE SCALE FACTOR
      WRITE(6,5) JASTC,JSCAL
    5 FORMAT(' EACH 'A1,' EQUALS 'I2,' POINTS'/)
C     FORM THE LINE TO BE PRINTED
      DO 6 J=1,NCEL
    6 JOUT(J)=JBLKC
      MAX=FMAX/JSCAL
      DO 7 J=1,MAX
      X=MAX-(J-1)
      DO 8 I=1,NCEL
      IF((JFREQ(I)/JSCAL -X).GE.0) JOUT(I)=JASTC
    8 CONTINUE
      LX=X*JSCAL
      IF(NCEL.EQ.2) WRITE(6, 9) LX,(JOUT(I),I=1,NCEL)
      IF(NCEL.EQ.3) WRITE(6,10) LX,(JOUT(I),I=1,NCEL)
      IF(NCEL.EQ.8) WRITE(6,11) LX,(JOUT(I),I=1,NCEL)
    7 CONTINUE
    9 FORMAT(I6,4X,2(4X,A1))
   10 FORMAT(I6,4X,3(4X,A1))
   11 FORMAT(I6,4X,8(4X,A1))
C     PRINT LOWER LINE AND INTERVAL NUMBERS
      WRITE(6,4)
      DO 12 I=1,NCEL
   12 JOUT(I)=I
      IF(NCEL.EQ.2) WRITE(6,13) (JOUT(I),I=1,NCEL)
      IF(NCEL.EQ.3) WRITE(6,14) (JOUT(I),I=1,NCEL)
      IF(NCEL.EQ.8) WRITE(6,15) (JOUT(I),I=1,NCEL)
   13 FORMAT(' INTERVAL ',2I5)
   14 FORMAT(' INTERVAL ',3I5)
```

```
   15 FORMAT(' INTERVAL ',8I5)
      WRITE(6,16)
   16 FORMAT(//10X,'INTERVAL',5X,'INTERVAL MIDPOINT')
C     PRINT MIDPOINT VALUES FOR EACH INTERVAL
      WRITE(6,17) (J,CMARK(J),J=1,NCEL)
   17 FORMAT(10X,I5,4X,F14.5)
      WRITE(6,18)
   18 FORMAT(///)
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+9
      CALL INTERP
      END
```

SUBROUTINE SKEW

Subroutine SKEW is used to calculate the skewness or kurtosis of an array. Skewness is calculated by the ratio of $\frac{\mu_3}{\sigma^3}$ where μ_3 is the third moment about the mean and σ is the standard deviation of the array. Kurtosis is calculated by $\frac{\mu_4}{\mu_2^2} - 3$ where μ_4 and μ_2 are the fourth and second moments about the mean respectively. (μ_2 is the variance of the array.)

Whichever quantity is asked for the value is printed along with a message and the variable name, pointer (JPOS) is updated and subroutine INTERP is called to decode the next instruction in the program. If the variable cannot be located in the NAME array a message 'ARRAY NOT IN MEMORY' is printed and subroutine BAD is called to skip the current instruction in the STIL program.

```
      SUBROUTINE SKEW
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ JOC,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      IK=K(JPOS+8)
C     VARIABLE IS LOCATED IN NAME ARRAY. IF IT IS NOT IN MEMORY A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THIS INSTRUCTION
      DO 10 I=1,MASZ
      IF( NAME(I) .EQ. IK ) GO TO 20
   10 CONTINUE
      WRITE(6,12) IK
   12 FORMAT(1X, 'ARRAY 'A1, ' IS NOT IN MEMORY' )
   15 CALL BAD
   20 KARRY=I
C     SECOND, THIRD AND FOURTH MOMENTS AROUND THE MEAN ARE CALCULATED
      LL=LNGT(KARRY)
      U2=0.
      U3=0.
      U4=0.
      IF( AVRG(KARRY) .NE. JPERC ) GO TO 30
      KW=KWAN
      KWAN=1
      CALL AVG
      KWAN=KW
   30 XB=AVRG(KARRY)
      IX=KST(KARRY)
      DO 40 I=1,LL
      D=DATA(IX)-XB
      D2=D*D
      D3=D2*D
      D4=D2*D2
      U2=U2+D2
      U3=U3+D3
      U4=U4+D4
   40 IX=NX(IX)
      D=LL
      U2=U2/D
      U3=U3/D
      U4=U4/D
      SD=SQRT(U2)
      STND(KARRY)=SD
      SK=U3/(U2*SD)
      RK=U4/(U2*U2)-3.
C     SKEWNESS AND KURTOSIS ARE PRINTED, JPOS IS UPDATED AND SUBROUTINE
C     INTERP IS CALLED TO WORK ON THE NEXT INSTRUCTION
      IF( K(JPOS+4) .EQ. JKC ) GO TO 45
      WRITE(6,42) IK,SK
   42 FORMAT( 1X, ' SKEWNESS OF ARRAY ',A1, ' IS ',G12.4 /)
      GO TO 50
   45 WRITE (6,44) IK,RK
   44 FORMAT( 1X, ' KURTOSIS OF ARRAY ',A1, ' IS ',G12.4 //)
```

```
50 JPOS=JPOS+9
   CALL INTERP
   END
```

SUBROUTINE CALREG

Subroutine CALREG sets the elements of the dependent array equal to the values on the regression line. It is called upon to execute instructions such as SET T = REG Y,X,Z. For the above example, first a regression line equation Y = A + BX + CZ will be found and then the elements of T array will be set equal to the values of Y calculated from that equation. It is possible that a variable name can appear on both sides of the equal sign such as SET Y = REG Y,X . In this case the old values of Y will be replaced by the new values of Y calculated from the Y = A + BX equation.

First the validity of the instruction is checked and if there is any error a proper message is printed and subroutine BAD is called to locate the next instruction in the program. If there aren't any errors KFLAG is set to 10 and subroutine REGR is called to calculate the coefficients of the regression equation. Then, with the help of the returned values, the values of the points on the regression line are calculated and stored in the appropriate elements of the specified array. The values of the elements are not printed since the user can get a display of these values by using a WRITE command if he wishes to do so.

After the completion of the task subroutine INTERP is called to work on the next instruction in the program.

```
      SUBROUTINE CALREG
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      COMMON/BLK9/ KORDER,CNST1,CNST2,CNST3,CNST4
C     LOCATE THE FIRST VARIABLE IN NAME ARRAY
      DO 10 I=1,MASZ
      IF( K(JPOS+8) .EQ. NAME(I)) GO TO 20
   10 CONTINUE
      WRITE(6,12) K(JPOS+8)
   12 FORMAT(  '  ARRAY NOT IN MEMORY   ', A1)
      CALL BAD
   20 LL=LNGT(I)
      KARYX=I
C     LOCATE THE VARIABLE TO BE SET TO REGRESSION VALUES IN NAME ARRAY
      DO 25 I=1,MASZ
      IF( K(JPOS+3) .EQ. NAME(I)) GO TO 90
   25 CONTINUE
C     AN EMPTY CELL IN NAME ARRAY IS LOCATED. IF NONE IS AVAILABLE, A MESSAGE
C     IS PRINTED AND SUBROUTINE BAD IS CALLED TO SKIP THE INSTRUCTION
      DO 30 I=1,MASZ
      IF( NAME(I) .EQ. 0 ) GO TO 40
   30 CONTINUE
      WRITE(6,32) K(JPOS+3)
   32 FORMAT(  '  TOO MANY ARRAYS  ', A1 )
      CALL BAD
   40 KARYW=I
      KSUM=0
C     CHECK TO SEE IF THERE IS ENOUGH EMPTY CELLS IN DATA ARRAY
      DO 45 I=1,MASZ
   45 KSUM=KSUM+LNGT(I)
      IF( KSUM+LL .LT. MDSZ ) GO TO 55
      WRITE(6,47) K(JPOS+3),KSUM,LL
   47 FORMAT( '  DATA ARRAY TOO FULL   ', A1,2I8 )
      CALL BAD
   55 IW=LWC+1
   60 IF( NX(IW) .EQ. JPERC ) GO TO 65
      IW=IW+1
      GO TO 60
   65 KST(KARYW)=IW
      NAME(KARYW)=K(JPOS+3)
      LNGT(KARYW)=LL
      LWC=IW
      N=LL-1
C     PROVIDE SPACE FOR THE NEW VARIABLE IN DATA ARRAY
      DO 80 I=1,N
   70 IW=IW+1
      IF( NX(IW) .EQ. JPERC ) GO TO 75
      GO TO 70
   75 NX(LWC)=IW
      LWC=IW
```

