POSTSCRIPT FOR TECHNICAL DRAWINGS
PSPLOT: A FORTRAN-CALLABLE
POSTSCRIPT PLOTTING LIBRARY

USER'S MANUAL

Kevin E. Kohler

Nova Southeastern University
Oceanographic Center
8000 North Ocean Drive
Dania, Florida  33004
1. INTRODUCTION

This manual is intended for users interested in generating two-dimensional technical drawings or graphics for technical journals in PostScript format. The manual describes a library of Fortran-callable subroutines which can be combined in a calling program to produce PostScript plot files. The purpose, syntax, and calling convention of each subroutine is presented, along with plotting examples.

PostScript is a very robust graphics and typesetting language with wide-ranging capabilities. Since the focus of the subroutines in this library is to produce technical drawings, many of the "artistic" features of PostScript have not been addressed or included here.

The name PostScript is a registered trademark of Adobe Systems Incorporated. All instances of the name PostScript in this manual are references to the PostScript language as defined by Adobe Systems Incorporated.

Send questions, comments, suggestions to kevin@ocean.nova.edu.
2. PLOTTING BASICS

This section will present the conventions used by the plotting subroutines in the PSPLLOT plotting library. This information can be used to create customized plotting subroutines.

A plotting session is a set of plotting instructions in a user’s application program which produces hardcopy graphic display output. The output can be either a single plot or graph or a set of graphs. Every plotting session must begin with a call to subroutine PSINIT (with the exception of a call to NEWDEV) and end with a call to subroutine PLOTND.

Pen movement is occasionally mentioned throughout this manual. While PostScript does not use an actual pen for graphic production, it is useful to visualize the plot commands as directing the movements of a pen of variable thickness to a specified (x,y) coordinate, with the pen being either up or down. The plotting subroutines are simple, user-callable commands which direct the movements of an imagined pen upon a plotting sheet.

The initial coordinate origin is approximately .5 inches from the bottom and left paper edges. The orientation of the page must be specified to be either portrait (short side horizontal) or landscape (long side horizontal) and is set by the call to PSINIT (which see). The figure below shows the paper orientation and beginning plot origin for portrait and landscape modes. The current plot origin can be relocated to other positions during the plotting session to provide new reference points for subsequent plotting commands. When the next graph is started, the new origin should be placed far enough away to avoid overlapping the just-completed graph.

![Plotting Sheet Orientation Diagram]
All plotting commands use an absolute plot coordinate system. This means that all coordinates passed to plotting subroutines are expected to be in terms of distance from the current plot origin. This is contrast to a relative plot coordinate system, in which coordinates are assumed to be in terms of distance from the current pen position.

In all plotting subroutines, plot coordinates, character heights, distances, etc. are measured in inches.

Unless otherwise noted, any subroutine argument which specifies an angle is stated in degrees relative to the X axis, with positive angles measured counterclockwise from the X axis.

Most character variables are specified as Hollerith, rather than character strings. The reason for this is primarily historical, combined with the fact that the enormous amount of code already written and in use at my site precluded a comprehensive revision. This should not cause any problems, however.

The initial font of a plot session is Helvetica, with a size of 12 points. This can be changed permanently in subroutine PSINIT. Of course, the current font can always be changed with subroutine SETFNT. PSPLOT supports the standard 35 fonts found on most PostScript printers.

PSPLOT supports color, although all examples in this manual are shown in grayscale for the purposes of reproduction. Color is specified in the relevant subroutines as red, green, and blue (RGB) values.

Continuation allows you to append a character string or number to the end of a previously plotted string or number. The coordinates of the appended string are automatically calculated. Whether or not a subroutine supports continuation is stated in subroutine description in the next section.

Continuation is specified by setting the X and/or Y coordinate argument in calls to the subroutines listed above to 999, and may be applied to X and Y coordinates separately. A subroutine call with continuation must immediately follow the previous plotting call. Continuation is useful when plotting strings containing variable values, such that the resultant string length is not known beforehand.

