Commands:
#
adr$b to set the next unused breakpoint (from 1 to 8) at location 'adr'.
#
x,,adr$B to set the next unused breakpoint which will automatically type out the contents of location x when the breakpoint is reached.
#
Ø$nB to remove specific breakpoint n.
#
$B to remove all breakpoints.
#
$P to proceed from a breakpoint - that is, after stopping at a breakpoint (and possibly examining cells, setting other breakpoints , removing breakpoints, etc.), $P causes IDDT to restart the program with the instruction that was broken.
#
n$P to proceed from the current breakpoint, and not stop at that breakpoint the next 'n' times it is reached, but to then break on the n+1'st time the breakpoint is reached. Note that unless the number n contains an 8 or a 9 digit or a decimal point that it will be interpreted as octal. Thus, 2Ø$P means to proceed past the current breakpoint 2Ø octal = 16 decimal times, while 2Ø.$P means 2Ø decimal.
#
adr$G to start the program at location adr. after a breakpoint it is usually safe to restart the program at a different place (instead of proceeding) if the new statement is labelled with a FORTRAN statement number. Starting within a statement will seldom be desirable and starting at an unnumbered statement may occasionally fail if a previous statement is needed to initialize some accumulator or temporary. An example of starting at a new numbered statement is 11ØP$G.
#
Examples of Use of the Commands with Reference to the Sample Program Above:
#
1. @IDDT
#
;Yank file: PROG.SAV$ ret
#
MAIN.$: 11ØP$B $$G
#
The user GETs the program, enters IDDT, establishes the main program as the context for names, sets a breakpoint at FORTRAN statement 11Ø (equivalent to label 2M in the IDDT output), and **starts** the program at the beginning.

The program will first request the input of the A array and B, and after the user has typed in the data, will proceed to the breakpoint and stop by typing:

```
$1B>>2M
```

2. At this point the user may want to insert a breakpoint at the statement 'B=1.2', but this statement does not have a FORTRAN statement number so the location of its first instruction is not known. The easiest course of action is to start typing out instructions starting at 110P (the nearest preceding statement with a number) until the start of the 'B=1.2' statement is recognized. The type-out is done by first typing '110P/' and then continuing with line feeds (alternatively, the user could start at 120P and go backwards with successive up arrows = shift-N). Even though the user may not know PDP-10 assembly language, it is fairly easy to recognize the statement boundaries because the instructions contains references to the user's FORTRAN variables and statement numbers (but only if they were requested during loading with the /S switch as described above in the section Preparation for Debugging). In this case, the user may recognize the 'JRST 120P' as the 'GO TO 120' at the end of the previous statement, or the 'MOVEM 2,B' as the storage of B in the assignment statement to be broken. We see that the instruction to be broken is at 2M+3 and so the IDDT command 2M+3$B will insert the breakpoint as desired.

3. It is frequently useful to stop the program as soon as it has entered a subroutine. This may be conveniently done by:

```
SUB$:     1M$B
```

The first command establishes the context for names as that of the subprogram (here named SUB), and the second command inserts the breakpoint at the first executable statement of the subprogram at label 1M (but see note 5 above for exceptions to the 1M label). This will be after the parameters have been set up during subprogram initialization, and so when the breakpoint is reached, the subprogram's parameters may be examined.

4. Although it is possible to set and remove specific breakpoints, the novice may find it easy to start by simply setting the next unused breakpoint (with the adr$B command), and when all eight breakpoints have been used (IDDT will give a message when the user attempts to set a ninth breakpoint), to remove them all with the $B command and then reset any that are still desirable.

FORTRAN-IV Library Subroutines.

MODIFICATIONS TO THE FORTRAN LIBRARY

1. IFILE and OFILE include the addition of an optional extension. The formats for these subroutines now are:

CALL IFILE(device,filename,extension)
CALL OFILE(device,filename,extension)

where device = file device number (required)

filename = ASCII filename of up to 5 ASCII characters (required)

extension = ASCII extension of up to 3 ASCII characters. If set to Ø for IFILE, system will search for DAT extension first and then blank extension. If set to Ø for OFILE, system will set extension to DAT. (optional only if a user-code is not to be specified.)

2. The format for DEFINE has been modified from

CALL DEFINE FILE(U,S,V,F,PJ,PG,CODE)

to

CALL DEFINE FILE(U,S,V,F,USER,CODE)

where USER is Ø and only required if it is to be followed by
CODE which is an optional protection code field.

3. The CalComp Plotter routines have been modified for operation on the TENEX CalComp Plotter, Model 665 which uses 12" wide paper and has 2 step sizes.

CalComp subroutines enable plotting of lines, curves, text, and graphs by calls from FORTRAN programs. The subroutines also control pen position, labeling of plots, scaling, and data identification. Detailed subroutine discussions follow standard FORTRAN notation conventions so that variables beginning with I,J,K.L,M, or N, are fixed point while all others are floating point.

OPEN PLT: causes about 6" of paper to be slewed. The system routines maintain a "fence" that prevents a user from going off the edge of the paper. The pen is assumed to be centered in the Y direction when PLT: is opened, and users should leave the pen in a similar place when PLT: is

closed.

The plotter accepts 5-bit bytes to direct its motion as shown below:

		+y				
7	33	Ø	32	1		PEN UP=11
31	27	2Ø	21	3Ø		PEN DOWN=12
6	26	.	22	2	+X	OTHERS ARE NOP'S
35	25	24	23	34		
5	37	4	36	3		

Moving in the +X direction slews paper on to the floor. The +Y direction is arranged so that the coordinate system is right handed.