```
   80 CONTINUE
      NX(LWC)=JASTC
      GO TO 125
   90 KARYW=I
      LLL=LNGT(I)
      IF( LLL .EQ. LL ) GO TO 115
      IF( LLL .GT. LL ) GO TO 100
      WRITE(6,92) K(JPOS+3),LNGT(I),K(JPOS+8),LL
   92 FORMAT('  VARIABLE LENGTH SMALLER THAN ARGUMENT  ',A1,3X,
     1 I4,5X,A1,3X,I4 )
      CALL BAD
C     ERASE THE EXCESS LENGTH
  100 IX=KST(I)
      DO 110 I=1,LLL
      IF( I .LE. LL ) GO TO 105
      IXN=NX(IX)
      NX(IX)=JPERC
      IX=IXN
      GO TO 110
  105 IX=NX(IX)
  110 CONTINUE
      LNGT(KARYW)=LL
  115 AVRG(KARYW)=JPERC
      STND(KARYW)=JPERC
C     DETERMINE THE NUMBER OF VARIABLES
  125 IF( K(JPOS+13) .EQ. JCOMC ) GO TO 196
      IF( K(JPOS+11) .EQ. JCOMC ) GO TO 166
      IF( K(JPOS+ 9) .EQ. JCOMC ) GO TO 141
      WRITE(6,127) K(JPOS+9)
  127 FORMAT( '  ILLEGAL INSTRUCTION  ',A1 )
      CALL BAD
C     PRINT THE HEADINGS
  141 WRITE(6,142) K(JPOS+3),K(JPOS+3),K(JPOS+10)
  142 FORMAT(1X,A1,' ARRAY IS SET TO ',2X,A1,4H=A+B,A1 )
      KORDR=2
      GO TO 129
  166 WRITE(6,167) K(JPOS+3),K(JPOS+3),K(JPOS+10),K(JPOS+12)
  167 FORMAT(1X,A1,' ARRAY IS SET TO ',2X,A1,4H=A+B,A1,2H+C,A1 )
      KORDR=3
      GO TO 129
  196 WRITE(6,197) K(JPOS+3),K(JPOS+3),K(JPOS+10),K(JPOS+12),K(JPOS+14)
  197 FORMAT(1X,A1,' ARRAY IS SET TO ',2X,A1,4H=A+B,A1,2H+C,A1,2H+D,A1 )
      KORDR=4
C     SET KFLAG AND CALL SUBROUTINE REGR TO FIND THE REGRESSION EQUATION
  129 KF=KFLAG
      KFLAG=10
      JPOS=JPOS+1
      CALL REGR
      JPOS=JPOS-1
      KFLAG=KF
      IW=KST(KARYW)
C     LOCATE THE SECOND VARIABLE IN NAME ARRAY
      DO 130 I=1,MASZ
      IF( K(JPOS+10) .EQ. NAME(I)) GO TO 135
  130 CONTINUE
      WRITE(6,12) K(JPOS+10)
```

```
      CALL BAD
  135 KARYY=I
      N=LL-1
      IY=KST(I)
      IF( KORDR .EQ. 4 ) GO TO 170
      IF( KORDR .EQ. 3 ) GO TO 150
C     CALCULATE AND STORE THE REGRESSION VALUES FOR TWO VARIABLES
      DATA(IW)=CNST1+CNST2*DATA(IY)
      DO 140 I=1,N
      IW=NX(IW)
      IY=NX(IY)
  140 DATA(IW)=CNST1+CNST2*DATA(IY)
      NX(IW)=JASTC
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+11
      CALL INTERP
C     LOCATE THE THIRD VARIABLE IN NAME ARRAY
  150 DO 155 I=1,MASZ
      IF( K(JPOS+12) .EQ. NAME(I)) GO TO 160
  155 CONTINUE
      WRITE(6,12) K(JPOS+12)
      CALL BAD
C     CALCULATE AND STORE THE REGRESSION VALUES FOR THREE VARIABLES
  160 KARYZ=I
      IZ=KST(I)
      DATA(IW)=CNST1+CNST2*DATA(IY)+CNST3*DATA(IZ)
      DO 165 I=1,N
      IW=NX(IW)
      IY=NX(IY)
      IZ=NX(IZ)
  165 DATA(IW)=CNST1+CNST2*DATA(IY)+CNST3*DATA(IZ)
      NX(IW)=JASTC
C     ADJUST JPOS AND CALL SUBROUTINE INTERP TO WORK ON THE NEXT INSTRUCTION
      JPOS=JPOS+13
      CALL INTERP
C     LOCATE THE THIRD VARIABLE IN NAME ARRAY
  170 DO 175 I=1,MASZ
      IF( K(JPOS+12) .EQ. NAME(I)) GO TO 180
  175 CONTINUE
      WRITE(6,12) K(JPOS+12)
      CALL BAD
  180 KARYZ=I
C     LOCATE THE FOURTH VARIABLE IN NAME ARRAY
      IZ=KST(I)
      DO 185 I=1,MASZ
      IF( K(JPOS+14) .EQ. NAME(I)) GO TO 190
  185 CONTINUE
      WRITE(6,12) K(JPOS+14)
      CALL BAD
C     CALCULATE AND STORE THE REGRESSION VALUES FOR FOUR VARIABLES
  190 KARYT=I
      IT=KST(I)
      DATA(IW)=CNST1+CNST2*DATA(IY)+CNST3*DATA(IZ)+CNST4*DATA(IT)
      DO 195 I=1,N
      IW=NX(IW)
      IY=NX(IY)
```

```
      IZ=NX(IZ)
      IT=NX(IT)
  195 DATA(IW)=CNST1+CNST2*DATA(IY)+CNST3*DATA(IZ)+CNST4*DATA(IT)
      NX(IW)=JASTC
C     ADJUST JPOS AND CALL SUBROUTINE INTERP
      JPOS=JPOS+15
      CALL INTERP
      END
```

SUBROUTINE SETARY

Addition, subtraction, division and multiplication of two arrays are done through the utilization of subroutine SETARY. It should be mentioned that multiplication of two arrays does not refer to vector multiplication; the elements of one array are multiplied by the corresponding elements of the second array and the results are stored into the proper elements of the dependent variable. A variable name can appear on both sides of the equal sign and manipulation of more than two arrays can be accomplished by the user through a series of instructions involving successive pairs of variables.

If the lengths of the independent variables do not agree, the operations cannot be performed and a message 'ARRAYS NOT OF EQUAL LENGTH' is printed along with the names and lengths of the variables, and subroutine BAD is called to skip the instruction. The dependent variable may or may not be in memory and its length may or may not agree with the independent variables' length. If its length is different it is destroyed and a new array of proper length is created.

If the operator is not one of (+), (-), (·) or (/) a message 'UNRECOGNIZED OPERATOR' is printed and subroutine BAD is called to skip the instruction. After a successful execution the pointer (JPOS) is updated and subroutine INTERP is called to decode the next instruction in the program.