For example,

\[
\begin{align*}
\text{ND}=10 \\
\text{CALL KEKNUM (1., 2., 0.15, FLOAT(ND), 0., -1, 0)} \\
\text{CALL KEKSYM (999., 999., .15, 5H DAYS, 0., .5, 0)}
\end{align*}
\]

produces "10 DAYS".
3. USER-LEVEL PLOTTING ROUTINES

This section describes the subroutines in the PSPLLOT plotting library at the time of this writing. These routines are Fortran-callable from an application program. Below is a brief summary of the subroutines, followed by an alphabetical listing of each subroutine containing a more detailed description of its function, syntax, and calling arguments.

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>Draws an arc of a specified radius and center.</td>
</tr>
<tr>
<td>AROHED</td>
<td>Draws an arrowhead at a specified location.</td>
</tr>
<tr>
<td>ARROW</td>
<td>Draws an arrow including the shaft.</td>
</tr>
<tr>
<td>AXIS</td>
<td>Draws an axis of a graph, with user-specified axis titles.</td>
</tr>
<tr>
<td>BLKSTP</td>
<td>Strips out embedded blanks in a character string.</td>
</tr>
<tr>
<td>BORDER</td>
<td>Draws a rectangular border with tick marks.</td>
</tr>
<tr>
<td>CHOPIT</td>
<td>Logically closes the current page and begins a new one.</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>Draws a circle with specified center, radius and fill.</td>
</tr>
<tr>
<td>CLIP</td>
<td>Inserts the PostScript command <em>clip</em> into the output file.</td>
</tr>
<tr>
<td>CLIPBOX</td>
<td>Defines the current clipping area.</td>
</tr>
<tr>
<td>COLBOX</td>
<td>Fills regions with specified color.</td>
</tr>
<tr>
<td>CONCOLR</td>
<td>Draws colored contour plots of a 2-dimensional array of regularly-spaced data.</td>
</tr>
<tr>
<td>CONFILL</td>
<td>Draws grayscale contour plots of a 2-dimensional array of regularly-spaced data.</td>
</tr>
<tr>
<td>CONREC</td>
<td>Draws contour plots of a 2-dimensional array of regularly-spaced data.</td>
</tr>
<tr>
<td>CURVE</td>
<td>Draws a curve between 2 points with specified beginning and ending slopes.</td>
</tr>
<tr>
<td>DRWCRV</td>
<td>Connects data points with a solid line.</td>
</tr>
<tr>
<td>DRWTRI</td>
<td>Draws a triangle.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>DSHCRV</td>
<td>Connects data points with a dashed curve.</td>
</tr>
<tr>
<td>DSHLIN</td>
<td>Draws a dashed line between two points.</td>
</tr>
<tr>
<td>FACTOR</td>
<td>Enlarges or reduces a plot by a specified ratio.</td>
</tr>
<tr>
<td>FAROHED</td>
<td>Draws a “fancy” arrowhead.</td>
</tr>
<tr>
<td>FILLBOX</td>
<td>Fills a shape with a specified grayscale level.</td>
</tr>
<tr>
<td>FILRGN</td>
<td>Same as FILLBOX, except it restores the current gray level after filling.</td>
</tr>
<tr>
<td>FILRGNCF</td>
<td>Fills a shape with the current gray level or color.</td>
</tr>
<tr>
<td>GREST</td>
<td>Calls the PostScript operator \textit{grestore}.</td>
</tr>
<tr>
<td>GRKSYM</td>
<td>Plots a Greek symbol.</td>
</tr>
<tr>
<td>GSAV</td>
<td>Calls the PostScript operator \textit{gsave}.</td>
</tr>
<tr>
<td>HILITEC</td>
<td>Prints text surrounded by a rectangular box (color).</td>
</tr>
<tr>
<td>HILITEG</td>
<td>Prints text surrounded by a rectangular box (grayscale).</td>
</tr>
<tr>
<td>INTEGRAL</td>
<td>Plots an integral with upper and lower limits.</td>
