SLAC-196
UC-32

DPAK AND HPAK
A VERSATILE DISPLAY AND HISTOGRAMMING PACKAGE

C.A. LOGG, A.M. BOYARSKI, A.J. COOK, R.L.A. COTTRELL
STANFORD LINEAR ACCELERATOR CENTER
STANFORD UNIVERSITY
Stanford, California 94305

PREPARED FOR THE
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
UNDER CONTRACT NO. E(04-3)-515

June 1976

Version 7.0

Printed in the United States of America. Available from National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161. Price: Printed Copy $5.00; Microfiche $2.25.

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS

CONTENTS

APPENDICES

CONTENTS PAGE

FIGURES

ABSTRACT

This document describes the features of a display and histogram package which requires a minimal number of subroutine calls in order to generate graphic output in many flavours on a variety of devices. Default options are preset to values that are generally most wanted, but the default values may be readily changed to the user's needs (e.g. size of plot, type of scales, scale factors, plot characters, super-imposing several plots, etc.). The description falls naturally into two parts, namely the set of routines (DPAK) for displaying data on some device, and the set of routines (HPAK) for generating histograms. HPAK provides a means of allocating memory for histograms, accumulating data into histograms and subsequently displaying the histograms via calls to the DPAK routines. Histograms and displays of either one or two independent variables can be made.

I. INTRODUCTION

The generation of displays by DPAK occurs at several levels. At the lowest level are the Unified Graphics subroutines (reference 2.) for generating vectors and characters. At the next level are the plotting subroutines (LPUT1, GRUT1, LPUT2, AND GRUT2) which accept the user data in the form of functions for the X and Y (or X, Y and Z) axes and generate graphs complete with scales, error bars and titles onto the designated device. Finally, a set of higher level interface routines (DISP1, DISP1A, DUT1, DISP2, DISP2A, and DUT2) are provided to make the package easier for the casual user. A set of supporting routines (OPTPUT and OPTGET) are provided for changing and examining the settings of the various options.

The histogram package was written with the view that it should be easy to add or delete histograms in a user's analysis program. For instance, in an interactive environment a user may need to delete one histogram to make room in core for another without reloading the job, or in a batch environment a user may wish to insert or delete histograms from one analysis pass to another. In HPAK, one block of memory serves as the data storage area for all histograms. A subroutine call is provided for allocating memory space for a new histogram and another subroutine call is provided for releasing memory space. Subroutines are also provided for accumulating, clearing and outputting either individual histograms or all the histograms that have been defined.

The numerous options and capabilities available may be at first confusing to a new user. We suggest that new users begin by using the default setting of options, and mastering the following subroutine calls:

HSIZE	for defining histogram storage block
HDEF1	for defining 1-D histogram properties (e.g. identifier, number of bins, bin width, title, etc.)
HCUM1	for accumulating into the 1-D histogram
HOUT	for outputting the histogram (use 'PRINTER' for the output unit)
HCLR	for clearing the contents of a histogram
HDEL	for releasing histogram storage space

By using the lineprinter for the output unit, and avoiding scatterplots, no special JCL statements are needed. These routines alone are sufficient in many cases. If graphs (which are not related to histogramming) are desired, then the plotting routines DISP1 and DISP1A (again with the output unit of 'PRINTER') can be used to plot functions and arrays respectively.

The program consists of about 9000 lines of source code. The majority (approximately 60%) of the code is in MORTRAN, a FORTRAN preprocessor (see reference 7). There are about 200 lines of IBM 360 Assembler language, and the remainder is in IBM FORTRAN H. The program though an outgrowth of a program to run on a Xerox Sigma 5 is designed to run on an IBM 360/370 and makes fairly extensive use of features such as LOGICAL*1 supplied in IBM FORTRAN H.

The following sections give detailed descriptions of the individual routines that may be called by the users. Many more subroutines, used internally are not described in detail, but are listed in Appendix V so that the user may avoid name conflicts.

II. DPAK

DPAK is a collection of FORTRAN callable subprograms which can be used to display graphs of data with one abscissa (henceforth referred to as 1-D data) and data with two abscissae (henceforth referred to as 2-D data) on the lineprinter and the various graphic devices available at SLAC. These subroutines evolved from the graphics display subprograms originally written for the SPEAR SIGMA-5 computer at SLAC (see reference 5.), and have been in regular use at SLAC since 1973. The data can be provided to the routines in either array or function form. The output format can be modified and tailored to the user's needs by means of the simple option setting routines which enable the user to override the default options.

Though primarily designed for display of graph type data, this package is very general in nature and has been sucessfully used for displaying pictures of charged particle tracks in magnetic detectors and ray tracing in optical systems.

The graphic devices currently available at SLAC are the CALCOMP drum plotters referred to a 'CALDRSM' and 'CALDRLG', the microfilm plotter (referred to as 'CAL16MU'), the microfiche plotter (referred to as 'CALFICH'), and the Tektronix 4013 storage scope (referred to as 'PDS4013'). All the graphic devices are programmed via the SLAC Unified Graphics software, and are described in detail in reference 2.

A. ONE ABSCISSA DISPLAY ROUTINES

1. DISP1

DISP1 is the subroutine for plotting 1-D data which are functions. The calling sequence is:

```
CALL DISP1(FX,FY,FERR,N,TITLE [,UNIT [,FB [,FC  [,FD  [,FE
                              [,FF  [,FG  [,FH  [,FP  [,FQ
                              ]]]]]]]]]])
```

where: FX, FY, FERR, FB, FC, FD, FE, FF, FG, FH, FP, FQ are REAL*4 functions which can be called in the form X=FX(I) for the Ith value
- FX is the abscissa function
- FY is the 1st ordinate function
- FERR is the error function for FY
- FA, FB, FC, FD, FE, FF, FG, FH, FP, and FQ are additional optional functions to be overplotted
- N is an I*4 value which is the number of points to be plotted
- TITLE is a string of up to 80 characters terminating with an '@'; this string may be broken up into sub-labels (separated by semi-colons (';')) of up to 32 characters each for the X and Y axes labels on graphic devices; for example: 'MAIN LABEL; X-AXIS LABEL; Y-AXIS LABEL@'; the string contains the main plot label followed by a ';', the X-axis label followed by a ';' and then the Y-axis label followed by the terminating '@'
- UNIT is a string indicating the output destination of plot (one of the following: 'PRINTER', 'CALDRSH', 'CALDRLG', 'CALFICH', 'CAL16HU', 'PDS4013', or 'PDSPDEV'); the unit argument is optional, and if it is not supplied the default unit is used (note that supplying a unit here does not change the default unit)

There is also another way of calling DISP1.

```
CALL DISP1(FX2,FY2,FERR2,N2,TITLE [,UNIT ])
```

where FX2, FY2, and FERR2 are doubly indexed functions of the form FX2(I,J), FY2(I,J), and FERR2(I,J) and the ranges L and M (for I and J respectively) are indicated by setting N2=M*2**16+L. When called in this manner, the M functions given by the points (FX2(I,J),FY2(I,J)) for I=1,L will be overplotted.

2. DISP1A

DISP1A is the subroutine for plotting 1-D data which are in arrays. The calling sequence is:

```
CALL DISP1A(X,Y,ERR,N,TITLE, [,UNIT [,B [,C  [,D  [,E  [,F
                              [,G [,H [,P [,Q ]]]]]]]]]])
```

where: X, Y, ERR, B, C, D, E, F, G, H, P, and Q are REAL*4 arrays of the form XI=X(I) for the Ith value
- X is the abscissa array
- Y is the 1st ordinate array
- ERR is the error array associated with the Y array
- B, C, D, E, F, G, H, P, and Q are additional optional arrays to be overplotted
- N, TITLE, and UNIT are the same as in the DISP1 call.

3. DUT1

DUT1 is a display utility subroutine with a slightly different calling sequence from DISP1. The functions may define a single curve, or they may define multiple curves that are to be overplotted.

The calling sequence is:

```
CALL DUT1(FX,FY,FERR,N,TITLE,SCALE)
```

where:
- FX, FY, FERR are REAL*4 functions for the X-axis, the Y-axis, and the error in Y respectively; they may be either singly indexed (FX(I), FY(I), and FERR(I)) or doubly indexed (FX(I,J), FY(I,J) and FERR(I,J)) as specified by N
- N is an integer specifying the number of data points to be plotted with I ranging from 1 to L; it may also specify the number of functions (M) to be plotted with L data points each, if N has the form:

  N = M*2**16 + L

  if M>0, then the doubly indexed functions must be supplied, with J ranging from 1 to M and I ranging from 1 to L.
- TITLE is the same as the title string for the call to DISP1
- SCALE is a real array (4 words) which supplies the manual scale factors in the order described in SCPUT1; note that these will override any scale factors given by a call to SCPUT1.

The output destination for plots generated by calls to DUT1 is the current default unit which may be altered by the call:

CALL OPTPUT('DEVICE',UNIT)

as described in section II.A.4.4.

4. Output Format Options Available

The options are described later in detail in the section on OPTPUT and OPTGET. This is a summary of those relevant to the different display media.

4.1 Options Applying to Both the Lineprinter and Graphic Device Output

1) 'YZERO' - force zero point on Y-axis scale
2) 'ERROR' - error bars on 1st ordinate
3) 'CHAR' - set character to be used in plot
4) 'MANUAL' - use manual scales (on the lineprinter manual scales are used only on the Yaxis)
5) 'YLOG' - log scale Y-axis
6) 'LOGS' - log scale both axes (on the lineprinter only the Y-axis is log scaled)
7) 'HIST' - display 1st ordinate as a bar histogram
8) 'OMIT' - omit 1st and last points when scaling
9) 'YMAN' - use manual scales for the Y-axis

4.2 Options Applying Just to the Lineprinter

1) 'GRAT' - put graticules on graph
2) 'LENGTH' - specify the total number of columns to be used in the lineprinter output

4.3 Options Applying Just to the Graphic Device Output

1) 'RESCALE' - recalculate the scale factors for next plot
2) 'XLOG' - log scale X-axis
3) 'FRAME' - plot or do not plot the frame on the graph (used to stop the frame from being repeatedly drawn when overplotting)
3) 'OVER' - overplot the output from subsequent calls as long as the 'OVER' option is TRUE
4) 'XMAN' - use manual scales on the X-axis
5) 'SIZE' - set size of plot
6) 'OFF' - offset plot
7) 'XTIC' - number of major tics on X-axis
8) 'YTIC' - number of major tics on Y-axis
9) 'ITICS' - the number of minor tic marks between the major tic marks

10) 'LINE' - draw a line between the points
11) 'CENTER' - bin alignment on bar histograms
12) 'XZERO' - force a zero on the X-axis

4.4 Setting the Default Output Unit

The default output unit is set by:

CALL OPTPUT('DEVICE',UNIT)

where UNIT is a character string indicating the device desired. Output devices available are:

'PRINTER' the currently assigned printer unit
'PDS4013' WYLBUR PDS
'CALDRSM' 10 inch CALCOMP
'CALDRLG' 29 inch CALCOMP
'CAL16MU' the CALCOMP microfilm unit
'CALFICH' the CALCOMP microfiche
'PDSPDEV' for the Unified Graphics PDSPDEV

5. Overplotting

The overplotting facilities in DPAK are quite versatile. There are two cases to consider: overplotting on the lineprinter and overplotting on graphic devices.

5.1 Overplotting on the Lineprinter

Overplotting on the lineprinter is done by using the optional arguments in the DISP1 and DISP1A calls or the doubly indexed functions in DISP1 and DUT1. All of the OPTPUT output options for the lineprinter still apply, however the 'CHAR', 'HIST', and 'ERROR' options apply only to the first ordinate function or array. The functions and/or arrays that are overplotted are done so with the character 'B' for the second one, 'C' for the third one, 'D' for the fourth one, etc.

An example is given in section II.A.7.1.

5.2 Overplotting on the Graphic Devices

Overplotting on the graphic devices can be done in either of two ways. The first way is to simply make a call as though one was overplotting on the lineprinter, but specify a graphic device unit instead of 'PRINTER'. The other way is through use of the OPTPUT 'OVER' option. To do overplotting by using the 'OVER' option, do the following:

1) turn overplotting on via a call to: OPTPUT('OVER',.TRUE.)
2) set up other options as desired
3) make the appropriate call to DISP1A or DISP1

4) repeat steps 2) and 3) until you have plotted all you want
5) turn overplotting off (CALL OPTPUT('OVER',.FALSE.))

Note that these two methods of overplotting can also be intermixed.

An example is given in section II.A.7.2.

6. Setting and Retrieving Manual Scale Factors(for DISP1 and DISP1A)

Manual scales, if required, are setup for the lineprinter and graphic device output via a call to SCPUT1. The calling sequence is:

```
CALL SCPUT1(SCALE)
```

where:
SCALE is a REAL*4 array of length 4 specifying the scale factors as follows:

- SCALE(1) = the minimum X-axis value if the X-axis is to be plotted linearly (if it is a log plot on the X-axis, then SCALE(1) is the minimum power of 10 as a floating point number)
- SCALE(2) = the increment between the X-axis tic marks if the plot is linear on the X-axis (if it is a log axis, then SCALE(2) is the number of decades as a floating point number)
- SCALE(3) = the minimum Y-axis value if the Y-axis is to be plotted linearly (if it is a log axis, then SCALE(3) is the minimum power of 10 as a floating point number)
- SCALE(4) = the increment between Y-axis tic marks if the plot is linear on the Y-axis (if it is log, then SCALE(4) is the number of decades as a floating point number)

To obtain the scale factors of the last plot made by a call to DISP1A, DISP1, or DUT1 use:

```
CALL SCGET1(SCALE [,IXTIC,IYTIC ])
```

where:
SCALE is a REAL*4 array of length 4 and the scale factors are returned as described in SCPUT1
IXTIC is an INTEGER*4 word and the number of tic marks used on the Y-axis will be returned
IYTIC is an INTEGER*4 word and the number of tic marks used on the y-axis will be returned
Note that IXTIC and IYTIC are optional, but that both must be supplied if either is.

7. Examples

Following are two examples which show some of the facilities available.