```
      SUBROUTINE SETARY
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      LOCATE THE INDEPENDENT VARIABLES IN NAME ARRAY
      NM=K(JPOS+5)
      CALL CKNAME(NM,ID)
   10 IF(ID .EQ. 0) GO TO 15
      KARYX=ID
      NM=K(JPOS+7)
      CALL CKNAME(NM,ID)
      IF(ID .NE. 0) GO TO 30
   15 WRITE(6,17) NM
   17 FORMAT(1X, ' ARRAY NOT IN MEMORY  ',A1)
   25 CALL BAD
   30 KARYY=ID
      COMPARE THE LENGTHS OF THE TWO VARIABLES
      IF( LNGT(KARYX) .EQ. LNGT(KARYY) ) GO TO 40
      WRITE (6,32) K(JPOS+5),LNGT(KARYX),K(JPOS+7),LNGT(KARYY)
   32 FORMAT(  '  ARRAYS NOT OF EQUAL LENGTH   ',A1, I6, 5X, A1, I6 )
      CALL BAD
   40 LL=LNGT(KARYX)
      CHECK TO SEE IF THE DEPENDENT VARIABLE IS ALREADY IN MEMORY
      NM=K(JPOS+3)
      IF( NM .EQ. K(JPOS+5)) GO TO 335
      IF( NM .EQ. K(JPOS+7)) GO TO 340
      CALL CKNAME(NM,ID)
      IF(ID .EQ. 0) GO TO 310
      IF IT IS IN MEMORY COMPARE ITS LENGTH  WITH INDEPENDENT VARIABLES LENGTH
      IF(LNGT(ID) .EQ. LL ) GO TO 345
      CALL DESTRY(ID)
      CHECK TO SEE IF THERE ARE ENOUGH EMPTY CELLS FOR THE NEW ARRAY
  310 KSUM=0
      DO 315 I=1,MASZ
  315 KSUM=KSUM+LNGT(I)
      IF(LL .LE. (MDSZ-KSUM)) GO TO 320
      WRITE(6,317) NM,LL
  317 FORMAT(1X, ' DATA ARRAY TOO FULL  ',A1,I8 )
      GO TO 25
      CHECK TO SEE IF THERE IS AN EMPTY CELL IN NAME ARRAY
  320 CALL ASNAME(ID)
      IF( ID .NE. 0 ) GO TO 330
      WRITE(6,322) NM
  322 FORMAT(1X, '  TOO MANY ARRAYS  ',A1)
      GO TO 25
  330 KARYZ=ID
*****CREATE THE NEW ARRAY   SET ASIDE CELLS ETC.
      NAME(KARYZ)=NM
      LNGT(KARYZ)=LL
      CALL ASCELL(KARYZ)
      GO TO 55
```

```
 335 KARYZ=KARYX
     GO TO 55
 340 KARYZ=KARYY
     GO TO 55
 345 KARYZ=ID
  55 IX= KST(KARYX)
     IY= KST(KARYY)
     IZ= KST(KARYZ)
     LL=LL-1
     IDENTIFY THE OPERATION
     IF( K(JPOS+6) .EQ. JDIVC ) GO TO 110
     IF( K(JPOS+6) .EQ. JPERC) GO TO 100
     IF( K(JPOS+6) .EQ. JMINC) GO TO 90
     IF( K(JPOS+6) .EQ. JPLSC) GO TO 70
  56 WRITE (6,57) K(JPOS+6)
  57 FORMAT(  '  UNRECOGNIZED OPERATOR   ', A1)
     CALL BAD
*****ADDITION OF TWO ARRAYS
  70 DO 75 I=1,LL
     DATA(IZ)=DATA(IX)+DATA(IY)
     IX=NX(IX)
     IY=NX(IY)
     IZ=NX(IZ)
  75 CONTINUE
     DATA(IZ)=DATA(IX)+DATA(IY)
     GO TO 115
*****SUBTRACTION OF TWO ARRAYS
  90 DO 95 I=1,LL
     DATA(IZ)=DATA(IX)-DATA(IY)
     IX=NX(IX)
     IY=NX(IY)
     IZ=NX(IZ)
  95 CONTINUE
     DATA(IZ)=DATA(IX)-DATA(IY)
     GO TO 115
*****MULTIPLICATION OF TWO ARRAYS
 100 DO 105 I=1,LL
     DATA(IZ)=DATA(IX)*DATA(IY)
     IX=NX(IX)
     IY=NX(IY)
     IZ=NX(IZ)
 105 CONTINUE
     DATA(IZ)=DATA(IX)*DATA(IY)
     GO TO 115
*****DIVISION OF TWO ARRAYS
 110 DO 113 I=1,LL
     DATA(IZ)=DATA(IX)/DATA(IY)
     IX=NX(IX)
     IY=NX(IY)
     IZ=NX(IZ)
 113 CONTINUE
     DATA(IZ)=DATA(IX)/DATA(IY)
 115 AVRG(KARYZ)=JPERC
     STND(KARYZ)=JPERC
     JPOS=JPOS+8
     CALL INTERP
     END
```

SUBROUTINE TRANSF

Common logarithm, natural logarithm, square root, inverse, exponential and power transformations are computed by subroutine TRANSF. Examples: $Y = \log_{10} X$; $Y = \log_e X$; $Y = X$; $Y = 1/X$; $Y = e^x$; $Y = X^{1.2}$.

If the calculated values are to be stored into a new array, availability of empty cells in the NAME and DATA arrays are checked and secured. If this phase is successfully completed the array is checked for illegal elements for the specified transformation. For instance, a negative number for square root transformation and a zero element for the inverse transformation are illegal transformations. If such an error is found, a proper message is printed and subroutine BAD is called to skip the current instruction. If the name of the dependent variable is already in memory information about the old array is erased from the memory.

If everything is correct transformations are carried out and the calculated values are stored in the proper elements. In this process, system functions SQRT, ALØG10, ALOG, ABS and EXP are used. Upon the completion of the task the pointer is updated and the subroutine INTERP is called to decode the next instruction.