</tr>
<tr>
<td>KEKEXP</td>
<td>Plots a floating point number in exponential format.</td>
</tr>
<tr>
<td>KEKFLT</td>
<td>Plots a floating point number in floating point format.</td>
</tr>
<tr>
<td>KEKNUM</td>
<td>Plots a floating point number.</td>
</tr>
<tr>
<td>KEKSYM</td>
<td>Plots a character string.</td>
</tr>
<tr>
<td>KEKSYMC</td>
<td>Plots a character string, with the text string being character rather than Hollerith.</td>
</tr>
<tr>
<td>KEKSYMO</td>
<td>Plots a character string with the characters outlined instead of filled.</td>
</tr>
<tr>
<td>LENSTR</td>
<td>Computes the number of characters in a character string.</td>
</tr>
<tr>
<td>NEWDEV</td>
<td>Specifies the name of the output PostScript file.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>Plots a floating point number. Similar to KEKNUM, but justification and continuation are not supported.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ONEHLF</strong></td>
<td>Draws the symbol $\frac{1}{2}$.</td>
</tr>
<tr>
<td><strong>OVERBAR</strong></td>
<td>Draws a character string with an overbar.</td>
</tr>
<tr>
<td><strong>OVRSBSP</strong></td>
<td>Draws an overbar over a subscripted and superscripted character string.</td>
</tr>
<tr>
<td><strong>OVRSBSPG</strong></td>
<td>Draws an overbar over a subscripted and superscripted Greek symbol.</td>
</tr>
<tr>
<td><strong>OVERSUB</strong></td>
<td>Draws an overbar over a subscripted character string.</td>
</tr>
<tr>
<td><strong>OVERSUBG</strong></td>
<td>Draws an overbar over a subscripted Greek symbol.</td>
</tr>
<tr>
<td><strong>OVERSUP</strong></td>
<td>Draws an overbar over a superscripted character string.</td>
</tr>
<tr>
<td><strong>OVERSUPG</strong></td>
<td>Draws an overbar over a superscripted Greek symbol.</td>
</tr>
<tr>
<td><strong>OVRGRK</strong></td>
<td>Draws an overbar over a Greek symbol.</td>
</tr>
<tr>
<td><strong>PLOT</strong></td>
<td>Directs &quot;pen&quot; movement to a specified (x,y) point with the &quot;pen&quot; either up or down.</td>
</tr>
<tr>
<td><strong>PLOTND</strong></td>
<td>Closes the output PostScript file and terminates the plotting session.</td>
</tr>
<tr>
<td><strong>PLSMIN</strong></td>
<td>Draws the $\pm$ character.</td>
</tr>
<tr>
<td><strong>PRIME</strong></td>
<td>Draws the ' character.</td>
</tr>
<tr>
<td><strong>PSINIT</strong></td>
<td>Called to begin a plotting session. It also specifies portrait or landscape mode.</td>
</tr>
<tr>
<td><strong>RECT</strong></td>
<td>Draws a rectangle.</td>
</tr>
<tr>
<td><strong>RECTFILC</strong></td>
<td>Draws a rectangle and fills it with specified red, green, and blue color levels.</td>
</tr>
<tr>
<td><strong>RECTFILG</strong></td>
<td>Draws a rectangle and fills it with specified gray level.</td>
</tr>
<tr>
<td><strong>ROTATE</strong></td>
<td>Rotates the current coordinate system by a specified angle. Essentially, ROTATE issues the PostScript <em>rotate</em> command.</td>
</tr>
<tr>
<td><strong>RRECT</strong></td>
<td>Draws a rectangle with rounded corners. It can then fill the rectangle with the current graylevel or RGB values.</td>
</tr>
<tr>
<td><strong>SETCOLR</strong></td>
<td>Sets the current color values.</td>
</tr>
</tbody>
</table>
SETFNT  Sets the current font.