7.1 Lineprinter Output Example

```
      EXTERNAL FX,FA,FB,FC,FERR
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('CHAR','*')
      CALL OPTPUT('ERROR',.TRUE.)
      CALL DISP1(FX,FA,FERR,25,
     1           '3 ARRAYS WITH HIST AND ERROR BARS@',
     2           'PRINTER',FB,FC)
```

It gives the following lineprinter plot:

```
ID=                               3 ARRAYS WITH HIST AND ERROR BARS
                                                        Y(I)*E -1
I-*-I ERR     B       C     X -25.         0.        25.        50.
 E -2 E -1    E -2    E -2  E   0 +---------+---------+---------++
 -199   14    -238    -278      1 B-*------                     I
 -196   14    -236    -275      2 CB*------                     I
 -191   13    -230    -270      3 CB*------                     I
 -183   13    -223    -263      4 CB-*-----                     I
 -174   13    -214    -254      5 CB-*-----                     I
 -163   12    -203    -243      6 C-B*------                    I
 -150   12    -190    -230      7 -CB-*-----                    I
 -135   11    -175    -215      8 -C-B-*---                     I
 -118   10    -158    -198      9 I-C-B*-----                   I
 -100   10    -139    -179     10 I -CB-*----                   I
  -79    8    -119    -159     11 I  -CB-*---                   I
  -56    7     -96    -136     12 I     CB-*---                 I
  -31    5     -71    -110     13 I      CB-*--                 I
   -4    2     -44     -84     14 I       CB-*-                 I
   25    5     -15     -55     15 I        CB-*--               I
   56    7      16     -23     16 I         C-B*---             I
   88    9      48       8     17 I          C-B-*---           I
  124   11      83      43     18 I           -CB-*----         I
  160   12     120      80     19 I           --C-B*------      I
  200   14     159     120     20 I            ---CB-*------    I
  241   15     201     161     21 I             ---C-B-*------  I
  283   16     243     203     22 I               ---C-B*------ I
  329   18     288     249     23 I                ----C-B*-----I
  376   19     336     296     24 I                 -----CB-*-----
  425   20     385     345     25 I                   -----CB-*----
 E -2 E -1    E -2    E -2  E   0 +---------+---------+---------++
I-*-I ERR     B       C     X -25.         0.        25.        50.
```

Figure 1

7.2 Graphic Device Output Example

```
      REAL SCALE(4),XA(25),G(25),ERR(25),A(25),YLOG(10)
      EXTERNAL FX,FD,FERR
C TURN OVERPLOTTING ON
      CALL OPTPUT('OVER',.TRUE.)
C MAKE A PLOT IN THE LOWER LEFT HAND CORNER
      CALL OPTPUT('HIST',.FALSE.)
      CALL OPTPUT('CHAR','*')
      CALL OPTPUT('YMAN',.TRUE.)
      CALL OPTPUT('XTIC',0)
      CALL OPTPUT('SIZE',.5,.5)
      CALL OPTPUT('ITICS',4)
      SCALE(3)=-3.
      SCALE(4)=1.
      CALL SCPUT1(SCALE)
      CALL DISP1A(XA,G,ERR,25,
     1            'LOWER LEFT FIGURE;X-AXIS;Y MAN SCALES@',
     2            'PDS4013',A)
C OVERPLOT A LINE PLOT
      CALL OPTPUT('FRAME',.FALSE.)
      CALL OPTPUT('ERROR',.FALSE.)
      CALL OPTPUT('LINE',.TRUE.)
      CALL OPTPUT('CHAR',' ')
      CALL DISP1(FX,FD,FERR,25,'@','PDS4013')
C MAKE A PLOT IN THE LOWER RIGHT HAND QUADRANT
      CALL OPTPUT('YMAN',.FALSE.)
      CALL OPTPUT('FRAME',.TRUE.)
      CALL OPTPUT('CHAR','X')
      CALL OPTPUT('ITICS',0)
      CALL OPTPUT('OFF',.5,0.)
      CALL OPTPUT('RESCALE',.TRUE.)
      CALL OPTPUT('YLOG',.TRUE.)
      CALL DISP1A(XA,YLOG,ERR,10,
     1            'LOWER RIGHT FIGURE;X-AXIS;LOG Y-AXIS@',
     2            'PDS4013')
C MAKE PLOT IN UPPER RIGHT HAND QUADRANT
      CALL OPTPUT('HIST',.TRUE.)
      CALL OPTPUT('LINE',.FALSE.)
      CALL OPTPUT('XTIC',0)
      CALL OPTPUT('OFF',.5,.5)
      CALL DISP1A(XA,YLOG,ERR,10,
     1            'HISTOGRAM;OPTIMAL X TICS;Y LOG AXIS@',
     2            'PDS4013')
C MAKE PLOT IN UPPER LEFT HAND QUADRANT
      CALL OPTPUT('YTIC',0)
      CALL OPTPUT('YLOG',.FALSE.)
      CALL OPTPUT('OFF',0.,.5)
      CALL OPTPUT('ERROR',.TRUE.)
      CALL OPTPUT('CHAR',' ')
      CALL DISP1A(XA,A,ERR,20,'HISTOGRAM WITH ERROR BARS@',
     1            'PDS4013')
```



```
C TURN OVERPLOTTING OFF
      CALL OPTPUT('OVER',.FALSE.)
C RESET SIZE OF GRAPHIC PICTURE
      CALL OPTPUT('SIZE',1.0,1.0)
      CALL OPTPUT('OFF',0.0,0.0)
```

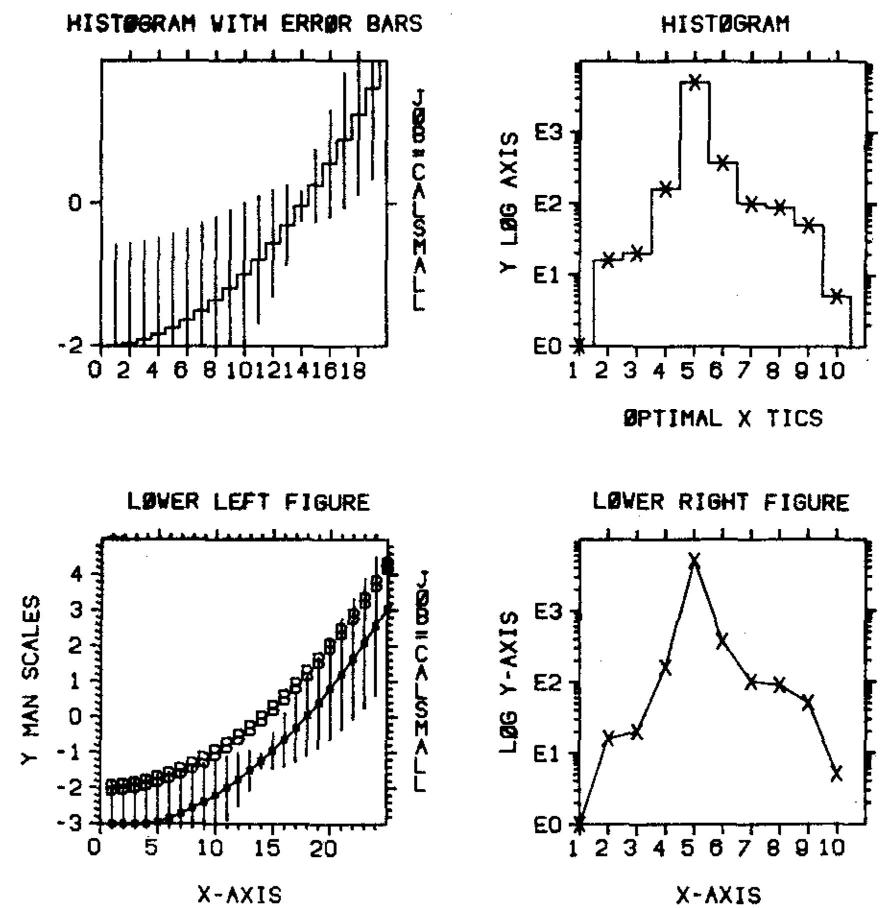

Figure 2

B. TWO ABSCISSA DISPLAY ROUTINES

The facilities available for displaying two abscissa data are discussed in this section. Note the presence of an argument for an error function (or array) in the calling sequence. In general this may just be a zero function or array, since for a 2-D display it is not used. It is provided in the argument list, because of the need for calculating the errors on the projection displays (though if no errors are requested on the projection, it may be a zero function or array).

1. DISP2

DISP2 is the subroutine for plotting 2-D data which are functions. The calling sequence is:

```
CALL DISP2(FX,FY,FZ,FZER,NX,NY,TITLE [,UNIT])
```

where: FX is the first abscissa function with a calling sequence of the form XI=FX(I) for I=1,NX
FY is the second abscissa function with a calling sequence of the form YJ=FY(J) for J=1,NY
FZ is the ordinate function with a calling sequence of the form ZIJ=FZ(I,J) for I=1,NX, and J=1,NY
FZER is the error function for function FZ and it is used only if the user is doing a projection; it has a calling sequence of the form ERRIJ=FZER(I,J) for I=1,NX, and J=1,NY
NX is the number of points along the first abscissa
NY is the number of points along the second abscissa
TITLE is a string of characters terminating with an '@' sign
UNIT is a string indicating the output destination of the plot(as in DISP1)

2. DISP2A

DISP2A is the subroutine for plotting 2-D data which are arrays. The calling sequence is:

```
CALL DISP2A(X,Y,Z,ZER,NX,NY,TITLE [,UNIT])
```

where: X is a REAL*4 array of the form XI=X(I) for I=1,NX
Y is a REAL*4 array of the form YJ=Y(J) for J=1,NY
Z is a REAL*4 array of the form ZIJ=Z(I,J) for I=1,NX, and J=1,NY
ZER is a REAL*4 array of the form ZERIJ=ZER(I,J) for I=1,NX, and J=1,NY
NX, NY, TITLE, and UNIT are as described in the section on DISP2

3. DUT2

DUT2 is another subroutine for plotting 2-D data which are functions. The calling sequence is:

```
CALL DUT2(FX,FY,FZ,FZER,NX,NY,TITLE,SCALE)
```

where: FX is the first abscissa function with a calling sequence of the form XI=FX(I) for I=1,NX
FY is the second abscissa function with a calling sequence of the form YJ=FY(J) for J=1,NY
FZ is the ordinate function with a calling sequence of the form ZIJ=FZ(I,J) for I=1,NX, and J=1,NY
FZER is the error function for function FZ and it is used only if the user is doing a projection; it has a calling sequence of the form ERRIJ=FZER(I,J) for I=1,NX, and J=1,NY
NX is the number of points along the first abscissa
NY is the number of points along the second abscissa
TITLE is a string characters terminating with an '@' sign
SCALE is an array of length 16 specifying the scale factors to be used when manual scaling has been called for; the order of the scale factors is the same as for the SCPUT2 call

The output destination of the plot generated by a DUT2 call is the current default unit which may be altered by the call:

```
CALL OPTPUT('DEVICE',UNIT)
```

See section II.B.4.4.

4. Output Options Available

The output options are discussed in detail in the section on OPTPUT and OPTGET. The following is merely a summary of which are relevant here.

4.1 Options Applying to Both the Lineprinter and Graphic Device Output

1) 'MANUAL' - use manual scale factors
2) 'OMIT' - omit first and last bins from scaling

4.2 Options Applying Just to the Lineprinter

1) 'SYMBOL' - number of symbols to be used for plot
2) 'ZAUTO' - automatic scaling of bin values
3) 'COLUMNS' - number of columns to be used for each bin value
4) 'TAUTO' - scale axes sums
5) 'LENGTH' - number of columns to be used on the lineprinter page
6) 'XYAUTO' - automatically calculate the number of column positions
7) 'SCIENTIFIC' - use scientific notation for bin values
8) 'PAGE' - number of lines per page to use

4.3 Options Applying Just to Graphic Device Output

1) 'YZERO' - force a zero on the Y-axis
2) 'FRAME' - draw a frame around the plot
3) 'SAVE' - save the scale factors used
4) 'XDIR' - draw lines parallel to the X-axis

4.4 Setting the Default Output Unit

The default output unit is set by:

```
CALL OPTPUT('DEVICE',UNIT)
```

where UNIT is a character string indicating the device desired. See section II.A.4.4 for a list of available units.

5. Displaying a Projection of 2-D Data

By setting the appropriate options with calls to OPTPUT the user may indicate that succeeding calls to DISP2, DISP2A, or DUT2 should display a projection of the 2-D data, rather than the regular 2-D data display. These four OPTPUT calls, and their purposes are:

1) CALL OPTPUT('SFLAG',.TRUE.) - sets a flag to indicate that for the succeeding calls to DISP2, DISP2A, or DUT2, the indicated projection is to be made; the CALL OPTPUT('SFLAG',.FALSE.) turns the option off
2) CALL OPTPUT('SDY',.FALSE.) - indicates the direction of the projection; FALSE causes a projection on the X-axis, while TRUE causes the projection to be on the Y-axis
3) CALL OPTPUT('SBEGIN',N) indicates that the summing for the projection should start with point Y(N) (or X(N) if slice is in Y direction)
4) CALL OPTPUT('SWIDTH',N) - indicates the number of abscissa values to be summed over

6. Setting and Retrieving Manual Scale Factors (for DISP2 and DISP2A)

Manual scales, if required, are set up for the graphic device output by way of a call to SCPUT2. The calling sequence is:

```
CALL SCPUT2(SCALE)
```

where: SCALE is a REAL*4 array of length 16 with the scale factors as follows:

- SCALE(1) - minimum X-axis value
- SCALE(2) - increment on the X-axis
- SCALE(3) - minimum Y-axis value
- SCALE(4) - increment on the Y-axis
- SCALE(5) - minimum Z-axis value
- SCALE(6) - increment on the Z-axis
- SCALE(7) - isometric rotation angle in degrees about the Z-axis
- SCALE(8) - isometric elevation angle in degrees about the horizontal line perpendictular to the line of sight
- SCALE(9)-SCALE(12) - XMIN, DX, YMIN, and DY for slicing on the X-axis
- SCALE(13)-SCALE(16) - XMIN, DX, YMIN, and DY for slicing on the Y-axis

Manual scales for 2-D lineprinter output are indicated by setting the appropriate options with calls to OPTPUT.

The manual scales used for the last 2-D graphic device display made may be obtained by the following call:

```
CALL SCGET2(SCALE)
```

where S is as described above.

7. Examples

7.1 Lineprinter Example - Integer Notation

```
      REAL XA(25),YY(25),ZZ(25,6),ZZE(25,25)
      CALL OPTPUT('LENGTH',70)
      CALL DISP2A(XA,YY(7),ZZ,ZZE,25,6,'INTEGER NOTATION@',
     1            ,'PRINTER')
```

It gives the following output:

```
            ID= INTEGER NOTATION
          J     1     2     3     4     5     6
       E  0    7.    8.    9.   10.   11.   12.Y(J).E  0
  I    X(I)+----+-----+-----+-----+-----+-----+-- TOTAL.E  0
  1      1.I                                    I      1
  2      2.I                                    I      2
  3      3.I    1     1     1     1     1     1 I      4
  4      4.I    1     1     1     1     1     1 I      7
  5      5.I    1     2     2     2     2     2 I     12
  6      6.I    2     3     4     4     4     3 I     20
  7      7.I    4     5     6     6     6     5 I     31
  8      8.I    6     7     9     9     9     7 I     47
  9      9.I    8    11    13    13    13    11 I     69
 10     10.I   12    15    18    19    18    15 I     98
 11     11.I   17    21    25    26    25    21 I    134
 12     12.I   22    28    33    34    33    28 I    178
 13     13.I   28    36    42    44    42    36 I    229
 14     14.I   35    45    38    35    38    45 I    236
 15     15.I   42    36    27    24    27    36 I    193
 16     16.I   41    27    17    13    17    27 I    143
 17     17.I   35    20     8     4     8    20 I     94
 18     18.I   30    13     1    -4     1    13 I     56
 19     19.I   27     9    -4    -8    -4     9 I     31
 20     20.I   26     8    -5   -10    -5     8 I     22
 21     21.I   27     9    -4    -8    -4     9 I     31
 22     22.I   30    13     1    -4     1    13 I     56
 23     23.I   35    20     8     4     8    20 I     94
 24     24.I   41    27    17    13    17    27 I    143
 25     25.I   42    36    27    24    27    36 I    193
  I    X(I)+----+-----+-----+-----+-----+-----+--
TOTAL.E   0   515   396   285   244   285   396     2125
```

Figure 3

7.2 Lineprinter Example Scientific Notation

```
      REAL XA(25),YY(25),ZZ(25,6),ZZE(25,25)
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('SCIENTIFIC',.TRUE.)
      CALL DISP2A(XA,YY(7),ZZ,ZZE,25,3,
     1            ,'SCIENTIFIC NOTATION@','PRINTER')
```

It gives the following output:

```
            ID=       SCIENTIFIC NOTATION
             J               1               2               3
        E    0              7.              8.              9.Y(J).E  0
  I     X(I)+        --------+       --------+       --------+ TOTAL.E  0
  1       1.I 0.1554E+00 0.1996E+00 0.2319E+00 I      1
  2       2.I 0.2880E+00 0.3698E+00 0.4296E+00 I      1
  3       3.I 0.5161E+00 0.6627E+00 0.7699E+00 I      2
  4       4.I 0.8945E+00 0.1149E+01 0.1334E+01 I      3
  5       5.I 0.1500E+01 0.1925E+01 0.2237E+01 I      6
  6       6.I 0.2431E+01 0.3122E+01 0.3627E+01 I      9
  7       7.I 0.3813E+01 0.4896E+01 0.5689E+01 I     14
  8       8.I 0.5784E+01 0.7427E+01 0.8629E+01 I     22
  9       9.I 0.8487E+01 0.1090E+02 0.1266E+02 I     32
 10      10.I 0.1204E+02 0.1546E+02 0.1797E+02 I     45
 11      11.I 0.1653E+02 0.2122E+02 0.2466E+02 I     62
 12      12.I 0.2194E+02 0.2818E+02 0.3274E+02 I     83
 13      13.I 0.2818E+02 0.3618E+02 0.4204E+02 I    106
 14      14.I 0.3499E+02 0.4493E+02 0.3780E+02 I    118
 15      15.I 0.4204E+02 0.3603E+02 0.2729E+02 I    105
 16      16.I 0.4116E+02 0.2729E+02 0.1714E+02 I     86
 17      17.I 0.3512E+02 0.1953E+02 0.8127E+01 I     63
 18      18.I 0.3035E+02 0.1341E+02 0.1012E+01 I     45
 19      19.I 0.2729E+02 0.9480E+01 -.3551E+01 I     33
 20      20.I 0.2624E+02 0.8127E+01 -.5123E+01 I     29
 21      21.I 0.2729E+02 0.9480E+01 -.3551E+01 I     33
 22      22.I 0.3035E+02 0.1341E+02 0.1012E+01 I     45
 23      23.I 0.3512E+02 0.1953E+02 0.8127E+01 I     63
 24      24.I 0.4116E+02 0.2729E+02 0.1714E+02 I     86
 25      25.I 0.4204E+02 0.3603E+02 0.2729E+02 I    105
  I     X(I)+        --------+       --------+       --------+
TOTAL.E  0               515             396             285     1198
```