PLOTTING CONTROL

PLOTS

CALL PLOTS(I) — OPENS the "Hardware" file PLT: If successful, I is set to Ø (a logical TRUE). Otherwise, I is set to -1.

Output goes directly to the plotter.

CALL PLOTS(I, 'FILE NAME') — Uses the named file instead of PLT: Any legal TENEX file name will do. I is returned as described above.

CALL PLOTS(I,Ø) — As above but the file name is requested from the Teletype similar to what TECO does with ;U$.

CALL PLOTS ('FILE NAME') — No value returned

In addition each of the above has an integer function counterpart which returns the success/fail value to IPLOTS as well as the argument I.

NOTE: Call PLOTS before any other plotting subroutine and use it once and only once in a plotting program. It prepares the system to do plotting, setting its current position as the origin (Ø,Ø) and raising the pen from the paper.

PLOT

CALL PLOT (X,Y,IPEN)

X,Y

moves the pen in a straight line from its current position to the position specified by the floating point coordinates X,Y, in inches.

IPEN

specifies pen position during motion:
1 - unchanged from previous action
2 - lowers pen before motion so a line is drawn
3 - raises pen before motion
-1,-2,-3 - same action as for positive values, except that after motion is completed, the pen position is taken as a new origin.

Call PLOT after completing all plotting in order to move the pen past the last plot.

WHERE

CALL WHERE (X,Y)

X,Y set to the current position of the pen, in inches, and returns these values to the program.

PLTEND

CALL PLTEND

Call PLTEND after completing all plotting to close the plotter output file.

LABELING

SYMBOL

```
CALL SYMBOL (X,Y,HEIGHT,CHAR,THETA,NS)
```

X,Y	coordinates of the lower lefthand corner of the first alphanumeric character.
HEIGHT	height of the character(s) in floating point inches
CHAR	an array containing a Hollerith string of letters, numbers of special characters.
THETA	direction, in degrees, of the base-line on which the text is plotted; normal orientation, X direction, is Ø.Ø.
NS	length of the string CHAR in characters.

NUMBER

```
CALL NUMBER (X,Y,HEIGHT,FPN,THETA,NN)
```

FPN	floating point number to be plotted; assumes the ASCII character set using the digits Ø through 9, "-" and ".".
NN	the number of decimal digits to be plotted to the right of the decimal point; a negative value suppresses the decimal point, printing only the integer part of the number.

SCALING

SCALE

```
CALL SCALE (X,N,AL,XMIN,DX)
```

X	a real vector containing the data to be plotted.
N	length or number of points in

	the array
AL	axis length in floating point inches
XMIN	set to the minimum value found in X, truncated to be multiple of DX
DX	set to the X-increment, a "reasonable" interval allowing X to be plotted in S inches and still be read, equal to A*1Ø**B, where A and B are integers and A is either 1,2,4,5 or 8.

Scans the data to be plotted for the maximum and minimum values, adjusting these to optimize the plot and determine scale values that are easy to interpolate between divisions.

DATA IDENTIFICATION

AXIS

```
CALL AXIS (XO,YO,TITLE,NC,AL,THETA,XMIN,DX)
```

XO,YO	the coordinates of the starting point of the axis, relative to the current origin.
TITLE	name of an alphabetic array for the axis title, may be of the form NH THIS IS AN N CHARACTER TITLE
NC	number of characters in the title; standard position is the counterclock-wise side of the axis, a negative NC places the title on the clockwise side

```
NORMAL      NEGATIVE NC
I           I
I           I
I           I
I           I
I TITLE     I----------
I--------     TITLE
```

AL	length of the axis in inches

THETA	angle, in degrees, of the axis; standard X-axis is 0, standard Y-axis is 90
XMIN	minimum value on the axis, determined by SCALE
DX	increment size, determined by SCALE

Draws a line with tic marks and scale values at one-inch intervals labeling the axis.

LINE OR CURVE GENERATION

LINE

```
CALL LINE (X,Y,N,K)
```

X	array of data for standard X direction
Y	array of data for standard Y direction
N	number of points to be plotted
K	skip factor, usually set to 1; for K>1, every Kth point is plotted.

Draws a line (or curve) through the specified points.

ADDITIONS TO THE FORTRAN LIBRARY

1. ACPT -- accept an arbitrary number of characters from the controlling teletype, float each ASCII code character and dump into an equal number of variables. The calling sequence is:

   CALL ACPT(CH1,CH2,...,CHn) where CH1,CH2,...,CHn are the variables into which the results are to be dumped.

2. ATN -- Arctangent Function which may be called with one or two floating point arguments. The subroutine decides whether it has one or two arguments and then calls the appropriate routine (ATAN or ATAN2). The format is:

   ATN(X) or ATN(X,Y)

3. COT -- Cotangent Function which computes COT(X) by the simple expedient of COT(X)=COS(X)/SIN(X). The format is:

   COT(X)

4. CPUTIM -- Subroutine to return the job CPU time in seconds. The calling sequence is:

   CALL CPUTIM(X)

5. CRAND -- Complex Random Number Generator which generates a complex random number between two given complex limits. The format is:

   CRAND((A,B),(C,D)) where (A,B) is the lower limit and (C,D) is the upper limit

6. DRAND -- Double Precision Random Number Generator which generates a double precision random number between two given double precision limits. The format is:

   DRAND(A,B) where A is the lower limit and B is the upper limit.