```
      SUBROUTINE TRANSF
C
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C
C  COMMON LOG, NATURAL LOG, SQUARE ROOT, EXPONENTIAL (E RAISED TO X POWER) AND
C  POWER TRANSFORMATIONS (X RAISED TO A POWER) ARE CALCULATED BY THIS ROUTINE
C
      M=K(JPOS+6)
C
C  TRANSFORMATIONS ARE IDENTIFIED
C
      IF( M.EQ. JGC .AND. K(JPOS+7) .EQ. JLPRC) GO TO 20
      IF( M.EQ. JNC .AND. K(JPOS+7) .EQ. JLPRC) GO TO 25
      IF( M .EQ. JQC ) GO TO 30
      IF( M .EQ. JNC ) GO TO 35
      IF( M .EQ. JEC ) GO TO 40
      IF( M .EQ. JXC ) GO TO 45
   10 CALL BAD
   20 INDC=1
      GO TO 50
   25 INDC=2
      GO TO 50
   30 INDC=3
      GO TO 50
   35 INDC=4
      GO TO 50
   40 INDC=6
      GO TO 50
   45 INDC=5
   50 N=JPOS+6
      DO 55 I=1,3
      IF( K(N+I) .EQ. JLPRC ) GO TO 60
   55 CONTINUE
      NMX=K(JPOS+5)
      GO TO 65
   60 NMX=K(N+I+1)
C
C  VARIABLE NAME IS FOUND IN NAME LIST
C
   65 CALL CKNAME(NMX,ID)
      IF( ID .NE. 0 ) GO TO 70
      WRITE(6,67) NMX
   67 FORMAT(1X, ' ARRAY IS NOT IN MEMORY  ',A1 /)
      GO TO 10
   70 KARYX=ID
      NMY=K(JPOS+3)
      LL=LNGT(KARYX)
      IF( NMX .EQ. NMY ) GO TO 95
      CALL CKNAME(NMY,ID)
```

```
C
C  DEPENDENT VARIABLE NAME IS SEARCHED IN NAME LIST
C
      IF( ID .EQ. 0 ) GO TO 75
C
C  IF THE DEPENDENT VARIABLE NAME IS IN MEMORY AND ITS LENGTH IS DIFFERENT THAN
C  THE INDEPENDENT VARIABLE, THE ARRAY IS ERASED FROM THE MEMORY
C
      IF( LL .EQ. LNGT(ID) ) GO TO 96
      CALL DESTRY(ID)
   75 KS=0
C
C  DATA ARRAY IS CHECKED TO SEE IF IT CONTAINES ENOUGH EMPTY CELLS FOR THE NEW
C  ARRAY TO BE CREATED
C
      DO 80 I=1,MASZ
   80 KS=KS+LNGT(I)
      IF( LL .LT. (MDSZ-KS)) GO TO 85
      WRITE(6,82) NMY,LL
   82 FORMAT(1X, ' DATA ARRAY TOO FULL  ',A1,I8 /)
      GO TO 10
C
C  AN EMPTY CELL IN NAME LIST IS SOUGHT FOR DEPENDENT VARIABLE NAME
C
   85 CALL ASNAME(ID)
      IF( ID .NE. 0 ) GO TO 90
      WRITE(6,87) NMY
   87 FORMAT(1X, ' TOO MANY ARRAYS  ',A1 /)
      GO TO 10
C
C  EMPTY DATA CELLS ARE ASSIGNED TO THE DEPENDENT VARIABLE
C
   90 KARYY=ID
      NAME(KARYY)=NMY
      LNGT(KARYY)=LL
      CALL ASCELL(KARYY)
      GO TO 100
   95 KARYY=KARYX
      GO TO 100
   96 KARYY=ID
  100 IX=KST(KARYX)
      IY=KST(KARYY)
C
C  DEPENDING ON THE VALUE OF 'INDC' AN APPROPRIATE SET OF CODING IS SELECTED
C  FOR THE TRANSFORMATIONS
C
      GO TO (110,110,160,175,190,200 ),INDC
C
C  CHECK TO SEE IF THERE IS AN ILLEGAL TRANSFORMATION
C
  110 DO 115 I=1,LL
      IF( DATA(IX) .LE. 0 ) GO TO 150
      IX=NX(IX)
  115 CONTINUE
      IX=KST(KARYX)
      IY=KST(KARYY)
```

```
      IF( INDC .EQ. 2 ) GO TO 130
C
C  COMMON LOG TRANSFORMATIONS
C
      DO 125 I=1,LL
      DATA(IY)=ALOG10(DATA(IX))
      IX=NX(IX)
  125 IY=NX(IY)
      GO TO 140
C
C  NATURAL LOG TRANSFORMATIONS
C
  130 DO 135 I=1,LL
      DATA(IY)=ALOG(DATA(IX))
      IX=NX(IX)
  135 IY=NX(IY)
  140 IX=KST(KARYX)
      IY=KST(KARYY)
      DO 145 I=1,LL
      IF( ABS(DATA(IY)) .LT. .000001 )  DATA(IY)=0.
      IY=NX(IY)
  145 CONTINUE
      JPOS=JPOS+10
      GO TO 400
  150 WRITE(6,152) DATA(IX)
  152 FORMAT(1X, ' ILLEGAL INSTRUCTION  ',G12.4 /)
      GO TO 300
C
C  SQUARE ROOT TRANSFORMATIONS
C
  160 DO 165 I=1,LL
      IF( DATA(IX) .LT. 0 ) GO TO 150
      IX=NX(IX)
  165 CONTINUE
      IX=KST(KARYX)
      IY=KST(KARYY)
      DO 170 I=1,LL
      DATA(IY)=SQRT(DATA(IX))
      IX=NX(IX)
  170 IY=NX(IY)
      JPOS=JPOS+12
      GO TO 400
C
C  INVERSE TRANSFORMATIONS
C
  175 DO 180 I=1,LL
      IF( DATA(IX) .EQ. 0. ) GO TO 150
      IX=NX(IX)
  180 CONTINUE
      IX=KST(KARYX)
      DO 185 I=1,LL
      DATA(IY)=1./DATA(IX)
      IX=NX(IX)
  185 IY=NX(IY)
      JPOS=JPOS+11
      GO TO 400
```

```
C
C  EXPONENTIAL TRANSFORMATIONS (Y IS SET TO E RAISED TO X POWER)
C
  190 DO 195 I=1,LL
      DATA(IY)=EXP(DATA(IX))
      IX=NX(IX)
  195 IY=NX(IY)
      JPOS=JPOS+11
      GO TO 400
C
C  POWER TRANSFORMATIONS (Y IS SET TO X RAISED TO A POWER )
C  THE VALUE OF THE POWER IS DETERMINED FIRST
C
  200 KB=JPOS+6
      N=0
      IF( K(KB+1) .EQ. JMINC .OR. K(KB+1) .EQ. JPLSC ) KB=KB+1
      DO 205 I=1,10
      IF( K(KB+I) .EQ. JSEMC ) GO TO 210
  205 N=N+1
      WRITE(6,207) K(KB+I)
  207 FORMAT(1X, 'ILLEGAL INSTRUCTION  ',A1 /)
      GO TO 300
  210 KE=KB+I
      NN=0
      E=0.
      DO 215 I=1,N
      IF( K(KB+I) .EQ. JPERC ) GO TO 225
  215 NN=NN+1
  220 E=INTEG(KB,N)
      GO TO 250
  225 KP=KB+I
      IF( NN .EQ. 0 ) GO TO 235
      IF( NN .EQ. (N-1)) GO TO 240
      E=INTEG(KB,NN)
      NN=N-NN-1
  230 E=E+FRAC(KP,NN)
      GO TO 250
  235 NN=N-1
      GO TO 230
  240 N=N-1
      GO TO 220
  250 IF( K(JPOS+7) .EQ. JMINC ) E=-E
      DO 255 I=1,LL
      DATA(IY)=DATA(IX)**E
      IX=NX(IX)
  255 IY=NX(IY)
      JPOS=KE
      GO TO 400
C
C  IF A TRANSFORMATION CAN NOT BE CARRIED OUT SUCCESSFULLY RESERVED SPACES FOR
C  THE NEW ARRAY ARE MADE AVAILABLE FOR FURTHER USE
C
  300 CALL DESTRY(KARYY)
      GO TO 10
C
C  SUB. INTERP IS CALLED TO DECODE THE NEXT INSTRUCTION IN THE PROGRAM
```