SETGRY  Sets the current gray level value.

SETLW   Sets the current line width.

SIGMA   Draws the symbol Σ, with upper and lower limits.

SLDCRV  Connects data points with a solid curve.

SLDLIN  Draws a solid line between two points.

SQRSGN  Draws the radical (square root) sign.

SQUARE  Draws a square.

STROKE  Calls the PostScript operator stroke.

SUBBER  Draws a subscript.

SUBBERSP Draws a subscript of “special” characters.

SUBSUP  Draws a subscript and superscript.

SUBSUPSP Draws subscripts and superscripts of “special” characters.

SUPER   Draws a superscript.

SUPERSP Draws a superscript of “special” characters.

SYMBOL  Plots a character string. Similar to KEKSYM, but justification and continuation are not supported.

The plotting subroutines in the PSPLLOT plotting library are written in Fortran and are called by user-written application programs. In the following subroutine descriptions, integer variables are denoted by arguments beginning with the letters I-N, as per Fortran convention. All other variables are real (4 byte) floating point Fortran and are called by user-written application programs. All of the subroutines are written in single precision. Hence, if your program generates data in double precision, any arguments sent to the plotting library subroutines must first be converted to single precision.
ARC

PURPOSE ARC draws an arc having a specified radius and arc center.

SYNTAX CALL ARC (XC, YC, RAD, ANG1, ANG2)

ARGUMENTS

XC, YC X,Y coordinates of the center of the arc.
RAD Radius of arc.
ANG1 Angle of arc beginning.
ANG2 Angle of arc ending.

Example: ARC

Sample code:
    call setlw(.02)
call arc(0., 2., .5, 0., 135.)
call arc(2., 2., .5, -45., 45.)
AROHEDE

PURPOSE  AROHED draws an arrowhead at a specified location.

SYNTAX   CALL AROHED (XPP, YPP, DIR, AROLNP, SPRANG, LOCXY )

ARGUMENTS

XPP,YPP  X,Y coordinates of the tip of the arrowhead.

DIR      Direction of arrowhead, measured east from north.

AROLNP   Length of arrowhead sides.

SPRANG   Half the angular spread of arrowhead.

LOCXY    Location of arrowhead point
         LOCXY=1  XPP,YPP at arrowhead point  (most commonly used)
         LOCXY=2  XPP, YPP at center of arrowhead.
         LOCXY=3  XPP,YPP at tail of arrowhead.

Example: AROHED

Sample code:
    call setlw(.02)
    call arohed(1., 1., 0., .08, 20., 0)
    call arohed(2., 2., 45., .15, 30., 0)

(0., 0.)
ARROW

PURPOSE  ARROW draws an arrow, with vector length, at a specified location.

SYNTAX  CALL ARROW (XSS, YSS, XPP, YPP, AROLNP, SPRANG, LOCXY )

ARGUMENTS

XSS,YSS   X,Y coordinates of the vector length origin.
XPP,YPP   X,Y coordinates of the tip of the arrowhead.
AROLNP   Length of arrowhead sides.
SPRANG   Half the angular spread of arrowhead.
LOCXY   Location of arrowhead point
        LOCXY=1        XPP,YPP at arrowhead point (most commonly used)
        LOCXY=2        XPP, YPP at center of arrowhead.
        LOCXY=3        XPP,YPP at tail of arrowhead.

Example: ARROW

Sample code:

    call setlw(0.)
    call arrow(.5,.5, 1., 1., .08, 20., 0)
    call setlw(.02)
    call arrow(2., 2., 2.5, 3., .15, 30., 0)
**AXIS**

**PURPOSE**  AXIS draws an axis of a graph, with user-specified axis titles. Tick marks are drawn with a spacing of one inch.

**SYNTAX**  CALL AXIS (XPP, YPP, IBCD, NC, AXLEN, THETA, RMIN, DEL)

**ARGUMENTS**

- **XPP, YPP**  X,Y coordinates of axis origin.
- **IBCD**  Axis title (Hollerith).
- **NC**  Number of characters in title. Its sign is used to specify on which side of the axis the title is to appear: positive for the counterclockwise side of the axis, negative for the clockwise side. Positive labeling is generally used for Y axes, negative for X axes.
- **AXLEN**  Length of axis.
- **THETA**  Angle of axis. Generally, 0 for X axes, 90 for Y axes.
- **RMIN**  The starting value and annotation of the first tick mark.
- **DEL**  The data value increment between tick marks.

**Example: AXIS**

Sample code:

```
xlen=2.
ylen=2.
call setlw(.01)
call axis(.7, .5, 13HDistance (km), -13, xlen , 0., 0., 1.)
call axis(4.5, .2, 15HVelocity (cm/s), 15, ylen, 135., 0., 20.)
```

![Graph](image)

Distance (km)  | Velocity (cm/s)
---|---|---|---
0.00 | 2.00 | 4.00 | 6.00 | 8.00 | 10.00 | 12.00 | 14.00 | 16.00 | 18.00 | 20.00
0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00

(0, 0)
**BLKSTP**

**PURPOSE**  BLKSTP strips out embedded blanks in a character string. It returns the new character string and the number of characters in the new string.