Figure 4

7.3 Lineprinter Slice Example

```
      REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
      CALL OPTPUT('ERROR',.FALSE.)
      CALL OPTPUT('HIST',.TRUE.)
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('SFLAG',.TRUE.)
      CALL OPTPUT('SWIDTH',6)
      CALL OPTPUT('SDY',.FALSE.)
      CALL OPTPUT('SBEGIN',7)
      CALL DISP2A(XA,YY,ZZ,ZZE,25,25,'SLICE TEST@',
     1            'PRINTER')
      CALL OPTPUT('SFLAG',.FALSE.)
```

It gives the following output:

```
                                         SLICE TEST
                                         Y(I)*E  2
 I-*-I ERR      SUM    X  0.         1.        2.        3.
  E -2 E  0     E -3  E  0 +---------+---------+---------++
   126    0     1262     1 *                              I
   233    0     3600     2 *                              I
   419    0     7791     3 *                              I
   726    0    15054     4 **                             I
  1217    0    27230     5 **                             I
  1974    0    46974     6 ***                            I
  3096    0    77938     7 ****                           I
  4696    0   124908     8 ******                         I
  6891    0   193820     9 ********                       I
  9779    0   291612    10 ***********                    I
 13422    0   425834    11 **************                 I
 17818    0   604022    12 *******************            I
 22879    0   832819    13 ************************       I
 23556    0  1068388    14 *************************      I
 19274    0  1261133    15 ********************           I
 14343    0  1404569    16 ***************                I
  9436    0  1498934    17 **********                     I
  5563    0  1554570    18 *******                        I
  3080    0  1585373    19 ****                           I
  2224    0  1607619    20 ***                            I
  3080    0  1638421    21 ****                           I
  5563    0  1694058    22 *******                        I
  9436    0  1788422    23 **********                     I
 14343    0  1931859    24 ***************                I
 19274    0  2124604    25 ********************           I
  E -2 E  0     E -3  E  0 +---------+---------+---------++
 I-*-I ERR      SUM    X  0.         1.        2.        3.
```

Figure 5

7.4 Lineprinter Example - Simulated Scatterplot

```
      REAL XF,YF,ZF,ZERR
      EXTERNAL YF,XF,ZF,ZERR
      CALL OPTPUT('COLUMNS',1)
      CALL OPTPUT('SYMBOL',16)
      CALL DISP2(XF,YF,ZF,ZERR,20,25,'SIMULATED SCATTERPLOT@'
     1           ,'PRINTER')
      STOP
      END
      FUNCTION XF(I)
      XF=I
      RETURN
      END
      FUNCTION YF(J)
      YF=J
      RETURN
      END
      FUNCTION ZF(I,J)
      ZF=35.*EXP(-0.5*(I-10)**2/9.0)*EXP(-0.5*(J-13)**2/16.0)
      RETURN
      END
      FUNCTION ZERR(I,J)
      ZERR=0.0
      RETURN
      END
```

It gives the following output:

```
          ID=      SIMULATED SCATTERPLOT
             1234567891123456789212345J,J=  1, 25
             *************************
                          1111111111222222
      E  0 1234567890123456789012345Y(J).E  0
  I    X(I)+------------------------- TOTAL.E  0
  1      1.I                          I      0,SYMBOL RANGE
  2      2.I                          I      0,1 =   1-   1
  3      3.I       11122222111        I     16,2 =   2-   2
  4      4.I      112234444432211     I     38,3 =   3-   3
  5      5.I     1123567888765321 1   I     74,4 =   4-   4
  6      6.I    113468ACDEDCA864311   I    130,5 =   5-   5
  7      7.I    12469C*******C96421   I    197,6 =   6-   6
  8      8.I   12369C*********C96321  I    270,7 =   7-   7
  9      9.I   1247AF*********FA7421  I    321,8 =   8-   8
 10     10.I   1247B***********B7421  I    337,9 =   9-   9
 11     11.I   1247AF*********FA7421  I    321,A =  10-  10
 12     12.I   12369C*********C96321  I    270,B =  11-  11
 13     13.I    12469C*******C96421   I    197,C =  12-  12
 14     14.I    113468ACDEDCA864311   I    130,D =  13-  13
 15     15.I     11235678887653211    I     74,E =  14-  14
 16     16.I      112234444432211     I     38,F =  15-  15
 17     17.I       11122222111        I     16,**=  16-****
 18     18.I                          I      0
 19     19.I                          I      0
 20     20.I                          I      0
  I    X(I)+-------------------------
                   11122222111
               1257148245428417521
TOTAL.E  0 0054615298255528925164500     2429
```

Figure 6

7.5 Graphic Device Output Example

```
      REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
      CALL OPTPUT('XDIR',.FALSE.)
      CALL DISP2A(XA,YY,ZZ,ZZE,25,25,'XDIR=FALSE EXAMPLE@'
     1          ,'PDS4013')
```

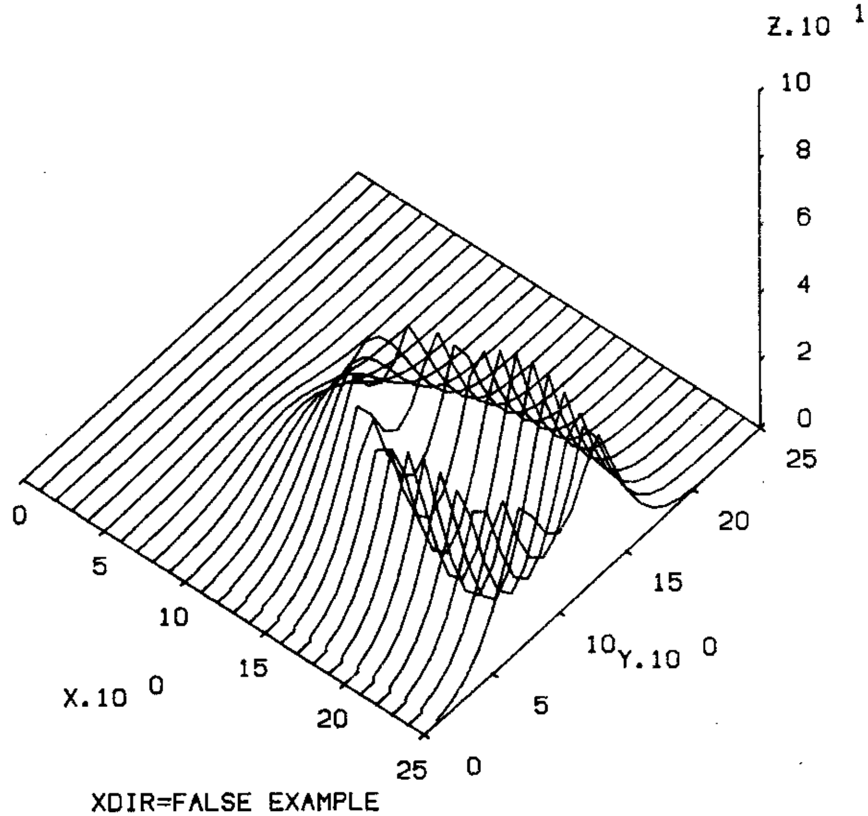

Figure 7

7.6 Graphic Device Output Example - Sliced 2-D Data

```
      REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
      CALL OPTPUT('ERROR',.FALSE.)
      CALL OPTPUT('HIST',.TRUE.)
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('SFLAG',.TRUE.)
      CALL OPTPUT('SWIDTH',6)
      CALL OPTPUT('SDY',.FALSE.)
      CALL OPTPUT('SBEGIN',7)
      CALL OPTPUT('CHAR','*')
      CALL DISP2A(XA,YY,ZZ,ZZE,25,25,'SLICE TEST@',
     1           'PDS4013')
      CALL OPTPUT('SFLAG',.FALSE.)
```

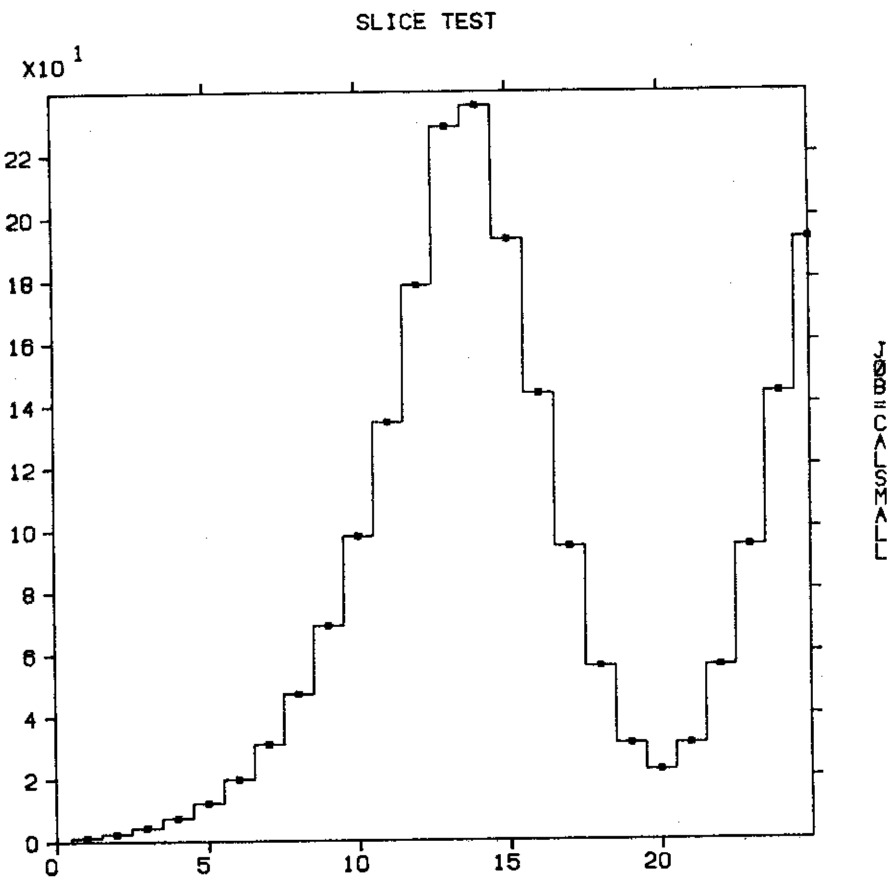

Figure 8

C. OPTPUT AND OPTGET - THE OPTION SETTING AND RETRIEVING ROUTINES

OPTPUT and OPTGET are the routines provided for setting and retrieving the value of various options used by the DPAK routines.

1. OPTPUT

The calling sequence is:

```
CALL OPTPUT(OPTION,VALUE1 [,VALUE2])
```

where:
- OPTION is a character string identifying the option to be set
- VALUE1 is the value it is to be set to
- VALUE2 is an optional argument not required for most options, but used for setting the few options requiring two values.

Unless otherwise indicated, options set by a call to OPTPUT remain set until changed by another call to OPTPUT.

Following is a list of option strings, the type of value required for them, their default values, and their function.

<u>Options for the 1-D and 2-D lineprinter output:</u>

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'EJECT'	TRUE	L*4	if true then a page is ejected for each new lineprinter plot; if false than no page is ejected
'EUNIT'	6	I*4	lineprinter logical unit on which any error messages are printed
'LENGTH'	133	I*4	the number of columns which are to be used on the lineprinter page; valid range: 50<VALUE<134; if one wishes to fetch the output and list it unnumbered on a TEKTRONIX 4013 scope, then the number of characters per line should be 70
'OUNIT'	6	I*4	logical unit for all lineprinter output except error messages

Options for the 2-D lineprinter output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'COLUMNS'	4	I*4	VALUE=number of columns in which to display each bin value; 0<VALUE<11
'PAGE'	60	L*1	2-D lineprinter output may be broken into sections because the Y-axis is too long for the page; a page eject is then made only if the remaining lines on the page are insufficient to include the next section; the number of lines per page for this purpose is specified by this option
'SCIENTIFIC'	FALSE	L*1	forces scientific notation in E10.4 format
'SYMBOL'	10	I*4	value (0<VALUE<37) is the number of symbols to be utilized in plotting the 2-D data on the lineprinter; the default is 10 (the characters 0-9); e.g. if called with VALUE=16 then the hexadecimal character set will be used (0-9 and A-F)
'TAUTO'	FALSE	L*4	if true, then scaled sums are output along the axes of 2-D lineprinter plots
'XYAUTO'	FALSE	L*4	automatically calculate the optimal number of column positions for displaying the bin values; this overrides the 'COLUMNS' option if true; if false, then the 'COLUMNS' value is utilized
'ZAUTO'	FALSE	L*4	controls automatic scaling of the bin values on the 2-D lineprinter display so that the maximum value just overflows the characters available and prints an asterisk; if true then this occurs; if false then it does not

Options for the 1-D and 2-D lineprinter and graphic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'MANUAL'	FALSE	L*4	if false, then the plots are automatically scaled on the X and Y axes; if true then the manual scale factors (supplied by the user through the appropriate scale setting routine) are used
'OMIT'	FALSE	L*4	if true, then the first and last points are omitted when the automatic scale factors are calculated

Options for the 1-D lineprinter and graphic device output and 2-D graphic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'YZERO'	FALSE	L*4	if true then the ordinate scale is constructed so that a zero point occurs at the origin; if false then there is no constraint

Options for the 1-D graphic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'CENTER'	TRUE	L*4	if true then the points will be plotted in the center of the bin; if false then the points will be plotted on the lower edge of the bins
'ITICS'	0	I*4	number of shorter intervening tic marks to be placed between major tic marks on a linear axis
'LINE'	FALSE	L*4	if true then lines are drawn to connect the points of the 1-D data; if false, then no lines are drawn to connect the points; this option effects only the graphic device output

'OFF'	0.0 0.0	2 R*4 values	offset as a fraction of the axis in the X and Y directions (e.g. CALL OPTPUT ('OFF',.5,.5) results in the upper right hand quadrant being used for graph)
'OVER'	FALSE	L*4	when OVER is set to TRUE, the user may overplot 1-D data on one graphic device at a time; see the section on overplotting for more information
'RESCALE'	FALSE	L*4	if true, then rescale for each overplot unless manual scales are set
'SIZE'	1.0 1.0	2 R*4 values	upper limits as a fraction of the graph, e.g., CALL OPTPUT ('SIZE', 1.5, .7) will create for the CALCOMP plotter a 15" (X-axis) by 7" graph; this call will not work for the TEKTRONIX scopes however. For any graphic devices except the CALCOMP plotters the largest legal sizes are 1.0,1.0.
'XLOG'	FALSE	L*4	if true, then log scale X-axis
'XMAN'	FALSE	L*4	if true, then use manual scales for X-axis
'XTIC'	8	I*4	number of tick marks on X-axis; the call CALL OPTPUT('XTIC',0) will cause the scaling routine to calculate the optimal number of tic marks
'XZERO'	FALSE	L*4	if true then the abscissa scale is constructed so that a zero point occurs at the origin; if false then there is no constraint
'YTIC'	8	I*4	number of tick marks on Y-axis; the call CALL OPTPUT('YTIC',0) will cause the scaling program to calculate the optimal number of tic marks

Options for the 1-D lineprinter and graphic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'CHAR'	Z40	L*4	specify the plot character to be used. (Note that: CALL OPTPUT('CHAR',' ') will result in a point being plotted on the graphic device output and an '*' on lineprinter output)
'ERROR'	TRUE	L*4	if true then the 1-D data points are displayed with error bars; if false, then no error bars are plotted
'HIST'	FALSE	L*4	if true then the 1-D data is displayed as a bar histogram
'LOGS'	FALSE	L*4	if true, then log scale both axes
'YLOG'	FALSE	L*4	if true, then log scale Y-axis
'YMAN'	FALSE	L*4	if true, then use manual scales for Y-axis

Options pertaining to 1-D and 2-D graphic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'FRAME'	TRUE	L*4	draw a labeled frame around plot

Options for the 2-D grpahic device output:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'XDIR'	FALSE	L*4	if true, the draw lines for the display parallel to the X-axis

Options for slicing the 2-D functions (output to either the lineprinter or a graphic device):

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'SBEGIN'	0	I*4	starting point of slice
'SDY'	FALSE	L*4	slice direction: if FALSE then slice in X direction, otherwise slice in Y direction
'SWIDTH'	0	I*4	number of points in slice

Option for setting the default output destination unit:

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING AND FUNCTION
'DEVICE'	'PDS4013'	R*8	default output device

2. OPTGET

OPTGET provides the user with a means of getting the current value of an option. The calling sequence is:

```
CALL OPTGET(OPTION,VALUE1 [,VALUE2])
```

where:
- OPTION is a character string identifying the option whose value is desired
- VALUE1 is the variable to contain the value when it is returned
- VALUE2 is an optional argument which only needs to be provided when the state of a two value option is sought

D. USING DPAK WITH THE NON-INTERACTIVE GRAPHIC DEVICES AT SLAC

There are several interactive and non-interactive graphic devices available at SLAC. DPAK has been used for making displays on both, but since the general user of DPAK will only deal with the non-interactive ones, they are discussed here.