7. DTE -- Subroutine to return the real result of the current date in the form MMDDYY. The calling sequence is:

   CALL DTE(X)

8. FILES -- A collection of file handling utility subroutines which include DELETE, FILOOK, PROTEC, RENAM, UNPROT.

   a. DELETE -- perform a deletion on a specified filename.

      The calling sequence is:

      CALL DELETE(NAME,EXT,IANS)

      where:

      NAME -- file name of up to 5 characters
      EXT -- file extension of up to 3 characters
      IANS -- indicates the result of the deletion
          =Ø -- Deletion successful
          =1 -- No such file
          =2 -- File open for output
          =3 -- Error in filename
          =4 -- Error during LOOKUP
          =5 -- Error in deletion

   b. FILOOK -- perform a LOOKUP on a specified filename.

The calling sequence is:

```
CALL FILOOK(NAME,EXT,IANS,IDCODE)
```

where:

```
NAME -- filename of up to 5 characters
EXT -- file extension of up to 3 characters
IANS -- indicates the result of the LOOKUP
    =Ø -- LOOKUP successful
    =1 -- No such file
    =2 -- File read-protected or open for output
    =3 -- Error in filename
    =4 -- Error during LOOKUP
IDCODE -- Optional 6-character ASCII IDcode
```

c. PROTEC -- set the protection code on a specified filename so that only the owner can access, modify or delete the file. The calling sequence is:

```
CALL PROTEC(NAME,EXT,IANS)
```

where:

```
NAME -- filename of up to 5 characters
EXT -- file extension of up to 3 characters
IANS -- indicates the result of setting the protection
    =Ø -- Protection set successfully
    =1 -- No such file
    =2 -- File open for output
    =3 -- Error in filename
    =4 -- Error during LOOKUP
    =5 -- Error in setting protection
```

d. RENAM -- performs a RENAME operation. The calling sequence

```
CALL RENAM(NEWNAM,NEWEXT,OLDNAM,OLDEXT,IANS)
```

where:

```
NEWNAM,NEWEXT -- filename and extension of resultant
file
OLDNAM,OLDEXT -- filename and extension of original
 file
IANS -- indicates the result of the RENAME
    =Ø -- RENAME successful
    =1 -- No such old file
    =2 -- Old file read-protected or open for
          output
    =3 -- Error in filename (old or new)
```

=4 -- New name in use
=5 -- Error during RENAME

e. UNPROT -- set the protection code on a specified filename so that anyone can access the file, but only the owner can modify or delete it. The calling sequence is:

CALL UNPROT(NAME,EXT,IANS)

where:

NAME -- filename of up to 5 characters
EXT -- file extension of up to 3 characters
IANS -- indicates the result of unprotecting the file
=Ø -- File unprotected successfully
=1 -- No such file
=2 -- File open for output
=3 -- Error in filename
=4 -- Error during LOOKUP
=5 -- Error in unprotecting the file

9. FPLOT -- a collection of FORTRAN Subroutines for the Calcomp Plotter. All the subroutines use the FORTRAN PLOT Routine.

a. CIRCLE -- draws a circle, arc or spiral. The calling sequence is:

CALL CIRCL(X,Y,ANS,ANF,RS,RF,DI)

where:

X -- X-coordinate of the starting point
Y -- Y-coordinate of the starting point
ANS -- Starting angle of the radius vector relative to the X-axis
ANF -- Finishing angle of the radius vector relative to the X-axis
RS -- Length of the radius at ANS
RF -- Length of the radius at ANF
DI -- Either a Ø or 1; where Ø means a solid line and 1 a dashed line

NOTE: The user can cause many types of curves to be drawn by changing various parameters.

RS=RF draws a circle
RS<RF draws a spiral

If RS and RF span zero, a curlycue shaped figure is

drawn.

b. DASHL -- draw dashed lines connecting a series of data points. The calling sequence is:

CALL DASHL(X,Y,N,INC)

where:

X -- Name of the array containing X-coordinates of the points
Y -- Name of the array containing Y-coordinates of the points
N -- Total number of points in the X and Y arrays
INC -- Skip factor, i.e. set to 1 for all points, set to 2 for every second point.

NOTE: If using INC greater than 1 be sure to start at correct point. That is the arguments X and Y should be X(INC), Y(INC) respectively.

c. DASHP -- draw a dashed line to a specified point. The calling sequence is:

CALL DASHP(X,Y,DASH)

where:

X -- X-coordinate of point
Y -- Y-coordinate of point
DASH -- Length of dash (also space length). If distance is less than dash length, DASH is automatically reset to 1/2 the distance.

d. ELIPS -- draw an ellipse or elliptical arc. The calling sequence is:

CALL ELIPS(X,Y,A,B,ANG,ANS,ANF,IPEN)

where:

X -- X-coordinate of the starting point
Y -- Y-coordinate of the starting point
A -- Length of semi-major axis
B -- Length of semi-minor axis
ANG -- Angle of semi-major axis with respect to the X-axis
ANS -- Starting angle of the radius vector relative to the X-axis
ANF -- Finishing angle of the radius vector

relative to the X-axis
IPEN -- Pen position for first move

3=Pen up
2=Pen down

e. GRID -- draw a linear grid. The calling sequence is:

CALL GRID(X,Y,DX,DY,NXS,NYS)

where:

X -- X-coordinate of lower left hand corner (origin)
Y -- Y-coordinate of lower left hand corner
DX -- Distance between lines parallel to Y-axis
DY -- Distance between lines parallel to X-axis
NXS -- Number of lines parallel to Y-axis
NYS -- Number of lines parallel to X-axis