```
C
  400 CALL INTERP
      END
```

SUBROUTINE DIST

Subroutine DIST sets the elements of an array to random numbers drawn from normal, binomial, uniform, and exponential distributions with specified parameters. The routine can also give random values from a given cumulative probability distribution function. A c.d.f. can be defined by a maximum of 15 points and straight lines will be assumed to connect these points. This feature enables the user to work with any distribution.

The execution of the routine goes as follows. First, the validity of the instruction is checked, and then the number of values requested is determined. If the array name is found in the NAME array, it is erased from memory and the availability of the required number of empty cells in NAME and DATA arrays are checked. Then, the number of parameters is determined and proper command positions are examined to determine the distribution. If all is correct a set of coding for the required distribution is executed. Otherwise, an appropriate error message is printed and subroutine BAD is called to skip the instruction?

Normal distribution requires two parameters to be defined - mean and the standard deviation. Binomial distribution is defined by two parameters - N and p and the uniform distribution is defined by two parameters - minimum and maximum. Exponential distribution is defined by one parameter - mean. Its lower limit is assumed to be at zero and its upper limit is assumed to be + infinity. Two methods of computation are used for binomial distribution depending on the value of N to increase the speed of execution. As it was mentioned earlier a cumulative probability distribution can be defined by three to fifteen points on the curve. The

first point must have the value of zero and the last must be one. There can be no decrease in the probability in going from one point to the next since the points represent cumulative probabilities.

The task is completed when the elements of the specified array is filled with the random values drawn from the required distributions. Once this is done subroutine INTERP is called to decode the next instruction in the user's program.

```
      SUBROUTINE DIST
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
      DIMENSION VAR(15),CDF(15)
C     SUBROUTINE DIST SETS THE ELEMENTS OF AN ARRAY TO RANDOM VALUES DRAWN FROM
C     SOME SPECIFIED DISTRIBUTIONS
C     'N' DESIGNATES A NORMAL DISTRIBUTION MEAN AND STANDARD DEVIATION SHOULD BE
C     DEFINED AS PARAMETERS
C     'B' DESIGNATES A BINOMIAL DISTRIBUTION, 'N' AND 'P' SOULD BE DEFINED
C     'U' DESIGNATES A UNIFORM DISTRIBUTION, MINIMUM AND MAXIMUM LIMITS  SOULD
C     BE DEFINED
C     'E' DESIGNATES AN EXPONENTIAL DISTRIBUTION, MEAN SHOULD BE DEFINED
C     IT IS ASSUMED TO BE SKEWED TO THE RIGHT WITH MINIMUM AT ZERO AND NO MAXIMUM
C     'C' DESIGNATES A GIVEN CUMULITIVE PROBOBILITY DISTRIBUTION. MAXIMUM NUMBER OF
C     POINTS ALLOWED IS 30.  EACH POINT MUST BE ON CDF.
      M=JPOS+4
      DO 10 I=1,6
      IF( K(M+I) .EQ. JLPRC ) GO TO 20
   10 CONTINUE
      WRITE(6,12) K(M+I)
   12 FORMAT(1X, ' ILLEGAL INSTRUCTION  ',A1,/)
   15 CALL BAD
   20 KLP=M+I
C     CALCULATE THE NUMBER OF VALUES TO BE DRAWN FROM THE DISTRIBUTIONS
      KB=M
      N=I-2
      IF( N .LE. 0 ) GO TO 25
      NVAL=INTEG(KB,N)
      GO TO 30
   25 WRITE(6,12) K(JPOS+6)
      GO TO 15
   30 M=K(JPOS+3)
C     CHECK THE VARIABLE NAME IN NAME LIST, IF IT IS THERE DESTROY THE ARRAY
      CALL CKNAME(M,ID)
      IF( ID .EQ. 0 ) GO TO 35
      CALL DESTRY(ID)
   35 KS=0
C     CHECK DATA ARRAY TO SEE IF THERE ARE ENOUGH EMPTY CELLS FOR THE NEW ARRAY
C     TO BE CREATED
      DO 40 I=1,MASZ
   40 KS=KS+LNGT(I)
      IF( NVAL .LT. (MDSZ-KS) ) GO TO 45
      WRITE(6,42) M,NVAL
   42 FORMAT(1X, ' DATA ARRAY TOO FULL  ',A1,I8 /)
      GO TO 15
C     CHECK NAME ARRAY TO SEE IF THERE IS AN EMPTY CELL
   45 CALL ASNAME(ID)
      IF( ID .NE. 0 ) GO TO 50
      WRITE(6,47) M
   47 FORMAT(1X, ' TOO MANY ARRAYS  ',A1 /)
```

```
      GO TO 15
   50 KARY=ID
      KNTR=0
C  DETERMINE THE NUMBER OF PARAMETERS
      DO 55 I=1,300
      IF( K(KLP+I) .EQ. JRPRC ) GO TO 60
      IF( K(KLP+I) .NE. JCOMC ) GO TO 55
      KNTR=KNTR+1
   55 CONTINUE
      WRITE(6,57) K(KLP)
   57 FORMAT(1X, 'NO CLOSING PARANTHESIS  ',A1 )
      GO TO 15
   60 KRP=KLP+I
C  IF THERE IS ONLY ONE PARAMETER BRANCH TO EXPONENTIAL DISTRIBUTION
C  IF THERE ARE TWO PARAMETERS BRANCH TO STATEMENT 130 OTHERWISE GO TO 270
      IF( KNTR .EQ. 0 ) GO TO 70
      IF( KNTR .EQ. 1 ) GO TO 130
      IF( KNTR .LE. 29 ) GO TO 270
      WRITE(6,12) K(KLP-1)
      GO TO 15
C  DETERMINE THE VALUE OF THE MEAN FOR EXPONENTIAL DISTRIBUTION
   70 N=KRP-KLP-1
      DO 75 I=1,N
      IF( K(KLP+I) .EQ. JPERC ) GO TO 80
   75 CONTINUE
      P1=INTEG(KLP,N)
      GO TO 95
   80 KP=KLP+I
      N=KP-KLP-1
      IF( N .EQ. 0 ) GO TO 85
      P1=INTEG(KLP,N)
      GO TO 90
   85 P1=0.
   90 N=KRP-KP-1
      IF( N .EQ. 0 ) GO TO 95
      P1=P1+FRAC(KP,N)
   95 M=K(KLP-1)
      IF( M .EQ. JEC .OR. M .EQ. JKC ) GO TO 100
      WRITE(6,12) M
      GO TO 15
C  CREATE THE NEW ARRAY (SET ASIDE THE CELLS ETC.)
  100 NAME(KARY)=K(JPOS+3)
      LNGT(KARY)=NVAL
      CALL ASCELL(KARY)
      IX=KST(KARY)
      IF( M .EQ. JKC ) GO TO 110
C  SET THE ELEMENTS OF THE ARRAY TO VALUES RANDOMLY DRAWN FROM EXPONENTIAL DIST.
      DO 105 I=1,NVAL
      U=RAN(NSEED)
      DATA(IX)=-P1*ALOG(U)
  105 IX=NX(IX)
      GO TO 500
  110 CONTINUE
  111 GO TO 500
C  DETERMINE THE VALUE OF THE FIRST PARAMETER
  130 DO 135 I=1,20
```