The current non-interactive devices at available at SLAC are:
1. the 10 inch CALCOMP drum plotter
2. the 29 inch CALCOMP drum plotter
3. the 16mm unsprocketed CALCOMP microfilm plotter
4. the 105mm CALCOMP microfiche plotter
5. the TEKTRONIX 4013 storage scope display - displays for this device are written out as individual members of a WYLBUR PDS; after his job has run, the user uses a member of the PDS, and then using a LIST UNNUMBERED command, displays the picture on the TEKTRONIX 4013 scope (see reference 1 for more on the WYLBUR PDS)
6. the disk PDS data set - this is not really a graphic device; it is merely a way of saving the pictures in a disk data set for future processing; see reference 2. for more information on this "device"

In general, the only thing the user must do to utilize one of these devices through DPAK is to include the appropriate DD statement with his program. If the default ddname is utilized, then the DPAK routines will take care of opening the specified device. If the user wishes however, he may explicitly open a device with the following call:

```
CALL DEVPIC(DEVICE [,DDNAME])
```

where:
- DEVICE is the character string
  - 'CALDRSM' for the 10" CALCOMP
  - 'CALDRLG' for the 29" CALCOMP
  - 'CAL16MU' for the 16mm unsprocketed microfilm
  - 'CALFICH' for the 105mm microfiche
  - 'PDS4013' for the TEKTRONIX 4013 PDS displays
  - 'PDSPDEV' for the disk PDS data set
- DDNAME is a an optional argument; if supplied, it must match the ddname on the DD statement for DEVICE.

The devices and their default ddnames are:

DEVICE	DDNAME DEFAULT
CALDRSM	CALDRSM
CALDRLG	CALDRLG
CAL16MU	CALFILM
CALFICH	CALFICH
PDS4013	PDS4013
PDSPDEV	PDSPDEV

The default DD statements for the various devices are:

For the 10" CALCOMP - 'CALDRSM'

```
//GO.CALDRSM DD DSN=&&DISP1,DISP=(NEW,PASS),VOL=SER=PLOT,
//    UNIT=T9-1600,LABEL=(1,SL),
//    DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
```

For the 29" CALCOMP - 'CALDRLG'

```
//GO.CALDRLG DD DSN=&&DISP2,DISP=(NEW,PASS),VOL=SER=PLOT,
//    UNIT=T9-1600,LABEL=(1,SL),
//    DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
```

For the 16mm unsprocketed microfilm - 'CAL16MU'

```
//GO.CAL16MU DD SYSOUT=X,
//    DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

For the 105mm microfiche - 'CALFICH'

```
//GO.CALFICH DD SYSOUT=Z,
//    DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

For the TEKTRONIX 4013 displayable PDS - 'PDS4013'

```
//GO.PDS4013 DD DSN=WYL.GG.UUU.NAME,UNIT=(SYSDA,2),
//    DISP=(NEW,CATLG),DCB=(RECFM=FB,BLKSIZE=1600,LRECL=80),
//    SPACE=(TRK,(100,10,10),RLSE)
```

This JCL will create the PDS on a scratch disk and catalog it. If the user wishes a more permanent file, he should modify the DD statement accordingly.

For a PDS - 'PDSPDEV'

```
//GO.PDSPDEV DD DSN=WYL.GG.UUU.NAME,DISP=(NEW,CATLG),
//    UNIT=(SYSDA,2),SPACE=(TRK,(10,10,3),RLSE),
//    DCB=(RECFM=U,BLKSIZE=4000)
```

Although it is not always necessary, it is a good practice to close all the graphic devices before terminating a job. This is done by the call:

```
CALL DEVCLO(UNIT)
```

where UNIT is the string representing the device to be closed. If more than one device is open (or as a general catch all), UNIT can be the string 'ALL'. This will result in all open devices being being closed.

E. OTHER USEFUL ROUTINES

Following is a description of some of the subprograms used by DPAK. They have been found to be useful for other purposes on several occasions and so are described here.

1. DCHAR

DCHAR is a LOGICAL*1 function which returns the plot character to be used for a given point. By providing his own LOGICAL*1 function the user may plot a different character for each point. The calling sequence used by the plot program is

```
LOGICAL*1 ICHR,DCHAR
ICHR=DCHAR(I)
```

where I is the index of the point to be plotted. Thus the user, by supplying his own DCHAR, has control over character choice.

2. DSCALE

DSCALE is the scaling routine used by the display routines to do automatic scaling. The algorithm used for the linear scaling is given in reference 6. The calling sequence is:

```
CALL DSCALE(F,I1,N,NINT,IYMIN,IDY,IPWR)
```

where: F is a REAL*4 function of the form F(I,J) or F(I) which gives the values to be scaled; F can be either singly or doubly indexed; if F is doubly indexed, then it is always called with J=1.

I1 is the point the scaling is to start from

N is the number of points to be scaled; if N is negative the a zero is forced into the scale

NINT is I*4 and the number of intervals to be allowed; if NINT is 0 when DSCALE is called, then DSCALE calculates the optimal number of intervals (from 2 to 15) and returns the value in NINT; if NINT=-1, then a log scale is calculated (the minimum power of 10 is returned in the variable IYMIN, and the number of decades is returned in the variable IDY)

IYMIN is the integer for the first tic mark label

IDY is the integer increment for the axis

IPWR is the power of 10 necessary to get the integerized IYMIN and IDY values to be the scale factors (not used for log scaling)

3. EQUAL

This is a logical function which compares the first N characters of two character strings STR1 and STR2. If the first N characters are the same, then EQUAL returns a value of .TRUE., otherwise it returns .FALSE.. The calling sequence is:

```
LOGICAL RESULT,EQUAL
LOGICAL*1 STR1(N),STR2(N)
RESULT = EQUAL(STR1,STR2,N)
```

4. HGHEAD

This routine sets up the title which is plotted along the right hand side of the 1-D graphic device output. If the user wishes, he may supply his own HGHEAD routine. The calling sequence necessary is:

```
CALL HGHEAD(TITLE,NCHAR)
```

where: TITLE is a string which the user fills in with the title desired
NCHAR is the number of characters to be printed

5. NTEXT

Given a string of text, this function returns the number of characters up to but not including a specified character. The calling sequence is:

```
LOGICAL*1 STRING(MAX),CHAR
INTEGER MAX
NCHARS=NTEXT(STRING,MAX,CHAR)
```

where: CHAR is the character to be searched for
MAX is the maximum number of characters to be searched

6. SET

The calling sequence for this routine is:

```
CALL SET(STR1,STR2,NCHAR)
```

where: STR1 and STR2 are arrays of at least length NCHAR bytes
SET will set the first NCHAR bytes of STR1 equal to the first NCHAR bytes of STR2.

II. HPAK

HPAK (for histogram package) is a collection of FORTRAN callable subprograms which are called by the user's program to perform convenient allocation, accumulation, and display of 1-D and 2-D histograms and 2-D scatterplots.

All the information concerning the histogram definitions, the storage for the histograms, and the storage for statistical analysis (means and moments etc.) are automatically pooled together into a single common block /HCOM/ whose length is defined by the user. This facilitates management of the histogram storage and allows for provision of dynamic storage allocation whereby one may allocate, and if required later reallocate histogram storage without recompilation. The user can optimize the use of the available space in /HCOM/ by specifying the number of bytes (1, 2, or 4) to be used per bin in a given histogram.

Accumulation of 1-D and 2-D histograms is made by calls to subroutines which automatically increment the appropriate bin of the selected histogram by a specified amount. In the case of scatterplots the accumulation call results in the coordinates being packed, buffered, and written out onto a scratch file.

The display of the histograms is done by calls to the DPAK routines. The form of the 1-D output may be at the user's option, bar histograms, points, or points connected by a line, with or without error bars. The ordinate and/or abscissa may be in linear or log form. 2-D histograms are output to the lineprinter in a tabular form that allows a variable number of characters per column entry, and replacement of absolute zero entries by blanks, so that the user may simulate scatterplots on the lineprinter. Output for 2-D histograms to the graphic devices is made in isometric form. True scatterplots are output only to graphic devices as a series of points of data at the prescribed coordinates.

A. USE OF HPAK

1. Defining the Length of /HCOM/

The first thing the user must do is to define the length (in words) of /HCOM/. This is done by including the following two statements in his program:

```
COMMON/HCOM/HCOM(N)
CALL HSIZE(N [,M])
```

The default size of N is 2000, however this will often not be enough, so the user should define it to be of sufficient length. A new user of HPAK can use 5000 words if he is initially uncertain of his storage requirement. A rule of thumb for estimating this length is given in Appendix II. The HSIZE call must be made before any other HPAK routine is called.

The optional argument M indicates the size of the common block SCPBUF (declaration COMMON/SCPBUF/NBUF,BUF(100)) which is used for the graphics. The default size for SCPBUF is NBUF=100, and this usually sufficient. However for interactive uses, a larger SCPBUF common block is usually required, and the length is set by calling HSIZE with two arguments. When specifying a larger SCPBUF and/or HCOM, the user must be sure to include the common block declaration for the larger common block(s).

2. Definition of Histograms

To define (allocate) a 1-D or 2-D histogram, a call to HDEF1 or HDEF2 respectively is made.

2.1 1-D Histogram definition:

```
CALL HDEF1(ID,MS,NB,XMIN,WIDTH,TITLE)
```

where:

ID	I*4	the histogram identifier, which is either an integer number (0<ID<10000) or a string of up to 4 characters (invalid ID's are ID=0 or any string starting with the 3 characters 'ALL'; e.g. 'ALL1', 'ALL', 'ALLA' are invalid)
MS	ANY	string of 3 characters to indicate the storage mode as follows:
	'L*1'	one byte (integer) per bin
	'I*2'	two bytes (integer) per bin
	'I*4'	four bytes (integer) per bin
	'R*4'	four bytes (real) per bin
	'E*4'	two real words (8 bytes) for storage of a value and an error per bin (described in HCUM1)
	'M*4'	two real words (8 bytes) for the storage of a value and an error(described in Appendix VIII)
	'W*4'	two real words (8 bytes) for the storage of a value and a weight(described in Appendix VII)
NB	I*4	number of bins; if NB is negative, the histogram definition is made, but space is not reserved for bin allocations; such a histogram is said to be deactivated (see HACT and HDEACT)
XMIN	R*4	value of the lower edge of the first bin
WIDTH	R*4	width of a bin
TITLE	ANY	string of up to 80 characters terminating with an '@'; this string may be broken down into sub labels of up to 32 characters each for the X and Y axes (of a graphic device plot) by the use of semi-colon characters within the string, as for example: 'MAIN LABEL; X-AXIS LABEL; Y-AXIS LABEL@' the string contains the main label followed by a semi-colon (';'), the X-axis label followed by a semi-colon, and the Y-axis label followed by the terminating '@'

For example:

```
CALL HDEF1(10,'I*2',50,1.0,.1,'HISTOGRAM NUMBER 10@')
```

will define the histogram with the identifier 10 to have 50 bins, with the low edge of the lowest bin having a value of 1.0, a bin width of .1 units, a title of "HISTOGRAM NUMBER 10" and 2 bytes of storage for each bin.

2.2 2-D Histogram and Scatterplot definitions

```
CALL HDEF2(ID,MS,NX,NY,XMINX,XMINY,WIDX,WIDY,TITLE)
```

where:

ID	I*4	the histogram identifier as described in HDEF1
MS	ANY	string of 3 (or 4) characters to indicate the storage mode; modes available are the same as in the 1-D histogram definition case; if a scatterplot of the histogram is desired, simply add an 'S' onto the string supplied in MS; i.e. 'L*1S', 'I*2S', 'I*4S', 'R*4S', 'E*4S', 'M*4S', 'W*4S'; a 2-D histogram is always accumulated (unless it was defined in deactivated mode), but the scatterplot information is only saved if the mode of storage has an 'S' on it

```
NX     I*4   number of X-bins: if NX is negative, then
             histogram is allocated in deactivated mode
NY     I*4   number of Y bins
XMINX  R*4   value of the lower edge of the lowest X-bin
XMINY  R*4   value of the lower edge of the lowest Y-bin
WIDX   R*4   width of each X-bin
WIDY   R*4   width of each Y bin
TITLE  ANY   title string as described in HDEF1
```

Examples:

```
CALL HDEF2('CTS','I*4',50,25,0.0,1.,.1,.5,'2-D HIST@')
```

will define the histogram identified by 'CTS' to have 50 X-bins each .1 units wide, 25 Y-bins each .5 units wide, and a title of "2-D HIST". It will allow four bytes of storage for each accumulation bin.

```
CALL HDEF2(102,'I*2S',-25,25,0.0,1.,0.1,.5,
          '2-D HISTOGRAM; X-AXIS ; Y-AXIS @')
```

will define 2-D histogram 102 to have 25 X-bins of width .1 with a lower bin edge of 0.0, 25 Y bins of width .5 with a lower bin value of 1.0, and a title of "2-D HISTOGRAM". Since the number of X bins is negative, the histogram will be allocated in deactivated mode. The "S" in "I*2S" specifies that a scatterplot is to be generated for a graphic device. The scatterplot will be labeled as in the 2-D histogram, but with a label on the X-axis of "X-AXIS" and a Y-axis label of "Y-AXIS".

When doing scatterplots for a graphic device, the following JCL is necessary for the scratch file on which the points are saved as they are accumulated.

```
//GO.FT19F001 DD DSN=&KIOWA,DISP=(NEW,DELETE),
//  DCB=(RECFM=VBS,LRECL=1805,BLKSIZE=3614),
//  SPACE=(TRK,(50,50),RLSE),UNIT=(SYSDA,2)
```

3. Accumulation

To accumulate a value in a 1-D histogram:

```
CALL HCUM1(ID,X,W)
```

where:

```
ID     I*4   the histogram identifier
X      R*4   coordinate for accumulation bin, numbers are
             determined so that the first bin contains all
             X from X=XMIN to, but not including
             X=XMIN+WIDTH; note also that underflows go
             into into first bin, and overflows into the
             last bin
W      R*4   weight to be accumulated (i.e. the amount by
             which the bin is to be incremented, e.g. 1.0)
```

If both a value and an error are to be accumulated (E*4 or M*4 option), then W(1) contains the value to be accumulated and W(2) contains for E*4 and M*4 the square of the error, and for W*4 the weight. E*4 histograms are accumulated as the sum of the values +/- SQRT(sum of the errors squared). See Appendix VII and VIII for a description of W*4 and M*4 respectively.