NOTE: Two calls to GRID can be used to emphasize major division lines. Example: First call with spacing of .1 inches, second call with spacing every inch with (X,Y) offset slightly from that of first call.

f. POLY -- draw an equilateral polygon. The calling sequence is:

CALL POLY(X,Y,SL,SN,ANG) where:

X -- X-coordinate of lower right hand corner of polygon
Y -- Y-coordinate of lower right hand corner of polygon
SL -- Side length
SN -- Number of sides
ANG -- Orientation angle of base with respect to X-axis and point (X,Y)

g. RECT -- draw a rectangle. The calling sequence is:

CALL RECT(X,Y,XW,YH,ANGLE,IPEN)

where:

X -- X-coordinate of the starting point
Y -- Y-coordinate of the starting point
XW -- Width of the rectangle
YH -- Height of the rectangle
ANGLE -- Angle
IPEN -- Pen position of first move

3=Pen up
2=Pen down

10. GETRAN -- get both halves of random number routine which gets the origin of the random number sequence and is used in conjunction with ZETRAN to allow a "random" number sequence to be repeated. The first random number generated after a call to GETRAN will be the same as the arguments. The calling sequence is:

    CALL GETRAN(X,Y) where X is the location to store the high part of the random number and Y is where to put the low part of the random number.

11. GRAND -- Gaussian Random Number Generator which uses the Gaussian method to generate a random number. The format is:

    GRAND(X,Y) where X is the mean and Y is the variance.

12. IRAND -- Integer Random Number Generator which gets a random integer uniformly distributed between two limits. The format is:

    IRAND(I,J) where I is the lower limit and J is the upper limit.

13. KJOB -- FORTRAN callable subroutine to logout the current job. The calling sequence is:

    CALL KJOB

14. RAND -- Random Bits Random Number Generator which generates a 36-bit integer random number. The result is returned in AC0. The calling sequence is:

    CALL RAND

15. RANDOM -- Random Real Number Generator which generates a random real number uniform between two limits. The format is:

    RANDOM(A,B) where A is the lower limit and B is the upper limit.

16. REENTR -- FORTRAN callable subroutine which sets a reentry point within a FORTRAN program to which control should be returned by the monitor command REENTER after an interruption by a Control-C or an error. The calling sequence is:

    CALL REENTR

which sets the reentry point at the next sequential FORTRAN statement.

17. RN -- Random Real Number Generator which generates a random real number between 1.Ø and 131Ø71.Ø. The calling sequence is:

    CALL RN(X)

18. SEC -- Subroutine to return the real result of the number of seconds since midnight. The calling sequence is:

    CALL SEC(X)

19. SEND -- type out an arbitrary number of characters on the controlling teletypewriter which have been converted to ASCII teletype code from an equal number of fixed and/or floating point arguments. Floating point arguments are first converted to fixed. The calling sequence is:

    CALL SEND (CH1,CH2,...,CHn) where CH1,CH2,...,CHn are the variables containing the arguments

2Ø. SETDEV -- makes an entry in the FORTRAN Device Table so that references to device number IDEV refer to device name DEV. The calling sequence is:

    CALL SETDEV('DEV',IDEV)

21. SHIFT -- shift a word a given number of bits. An optional Mask may also be specified. (SHIFT is convenient for character manipulation in FORTRAN.)

    The calling sequence is:

    CALL SHIFT(ARG,ARG2,ARG3,ARG4)

    where:

    ARG = Location of word to be shifted
    ARG2 = Shift factor and direction of shift
           (A "+" indicates left and a "-" indicates right)
    ARG3 = Destination location for shifted word
    ARG4 = Optional Mask factor

22. SUBTMP -- contains the FORTRAN subroutines RDTMP and WRTMP
for reading and writing temporary files. The calling sequences are:

    CALL WRTMP(FILNAM,ADDR,LENGTH,IERROR)

CALL RDTMP(FILNAM,ADDR,LENGTH,IREAD,IERROR)

where:

FILNAM -- hollerith string or any variable containing the one to three character alphanumeric name of the TMP file

ADDR -- array name containing or receiving the contents of the TMP file

LENGTH -- number of elements in array to be transferred

IREAD -- optional return argument giving the actual length in words of the TMP file which was read. If the array is too small for the TMP file, as much of the file as can fit is transferred. If the file is shorter than the array, the rest of the array is filled with zeroes.

IERROR -- optional return argument giving the following error codes:

Ø -- no errors
1 -- FILNAM not alphanumeric
2 -- an argument was of the wrong type
3 -- length out of range
4 -- core had insufficient space and disk not available for TMP files
5 -- RDTMP: file not found
6 -- transmission error

23. TAN -- Tangent Function which computes TAN(X) by the simple expedient of TAN(X)=SIN(X)/COS(X). The format is:

TAN(X)

24. TIM -- Subroutine to return the real result of the time of day as HHMM. The calling sequence is:

CALL TIM(X)

25. ZETRAN -- sets the random number "initial value" and is used to set the origin of the random number sequence and in conjunction with GETRAN to allow a "random" number sequence to be repeated. The first random number generated after a call to GETRAN will be the same as the first random number generated after a call to ZETRAN with the same arguments. The calling sequence is:

CALL ZETRAN(X,Y) where X contains the high order part and Y contains the low order part

OTHER FORTRAN SUBROUTINES

See DECsystem10 FORTRAN-IV Manual for further information on subroutines.