**SYNTAX**  CALL BLKSTP (STR1, NDIM, STR2, NC)

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR1</td>
<td>Input character string.</td>
</tr>
<tr>
<td>NDIM</td>
<td>Maximum length of STR1 as stated in calling program.</td>
</tr>
<tr>
<td>STR2</td>
<td>Output character string with embedded blanks removed.</td>
</tr>
<tr>
<td>NC</td>
<td>Number of characters in STR2.</td>
</tr>
</tbody>
</table>
**BORDER**

**PURPOSE**  
*BORDER* draws a rectangular border with tick marks. The lower left corner of the border is located at the current plot origin.

**SYNTAX**  
CALL BORDER (XLEN, YLEN, ITIC, IBRD, MAJX, MINX, MAJY, MINY)

**ARGUMENTS**

- **XLEN**  
  Length of x-direction side of border.

- **YLEN**  
  Length of y-direction side of border.

- **ITIC**  
  Four digit number which determines which of the border sides will contain tick marks. Each of the digits is either 0 or 1, and if set to 1, that side will contain tick marks. The border sides are ordered as follows:

  - Left-vertical
  - Bottom
  - Right-vertical
  - Top

  For example, if ITIC = 1011, all sides except the bottom would contain tick marks. Additionally, if ITIC<0, the tick marks will be drawn on the outside of the border than on the inside (default).

- **IBRD**  
  IBRD is similar to ITIC except that it determines which sides of the border will be drawn. Hence, you can have tick marks with no border, and vice versa.

- **MAJX**  
  Number of major divisions in the x-direction. A longer tick mark is drawn for these divisions.

- **MINX**  
  Number of minor divisions in the x-direction, i.e. the number of divisions per major division. A shorter tick mark is drawn for these divisions.

- **MAJY**  
  Same as MAJX in y-direction.

- **MINY**  
  Same as MINX in y-direction.

---

**Example: BORDER**

Sample code:
```
call setlw(.01)
call border(1.5, 1.5, 1111, 1001, 4, 2, 3, 4)
call plot(2., 0., -3)
call border(1., 1.7, -1010, 1001, 4, 1, 5, 3)
```
CHOPIT

PURPOSE   CHOPIT logically closes the current graphics page and begins a new one.

SYNTAX    CALL CHOPIT (XPP, YPP)

ARGUMENTS

XPP, YPP   X,Y coordinates of the initial plot origin of the next plot. This allows
you to begin all plots at the same origin on each page.

Note: XPP and YPP are coordinates independent of the current scaling factor.
That is, XPP and YPP represent actual inches, not scaled coordinates. The current
scaling factor is reinstated after the new coordinate origin is set.

Subroutine CHOPIT should not be confused with subroutine PLOTND, which is
called once at the end of each plotting session and which closes the entire output
PostScript file.
CIRCLE

PURPOSE  CIRCLE draws a circle.

SYNTAX    CALL CIRCLE (XC, YC, RAD, FILL)

ARGUMENTS

XC,YC     X,Y coordinates of the center of the circle.

RAD       Radius of the circle.

FILL      Logical value. If FILL=.TRUE., circle is filled with the current color or gray scale; otherwise it is outlined.

Example: CIRCLE

Sample code:
    call circle(1.,1.,.25,.true.)
    call setlw(.02)
    call circle(3.,3.,.5,.false.)
CLIP

PURPOSE  CLIP inserts the PostScript command \textit{clip} into the output file.

SYNTAX  CALL CLIP

ARGUMENTS

NONE.

Note: Most uses of this command are bracketed by calls to GSAV and GREST.
PURPOSE  CLIPBOX defines a clipping region.

SYNTAX  CALL CLIPBOX (XPTS, YPTS, NPTS)

ARGUMENTS

XPTS    Array containing x-coordinates of region path.