To accumulate a value in a 2-D histogram:

```
CALL HCUM2(ID,X,Y,W)
```

where:

```
ID     I*4   the histogram identifier
X      R*4   X-coordinate for accumulation
Y      R*4   Y-coordinate for accumulation
W      R*4   weight to be accumulated (i.e. the amount by
             which the bin is to be incremented); both
             weights and errors can be accumulated, as
             described in HCUM1
```

4. Output

4.1 Basic Histogram Output

To output a histogram or scatterplot:

```
CALL HOUT(ID [,UNIT])
```

where:

```
ID     I*4   the histogram identifier, or the string 'ALL';
             if ID='ALL' then all the histograms, both 1-D
             and 2-D will be output on the specified unit
UNIT   ANY   string or value which specifies the output
             device; possible unit values are: a logical
             unit number >0 and <100 to which printed
             output will be routed; in this case the unit
             number will be assigned as the printout unit
             for later HPAK subroutine calls
  'PRINTER'  the currently assigned printer unit
  'PDS4013'  WYLBUR PDS
```

'CALDRSM' 10 inch CALCOMP
'CALDRLG' 29 inch CALCOMP
'CAL16MU' the CALCOMP microfilm unit
'CALFICH' the CALCOMP microfiche
'NOUNIT' no output is made to any unit
'PDSPDEV' for the Unified Graphics PDSPDEV
if the UNIT argument is not supplied, then the output will be routed to all devices specified in the JCL, plus the lineprinter

4.2 Projection of 2-D Histograms - HSLICE

This routine allows the user to take a slice of a 2-D histogram, and then display in projection that slice as a 1-D histogram. The calling sequence is:

```
CALL HSLICE(ID,XORY [,M1 [,NS [,UNIT [,TITLE ] ] ] ])
```

where:

ID	I*4	is the identifier of the histogram to be sliced
XORY	L*1	the string 'X' or 'Y'; if XORY='X' then the slice is projected onto the X-axis; if XORY='Y' then the slice is projected onto the Y-axis
M1	I*4	index of the first bin of the slice
NS	I*4	number of bins to be summed in the slice (0 for all bins beyond M1)
UNIT	ANY	a string or value to specify the output device as in the call to HOUT; if UNIT='NOUNIT' then no output of the slice will be done, but the slice functions will be initialized to allow the user access to them (see section on HFSLC and HERSLC)
TITLE	ANY	this argument is optional; if supplied, this title will be substituted for the normal title on the output; must be a text string terminating with an '@'

The arguments M1, NS, and UNIT are optional. The defaults are:

M1 = 1
NS = total number of bins along specified axis
UNIT = 'PDS4013'

4.3 Manual Scaling of Histogram Output - HMSC

Manual scales for histogram output are set by a call to the subroutine HMSC. This results in the scale factors being saved in the common block /HCOM/. The calling sequence is:

```
CALL HMSC(ID,SCALE)
```

where: SCALE is a REAL*4 array of length 16 for 2-D histograms and of length 4 for 1-D histograms; the order of the scale factors is the same as in the SCPUT2 and SCPUT1 calls which are described in the DPAK section of this writeup
ID is the identifier of the histogram the scale factors are to be associated with

4.4 Overplotting 1-D Histogram Output

Overplotting of histogram output may be done on either the lineprinter or graphic devices. It is done by doing things in the following sequence:

1) CALL OPTPUT('OVER',.TRUE.)
2) set up other desired options via the appropriate OPTPUT calls
3) make the appropriate HOUT call
4) repeat the sequences 2) and 3) until you are through
5) CALL OPTPUT('OVER',.FALSE.)

Up to 10 histograms may be overplotted on the lineprinter, and any number on the graphic devices.

5. Example

Consider the following:

```
      INTEGER CLOCK1
      COMMON/HCOM/HCOM(7000)
      CALL HSIZE(7000)
      CALL HDEF2('SCAT','I*4S',20,25,10.,0.,1.,1.,
     1          'SIMULATED SCATTERPLOT$')
      CALL RAN1A(CLOCK1(2))
      DO 10 I=1,40
      X=I
      DO 11 J=1,25
      Y=J
      IZ=ZF2(I,J)
      IF(IZ.EQ.0) GO TO 11
      DO 12 K=1,IZ
      XD=RAN1(0)+X
      YD=RAN1(0)+Y
      CALL HCUM2('SCAT',XD,YD,1.0)
12    CONTINUE
11    CONTINUE
10    CONTINUE
C
C OUTPUT TO LINEPRINTER
C
      CALL OPTPUT('COLUMNS',1)
      CALL OPTPUT('SYMBOL',16)
      CALL HOUT('SCAT','PRINTER')
      CALL HOUT('SCAT','PDS4013')
      RETURN
      END
      FUNCTION ZF2(I,J)
      ZF=INT(35.*EXP(-.5*(I-19)**2/9.)
     1   *EXP(-5.*(J-13)**2/16.))
      RETURN
      END
```


The HOUT('SCAT','PRINTER') call gives the following:

```
          ID=SCAT SIMULATED SCATTERPLOT
            1234567891123456789212345J,J=  1, 25
            *************************
                         1111111111122222
        E  0 0123456789012345678901234Y(J).E  0
  I     X(I)+------------------------- TOTAL.E  0
  1      10.I                          I     0,SYMBOL RANGE
  2      11.I                          I     0,1 =   1-   1
  3      12.I        11122222111       I    16,2 =   2-   2
  4      13.I      112234444432211     I    38,3 =   3-   3
  5      14.I     11235678887653211    I    74,4 =   4-   4
  6      15.I    113468ACDEDCA864311   I   130,5 =   5-   5
  7      16.I    12469C*******C96421   I   197,6 =   6-   6
  8      17.I   12369C*********C96321  I   270,7 =   7-   7
  9      18.I   1247AF*********FA7421  I   321,8 =   8-   8
 10      19.I   1247B***********B7421  I   337,9 =   9-   9
 11      20.I   1247AF*********FA7421  I   321,A =  10-  10
 12      21.I   12369C*********C96321  I   270,B =  11-  11
 13      22.I    12469C*******C96421   I   197,C =  12-  12
 14      23.I    113468ACDEDCA864311   I   130,D =  13-  13
 15      24.I     11235678887653211    I    74,E =  14-  14
 16      25.I      112234444432211     I    38,F =  15-  15
 17      26.I        11122222111       I    16,**=  16-****
 18      27.I                          I     0
 19      28.I                          I     0
 20      29.I                          I     0
  I     X(I)+-------------------------
                     11122222111
                  1257148245428417521
TOTAL.E  0 0005461529825552892516450     2429
```

Figure 9

The HOUT('SCAT','PDS4013') call gives the following:

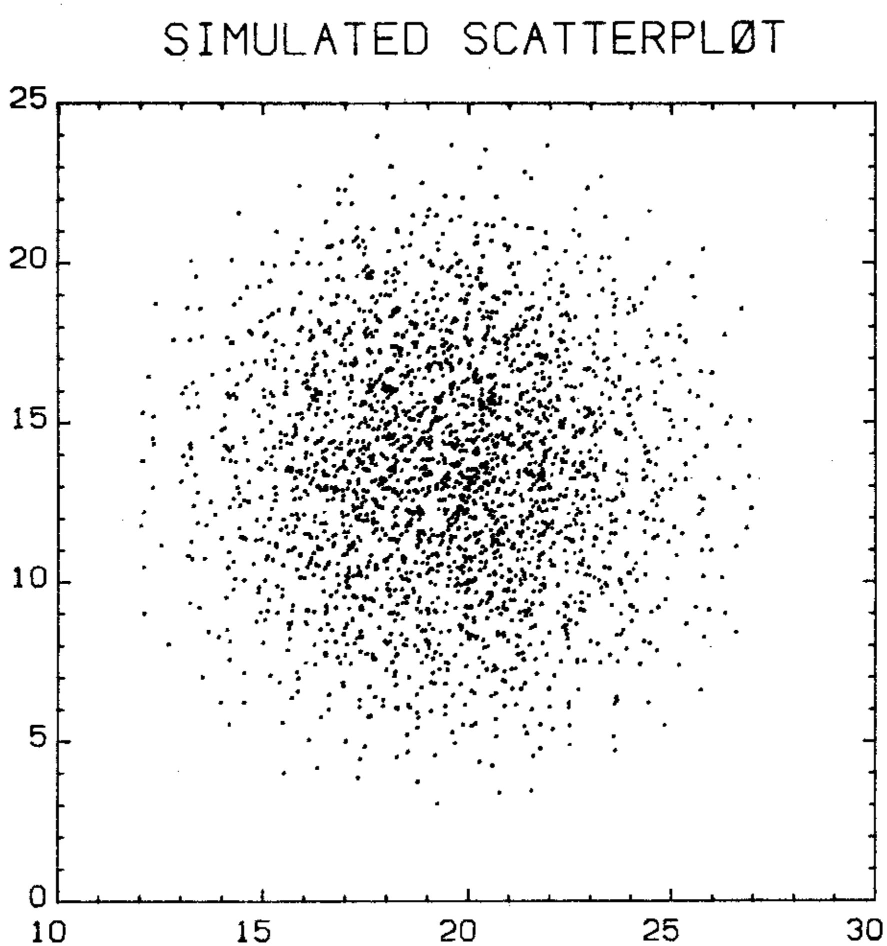

Figure 10

B. LOCATING HISTOGRAMS IN /HCOM/ - NOSUCH

NOSUCH is a logical function for locating a particular histogram in /HCOM/. HCOM(3) points to the location in /HCOM/ of the histogram last accessed. If the user wishes to access a histogram on his own, he must set up HCOM(3). To do this use a call like:

```
C N IS THE LENGTH OF /HCOM/
      COMMON/HCOM/HCOM(N)
      INTEGER HCOM
      LOGICAL NOSUCH
      IF(NOSUCH(ID)) RETURN
        ...ID found - continue processing...
```

If NOSUCH is returned with a value of TRUE, then no histogram with that ID is currently defined. If NOSUCH returns with a value of FALSE, then the histogram exists and HCOM(3) points to its record. A detailed description of a histogram record is contained in Appendix II.

C. ACCESSING HISTOGRAM CONTENTS

1. HFSLC and HEFSLC

These REAL*4 functions allow the user access to contents of the sliced bins and their errors. To intialize these functions the following call must be made immediately before using them:

```
CALL HSLICE(ID,XORY,M1,NS,'NOUNIT')
```

then one may use HFSLC(I) as a function to get the contents of sliced bins and HEFSLC(I) as its error.

2. HGET

This routine allows the user to fetch the current specifications of histograms or print them on the currently defined lineprinter unit. The calling sequence is:

```
CALL HGET(ID,IOPT [,ND,NB,MS,XMIN,WIDTH [,IH [,IE
              [,TITLE]]]])
```

where:

ID	I*4	is the identifier of the histogram whose specifications are desired; if ID='ALL', then the specifications of all the currently defined histograms are printed on the currently defined lineprinter unit
IOPT	ANY	a string which specifies whether the specifications are to be be printed or returned to the user; option strings are:
'PRINTER'		print the specifications on the currently defined lineprinter unit; the remaining arguments are not necessary
'GET'		return the specifications to the user in the variables he has provided as follows:
ND	I*4	number of dimensions (1 or 2)
NB	I*4	number of bins; if ND=2 then NB must be an array of length 2 and the number of X bins is returned in NB(1), while the number of Y bins is returned in NB(2)
MS	I*4	mode of storage; a character string specifying the mode of storage as in the HDEF1 and HDEF2 calls is returned
XMIN	R*4	the value of the lowest bin; if ND=2 then XMIN must be an array of length 2, and the lowest X bin value is returned in XMIN(1) while the lowest Y bin is returned in XMIN(2)
WIDTH	R*4	the bin width; if ND=2 then WIDTH must be an array of length 2, and the X bin width is returned in WID(1) while the Y bin width is returned in WID(2)

Note that when calling HGET with IOPT='GET', the arguments ND, NB, MS, XMIN, and WID are mandatory. The following will return additional information, but are not necessary if the information is not desired:

IH	I*4	is an optional argument; if supplied, it will contain the index value with 1 subtracted from it of the place in HCOM where the histogram array is stored; i.e. HCOM(IH+1) will get the user the first bin value
IE	I*4	is an optional argument; if supplied, and MS='E*4' then it will contain the index value with 1 subtracted from it of the place in HCOM where the error array is stored; i.e. HCOM(IE+1) will contain the square of the error on the first bin
TITLE	ANY	will contain the character string which was passed as a title

Examples of HGET calls:

a) `CALL HGET('ALL','PRINTER')`

will print out specifications of all defined histograms.

b) `CALL HGET(1,'GET',ND,NB,MS,XMIN,WIDTH,IH)`

will return the number of dimensions, number of bins, mode of storage, minimum bin value, and the width of the bins for the histogram identified by the integer 1. Also if say MS='L*1' and the histogram is a 1-D histogram, then we may obtain the histogram bin contents as follows:

```
       COMMON/HCOM/M(2000)
       LOGICAL*1 L(1),H(NB)
       EQUIVALENCE(L(1),M(1))
       DO 200 I=1,NB
       H(I)=L(IH*4+3+I)
 200   CONTINUE
```

c) `CALL HGET(1,'GET',ND,NB,MS,XMIN,WIDTH,IH,IE,TITLE)`

will return the number of dimensions, number of bins, mode of storage, minimum bin value, the width of the bins, the pointer to the histogram in HCOM, the pointer to its errors (if MS='E*4'), and its title.

d) Note that the following call will not work:

```
CALL HGET(1,'GET',ND,NB,MS,XMIN,WIDTH,TITLE)
```

that is the title will not be returned. In order for the title to be returned, the dummy arguments IH and IE must be supplied.

3. HSPACE

The subroutine HSPACE prints out on the error unit number the number of words left unused in /HCOM/ at any point. The call is simply:

```
CALL HSPACE
```

4. HSUM and HSUMS

These functions enable the user to output or fetch the statistics that have been accumulated for a particular histogram. The calling sequence is:

```
LOGICAL HSUM,EXIST
EXIST=HSUM(ID,IOPT [,D])
```

where the result .TRUE. is returned if the histogram exists, and .FALSE. is returned otherwise.

The arguments are:

ID	I*4	the histogram identifier
IOPT	ANY	a string or value which specifies whether the statistics are to be printed or to be returned to the user; the printout occupies 2 lines if it is a 1-D histogram, and 5 lines if it is a 2-D histogram; if 0<IOPT<100 then logical unit IOPT is used to print the statistics

'PRINT' print statistics for the histogram on the currently defined lineprinter unit: the following statistics are included:
if it is a 1-D histogram: # calls; sum of weights; <x>; d<x>; o(x)=standard deviation in x; do(x); g(x)=skew (third moment about <x>/o**3); dg(x); xlo=lowest value of x; xhi=highest value of x; r=resolution=o/<x>; dr;
if it is a 2-D histogram: ylo; yhi; <y>; d<y>; o(y); do(y); g(y); dg(y);
covariance matrix: o**2(x) cov(x,y)
cov(x,y) o**2(y)
correlation coef. p=cov(x,y)/(o(x)*o(y));
dp is the error in correlation coefficient;
major and minor axes semi-diameters of the covariance ellipse D1, D2; and the angle of the ellipse a; (see Ref 4 for the formulations of the various statistics)

'GET' This will return the statistics to the user in array D;
for the 1-D histogram, D should be of dimension REAL*4 D(10), and the statistics returned are:
D(1) = # calls
D(2) = sum of weights
D(3) = <x>
D(4) = o(x)
D(5) = xlo
D(6) = xhi
D(7) = g(x)
D(8) = dg(x)
D(9) = underflow
D(10) = overflow
for the 2-D histogram, D should be of dimension REAL*4 D(23), and the returned statistics are:
D(1) through D(8) are the same as the 1-D case
D(9) = <y>
D(10) = o(y)
D(11) = ylo
D(12) = yhi
D(13) = g(y)
D(14) = dg(y)
D(15) = o**2(x)
D(16) = cov(x,y)
D(17) = cov(x,y)
D(18) = o**2(y)
D(19) = p
D(20) = dp
D(21) = D1
D(22) = D2
D(23) = a

D R*4 argument required if using the 'GET' option to return the statistics to the user

For a slice of a 2-D histogram the call is:

```
LOGICAL EXIST,HSUMS
EXIST=HSUMS(ID,IOPT [,D])
```

This call performs the same function for the sliced histogram as HSUM does for the 1-D case. Note that this call must be preceeded by a call to HSLICE.