FORTRAN Scientific Subroutine Package

The FORTRAN Scientific Subroutine Package is now available on the TENEX System. To incorporate routines from the Package into a FORTRAN Program, use the following command when loading with LOADER after the main program has been loaded and before the alt mode ($) key is struck:

```
SYS:SSP/L
```

This will invoke a search of the Scientific Subroutine Package. The alt mode ($) invokes a search of the Standard FORTRAN Library, and also terminates the loading procedure.

A complete description of all the subroutines is contained in the IBM application Program Manual entitled, "System/360 Scientific Subroutine Package, (360A-CM-03X) Version III, Programmer's Manual". The IBM Manual Number is H20-0205-4. Copies of the manual can be purchased from IBM and also from bookstores of many of the universities.

DATA SCREENING

TALLY - Totals, means, standard deviations, minimums, and maximums
BOUND - Selection of observations within bounds
SUBST - Subset selection from observation matrix
ABSNT - Detection of missing data
TAB1 - Tabulation of data (one variable)
TAB2 - Tabulation of data (two variables)
SUBMX - Building of subset matrix

CORRELATION AND REGRESSION

CORRE - Means, standard deviations, and correlations
MISR - Means, standard deviations, third and fourth moments, correlations, simple regression coefficients and their standard errors; considers that data may be missing
ORDER - Rearrangement of intercorrelations
MULTR - Multiple linear regression
GDATA - Data matrix generation for polynomial regression
STPRG - Stepwise multiple linear regression
PROBT - Probit analysis
CANOR - Canonical correlation

DESIGN ANALYSIS

AVDAT - Data storage allocation
AVCAL - Sigma and Delta operation
MEANQ - Mean square operation

DISCRIMINANT ANALYSIS

DMATX - Means and dispersion matrix
DISCR - Discriminant functions

TRACE - Cumulative percentage of eigenvalues
LOAD - Factor loading
VARMX - Varimax rotation

TIME SERIES

AUTO - Autocovariances
CROSS - Cross covariances
SMO - Application of filter coefficients (weights)
EXSMO - Triple exponential smoothing

NONPARAMETRIC STATISTICS

KOLMO - Kolmogorov-Smirnov one-sample test
KOLM2 - Kolmogorov-Smirnov two-sample test
SMIRN - Kolmogorov-Smirnov limiting distribution values
CHISQ - Chi-square test for contingency tables
KRANK - Kendall Rank correlation
MPAIR - Wilcoxin's signed ranks test
QTEST - Cochran Q-test
RANK - Rank observations
SIGNT - Sign test
SRANK - Spearman rank correlation
TIE - Calculation of ties in ranked observations
TWOAV - Friedman two-way analysis of variance statistic UTEST -
Mann-Whitney U-Test
WTEST - Kendall coefficient of concordance

GENERATION OF RANDOM VARIATES - DISTRIBUTION FUNCTIONS

NDTR - Normal Distribution function
BDTR - Beta distribution function
CDTR - Chi-square distribution function
NDTRI - Inverse of normal distribution function

ELEMENTARY STATISTICS AND MISCELLANY

MOMEN - First four moments
TTEST - Test on population means
BISER - Biserial correlation coefficient
PHI - PHI coefficient
POINT - Point-biserial correlation coefficient

TETRA - Tetrachoric correlation coefficient
SRATE - Survival rates

MATRICES: STORAGE

MCPY - Matrix copy
RCPY - Copy row of matrix into vector
CCPY - Copy column of matrix into vector
DCPY - Copy diagonal of matrix into vector
XCPY - Copy submatrix from given matrix
MSTR - Storage conversion
LOC - Location in compressed-stored matrix
CONVT - Single-precision/double-precision conversion
ARRAY - Vector storage/double-dimensioned storage conversion

MATRICES: OPERATIONS

GMADD - Add two general matrices
GMSUB - Subtract two general matrices
BMPRO - Product of two general matrices
GMTRA - Transpose of a general matrix
GTPRO - Transpose product of two general matrices
MADD - Add two matrices
MSUB - Subtract two matrices
MPRO - Matrix product (row into column)
MTRA - Transpose a matrix
TPRO - Transpose product
MATA - Transpose product of matrix by itself
SADD - Add scalar to matrix
SSUB - Subtract scalar from a matrix
SMPY - Matrix multiplied by a scalar
SDIV - Matrix divided by a scalar
SCLA - Matrix clear and add scalar
DCLA - Replace diagonal with scalar
RADD - Add row of one matrix to row of another matrix
CADD - Add column of one matrix to column of another matrix
SRMA - Scalar multiply row and add to another row
SCMA - Scalar multiply column and add to another column
RINT - Interchange two rows
CINT - Interchange two columns
RSUM - Sum the rows of a matrix
CSUM - Sum the columns of a matrix
RTAB - Tabulate the rows of a matrix
CTAB - Tabulate the columns of a matrix
RSRT - Sort matrix rows
CSRT - Sort matrix columns
RCUT - Partition by row
CCUT - Partition by column
RTIE - Adjoin two matrices by row
CTIE - Adjoin two matrices by column
MPRC,DMPRC - Permute rows or columns