```
      IF( K(KLP+I) .EQ. JCOMC ) GO TO 140
  135 CONTINUE
      WRITE(6,12) K(KLP+I)
      GO TO 15
  140 KC=KLP+I
      N=KC-KLP-1
      DO 145 I=1,N
      IF( K(KLP+I) .EQ. JPERC ) GO TO 150
  145 CONTINUE
      P1=INTEG(KLP,N)
      GO TO 165
  150 KP=KLP+I
      N=KP-KLP-1
      IF( N .EQ. 0 ) GO TO 155
      P1= INTEG(KLP,N)
      GO TO 160
  155 P1=0.
  160 N=KC-KP-1
      IF( N .EQ. 0 ) GO TO 165
      P1=P1+FRAC(KP,N)
C  DETERMINE THE VALUE OF THE SECOND PARAMETER
  165 N=KRP-KC-1
      DO 170 I=1,N
      IF( K(KC+I) .EQ. JPERC ) GO TO 175
  170 CONTINUE
      P2=INTEG(KC,N)
      GO TO 190
  175 KP=KC+I
      N=KP-KC-1
      IF( N .EQ. 0 ) GO TO 180
      P2=INTEG(KC,N)
      GO TO 185
  180 P2=0.
  185 N=KRP-KP-1
      IF( N .EQ. 0 ) GO TO 190
      P2=P2+FRAC(KP,N)
  190 M=K(KLP-1)
C  CHECK THE VALIDITY OF THE INSTRUCTION
      IF( M .EQ. JNC .OR. M .EQ. JBC .OR. M .EQ. JUC ) GO TO 200
      WRITE(6,12) M
      GO TO 15
C  CREATE THE NEW ARRAY (SET ASIDE CELLS ETC.)
  200 NAME(KARY)=K(JPOS+3)
      LNGT(KARY)=NVAL
      CALL ASCELL(KARY)
      IX=KST(KARY)
C  IF IT IS A NORMAL DISTRIBUTION BRANCH TO STATEMWNT 250
C  IF IT IS A BINOMIAL DISTRIBUTION BRANCH TO 210 OTHERWISE SET THE ELEMENTS OF
C  THE ARRAY TO RANDOM NUMBERS DRAWN FROM A UNIFORM DISTRIBUTION
      IF( M .EQ. JNC ) GO TO 250
      IF( M .EQ. JBC ) GO TO 210
      DO 205 I=1,NVAL
      DATA(IX)=P1+(P2-P1)*RAN(NSEED)
  205 IX=NX(IX)
      GO TO 500
C  IF 'N' IS SMALL (50 OR LESS) BRANCH TO 235 OTHERWISE EXECUTE FOLLOWING
```

```
C  INSTRUCTIONS FOR THE BINOMIAL DISTRIBUTION
  210 IF( P1 .LE. 50.) GO TO 235
      N=P1
      DO 230 II=1,NVAL
      R=(1.-P2)**N
      U=RAN(NSEED)
      IF( U .LE. R ) GO TO 225
      SUM=R
      DO 215 J=1,N
      I=J-1
      AN=N-I
      AI=I+1
      R=AN/AI*(P2/(1.-P2))*R
      SUM=SUM+R
      IF( U .LE. SUM ) GO TO 220
  215 CONTINUE
      DATA(IX)=N
      GO TO 230
  220 DATA(IX)=J
      GO TO 230
  225 DATA(IX)=0.
  230 IX=NX(IX)
      GO TO 500
C  IF 'N' IS LARGE (GREATER THAN 50) EXECUTE FOLLOWING INSTRUCTIONS FOR THE
C  BINOMIAL DISTRIBUTION
  235 N=P1
      DO 245 II=1,NVAL
      KVAL=0
      DO 240 I=1,N
      U=RAN(NSEED)
      IF( U .GT. P2 ) GO TO 240
      KVAL=KVAL+1
  240 CONTINUE
      DATA(IX)=KVAL
  245 IX=NX(IX)
      GO TO 500
C  SET THE ARRAY TO RANDOM NUMBERS DRAWN FROM THE SPECIFIED NORMAL DISTRIBUTION
  250 DO 260 II=1,NVAL
      SUM=0.
      DO 255 I=1,12
  255 SUM=SUM+RAN(NSEED)
      DATA(IX)=(SUM-6.)*P2+P1
  260 IX=NX(IX)
      GO TO 500
C  CHECK THE VALIDITY OF THE INSTRUCTION
  270 IF( KNTR .GE. 3 ) GO TO 280
  275 WRITE(6,12) K(KLP-1)
      GO TO 15
  280 IF( KNTR/2*2 .EQ. KNTR ) GO TO 275
      NPAIR=(KNTR+1)/2
      KB=KLP
C  DETERMINE THE VALUES OF THE POINTS ON THE CUMULITIVE PROBOBILITY DISTRIBUTION
      DO 330 J=1,NPAIR
      KFL=0
  285 N=0
      DO 290 I=1,10
```

```
      M=K(KB+I)
      IF( M .EQ. JCOMC .OR. M .EQ. JRPRC ) GO TO 295
  290 N=N+1
  291 WRITE(6,12) M
      GO TO 15
  295 IF( N .EQ. 0) GO TO 291
      VAL=0.
      NN=0
      DO 300 I=1,N
      IF( K(KB+I) .EQ. JPERC ) GO TO 305
  300 NN=NN+1
      VAL=INTEG(KB,N)
      GO TO 315
  305 KP=KB+I
      IF( NN .EQ. 0 ) GO TO 310
      VAL=INTEG(KB,NN)
      GO TO 311
  310 VAL=0.
  311 NN=N-NN-1
      IF( NN .EQ. 0 ) GO TO 315
      VAL=VAL+FRAC(KP,NN)
  315 IF( KFL .EQ. 0 )  GO TO 320
      VAR(J)=VAL
      KFL=0
      GO TO 325
  320 CDF(J)=VAL
      KFL=1
  325 KB=KB+N+1
      IF( KFL .EQ. 0 ) GO TO 330
      GO TO 285
  330 CONTINUE
C  CHECK THE VALIDITY OF CDF
      DO 335 I=1,NPAIR
      IF( CDF(I) .GT. 1. ) GO TO 340
  335 CONTINUE
      GO TO 345
  340 WRITE(6,342)
  342 FORMAT(1X, ' BAD CDF DATA  ' )
      GO TO 15
  345 IF( K(KLP-1) .EQ. JCC ) GO TO 350
      WRITE(6,12) K(KLP-1)
      GO TO 15
C  CREATE THE NEW ARRAY (SET ASIDE CELLS ETC. )
  350 NAME(KARY)=K(JPOS+3)
      LNGT(KARY)=NVAL
      CALL ASCELL(KARY)
      IX=KST(KARY)
C  SET THE ELEMENTS OF THE ARRAY TO RANDOM NUMBERS DRAWN FROM THE GIVEN
C  CUMULATIVE PROBOBILITY DISTRIBUTION
      DO 365 J=1,NVAL
      U=RAN(NSEED)
      DO 355 I=2,NPAIR
      IF( U .LE. CDF(I) ) GO TO 360
  355 CONTINUE
      GO TO 340
  360 A=(VAR(I)-VAR(I-1))/(CDF(I)-CDF(I-1))
```

```
      DATA(IX)=(U-CDF(I-1))*A+VAR(I-1)
  365 IX=NX(IX)
C  SET THE POINTER AND CALL SUB. INTERP TO DECODE THE NEXT INSTRUCTION
  500 JPOS=KRP+1
      CALL INTERP
      END
```

SUBROUTINE BAD

Subroutine BAD accomplishes the following functions:

- Execution is terminated if the program is processed and the pointer goes beyond the end of the user's program.
- Unidentified instructions or comments are skipped. The beginning of the next instruction is located and subroutine INTERP is called to decode this instruction.
- If an asterisk is found in the current position of the program, subroutine INPUT is called to take another instruction in a conversational environment.