Example:
If the user wishes to normalize a histogram by the sum of the weights, he might use:

```
REAL*4 D(23)
LOGICAL HSUM
IF(.NOT.HSUM(ID,'GET',D)) RETURN
CALL HNORM(ID,D(2))
```

5. HX, HY, HYE, H2Y, HZ, and HZE

These functions are provided to allow the user access to the individual histogram bins. A call to NOSUCH for the particular histogram the user wishes to access must be made before any of these functions are called. The functions and their values are as follows:

HX(i)	REAL*4	returns the X-coordinate value of the ith bin (1-D or 2-D)
HY(i)	REAL*4	returns the value of the ith bin in real form (1-D)
HYE(i)	REAL*4	returns the error of the ith bin in real form (1-D)
H2Y(j)	REAL*4	returns the value of the jth bin (2-D)
HZ(i,j)	REAL*4	returns the value of the (i,j)th 2-D histogram bin in real form
HZE(i,j)	REAL*4	returns the error on the (i,j)th bin of the 2-D histogram in real form

6. IHX and IHY

These integer functions return to the user the bin index of a value. A call to NOSUCH(ID) where ID is the desired histogram must preceed the IHX and IHY call. The calling sequences are:

J=IHX(VALX) where VALX is the X-coordinate whose bin number is desired (may be either a 1-D or 2-D histogram)

K=IHY(VALY) where VALY is the Y-coordinate of a 2-D histogram whose bin index is desired

If the coordinate lies outside the histogram range, then the index of the first(underflow) or last(overflow) bin is returned.

An example of how these might be used is: Given a 2-D histogram (ID='TST') with X-axis range [0.0,10.0] and Y-axis range [100.0,200.0], the user wishes a projection along the X-axis of the bins in the Y range of [125.,150.]. The following set of calls would accomplish this:

```
      LOGICAL NOSUCH
      IF(NOSUCH('TST'))RETURN
      CALL HSLICE('TST','X',IHY(125.),IHY(150.)-IHY(125.)+1,
     1            'PDS4013')
```

D. EDITING HISTOGRAMS

In this section are outlined the calls which may be used to modify a histogram definition or its contents.

1. HACT and HDEACT

All histograms have two possible states. They may be either active or inactive. An active histogram is one that is fully defined and has storage space with it for accumulations. An inactive histogram is one which has its definition stored in the common block /HCOM/ but has no storage space for accumulations allocated to it (therefore accumulations are not done). To activate a histogram:

```
CALL HACT(ID)
```

where ID is the histogram identifier. To deactivate a histogram, that is release its accumulation space:

```
CALL HDEACT(ID).
```

If ID='ALL' then all the histograms are activated or deactivated.

2. HCLR

This subroutine allows the user to clear a specific histogram. The calling sequence is:

```
CALL HCLR(ID)
```

where:

ID	I*4	the identifier of the histogram to be cleared or the string 'ALL': if ID='ALL', then all the histograms are cleared

This routine does not delete the histogram specifications, but merely zeros all bin values and statistics.

3. HDEL

This subroutine deletes a histogram. The space is then freed and may be reallocated if desired. The calling sequence is:

```
CALL HDEL(ID)
```

where:

ID	I*4	the identifier of the histogram to be deleted; if ID='ALL', then all histograms are deleted

4. HNORM

This subroutine enables the user to set a normalizing factor for a histogram (the histogram contents are not changed). During display, the histogram contents are divided by this factor.

```
CALL HNORM(ID,VALUE)
```

where:

ID	I*4	is the identifier of the histogram to be normalized
VALUE	R*4	the normalizing factor for the histogram

5. HSET1

This subroutine allows the user to set a specific bin of a 1-D histogram to a desired value. The calling sequence is:

```
CALL HSET1(ID,X,VAL)
```

where:

ID	I*4	is the identifier of the histogram whose bin is to be set
X	R*4	the coordinate of the bin whose value is to be set
VAL	R*4	the value to be stored in the bin; if MS=E*4 then VAL must be a two word array (see HCUM1)

6. HSET2

This subroutine allows the user to set a specific bin of a 2-D histogram. The calling sequence is:

```
CALL HSET2(ID,X,Y,VAL)
```

where:

ID	I*4	is the identfier of the histogram whose bin is to be set
X	R*4	the X-coordinate of the bin to be set
Y	R*4	the Y-coordinate of the bin to be set
VAL	R*4	the value to be stored in the bin; if MS=E*4 then VAL must be a two word array (see HCUM1)

E. OVERWRITING DEFAULT OPTIONS - HOPTN

HOPTN allows the user to dynamically override the default options that control the accumulation and output of the histograms. The call is:

```
CALL HOPTN(IOPT,VALUE [,ID])
```

where:

IOPT	I*4	is a string specifying the option to be set
VALUE	VARIOUS	the value which the option being specified is expecting
ID	I*4	optional argument used for some calls

The following table gives IOPT values, VALUE type expected, the meaning of the VALUE, and the default.

<u>Options for controling the accumulation of 1-D AND 2-D histograms</u>

IOPT STRING	DEFAULT	VARIABLE TYPE OF VALUE	VALUE MEANING
'EXCLUDE'	FALSE	L*4	the calling sequence for this option is: CALL HOPTN('EXCL',LVAL,ID) where: LVAL L*4 indicates whether to exclude underflows and overflows; ID I*4 the ID of the histogram to be affected; if ID='ALL', then all 1-D and 2-D histograms are set as indicated by LVAL. if FALSE for a given ID, then all values are used in the statistical calculations; if TRUE then overflows and underflows are not included
'INTEG'	FALSE	L*4	when INTEG is TRUE, the running sum of the bins is plotted instead of the normal histogram
'HSUMS'	FALSE	L*4	if true, indicates that a call to HSUM is to be made after the histogram has been output; the calling sequence is: CALL HOPTN('HSUM',LVAL,ID) where: LVAL is an L*4 value to indicate the state the option is to be set to; ID is the histogram identifier; if ID='ALL' then the option is set for all histograms
'MTBIN'	0	I*4	the error on an empty bin; this is an integer which is converted to a real if the histogram binning is real; the VALUE is either 0 or 1 (for 0.0 or 1.0 in the real case)
'STAT'	TRUE	L*4	when STAT is true the statistics necessary for the HSUM calculations are built up as the histogram is accumulated; if STAT is FALSE they are not and therefore an HSUM call will not produce any statistics; the calling sequence is: CALL HOPTN('STAT',LVAL,ID) where: LVAL is an L*4 value indicating the state the option is to be set to; ID is the identifier of the histogram to be affected; if ID='ALL' then all histograms are so set

APPENDIX I - SYNTAX USED FOR DESCRIBING SUBROUTINE ARGUMENTS, ETC.

In general the standard FORTRAN convention applies for determining the type of variables assigned to symbol names (i.e. I-N are integers and all others are reals).

Data types are described as:

L*1	LOGICAL*1	logical variable (used as an integer) occupy one byte
I*2	INTEGER*2	integer variable, occupy two bytes
I*4	INTEGER*4	integer variable, occupy four bytes
L*4	LOGICAL*4	logical variable, occupy four bytes
R*4	REAL*4	real variable, occupy 4 bytes
R*8	REAL*8	real variable, occupy 8 bytes
ANY		means data type is irrelevant

Optional arguments to subroutine calls are enclosed in square brackets. For example:

```
    CALL DEVPIC(DEVICE [,DDNAME])
```


APPENDIX II - STRUCTURE OF /HCOM/

The work area common block /HCOM/ must be defined by the user to be of sufficient length. In general, the length N should be:

$$N = 10 + \sum_{i=1}^{M} \left(52 + \frac{NX(i)*MAX(NY(i),1)*B(i)}{4}\right)$$

where:
- M = number of histograms
- NX(i) = number of X bins in ith histogram
- NY(i) = number of Y bins in ith histogram
- MAX = FORTRAN maximum function
- B(i) = number of bytes in the storage mode for the ith histogram

/HCOM/ is set up as a linked list with a main header section which contains a description of the minor records where the individual histogram specifications and accumulations are kept. The overall structure of /HCOM/ is as follows:

HCOM WORD	TYPE	CONTENTS
1	I*4	total number of currently booked histograms
2	I*4	number of header words in each record for the individual 2-D histogram specifications
3	I*4	pointer to the record of the histogram most recently accessed
4	I*4	maximum number of words in /HCOM/
5	I*4	pointer to the beginning of free space in /HCOM/
6	I*4	pointer to the start of the first record; this is equal to M+1 where M=number of words in the header section
7	I*4	total number of booked scatterplots
8	I*4	word containing the unit number for the scatterplotting scratch file in the left half

9	I*4	number of header words in each record used to hold the Y bin statistics for the 2-D histograms

10 through N=HCOM(6)-1 are currently reserved for the error processing subsystem

- -

Let N=HCOM(3) (HCOM(3) points to a record of one of the histograms). If this is a 1-D histogram, its record has the following structure:

HCOM WORD	TYPE	CONTENTS
N	I*4	pointer to the next consecutive histogram record
N+1	I*4	the histogram identifier
N+2	4(L*1)	mode of storage as follows: VALUE MODE OF STORAGE 1 L*1 2 I*2 3 I*4 4 R*4 5 R*4 6 R*8
N+3		not used
N+4		not used
N+5	I*4	number of X bins
N+6	R*4	lowest X bin value
N+7	R*4	width of the X bins
N+8	I*4	number of Y bins
N+9	R*4	not used
N+10	R*4	not used
N+11	I*4	pointer to the scale factors
N+12	I*4	pointer to the title
N+13	I*4	pointer to the accumulation bins
N+14	I*4	total number of bins

N+15	R*4	normalization value
N+16		not used
N+17	R*4	number of calls
N+18	R*4	sum of weights
N+19	R*4	lowest X value binned
N+20	R*4	highest X value binned
N+21	R*8	first moment (if R*8 RCOM.. address as RCOM(HCOM(3)/2+11))
N+23	R*8	second moment (if R*8 RCOM.. address as RCOM(HCOM(3)/2+12))
N+25	R*8	third moment (if R*8 RCOM.. address as RCOM(HCOM(3)/2+13))
N+27	R*8	sixth moment (if R*8 RCOM.. address as RCOM(HCOM(3)/2+14))

N+HCOM(2)-HCOM(9) through HCOM(N)-1 is the storage area for the scale factors, title, and the accumulation bins.

If this histogram is a 2-D histogram, then the structure of the record is the following:

HCOM WORD	TYPE	CONTENTS
N through N+8 are the same as in the 1-D case		
N+9	R*4	the low Y bin value
N+10	R*4	the width of the Y bins
N+11 through N+15 are the same as in the 1-D case		
N+16	I*4	the scatterplot number: if no scatterplot is being created for this 2-D histogram, then this is 0
N+17 through N+28 are the same as in the 1-D case		
N+29	R*4	lowest Y value binned

LMODE(34)	FR	L*1	TRUE	frame
LMODE(35)	AX	L*1	0	auto set X tics
LMODE(36)	AY	L*1	0	auto set Y tics
LMODE(37)	XT	L*1	8	number of X tic marks
LMODE(38)	YT	L*1	8	number of Y tic marks
LMODE(39)	XZ	L*1	FALSE	force a zero on X-axis
LMODE(40)	YZ	L*1	FALSE	force a zero on Y-axis
MODE(11)	DI	I*4	2000	default /HCOM/ size
MODE(12)	BU	I*4	100	default /SCPBUF/ size
MODE(13)	SL	L*4	FALSE	optional slice title flag
MODE(14)	SP	I*4	FALSE	slice flag
MODE(15)		I*4	FALSE	slice flag
MODE(16)	SD	I*4	0	slice direction 0 is X direction 1 is Y direction
MODE(17)	SW	I*4	0	width of slice
MODE(18)	SB	I*4	0	beginning of slice
MODE(19)	ID	I*4	0	ID
MODE(20)		I*4	0	not used
MODE(21)		I*4	0	not used
MODE(22)		I*4	0	not used
MODE(23)	OU	I*4	6	lineprinter output unit
MODE(24)	CE	L*4	TRUE	bin alignment flag
LMODE(97)	GR	L*4	FALSE	graticules on LPUT1 plots
LMODE(98)		L*1	FALSE	not used
LMODE(99)	IT	L*1	0	# of intervening tics (GRUT1)
LMODE(100)	EJ	L*1	TRUE	page eject
MODE(26)	EU	I*4	6	logical unit for error messages
MODE(27)	SI	R*4	1.0	size of X-axis (GRUT1)
MODE(28)	SI	R*4	1.0	size of Y-axis (GRUT1)
MODE(29)	OF	R*4	0.0	X-axis offset
MODE(30)	OF	R*4	0.0	Y-axis offset
MODE(31)	EN	I*4	0	graphic element number
MODE(32)	DU	I*4	-1	buffer dump flag for scatterplots
MODE(33)	HT	I*4	0	HTRAP counter
MODE(34)	AI	I*4	0	ID as it is stored in HCOM
MODE(35)	MT	I*4	0	error value for a zero bin
LMODE(141)	SY	L*1	Z0A	symbol range (LPUT2)
LMODE(142)	ZA	L*1	0	ZAUTO flag (LPUT2)
LMODE(143)	LE	L*1	Z85	length (LPUT2)
LMODE(144)	XY	L*1	0	XYAUTO flag (LPUT2)
LMODE(145)	CO	L*1	Z04	# OF columns (LPUT2)
LMODE(146)	TA	L*1	FALSE	TAUTO flag (LPUT2)
LMODE(147)	SC	L*1	FALSE	scientific notation (LPUT2)
LMODE(148)	PA	L*1	60	lines per page (LPUT2)
MODE(38)	#G	I*4	0	number of open graphic device
MODE(39)	DE	I*4	PDS4	default output device
MODE(40)		I*4	013	continuation of MODE(39)

APPENDIX IV - ERROR CONDITIONS - PROBABLE CAUSES AND POSSIBLE CURES

The following is a list of some of the problems DPAK and HPAK users have had. If the user's program should abend or not work correctly, I suggest looking through this list.

1. OC6 ABEND - user has supplied a subroutine, entry point or common block with a name that is the same as one of the DPAK or HPAK subroutine, common block, or entry point names.

2. OC1 or OC4 - the common block /HCOM/ or /SCPBUF/ is not as long as the user specified it would be in his HSIZE call.

3. The message 'SCALING PROBLEM WITH THIS PLOT' - user has specified that manual scales are to be used, but failed to set them up; note that manual scales for histogram output via a HSLICE or HOUT call are set up by calling HMSC; DISP1A or DISP1 scales are set up via a call to SCPUT1, and DISP2 and DISP2A scales are set up via a call to SCPUT2.

4. Incomplete picture on graphic device - user has used the wrong UGXERR routine. Check that the one you have used is from the HPAK library.

5. ABEND code of 806 - user has failed to provide the go step library for the Unified Graphics routines.

6. Error message saying 'MISSING DD CARD' - user has failed to provide the JCL for the scatterplot scratch file.

7. Divide checks, overflows, or underflows in HCOM - check that values passed for accumulation are valid (REAL*4 values).

APPENDIX V - SUBROUTINES ENTRY POINTS AND COMMON BLOCK NAMES USED IN DPAK AND HPAK

Often one of the problems that arises in using a large program is accidently using one of its CSECT names in the user's program. In the following list are the common block, subroutine, and entry point names used in DPAK and HPAK.