MFUN - Matrix transformation by a function
RECP - Reciprocal function for MFUN

MATRICES: INVERSION, SYSTEMS OF LINEAR EQUATIONS & RELATED TOPICS

MINV - Matrix inversion
SINV, DSINV - Invert a symmetric positive definite matrix
SIMQ - Solution of simultaneous linear, algebraic equations
GELG, DGELG - System of general simultaneous linear equations by gauss elimination
RSLMC - Solution of simultaneous linear equations with iterative refinement
FACTR - Triangular factorization of a nonsingular matrix
MFGR,DMFGR - Matrix factorization and rank determination
GELS,DGELS - System of general simultaneous linear equations with symmetric coefficients
GELB,DGELB - System of general simultaneous linear equations with band structured coefficients
MTDS,DMTDS - Divide a matrix by a triangular matrix
MLSS,DMLSS - Solution of simultaneous linear equations with symmetric positive semidefinite matrix
MCHB,DMCHB - Triangular factorization of a symmetric positive definite band matrix
MFSS,DMFSS - Triangular factorization and rank determination of a symmetric positive semidefinite matrix
MFSD,DMFSD - Triangular factorization of a symmetric positive definite matrix
LLSQ,DLLSQ - Solution of linear least-squares problems

MATRICES: EIGENANALYSIS AND RELATED TOPICS

EIGEN - EIGENVALUES and EIGENVECTORS of a real, symmetric matrix
NROOT - Eigenvalues and eigenvectors of a special nonsymmetric matrix
ATEIG - Eigenvalues of a real almost triangular matrix
HSBG - Reduction of a real matrix to almost triangular form

POLYNOMIALS: OPERATIONS

PADD - Add two polynomials
PSUB - Subtract one polynomial from another
PMPY - Multiply two polynomials
PDIV - Divide one polynomial by another
PCLA - Replace one polynomial by another
PADDM - Multiply polynomial by constant and add to another polynomial
PVAL - Value of a polynomial

PVSUB - Substitute variable of polynomial
PILD - Evaluate polynomial and its first derivative
PDER - Derivative of a polynomial
PINT - Integral of a polynomial
PQSD - Quadratic synthetic division of a polynomial
PCLD - Complete linear synthetic division
PGCD - Greatest common divisor of two polynomials
PNORM - Normalize coefficient vector of polynomial
PECN,OPECN - Economization of a polynomial for symmetric range
PECS,DPECS - Economization of a polynomial for unsymmetric range

POLYNOMIALS: ROOTS

POLRT - Real and complex roots of a real polynomial
PRQD,DPRQD - Roots of a real polynomial by QD algorithm with displacement
PRBM,DPRBM - Roots of a real polynomial by Bairstow's algorithm
PQFB,DPQFB - Determine a quadratic factor of a real polynomial

POLYNOMIALS: SPECIAL TYPES

CNP,DCNP - Value of N(th) Chebyshev polynomial
CNPS,DCNPS - Value of series expansion in Chebyshev polynomials
TCNP,DTCNP - Transform series expansion in Chebyshev polynomials to a polynomial
CSP,DCSP - Value of N(th) shifted Chebyshev polynomial
CSPS,DCSPS - Value of series expansion in shifted Chebyshev polynomials
TCSP,DTCSP - Transform series expansion in shifted Chebyshev polynomials to a polynomial
HEP,DHEP - Value of hermite polynomial
HEPS,DHEPS - Value of series expansion in hermite polynomials
THEP,OTHEP - Transform series expansion in hermite polynomials to a polynomial
LAP,DLAP - Value of Laguerre polynomial
LAPS,DLAPS - Value of series expansion in Laguerre polynomials
TLAP,DTLAP - Transform series expansion in Laguerre polynomials to a polynomial
LEP,DLEP - Value of Legendre polynomial
LEPS,DLEPS - Value of series expansion in Legendre polynomials
TLEP,DTLEP - Transform a series expansion in Legendre polynomials to a polynomial

ROOTS OF NONLINEAR EQUATIONS

RTWI,DRTWI - Refine estimate of root by Wegstein's iteration
RTMI,DRTMI - Determine root within a range by Mueller's iteration
RTNI,DRTNI - Refine estimate of root by Newton's iteration

EXTREMUM OF FUNCTIONS

FMFP,DFMFP - Unconstrained minimum of a function of several variables -- Davidon method
FMCG,DFMCG - Unconstrained minimum of a function of several variables -- conjugate gradient method

PERMUTATIONS

PPRCN - Composition of permutations
PERM - Operations with permutations and transpositions

SEQUENCES: SUMS AND LIMITS

TEAS,DTEAS - Limit of a given sequence

INTERPOLATION, APPROXIMATION, AND SMOOTHING

ALI,DALI - Aitken-Lagrange Interpolation
AHI,DAHI - Aitken-Hermite Interpolation
ACFI,DACFI - Continued fraction interpolation
ATSG,DATSG - Table selection out of a general table
ATSM,DATSM - Table selection out of a monotonic table
ATSE,DATSE - Table selection out of an equidistant table
SG13,DSG13 - Local least-squares smoothing of tabulated functions

SE13,DSE13
SE15,DSE15
SE35,DSE35 - Local least-squares smoothing of equidistantly tabulated functions
APFS,DAPFS - Solve normal equations for least-squares fit
APCH,DAPCH - Least-squares polynomial approximation
ARAT,DARAT
FRAT,DFRAT - Rational least-squares approximation
APLL,DAPLL - Linear least-squares approximation
FORIF - Fourier analysis of a given function
FORIT - Fourier analysis of a tabulated function
HARM,DHARM - Complex three-dimensional fourier analysis
RHARM,DRHARM - Real one-dimensional Fourier Analysis
APMM,DAPMM - Linear Chebyshev approximation over a discrete range