```
      SUBROUTINE BAD
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C     COMMENTS AND ILLEGAL INSTRUCTIONS ARE SKIPPED, CONTROL IS GIVEN TO
C     SUBROUTINE INTERP WHENEVER A SEMICOLON IS FOUND IN K ARRAY
      WRITE(6,2)
    2 FORMAT(//)
   10 IF( K(JPOS) .EQ. JASTC ) GO TO 30
      IF( K(JPOS) .EQ. JSEMC ) CALL INTERP
      JPOS=JPOS+1
      IF( JPOS .GT. KEND ) GO TO 20
      GO TO 10
   30 JPOS=2
      CALL INPUT
   20 STOP 111
      END
```

BLOCK DATA

The variables which are used by most of the routines are put into block commons. Definitions of the letters of the alphabet, numerals, special symbols and the constants which do not change over the course of an execution are made in this routine.

```
BLOCK DATA
COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
COMMON/BLK5/ JOC,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
DATA JAC,JBC,JCC,JDC,JEC,JFC,JGC/ 'A','B','C','D','E','F','G' /
DATA JHC,JIC,JJC,JKC,JLC,JMC,JNC/ 'H','I','J','K','L','M','N' /
DATA JOC,JPC,JQC,JRC,JSC,JTC,JUC/ 'O','P','Q','R','S','T','U' /
DATA JVC,JWC,JXC,JYC,JZC,JOC,J1C/ 'V','W','X','Y','Z','0','1' /
DATA J2C,J3C,J4C,J5C,J6C,J7C,J8C/ '2','3','4','5','6','7','8' /
DATA J9C,JPLSC,JMINC,JPERC,JDIVC,JASTC /  '9','+','-','.','/','*'/
DATA JEQLC,JBLKC,JCOMC,JSEMC,NSEED,KWAN / '=',' ',',',' ',5179,0/
DATA JPOS,LWC,MPSZ,MDSZ,MASZ,KFLAG,LSEM/1,0,4000,2000,20,0,4000 /
DATA JLPRC,JRPRC / '(',')' /
END
```

SUBROUTINE DECODE (M,M1)

Subroutine DECODE determines the value of a decimal digit which is residing in the computer memory in alphanumeric code. The first argument gives the character and is defined by the calling routine. Subroutine DECODE stores the value of the digit (0-9) into the second argument and returns to the calling routine. If the character is not a decimal digit a value of ten is stored into the second argument.

```
      SUBROUTINE DECODE(M,M1)
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
C  IDENTIFY THE DECIMAL DIGIT AND RETURN ITS VALUE
C  IF THE CHARACTER IS NOT A DECIMAL DIGIT RETURN WITH A VALUE OF ZERO
      IF( M .EQ. J0C ) GO TO 10
      IF( M .EQ. J1C ) GO TO 11
      IF( M .EQ. J2C ) GO TO 12
      IF( M .EQ. J3C ) GO TO 13
      IF( M .EQ. J4C ) GO TO 14
      IF( M .EQ. J5C ) GO TO 15
      IF( M .EQ. J6C ) GO TO 16
      IF( M .EQ. J7C ) GO TO 17
      IF( M .EQ. J8C ) GO TO 18
      IF( M .EQ. J9C ) GO TO 19
      M1=10
      GO TO 20
   10 M1=0
      GO TO 20
   11 M1=1
      GO TO 20
   12 M1=2
      GO TO 20
   13 M1=3
      GO TO 20
   14 M1=4
      GO TO 20
   15 M1=5
      GO TO 20
   16 M1=6
      GO TO 20
   17 M1=7
      GO TO 20
   18 M1=8
      GO TO 20
   19 M1=9
   20 RETURN
      END
```

FUNCTION INTEG (KB,N)

Function INTEG evaluates the whole portion of a decimal number written in alphanumeric code. The first argument defines the position prior to the first decimal digit and the second argument gives the number of decimal digits before the decimal point. Both arguments must be defined by the calling routine. The value of the number is stored in the function name.

If there is a character other than a decimal digit within the specified field, the evaluation is stopped, an error message 'BAD DATA' is printed along with the illegal character and subroutine BAD is called to locate the next instruction in the user's program.

The maximum number of decimal digits allowed in a number is ten, and the arguments of the function are integers.

```
      FUNCTION INTEG(KB,N)
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      DIMENSION NUM(10)
C  FUNCTION INTEG(KB,N) DETERMINES THE VALUE OF THE WHOLE PORTION OF A NUMBER
C  KB DEFINES THE LOCATION PRIOR TO THE FIRST DECIMAL DIGIT
C  N DEFINES THE NUMBER OF DECIMAL DIGITS
      DO 10 I=1,N
      M=K(KB+I)
      CALL DECODE(M,MM)
      IF( MM .GT. 9 ) GO TO 20
   10 NUM(I)=MM
      INTEG=0
      DO 15 I=1,N
   15 INTEG=INTEG+NUM(I)*10**(N-I)
C  THE VALUE IS RETURNED TO THE CALLING ROUTINE
      RETURN
C  IF THERE IS A CHARACTER OTHER THAN DECIMAL DIGITS IN THE FIELD AN ERROR
C  MESSAGE WILL BE GIVEN AND THE INSTRUCTION WILL BE SKIPPED BY CALLING SUB. BAD
   20 WRITE(6,22) M
   22 FORMAT(1X, ' BAD DATA  ',A1 /)
      CALL BAD
      END
```

FUNCTION FRAC (KB,N)

Function FRAC evaluates the fractional portion of a decimal number written in alphanumeric code. The first argument gives the location of the decimal point and the second argument defines the number of decimal digits following the decimal point. Both arguments must be defined by the the calling routine. The value of the number is returned under the function name after the completion of the evaluation.

If there is an illegal character (any character other than a decimal digit) within the field an error message 'BAD DATA' is printed along with the illegal character, and subroutine BAD is called to locate the next instruction in the user's program.

Both arguments must be integers and the number of decimal digits in the field should not exceed ten.

```
      FUNCTION FRAC(KB,N)
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      DIMENSION NUM(10)
C  FUNCTION FRAC(KB,N) DETERMINES THE FRACTIONAL PORTION OF A NUMBER
C  KB DEFINES THE LOCATION OF THE DECIMAL POINT
C  N DEFINES THE NUMBER OF DECIMAL DIGITS
      DO 10 I=1,N
      M=K(KB+I)
      CALL DECODE(M,MM)
      IF( MM .GT. 9 ) GO TO 20
   10 NUM(I)=MM
      FRAC=0.
      DO 15 I=1,N
   15 FRAC=FRAC+NUM(I)/10. **I
      RETURN
C  IF THERE IS AN ILLEGAL CHARACTER IN THE FIELD SUB. BAD IS CALLED AND
C  INSTRUCTION IS SKIPPED
   20 WRITE(6,22) M
   22 FORMAT(1X, ' BAD DATA  ',A1 /)
      CALL BAD
      END
```

FUNCTION RAN (NSEED)

Function RAN generates and returns a random number between zero and one. The argument NSEED is changed by multiplying it by 65539 before generating a random number. Function DABS and the constant $2^{31} - 1$ (214748364.7 D + 1) are used in the process. If NSEED ever goes to zero it is set to 12345. The constant $2^{31} - 1$ must be changed for machines with a word size other than 32 bits.