PROGRAM NAME	ALIASES	DESCRIPTION
BUFRED		reads back scatterplot information
DCHAR		returns plot character
DEVPIC		opens graphic device
	DEVCLO	closes graphic device
	DEVNAM	returns name of graphic device
	DEVNUM	returns identifier for graphic device
DEVSET		activates output destination
DISP0		functional interface for GRUT1 and LPUT1
DISP1		function plotting routine
DISP1A		array plotting routine
DISP2		2-D function plotting routine
DISP2A		2-D array plotting routine
DLAB		partitions title
DPPICT		outputs graphic device picture
DSCALE		scaling routine
DSETOK		sets up labels for the scatterplot
DTAB		table formatting subprogram
DUGPUT		outputs graphic element
DUSETS		sets up slicing function
	DUESLC	slice error function
	DUFSLC	slice function
	DUSLIM	sets up slice limits
	DUSTIT	sets up slice title
	DUSTXT	returns slice title
	HERSLC	HPAK alias for slice error function
	HFSLC	HPAK alias for slice function
	DUTITR	sets up title for slicing
DUTXP		calculates an exponent
DUT1		1-D function plotting routine
DUT2		2-D functions plotting routine
EQUAL		compares two character strings
GRISO		internal 2-D plot function
GRUT0		part of 1-D graphics routine
GRUT1		main 1-D graphics routine
GRUT2		main 2-D graphics routine
GTMAIN		routine which gets core
HACT		activates a histogram
	HDEACT	deactivates a histogram
HCBLNK		clears blanks out of a character string
HCLR		clears a histogram

PROGRAM NAME	ALIASES	DESCRIPTION
HCUM		accumulates a histogram
	HCUM1	accumulates a 1-D histogram
	HCUM2	accumulates a 2-D histogram
HDEF		defines a histogram
HDEF2		defines a 2-D histogram
	HDEF1	defines a 1-D histogram
HDEL		delete a histogram
HEXALL		outputs all histograms
HGET		get histogram definitions
HGHEAD		header for right side of graphic plot
HGR1		dummys for backward compatibility
	HGGET	
	HGGET2	
	HGRUT1	
	HGRUT2	
	HGR2	
	HUNIT	
	HGSET	
	HGSET2	
	HOPEN	
	HSCALE	
HMSC		sets manual scales for histogram
HNORM		normalizes a histogram
HOPTN		HPAK option setting routine
HOPUS		overplots multiple histograms
	HOPUS0	passes histogram id's to hopus
HOUT		outputs histograms
HSET1		sets a bin in a 1-D histogram
	HSET2	sets a bin in a 2-D histogram
HSIZE		sets histogram space size
HSLICE		outputs a slice of a histogram
HSPACE		tells how much space is used
HSUM		outputs histogram statistics
	HSUMS	outputs slice statistics
HTODAY		returns date in printable format
HX		returns X value for a bin
	HLA	bin function for lineprinter plot
	HLERR	bin error for lineprinter plot
	HLX	X value for lineprinter plot
	HLO	entry point for id's
	HY	returns bin value
	HYE	returns error on bin value
	HZ	2-D bin value
	HZE	2-D error value
	H2Y	2-D Y bin value
IHX		returns bin number for an X value
IHY		returns bin number for a Y value
I4TOA		takes integer into character string
JOBPRM		returns job name
LCOMP		compares two LOGICAL*1 values
LPHEAD		header routine for LPUT1
	LPTAIL	trailer routine for LPUT1

PROGRAM NAME	ALIASES	DESCRIPTION
LPUT1		1-D lineprinter output routine
LPUT2		2-D lineprinter output routine
LP1OUT		outputs line of text for LPUT1
LP1SMO		zeros sum in LPUT1
	LP1SUM	sums Y values for LPUT1
LP1VAR		sets up variable format for LPUT1
NARGS		counts number of arguments to a call
NOGRUT		dummies out graphics calls
NOLPUT		dummies out lineprinter output routine
NOSCAT		dummies out scatterplot routines
NOSUCH		locates a histogram
NTEXT		counts characters in a string
NWPAGE		puts out page heading
OPTGET		gets output option
	OPTIDX	internal option identifier
OPTPUT		sets an option
PLFUN1		function fixer-upper for LPUT1
	PLFUPY	entry point for multiple functions
PLOT2A		2-D function plotting routine
PLSET1		1-D array to function routine
	PL1FX	X array to function
	PL1FY	Y array to function
	PL1FYE	error array to function
PLSET2		2-D array to function translator
	PL2FX	X array to function
	PL2FY	Y array to function
	PL2FZ	Z array to function
	PL2FZE	error array to function
PXBOX		scatterplot output routine
PXBUFI		scatterplot output routine
PXCURV		scatterplot output routine
PXHSET		scatterplot output routine
PXPLOT		scatterplot output routine
PXSORT		scatterplot output routine
PZDOT		scatterplot output routine
PZELEM		scatterplot output routine
PZLINE		scatterplot output routine
PZNEXT		scatterplot output routine
PZTEXT		scatterplot output routine
RDJFCB		routine to check for ddcard presence
ROUND		scatterplot output routine
RTOIE		real number to integer and exponent
SCBUF		buffers scatterplot information
	SCBDMP	dumps final scatterplot information
SCGET1		returns last used 1-D scale factors
	SCPUT1	sets up 1-D output scale factors
SCGET2		returns 2-D output scale factors
	SCPUT2	sets up 2-D output scale factors
SCOUT		outputs the scatterplots
SET		sets one character string to another
UGXERR	HPUGER	error processing routine for U.G.
UPLOT	HPUPLT	dummy for user overplot routine
VPRINT		scatterplot output routine

APPENDIX VI - OVERPLOTTING OF USER SUPPLIED FUNCTIONS ON SCATTERPLOTS AND 2-D HISTOGRAM LINEPRINTER OUTPUT

Facilities exist in HPAK for overplotting of a user supplied function on scatterplots and 2-D lineprinter histogram output. The format of the function is:

```
REAL FUNCTION UPLOT(X,Y,ID)
```

where:

X	R*4	the X position the function is to be calculated for
Y	R*4	the Y position the function is to be calculated for
ID	I*4	the histogram identifier of the scatterplot/2-D histogram currently being plotted

The calling program assumes the function UPLOT changes sign at those values of X and Y that define the curve. Thus, if the curve can be defined by

$$f(x,y)=0$$

where f is a functional form of X and Y, then the FORTRAN statement

```
UPLOT=f(X,Y)
```

will serve to define the curve for the calling program. If the conditions which define the curve cannot be expressed in a simple closed form, the user can set UPLOT=+1.0 or -1.0, depending upon whether the point (X,Y) is inside or outside the curve.

The calling program calls UPLOT for each point. Thus, for those plots which are not to have a curve, the user should set UPLOT=1.0 before returning.

If several curves are desired on a single scatter plot, then the user should set

$$\text{UPLOT} = \prod_{i=1}^{NC} Fi(X,Y) \ ,$$

where NC different curves are to be superimposed on the scatter plot and Fi(X,Y)=0 defines the ith curve. Note that for graphic devices each curve must close back on itself, also the curve must not intersect itself, though a curve may intersect other curves on the plot any number of times. If for a given scatterplot/2-D histogram there is no function to be overplotted simply set UPLOT=HNONE before returning where HNONE='NONE'. The function is overplotted on both the lineprinter output and the graphic output.

For example:

Suppose three 2-D histograms have been defined with ID's of 'PRO', 'DEU', and 3 respectively, and the user wishes to overplot a function on the 2-D histogram identified by 3. The following is an example of the function format:

```
      REAL FUNCTION UPLOT(X,Y,ID)
      REAL HNONE/'NONE'/
      IF (ID.NE.3) GO TO 10
C     CALCULATE FUNCTION HERE AND RETURN
           ...
      RETURN
C NO OVERPLOT
10    UPLOT=HNONE
      RETURN
      END
```


APPENDIX VII - DESCRIPTION OF W*4 MODE OF STORAGE

The W*4 mode of storage is used for building histograms by accumulating the counts/bin and the weights/bin. More explicitly we may define this as follows. Let us select the jth bin and let V(i)= value of the jth bin after accumulating i values in that bin, also d(i)= error of the jth bin after accumulating i values in that bin. Let us suppose we have a call to HCUM1

```
    W(1)= NCOUNTS
    W(2)= WEIGHT
    CALL HCUM1(ID,RJ,W)
```

where RJ indicates the jth bin.

then

V(0)= 0, dV(0)= 0

and

V(i)= V(i-1) + W(1) = sum of counts

dV(i)= dV(i-1) + W(2) = sum of weights

At output time

V(N)/dV(N)+/- SQRT(V(N))/dV(N)

is output for this jth bin where N indicates that the bin has been accumulated into N times.

The N*4 mode is used for building histograms by accumulating the weighted means and errors per bin. Using the same notation as for W*4, then a call to HCUM1 will appear as

```
    W(1)= value
    W(2)= (error in value)**2
    CALL HCUM1(ID,RJ,W)
then
    V(0)=0, dV(0)=0
    dV(i)**2= 1/(1/W(2)+ 1/dV(i-1)**2)
    V(i)= (W(1)/W(2)+V(i-1)/dV(i-1)**2)*dV(i)**2
At output time
    V(N)+/-dV(N) is output
```


PLOTTING CALLS:

1-D plotting calls:

```
 CALL DISP1(FX,FY,FERR,N,TITLE [,UNIT [,FB [,FC [,FD [,FE
          [,FF [,FG [,FH [,FP [,FQ]]]]]]]]]])
 CALL DISP1A(X,Y,ERR,N,TITLE [,UNIT [,B [,C [,D [,E [,F
          [,G [,H [,P [,Q]]]]]]]]]])
 CALL DUT1(FX,FY,FERR,N,TITLE,SCALE)
```

2-D plotting calls:

```
 CALL DISP2(FX2,FY2,FZ2,FERR2,NX,NY,TITLE [,UNIT])
 CALL DISP2A(X,Y,Z,ZERR,NX,NY,TITLE [,UNIT])
 CALL DUT2(FX,FY,FZ,FZERR,NX,NY,TITLE,SCALE)
```

SETTING AND RETRIEVING MANUAL SCALES:

for the 1-D case:

```
  CALL SCPUT1(SCALE) and CALL SCGET1(SCALE)
```

for the 2-D graphics case:

```
  CALL SCPUT2(SCALE) and CALL SCGET2(SCALE)
```

OPTPUT AND OPTGET CALLS (WITH DEFAULTS):

set default output unit: CALL OPTPUT('DEVICE','PDS4013')

options for 1-D and 2-D lineprinter:

```
  CALL OPTPUT('EJECT',.TRUE.)
  CALL OPTPUT('EUNIT',6)
  CALL OPTPUT('OUNIT',6)
```

options for the 2-D lineprinter output:

```
  CALL OPTPUT('COLUMNS',4)
  CALL OPTPUT('LENGTH',133)
  CALL OPTPUT('PAGE',60)
  CALL OPTPUT('SCIENTIFIC',.FALSE.)
  CALL OPTPUT('SYMBOL',10)
  CALL OPTPUT('TAUTO',.FALSE.)
  CALL OPTPUT('XYAUTO',.FALSE.)
  CALL OPTPUT('ZAUTO',.FALSE.)
```

options for both the 1-D and 2-D lineprinter and graphic device output:

```
  CALL OPTPUT('MANUAL',.FALSE.)
  CALL OPTPUT('OMIT',.FALSE.)
```

options for 1-D lineprinter and graphic device output and 2-D graphic device output:

```
  CALL OPTPUT('YZERO',.FALSE.)
```

options for the 1-D lineprinter and graphic device output:

```
CALL OPTPUT('CHAR',' ')
CALL OPTPUT('ERROR',.TRUE.)
CALL OPTPUT('HIST',.FALSE.)
CALL OPTPUT('LOGS',.FALSE.)
CALL OPTPUT('YLOG',.FALSE.)
CALL OPTPUT('YMAN',.FALSE.)
```

options for the 1-D graphic device output:

```
CALL OPTPUT('CENTER',.TRUE.)
CALL OPTPUT('ITICS',0)
CALL OPTPUT('OFF',0.0,0.0)
CALL OPTPUT('OVER',.FALSE.)
CALL OPTPUT('RESCALE',.FALSE.)
CALL OPTPUT('SIZE',1.0,1.0)
CALL OPTPUT('XLOG',.FALSE.)
CALL OPTPUT('XMAN',.FALSE.)
CALL OPTPUT('XTIC',8)
CALL OPTPUT('XZERO',.FALSE.)
CALL OPTPUT('YTIC',8)
```

options for the 1-D and 2-D graphic device output:

```
CALL OPTPUT('FRAME',.TRUE.)
```

options for the 2-D graphic device output:

```
CALL OPTPUT('XDIR',.FALSE.)
```

options for the slicing of 2-D functions:

```
CALL OPTPUT('SBEGIN',0)
CALL OPTPUT('SDY',.FALSE.)
CALL OPTPUT('SFLAG',.FALSE.)
CALL OPTPUT('SWIDTH',0)
```


APPENDIX X - SUMMARY OF HPAK CALLS

<u>SETTING STORAGE SPACE SIZE</u>:

```
COMMON/HCOM/H(NWORDS)
CALL HSIZE(2000)
```

<u>SETTING UP HISTOGRAM SPECIFICATIONS</u>:

```
1-D case: CALL HDEF1(ID,MS,NBINS,BINLOW,BINWID,TITLE)
2-D case: CALL HDEF2(ID,MS,NBINSX,NBINSY,XLOW,YLOW,XWID,
       1            ,YWID,TITLE)
```

<u>ACCUMULATING HISTOGRAMS</u>:

```
1-D case: CALL HCUM1(ID,XVALUE,INCREMENT)
2-D case: CALL HCUM2(ID,XVALUE,YVALUE,INCREMENT)
```

<u>OUTPUTTING HISTOGRAMS</u>:

```
CALL HOUT(ID,UNIT)
```

<u>PROJECTION OF A 2-D HISTOGRAM</u>:

```
CALL HSLICE(ID,XORY,FIRST,NUMBINS,UNIT)
```

<u>SET MANUAL SCALES FOR A HISTOGRAM</u>:

```
CALL HMSC(ID,SCALE)
```

<u>OTHER HISTOGRAM RELATED CALLS</u>:

```
CALL HACT(ID)
CALL HCLR(ID)
CALL HDEACT(ID)
CALL HDEL(ID)
CALL HGET(ID,OPT,NUMDIM,NBINS,MS,BINLOW,BINWID,IPTR)
CALL HNORM(ID,VALUE)
CALL HSET1(ID,XVALUE,VALUE)
CALL HSET2(ID,XVALUE,YVALUE,VALUE)
CALL HSUM(ID,OPT,D)
```

<u>OPTION SETTING</u>:

```
CALL HOPTN('EXCLUDE',.FALSE.)
CALL HOPTN('HSUMS',.FALSE.)
CALL HOPTN('INTEG',.FALSE.)
CALL HOPTN('MTBIN',0)
CALL HOPTN('STAT',.TRUE.)
```

LINKAGE EDITOR OR LOADER LIBRARIES:

```
WYL.EA.PUB.HPAKLM - contains the DPAK and HPAK modules
WYL.CG.RCB.UGPTNLIB - contains the Unified Graphics modules
```

GO STEP LIBRARY:

```
WYL.CG.RCB.UGRUNLIB contains the run time modules needed
                    for Unified Graphics
```

JCL NEEDED FOR GRAPHIC DEVICE OUTPUT:

For the 10" CALCOMP - 'CALDRSM'

```
//GO.CALDRSM DD DSN=&&DISP1,DISP=(NEW,PASS),VOL=SER=PLOT,
//    UNIT=T9-1600,LABEL=(1,SL),
//    DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
```

For the 29" CALCOMP - 'CALDRLG'

```
//GO.CALDRLG DD DSN=&&DISP2,DISP=(NEW,PASS),VOL=SER=PLOT,
//    UNIT=T9-1600,LABEL=(1,SL),
//    DCB=(RECFM=F,LRECL=480,BLKSIZE=480,DEN=3)
```

For the 16mm unsprocketed microfilm - 'CAL16MU'

```
//GO.CAL16MU DD SYSOUT=X,
//    DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

For the 105mm microfiche - 'CALFICH'

```
//GO.CALFICH DD SYSOUT=Z,
//    DCB=(RECFM=F,LRECL=1480,BLKSIZE=1480)
```

For the TEKTRONIX 4013 displayable PDS - 'PDS4013'

```
//GO.PDS4013 DD DSN=WYL.gg.uuu.name,UNIT=(SYSDA,2),
//    DISP=(NEW,CATLG),DCB=(RECFM=FB,BLKSIZE=1600,LRECL=80),
//    SPACE=(TRK,(100,10,10),RLSE)
```

For a PDS - 'PDSPDEV'

```
//GO.PDSPDEV DD DSN=WYL.gg.uuu.name,DISP=(NEW,CATLG),
//    UNIT=(SYSDA,2),SPACE=(TRK,(10,10,3),RLSE),
//    DCB=(RECFM=U,BLKSIZE=4000)
```

HPAK SCATTERPLOT SCRATCH FILE:

```
//GO.FT19F001 DD DSN=&KIOWA,DISP=(NEW,DELETE),
//   SPACE=(TRK,(50,50),RLSE),UNIT=(SYSDA,2),
//  DCB=(RECFM=VBS,LRECL=1805,BLKSIZE=3614)
```


The entire DPAK and HPAK requires a substantial amount of core. However much of it can be saved by dummying up various routines which are known not to be needed by a job. The following dummies are provided in the load module containing the DPAK and HPAK subroutines. To make use of them, just place a call to the desired dummy at the beginning of your main program.

1. NOLPUT

If no lineprinter output is desired, the call to NOLPUT will cause dummies to be loaded for the following routines:

LPUT1
LPUT2

This will results in a savings of approximately 24000 decimal bytes.

2. NOSCAT

If no scatterplots are desired, the call to NOSCAT will dummy out the following routines:

SCBUF
SCOUT

This will result in a savings of approximately 24000 decimal bytes.

3. NOGRUT

If no graphic device output is desired, the call to NOGRUT will result in the following subroutines being dummied out:

GRUT1
GRUT2
DEVPIC
DPPICT
PXPLOT

This will result in a savings of approximately 54000 decimal bytes.

APPENDIX XIII- ACTUAL PROGRAM WHICH GENERATED THE EXAMPLES

The following is the actual program which generated the examples in the writeup. The comments indicated with 'C+' were included to make examples somewhat self contained.