NUMERICAL QUADRATURE

QTFG,DQTFG - Integration of monotonically tabulated function by trapezoidal rule
QTFE,DQTFE - Integration of equidistantly tabulated function by trapezoidal rule

QSF.DQSF - Integration of equidistantly tabulated function by Simpson's rule
QHFG,DQHFG - Integration of monotonically tabulated function with first derivative by Hermitian formula of first order
QHFE,DQHFE - Integration of equidistantly tabulated function with first derivative by Hermitian formula of first order
QHSG,DQHSK - Integration of monotonically tabulated function with first and second derivatives by Hermitian formula of first order
QHSE,DQHSE - Integration of equidistantly tabulated function with first and second derivatives by Hermetian formula of second order
QATR,DQATR - Integration of a given function by trapezoidal rule together with Romberg's extrapolation method
QG2,QG1Ø,DQG4-DQG32 - Integration of a given function by Gaussian Quadrature formulas
QL2-QL1Ø,DQL4-DQL32 - Integration of a given function by Gaussian-Laguerre quadrature formulas
QH2-QH1Ø,DQH8-DQH64 - Integration of a given function by Gaussian-Hermite quadrature formulas
QA2-QA1Ø,DQA4,DQA32 - Integration of a given function by associated Gaussian-Laguerre quadrature formulas

NUMERICAL DIFFERENTIATION

DGT3,DDGT3 - Differentiation of a tabulated function by parabolic interpolation
DET3,DDET3
DET5,DDET5 - Differentiation of an equidistantly tabulated function
DCAR,DDCAR - Derivative of a function at the center of an interval
DBAR,DDBAR - Derivative of a function at the border of an interval

ORDINARY DIFFERENTIAL EQUATIONS

RK1 - Solution of first-order differential equation by Runge-Kutta method
RK2 - Tabulated solution of first-order differential equation by Runge-Kutta method
RKGS,DRKGS - Solution of system of first-order ordinary differential equations with given initial values by the Runge-Kutta method
HPCG,DHPCG - Solution of general system of first-order ordinary differential equations with given initial values by Hamming's modified predictor-corrector method
HPCL,DHPCL - Solution of linear system of first-order ordinary differential equations with given initial values by Hamming's modified predictor-corrector method
LBVP,DLBVP - Solution of system of linear first-order ordinary

differential equations with linear boundary conditions by method of adjoint equations

SPECIAL FUNCTIONS

GMMMA - Gamma function
DLGAM - Log of Gamma function
BESJ - J Bessel function
BESY - Y Bessel function
BESI - I Bessel function I(Ø)
BESK - K Bessel function
EXPI - Exponential integral
SICI - Sine cosine integral
CS - Fresnel integrals
CEL1,DCEL1 - Complete elliptic integral of the first kind
CEL2,DCEL2 - Complete elliptic integral of the second kind
ELI1,DELI1 - Generalized elliptic integral of the first kind
ELI2,DELI2 - Generalized elliptic integral of the second kind
JELF,DJELF - Jacobian elliptic functions

EISPACK

EISPACK is the FORTRAN Eigensystem Subroutine Package which was developed at Argonne National Laboratory and is now available on BBN/TENEX. To incorporate routines from the package into a FORTRAN Program, use the following command when loading with LOADER after the main program has been loaded and before the altmode ($) key is struck:

SYS:EISPACK/L

This will invoke a search of the Eigensystem Subroutine Package. The altmode ($) invokes a search of the Standard FORTRAN Library, and also terminates the loading procedure.

A list of the routines in the Eigensystem Subroutine Package follows.

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

BALANC BALANCES A REAL GENERAL MATRIX.

BALBAK BACK TRANSFORMS THE EIGENVECTORS OF THAT REAL MATRIX TRANSFORMED BY BALANC.

CBAL BALANCES A COMPLEX GENERAL MATRIX.

CBABK2 BACK TRANSFORMS THE EIGENVECTORS OF THAT COMPLEX MATRIX TRANSFORMED BY CBAL.