YPTS    Array containing y-coordinates of region path.

NPTS    Number of points in region path.

Example: CLIPBOX

Sample code:
    dimension xa(4), ya(4)
    data xa/1.5,3.,3.,1.5/
    data ya/1.,1.,3.,3./
    call clipbox(xa, ya, 4)
    call keksym(.5, .5, .2, 23hA very long text string, 45., 23,0)

Code to outline clipping region not shown.
**COLBOX**

*PURPOSE* COLBOX fills a region with a specified color.

*SYNTAX* CALL COLBOX (XPTS, YPTS, NPTS, RED, GREEN, BLUE)

*ARGUMENTS*

- **XPTS** Array containing x-coordinates of region path.
- **YPTS** Array containing y-coordinates of region path.
- **NPTS** Number of points in region path.
- **RED** Red value of specified color in RGB color model.
- **GREEN** Green value of specified color in RGB color model.
- **BLUE** Blue value of specified color in RGB color model.

Arguments RED, GREEN, and BLUE must lie between 0. and 1.0, inclusive.
CONCOLR

PURPOSE  CONCOLR draws color contour plots of a two-dimensional array of regularly spaced data.

SYNTAX  CALL CONCOLR  (ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, COLOR, NVAL, IOFFP, SPVAL)

ARGUMENTS

ARR  Two-dimensional array containing regularly spaced data to be contoured. CONCOLR assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.

IMAX  The first dimension of ARR in the calling program.

IEXT  Number of points in x-direction of ARR to be contoured.

JEXT  Number of points in y-direction of ARR to be contoured.

XLEN  X-direction length of plotting area.

YLEN  Y-direction length of plotting area.

CVAL  Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. Unlike in subroutine CONREC, NVAL cannot equal 0.

COLOR  Array dimensioned (3,NVAL) containing the red, green, and blue values for each of the contour levels. The red value is stored in (1,n), the green value in (2,n) and the blue value in (3,n), where n specifies the contour index corresponding to CVAL. Regions less than or equal to the corresponding contour value CVAL are filled with the corresponding RBG values.

NVAL  Number of contour intervals. NVAL must be less than or equal to 100, and unlike in CONREC, NVAL cannot equal 0.

IOFFP  Flag indicating that grid boxes whose vertices have the value SPVAL are to be ignored during contouring.

SPVAL  Special value denoting which grid boxes are to be ignored during contouring.

See coding example for CONFILL.
CONFILL

PURPOSE CONFILL draws grayscale contour plots of a two-dimensional array of regularly spaced data.

SYNTAX CALL CONFILL(ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, GRYLEV, NVAL, IOFFP, SPVAL)

ARGUMENTS

ARR Two-dimensional array containing regularly spaced data to be contoured. CONFILL assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.

IMAX The first dimension of ARR in the calling program.

IEXT Number of points in x-direction of ARR to be contoured.

JEXT Number of points in y-direction of ARR to be contoured.

XLEN X-direction length of plotting area.

YLEN Y-direction length of plotting area.

CVAL Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. Unlike subroutine CONREC, NVAL cannot equal 0.

GRYLEV Array dimensioned (NVAL) containing the grayscale values for each of the contour levels. Regions less than or equal to the corresponding contour value CVAL are filled with the corresponding grayscale values. Grayscale values range from 0. (black) to 1.0 (white).

NVAL Number of contour intervals. NVAL must be less than or equal to 100, and unlike in CONREC, NVAL cannot equal 0.

IOFFP Flag indicating that grid boxes whose vertices have the value SPVAL are to be ignored during contouring.

SPVAL Special value denoting which grid boxes are to be ignored during contouring.
Example: CONFILL

Sample code:

```fortran
    do 10 j=1,10
      do 10 i=1,10
          f(i,j)=i*j
    10   xlen=5.
          ylen=5.
      nval=6
      do 20 n=1,nval
          cval(n)=(n-1)*20
          if(cval(n).le.20.) then
              grylev(n)=.3
          elseif(cval(n).le.60.) then
              grylev(n)=.6
          else
              grylev(n)=.9
          endif
      20     continue
      call confill(f,10,10,10,xlen,ylen,cval,grylev,nval,ioffp,spval)

      Code to draw grid boxes not shown.
```
CONREC

PURPOSE  CONREC draws contour plots of a two-dimensional array of regularly spaced data.