```
// JOB
/// DELETE WYL.EA.CAL.TEST1
// EXEC FORTHCG,
// GORGN=300K,LKEDPRM='SIZE=300000',
//  LKEDLB1='WYL.EA.PUB.HPAKLM',
//  LKEDLB2='WYL.CG.RCB.UGFTNLIB',
//  GOSL1='WYL.CG.RCB.UGRUNLIB'
//FORT.SYSIN DD *
      INTEGER CLOCK1
      EXTERNAL ZF,XF,YF,ZERR,FX,FY,FERR,FA,FB,FC,FD
      COMMON/HCOM/HCOM(7000)
      REAL XA(25),ERR(25),A(25),C(25),E(25),SCALE(4)/4*0./
     1    ,YLOG(10)/1.0,1.6E1,2.0E1,1.6E2,5.0E3,3.7E2,1.0E2
     2    ,8.9E1,5.0E1,5.0E0/,G(25)
      REAL ZZE(25,25)/625*0.0/,ZZ(25,25),YY(25),WW(25,6)
C
C ACCUMULATE ARRAYS FOR DPAK FIGURES
C
      DO 110 I=1,25
             X=I
             YY(I)=I
             XA(I)=X
             A(I)=(I*.1)**2-2.
             ERR(I)=SQRT(ABS(A(I)))
             C(I)=A(I)-.4
             E(I)=A(I)-.8
             G(I)=A(I)-1.2
             DO 100 J=1,25
                    Y=J
                    TEM=100./EXP(((I-20)**2+3.*(J-10)**2)/60.)
                    IF(TEM .GT. 45.)TEM=90-TEM
                    ZZ(I,J)=TEM
                    IF(J-6.GT.0.AND.J-6.LE.6)WW(I,J-6)=TEM
 100         CONTINUE
 110  CONTINUE
      CALL FSET(XA,YY,ERR,A,C,E,G)
C
C EXAMPLE II.A.6.1 - LINEPRINTER OUTPUT EXAMPLE
C
C+    EXTERNAL FX,FA,FB,FC,FERR
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('CHAR','*')
```



```
      CALL OPTPUT('ERROR',.TRUE.)
      CALL DISP1(FX,FA,FERR,25,
     1          '3 ARRAYS WITH HIST AND ERROR BARS@',
     2          'PRINTER',FB,FC)
C
C EXAMPLE II.A.6.2 - GRAPHIC DEVICE OUTPUT EXAMPLE
C
C+    REAL SCALE(4),XA(25),G(25),ERR(25),A(25),YLOG(10)
C+    EXTERNAL FX,FD,FERR
C
C TURN OVERPLOTTING ON
C
      CALL OPTPUT('OVER',.TRUE.)
C
C MAKE A PLOT IN THE LOWER LEFT HAND CORNER
C
      CALL OPTPUT('HIST',.FALSE.)
      CALL OPTPUT('CHAR','*')
      CALL OPTPUT('YMAN',.TRUE.)
      CALL OPTPUT('XTIC',0)
      CALL OPTPUT('SIZE',.5,.5)
      CALL OPTPUT('ITICS',4)
      SCALE(3)=-3.
      SCALE(4)=1.
      CALL SCPUT1(SCALE)
      CALL DISP1A(XA,G,ERR,25,
     1          'LOWER LEFT FIGURE;X-AXIS;Y MAN SCALES@',
     2          'PDS4013',A)
C OVERPLOT A LINE PLOT
      CALL OPTPUT('FRAME',.FALSE.)
      CALL OPTPUT('ERROR',.FALSE.)
      CALL OPTPUT('LINE',.TRUE.)
      CALL OPTPUT('CHAR',' ')
      CALL DISP1(FX,FD,FERR,25,'@','PDS4013')
C
C MAKE A PLOT IN THE LOWER RIGHT HAND QUADRANT
C
      CALL OPTPUT('YMAN',.FALSE.)
      CALL OPTPUT('FRAME',.TRUE.)
      CALL OPTPUT('CHAR','X')
      CALL OPTPUT('ITICS',0)
      CALL OPTPUT('OFF',.5,0.)
      CALL OPTPUT('RESCALE',.TRUE.)
      CALL OPTPUT('YLOG',.TRUE.)
      CALL DISP1A(XA,YLOG,ERR,10,
     1          'LOWER RIGHT FIGURE;X-AXIS;LOG Y=AXIS@',
     2          'PDS4013')
```

```
C
C MAKE PLOT IN UPPER RIGHT HAND QUADRANT
C
      CALL OPTPUT('HIST',.TRUE.)
      CALL OPTPUT('LINE',.FALSE.)
      CALL OPTPUT('XTIC',0)
      CALL OPTPUT('OFF',.5,.5)
      CALL DISP1A(XA,YLOG,ERR,10,
     1           'HISTOGRAM;OPTIMAL X TICS;Y LOG AXIS@',
     2           'PDS4013')
C
C MAKE PLOT IN UPPER LEFT HAND QUADRANT
C
      CALL OPTPUT('YTIC',0)
      CALL OPTPUT('YLOG',.FALSE.)
      CALL OPTPUT('OFF',0.,.5)
      CALL OPTPUT('ERROR',.TRUE.)
      CALL OPTPUT('CHAR',' ')
      CALL DISP1A(XA,A,ERR,20,'HISTOGRAM WITH ERROR BARS@',
     1           'PDS4013')
C
C TURN OVERPLOTTING OFF
C
      CALL OPTPUT('OVER',.FALSE.)
C
C RESET SIZE OF GRAPHIC PICTURE
C
      CALL OPTPUT('SIZE',1.0,1.0)
      CALL OPTPUT('OFF',0.0,0.0)
C
C EXAMPLE II.B.6.1 - LINEPRINTER EXAMPLE - INTEGER NOTATION
C
C+    REAL XA(25),YY(25),WW(25,6),ZZE(25,25)
      CALL OPTPUT('LENGTH',70)
      CALL DISP2A(XA,YY(7),WW,ZZE,25,6,'INTEGER NOTATION@',
     1           'PRINTER')
C
C EXAMPLE II.B.6.2 - LINEPRINTER EXAMPLE SCIENTIFIC NOTATION
C
C+    REAL XA(25),YY(25),WW(25,6),ZZE(25,25)
      CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('SCIENTIFIC',.TRUE.)
      CALL DISP2A(XA,YY(7),WW,ZZE,25,3,
     1           'SCIENTIFIC NOTATION@','PRINTER')
      CALL OPTPUT('SCIENTIFIC',.FALSE.)
```



```
C
C EXAMPLE II.B.6.3 - LINEPRINTER SLICE EXAMPLE
C
C+    REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
C+    CALL OPTPUT('ERROR',.FALSE.)
C+    CALL OPTPUT('HIST',.TRUE.)
C+    CALL OPTPUT('LENGTH',70)
      CALL OPTPUT('SFLAG',.TRUE.)
      CALL OPTPUT('SWIDTH',6)
      CALL OPTPUT('SDY',.FALSE.)
      CALL OPTPUT('SBEGIN',7)
      CALL DISP2A(XA,YY,ZZ,ZZE,25,25,' SLICE TEST@',
     1           'PRINTER')
      CALL OPTPUT('SFLAG',.FALSE.)
C
C EXAMPLE II.B.6.4 - LINEPRINTER EXAMPLE - SIMULATED
C SCATTERPLOT
C
C+    REAL XF,YF,ZF,ZERR
      CALL OPTPUT('COLUMNS',1)
      CALL OPTPUT('SYMBOL',16)
      CALL DISP2(XF,YF,ZF,ZERR,20,25,
     1           'SIMULATED SCATTERPLOT@','PRINTER')
C+    STOP
C+    END
C+    FUNCTION XF(I)
C+    XF=I
C+    RETURN
C+    END
C+    FUNCTION YF(J)
C+    YF=J
C+    RETURN
C+    END
C+    FUNCTION ZF(I,J)
C+    ZF=35.*EXP(-0.5*(I-10)**2/9.0)*EXP(-0.5*(J-13)**2/16.0)
C+    RETURN
C+    END
C+    FUNCTION ZERR(I,J)
C+    ZERR=0.0
C+    RETURN
C+    END
C
C EXAMPLE II.B.6.5 - GRAPHIC DEVICE OUTPUT EXAMPLE
C
C+    REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
      CALL OPTPUT('XDIR',.FALSE.)
      CALL DISP2A(XA,YY,ZZ,ZZE,25,25,'XDIR=FALSE EXAMPLE@',
     1           'PDS4013')
```

```
C
C EXAMPLE II.B.6.6 - GRAPHIC DEVICE OUTPUT EXAMPLE - SLICED
C 2-D DATA
C
C+     REAL XA(25),YY(25),ZZ(25,25),ZZE(25,25)
C+     CALL OPTPUT('ERROR',.FALSE.)
C+     CALL OPTPUT('HIST',.TRUE.)
       CALL OPTPUT('LENGTH',70)
       CALL OPTPUT('SFLAG',.TRUE.)
       CALL OPTPUT('SWIDTH',6)
       CALL OPTPUT('SDY',.FALSE.)
       CALL OPTPUT('SBEGIN',7)
       CALL OPTPUT('CHAR','*')
       CALL DISP2A(XA,YY,ZZ,ZZE,25,25,'SLICE TEST@','PDS4013')
       CALL OPTPUT('SFLAG',.FALSE.)
C
C EXAMPLE III.A.5 - HISTOGRAM EXAMPLE
C
C+     INTEGER CLOCK1
C+     COMMON/HCOM/HCOM(7000)
       CALL HSIZE(7000)
       CALL HDEF2('SCAT','I*4S',20,25,10.,0.,1.,1.
      1          ,'SIMULATED SCATTERPLOT@')
       CALL RAN1A(CLOCK1(2))
       DO 10 I=1,40
       X=I
       DO 11 J=1,25
       Y=J
       IZ=ZF2(I,J)
       IF(IZ.EQ.0) GO TO 11
       DO 12 K=1,IZ
       XD=RAN1(0)+X
       YD=RAN1(0)+Y
       CALL HCUM2('SCAT',XD,YD,1.0)
12     CONTINUE
11     CONTINUE
10     CONTINUE
C
C OUTPUT TO LINEPRINTER
C
       CALL OPTPUT('COLUMNS',1)
       CALL OPTPUT('SYMBOL',16)
       CALL HOUT('SCAT','PRINTER')
       CALL HOUT('SCAT','PDS4013')
C+     RETURN
C+     END
C+     REAL FUNCTION ZF2(I,J)
C+     ZF2=INT(35.*EXP(-.5*(I-19)**2/9.)*EXP(-.5*(J-13)**2/16.))
C+     RETURN
C+     END

C

       RETURN
       END
```



```
C
C SUBROUTINE TO OVERRIDE DEFAULT HGHEAD
C
       SUBROUTINE HGHEAD(TITLE,N)
       LOGICAL*1 TITLE(1)
       REAL*8 JOBNAM,JOBPRM
       LOGICAL*1 JOB(40)/40*' '/
       LOGICAL JOBL/'JOB='/
       EQUIVALENCE (JOB(5),JOBL),(JOB(9),JOBNAM)
       JOBNAM=JOBPRM(0)
       DO 100 I=1,32
              TITLE(I)=JOB(I+4)
 100   CONTINUE
       N=12
       RETURN
       END
C
C FUNCTIONS FOR FUNCTION PLOTTING EXAMPLES
C
       REAL FUNCTION FSET(X,Y,ERR,A,B,C,D)
       REAL X(1),Y(1),ERR(1),A(1),B(1),C(1),D(1)
       RETURN
       ENTRY FX(I)
       FX=X(I)
       RETURN
       ENTRY FY(I)
       FY=Y(I)
       RETURN
       ENTRY FERR(I)
       FERR=ERR(I)
       RETURN
       ENTRY FA(I)
       FA=A(I)
       RETURN
       ENTRY FB(I)
       FB=B(I)
       RETURN
       ENTRY FC(I)
       FC=C(I)
       RETURN
       ENTRY FD(I)
       FD=D(I)
       RETURN
       END
       FUNCTION XF(I)
       XF=I
       RETURN
       END
```

```
      FUNCTION YF(J)
      YF=J
      RETURN
      END
      FUNCTION ZF(I,J)
      ZF=35.*EXP(-0.5*(I-10)**2/9.0)*EXP(-0.5*(J-13)**2/16.0)
      RETURN
      END
      FUNCTION ZERR(I,J)
      ZERR=0.0
      RETURN
      END
      REAL FUNCTION ZF2(I,J)
      ZF2=INT(35.*EXP(-.5*(I-19)**2/9.)*EXP(-.5*(J-13)**2/16.))
      RETURN
      END
/*
//GO.PDS4013 DD DSN=WYL.EA.CAL.TEST1,UNIT=DISK,
//    VOL=SER=SCFEV4,SPACE=(TRK,(10,1,10),RLSE),
//    DCB=(RECFM=FB,BLKSIZE=1600,LRECL=80),DISP=(NEW,CATLG)
//GO.FT19F001 DD DSN=&KIOWA,DISP=(NEW,DELETE),
/   SPACE=(TRK,(50,50),RLSE),UNIT=(SYSDA,2),
//  DCB=(RECFM=VBS,LRECL=1805,BLKSIZE=3614)
```


REFERENCES

1. WYLBUR/370, The Stanford Timesharing System, Reference Manual, Third Edition, November 1975.

2. CGTM NO. 170 - THE SLAC UNIFIED GRAPHICS SYSTEM - VERSION II, by Robert C. Beach, Computation Group, SLAC.

3. CGTM No. 149 - GRAPHIC KIOWA, by Roger Chaffee, Computation Group, SLAC. The code for the HPAK scatterplot output is a slight modification of the code used by Roger Chaffee in his GRAPHIC KIOWA.

4. Brandt, Siegmund; Statistical and Computation Methods in Data Analysis.

5. SPEAR Program SPGM-5, DISPLAY UTILITY PROGRAMS-DUT1 AND DUT2 et al., A. Boyarski, Experimental Group C, SLAC.

6. The algorithm used by HSCALE was given in the <u>Journal of the ACM</u>, Vol. II, No. 1 (January 1964) by T. Giammo in an artical entitled 'A Mathematical Method for Automatic Scaling of a Function'.

7. CGTM No. 165 - A USER'S GUIDE TO MORTRAN2, by A. James Cook and L. J. Shustek, Computation Group, SLAC. DPAK and HPAK are partially coded in MORTRAN2.

INDEX

X

'X', 10,39,49,A22
'XDIR', 14,21,27,A17,A24
'XLOG', 6,26,A17
'XMAN', 6,26,A17
'XTIC', 6,10,26,A17,A22,A23
'XYAUTO', 14,24,A16
'XZERO', 7,26,A17

Y

'Y', 38
'YLOG', 6,10,27,A16,A22,A23
'YMAN', 6,10,27,A16,A22
'YTIC', 6,10,26,A17,A23
'YZERO', 6,14,25,A16

Z

'ZAUTO', 14,24,A16

1-D, 2,3,4,5,23,25,26,27,32,33,34,35,36,37,38,39,45,46,47,
48,51,52,A3,A4,A9,A10,A11,A16,A17,A18

2-D, 3,12,13,14,15,22,23,24,25,27,33,34,35,36,37,38,39,46,
47,48,49,51,52,A2,A3,A4,A6,A9,A10,A11,A12,A13,A16,A17,
A18,A25