ELMHES REDUCES A REAL GENERAL MATRIX TO UPPER HESSENBERG FORM

```
#           USING ELEMENTARY TRANSFORMATIONS.
#
# ELMBAK    BACK TRANSFORMS THE EIGENVECTORS OF THAT UPPER
#           HESSENBERG MATRIX DETERMINED BY ELMHES.
#
# ORTHES    REDUCES A REAL GENERAL MATRIX TO UPPER HESSENBERG FORM
#           USING ORTHOGONAL TRANSFORMATIONS.
#
# ORTBAK    BACK TRANSFORMS THE EIGENVECTORS OF THAT UPPER
#           HESSENBERG MATRIX DETERMINED BY ORTHES.
#
# TRED1     REDUCES A REAL SYMMETRIC MATRIX TO A SYMMETRIC
#           TRIDIAGONAL MATRIX USING ORTHOGONAL TRANSFORMATIONS.
#
# TRED2     REDUCES A REAL SYMMETRIC MATRIX TO A SYMMETRIC
#           TRIDIAGONAL MATRIX ACCUMULATING THE ORTHOGONAL
#           TRANSFORMATIONS.
#
# TRBAK1    BACK TRANSFORMS THE EIGENVECTORS OF THAT SYMMETRIC
#           TRIDIAGONAL MATRIX DETERMINED BY TRED1.
#
# FIGI      TRANSFORMS A CERTAIN REAL NON-SYMMETRIC TRIDIAGONAL
#           MATRIX TO A SYMMETRIC TRIDIAGONAL MATRIX.
#
# BAKVEC    BACK TRANSFORMS THE EIGENVECTORS OF THAT SYMMETRIC
#           TRIDIAGONAL MATRIX DETERMINED BY FIGI.
#
# COMHES    REDUCES A COMPLEX GENERAL MATRIX TO COMPLEX UPPER
#           HESSENBERG FORM USING ELEMENTARY TRANSFORMATIONS.
#
# COMBAK    FORMS THE EIGENVECTORS OF A COMPLEX GENERAL MATRIX FROM
#           THE EIGENVECTORS OF THAT UPPER HESSENBERG MATRIX
#           DETERMINED BY COMHES.
#
# HTRIDI    REDUCES A COMPLEX HERMITIAN MATRIX TO A REAL SYMMETRIC
#           TRIDIAGONAL MATRIX USING UNITARY TRANSFORMATIONS.
#
# HTRIBK    BACK TRANSFORMS THE EIGENVECTORS OF THAT SYMMETRIC
#           TRIDIAGONAL MATRIX DETERMINED BY HTRIDI.
#
# HQR       DETERMINES THE EIGENVALUES OF A REAL UPPER HESSENBERG
#           MATRIX.
#
# HQR2      DETERMINES THE EIGENVALUES AND EIGENVECTORS OF A REAL
#           UPPER HESSENBERG MATRIX.
#
# INVIT     DETERMINES THOSE EIGENVECTORS OF A REAL UPPER
#           HESSENBERG MATRIX CORRESPONDING TO SPECIFIED
#           EIGENVALUES.
#
# TQL1      DETERMINES THE EIGENVALUES OF A SYMMETRIC TRIDIAGONAL
```

	MATRIX.
TQL2	DETERMINES THE EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX.
IMTQL1	DETERMINES THE EIGENVALUES OF A SYMMETRIC TRIDIAGONAL MATRIX.
IMTQL2	DETERMINES THE EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX.
TSTURM	DETERMINES SOME EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX.
BISECT	DETERMINES SOME EIGENVALUES OF A SYMMETRIC TRIDIAGONAL MATRIX.
COMLR	DETERMINES THE EIGENVALUES OF A COMPLEX UPPER HESSENBERG MATRIX.
COMLR2	DETERMINES THE EIGENVALUES AND EIGENVECTORS OF A COMPLEX UPPER HESSENBERG MATRIX.
CINVIT	DETERMINES THOSE EIGENVECTORS OF A COMPLEX UPPER HESSENBERG MATRIX CORRESPONDING TO SPECIFIED EIGENVALUES.
RATQR	DETERMINES SOME EXTREME EIGENVALUES OF A SYMMETRIC TRIDIAGONAL MATRIX.
ELTRAN	ACCUMULATES THE TRANSFORMATIONS IN THE REDUCTION OF A REAL GENERAL MATRIX BY ELMHES.
ORTRAN	ACCUMULATES THE TRANSFORMATIONS IN THE REDUCTIONS OF A REAL GENERAL MATRIX BY ORTHES.
FIGI2	TRANSFORMS A CERTAIN REAL NON-SYMMETRIC TRIDIAGONAL MATRIX TO A SYMMETRIC TRIDIAGONAL MATRIX ACCUMULATING THE DIAGONAL TRANSFORMATIONS.
TINVIT	DETERMINES SOME EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX.

FRKCOM

The program FRKCOM allows the user to compare an address space with the address space of a file (such as a subsystem) whether or not the file has shared pages (BINCOM does the wrong thing on such files). FRKCOM is primarily useful for determining if the loading of a number of .REL files matches a SAVE or SSAVE file.

FRKCOM will complain if there are any significant differences in the spaces by telling the starting address where changes begin in a page, then hopping to the next page. No attempt is made to get things back into sync a la SRCCOM since this is virtually impossible with relocated binary code. Comparisons begin at location 140 and end at 775777(inclusive). The top two pages of the EXEC's immediately inferior fork's address space are used by FRKCOM itself. An example of FRKCOM's use follows.

```
@GET <CHIPMAN>RUNOFF
@MERGE <SUBSYS>FRKCOM
@GO 777000

TYPE FILE TO COMPARE WITH FORK CONTENTS:  <SUBSYS>RUNOFF.SAV%

COMPARE FINISHED
@
```

FUDGE2

The FUDGE2 program is used to update files containing one or more relocatable binary programs and to manipulate programs within program files. Three files are used in the updating process.

1. A master file containing the file to be updated.

2. A transaction file containing the file of programs to be used when updating.

3. An output file containing the updated file.

All three files can be on the same device if the device is DSK. The two input files can be on the same DECtape.

The desired function of FUDGE2 is specified by a command code at the end of the command string. Only one command code can be specified in each command string. The command string is then terminated with an ESCApe (ALTMODE). Switches can also be used to manipulate file directories and to position a magnetic tape.

For further information on FUDGE2 please refer to DECsystem10 Assembly Language Handbook, UTILITY Section.

@FUDGE2

*output dev:file.ext=master dev:file.ext<programs>, transaction dev:file.ext<programs> (command)$

output dev: = the device on which the updated file is written. If omitted, DSK is assumed.

master dev: = the device containing the file to be updated. If omitted, the default is DSK.

transaction dev: = the device containing the files of programs to be used in the updating process. When more than one file is transferred from magnetic tape or paper tape, a colon must follow the device name for each file. For example,

MTA: : : Transfer 3 files

If the device is omitted, DSK is assumed.

file.ext = the file name and extension of each file.