SYNTAX    CALL CONREC (ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, NVAL)

ARGUMENTS

ARR       Two-dimensional array containing regularly spaced data to be contoured. CONREC assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.

IMAX      The first dimension of ARR in the calling program.

IEXT      Number of points in x-direction of ARR to be contoured.

JEXT      Number of points in y-direction of ARR to be contoured.

XLEN      X-direction length of plotting area.

YLEN      Y-direction length of plotting area.

CVAL      Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. If NVAL = 0, the individual contour values are automatically computed.

NVAL      Number of contour intervals. If NVAL = 0, the number of contour intervals and the contour interval values are automatically calculated. This is helpful if the range of data values is not known beforehand. NVAL must be less than or equal to 100.

Special features of CONREC:

If NVAL < 0, only high and lows are plotted instead of contours.

Common block CONPAR is used to control various characteristics of the contour plot from the calling program. The variables in CONPAR are shown below, along with their default values:

COMMON/CONPAR/ ISPEC, IOFFP, SPVAL, ILEGG, ILABB, NHII, NDECCN, NLBLL, LSCAL, LDASH, HGTLAB

ISPEC      If 0, variables in CONPAR have no effect. Default = 0.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOFFP</td>
<td>If 1, you can specify regions not to contour by using SPVAL. Grid boxes with any corner values equal to SPVAL are not contoured. Default = 0.</td>
</tr>
<tr>
<td>SPVAL</td>
<td>Special value to denote areas not to contour (usually set to 999.). Default = 0.</td>
</tr>
<tr>
<td>ILEGG</td>
<td>If 0, legend below contour plot showing contour values, scale factors, etc. will not be printed. Default = 1.</td>
</tr>
<tr>
<td>ILABB</td>
<td>If 0, contour labels are not plotted. Default = 1.</td>
</tr>
<tr>
<td>NHII</td>
<td>If 0, H and L mark relative highs and lows, with the data value plotted underneath the symbol. If &gt; 0, values are plotted at each array point. If &lt; 0, neither of the above is done. Default = 0.</td>
</tr>
<tr>
<td>NDECCN</td>
<td>Number of digits to the right of the decimal point in the contour labels. Default = 1.</td>
</tr>
<tr>
<td>NLBLL</td>
<td>Number of unlabeled contours between labelled contours. Default = 3.</td>
</tr>
<tr>
<td>LSCAL</td>
<td>If LSCAL = 0, the contours are scaled such that 0 &lt; labeled values &lt; 100. If LSCAL = 1, the contours are not scaled, i.e. the contour values are the actual data values. Default = 0.</td>
</tr>
<tr>
<td>LDASH</td>
<td>Specifies whether contours are solid (LDASH = 0) or dashed (LDASH NE 0). If LDASH is not equal to 0, then LDASH specifies the type of dash line to use to draw the contours. The numeric value of LDASH is the number of times the dashed pattern (solid line/blank space) is repeated per inch. Default= 0.</td>
</tr>
<tr>
<td>HGTLAB</td>
<td>Specifies the height of contour labels. If HGTLAB=0, the height of the contours is .11 inches.</td>
</tr>
</tbody>
</table>
Example: CONREC

Sample code:

```plaintext
do 10 j=1,10
   do 10 i=1,10
      f(i,j)=i*j
   xlen=5.
ylen=5.
nval=6
   call border(xlen, ylen, 1111, 1111, 4, 2, 5, 3)
do 20 n=1,nval
cval(n)=(n-1)*20
20 continue
   nlbl=0
   lscal=1
   call conrec(f, 10, 10, 10, xlen, ylen, cval, nval)
ldash=6
   ndecn=-1
   do 25 n=1,nval
cval(n)=10.+(n-1)*20
25 continue
   call conrec(f, 10, 10, 10, xlen, ylen, cval, nval)
```
CURVE

PURPOSE  CURVE draws a curve between two points, given the beginning and ending curve slopes.