```
      FUNCTION RAN(NSEED)
C  FUNCTION RAN  GENERATES AND RETURNS A RANDOM NUMBER BETWEEN 0. AND 1.
      RAN=DABS(NSEED/214748364.7D+1 )
   10 NSEED=NSEED*65539
      IF( NSEED .NE. 0 ) RETURN
      NSEED=12345
      GO TO 10
      END
```

SUBROUTINE DESTRY (ID)

Subroutine DESTRY erases an array from the memory and makes the cells used by that array available for future use. The argument gives the position of the variable in the NAME vector and must be defined by the calling routine. All the information about the variable is destroyed including its name.

```
      SUBROUTINE DESTRY(ID)
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C   SUBROUTINE DESTRY ERASES AN ARRAY FROM THE MEMORY
C   ID DEFINES THE LOCATION OF THE ARRAY  IN NAME LIST
C   THE LENGTH OF THE ARRAY MUST PREVIOUSLY BE DEFINED IN CALLING ROUTINE
      IX=KST(ID)
      KST(ID)=0
      NAME(ID)=0
      AVRG(ID)=JPERC
      STND(ID)=JPERC
      LL=LNGT(ID)
      LNGT(ID)=0
      IF( LL .LT. 2 ) GO TO 20
C   THE CELLS BELONGING TO THE DESTROYED ARRAY ARE MADE AVAILABLE BY SETTING
C   NX ARRAY TO PERIODS
      DO 10 I=2,LL
      IXN=NX(IX)
      NX(IX)=JPERC
   10 IX=IXN
   20 NX(IX)=JPERC
      RETURN
      END
```

SUBROUTINE CKNAME (NM,ID)

Subroutine CKNAME searches the NAME array for the name of a variable. The name of the variable is given by the first parameter and must be defined by the calling routine. If the name is found in the list, its position is stored in the second argument. If the name is not in the list, a value of zero is stored in the second argument before returning to the calling routine. The arguments must be in integer mode.

```
      SUBROUTINE CKNAME(NM,ID)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
C   SUBROUTINE CKNAME LOCATES A VARIABLE IN NAME LIST
C   FIRST ARGUMENT 'NM' DEFINES THE NAME OF THE VARIABLE AND MUST BE FURNISHED
C   BY THE CALLING ROUTINE
C   SECOND ARGUMENT 'ID' DEFINES THE LOCATION OF THE VARIABLE IN NAME ARRAY
C   IF THE NAME IS NOT IN THE LIST A VALUE OF ZERO IS RETURNED FOR THE SECOND
C   ARGUMENT
      DO 10 I=1,MASZ
      IF( NAME(I) .EQ. NM ) GO TO 20
   10 CONTINUE
      ID=0
      GO TO 30
   20 ID=I
   30 RETURN
      END
```

SUBROUTINE ASNAME (ID)

Subroutine ASNAME looks for an empty cell in the NAME array. If an empty cell is found, its position is stored in the argument. If there are no empty cells in the list a value of zero is stored in the argument before returning to the calling routine. The argument must be an integer variable.

```
      SUBROUTINE ASNAME(ID)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
C  SUBROUTINE ASNAME LOCATES AN EMPTY CELL IN NAME ARRAY AND RETURNS IT AS THE
C  VALUE OF THE ARGUMENT TO THE CALLING ROUTINE
C  IF THERE ARE NO EMPTY CELLS IN NAME ARRAY A VALUE OF ZERO IS RETURNED
      DO 10 I=1,MASZ
      IF( NAME(I) .EQ. 0 ) GO TO 20
   10 CONTINUE
      ID=0
      GO TO 30
   20 ID=I
   30 RETURN
      END
```

SUBROUTINE ASCELL (ID)

Subroutine ASCELL allocates empty cells to a variable in the DATA array. The argument gives the position of the variable in the NAME vector and must be defined by the calling routine. The length of the array, LNGT(ID), (the number of cells to be allocated) has to be defined and the availability of the required number of empty cells in DATA array must be checked before calling this routine.

```
      SUBROUTINE ASCELL(ID)
      COMMON/BLK1/ L(80),K(4000),DATA(2000),NX(2000)
      COMMON/BLK2/ NAME(20),KST(20),LNGT(20),AVRG(20),STND(20)
      COMMON/BLK3/ JAC,JBC,JCC,JDC,JEC,JFC,JGC,JHC,JIC,JJC,JKC,JLC,JMC
      COMMON/BLK4/ JNC,JOC,JPC,JQC,JRC,JSC,JTC,JUC,JVC,JWC,JXC,JYC,JZC
      COMMON/BLK5/ J0C,J1C,J2C,J3C,J4C,J5C,J6C,J7C,J8C,J9C
      COMMON/BLK6/ JPLSC,JMINC,JPERC,JDIVC,JASTC,JEQLC,JBLKC,JCOMC,JSEMC
      COMMON/BLK7/ JPOS,KEND,JR,MPSZ,MDSZ,MASZ,NSEED,KWAN,KFLAG,LCLM
      COMMON/BLK8/ LSEM,LWC,JLPRC,JRPRC
C   SUBROUTINE ASCELL ASSIGNS CELLS IN DATA ARRAY FOR A SPECIFIED VARIABLE
C   THE ARGUMENT 'ID' AND THE LENGTH OF THE ARRAY MUST BE DEFINED BEFORE CALLING
C   AVAILIBILITY OF ENOUGH EMPTY CELLS MUST BE CHECKED  BEFORE CALLING
      IX=LWC+1
   10 IF( IX .GT. MDSZ ) IX=1
      IF( NX(IX) .EQ. JPERC ) GO TO 20
      IX=IX+1
      GO TO 10
   20 KST(ID)=IX
      LWC=IX
      LL=LNGT(ID)
      IF( LL .EQ. 1 ) GO TO 60
      DO 50 I=2,LL
   30 IX=IX+1
      IF( IX .GT. MDSZ ) IX=1
      IF( NX(IX) .EQ. JPERC ) GO TO 40
      GO TO 30
   40 NX(LWC)=IX
   50 LWC=IX
   60 NX(LWC)=JASTC
      RETURN
      END
```

REFERENCES

1. Donaghey, Charles E. and Ozkul, Osman S., STIL USERS MANUAL, Technical report no. RE 8-69, Office of Naval Research Publication, August, 1969.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* University of Houston	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP Unclassified

3. REPORT TITLE

Stil Systems Manual

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

Charles E. Donaghey
Osman S. Ozkul

6. REPORT DATE	7a. TOTAL NO. OF PAGES 114	7b. NO. OF REFS 1
8a. CONTRACT OR GRANT NO. N00014-68-A-0151 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) RE 12-69 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

Reproduction in whole or in part is permitted for any purpose of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

There are an abundance of statistical programs and subroutines available to the computer user. However, for the occasional computer user, or the beginning statistics student, the use of these programs and subroutines can prove to be quite complex. For this reason STIL (STatistical Interpretive Language) has been developed. This language allows a user to quickly and easily write programs that solve a moderate range of statistical and probability problems. This manual lists the STIL interpreter and describes how the system operates. The interpreter consists of a main program and 33 sub-programs all written in FORTRAN IV.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

UNCLASSIFIED

Security Classification

A-31408

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Statistics						
Probability						
Language						
Interpreter						
FORTRAN						
Computers						
System						

DD FORM 1 NOV 65 1473 (BACK)

S/N 0101-807-6821

Unclassified

Security Classification

A-31409