SYNTAX  CALL CURVE (X1, Y1, X2, Y2, X3, Y3, X4, Y4, CONTIN)

ARGUMENTS

X1,Y1  X,Y coordinates of the beginning of the curve.

X2,Y2  X, Y coordinates specifying the beginning curve slope. The initial slope of the curve is given by (Y2-Y1)/(X2-X1).

X3,Y3  X,Y coordinates specifying the ending curve slope. The end slope of the curve is given by (Y4-Y3)/(X4-X3).

X4,Y4  X,Y coordinates of the endpoint of the curve.

CONTIN  Logical value. If CONTIN=.false. then the curve drawn starts a new path; otherwise, the curve is appended to the current path.

Note: The curve produced by CURVE is not automatically stroked.

Example: CURVE

Sample code:

data x1, x2, x3, x4/2., 3., 3.5, 5./
data y1, y2, y3, y4/1., 1.5, 0., 1.5/
call setlw(.02)
call curve(x1,y1,x2,y2,x3,y3,x4,y4,.false.)
call stroke

Code to indicate points and dashed lines not shown.
**DRWCRV**

**PURPOSE**  
DRWCRV draws a solid curve.

**SYNTAX**  
CALL DRWCRV (XARR, YARR, NPTS, THK, CLOSER)

**ARGUMENTS**

- **XARR**  
Array containing the x-coordinates of the points to be connected.

- **YARR**  
Array containing the y-coordinates of the points to be connected.

- **NPTS**  
Number of points in the curve.

- **THK**  
Thickness of the curve. If THK = 0., the current linewidth is used.

- **CLOSER**  
Logical variable. If CLOSER = .TRUE., the last point in the data arrays is connected to the first point, i.e. the curve is closed; otherwise, the curve is open.

---

**Example: DRWCRV**

**Sample code:**

```fortran
  dimension xp1(6), yp1(6)
  dimension xp2(3), yp2(3)
  data xp1/ .5, 1, 1.5, 2., 2.5, 3. /
  data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
  data xp2/ 4., 4.5, 5. /
  data yp2/ .5, 1.5, 1. /
  call drwcrv(xp1, yp1, 6, .02, .false. )
  call drwcrv(xp2, yp2, 3, .03, .true. )
```

Code to draw axes not shown.
**PURPOSE**  **DRWTRI** draws an equilateral triangle.

**SYNTAX**  CALL DRWTRI (XC, YC, SIDE, THK)

**ARGUMENTS**

XC, YC  X, Y coordinates of triangle center.

SIDE  Length of triangle side.

THK  Thickness of line used to draw the triangle. If THK = 0., the current line width is used.

---

**Example: DRWTRI**

Sample code:

xc=2.
yc=2.
tlen=1.
call drwtri(xc, yc, tlen, .02)

---

[Diagram of an equilateral triangle drawn with the specified arguments]
DSHCRV

PURPOSE  DSHCRV connects points in two data arrays using a line with a specified dash pattern.

SYNTAX   CALL DSHCRV (XARR, YARR, NPTS, IDSHPN, THK)

ARGUMENTS

XARR  Array containing the x-coordinates of the points to be connected.

YARR  Array containing the y-coordinates of the points to be connected.

NPTS  Number of points in the curve.

IDSHPN  Dashed pattern of the line used to connect the data points. IDSHPN is specified as the number of times the dashed pattern (solid line/blank space) is repeated per inch.

THK  Thickness of the dashes in the curve. If THK = 0., the current line width is used.

Example: DSHCRV

Sample code:

dimension xp1(6), yp1(6)
dimension xp2(5), yp2(5)
data xp1/ .5, 1., 1.5, 2., 2.5, 3. /
data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
data xp2/ 3.5, 4., 4.5, 5., 5.5 /
data yp2/ .5, 1., 2., .5, 1. /
call dshcrv(xp1, yp1, 6, 5, .02)
call dshcrv(xp2, yp2, 5, 8, .01)

Code to draw axes not shown.
DSHLIN

PURPOSE  DSHELIN connects two points using a line with a specified dash pattern.

SYNTAX   CALL DSHELIN (X1, Y1, X2, Y2, IDSHPN, THK)

ARGUMENTS

X1,Y1    X,Y coordinates of first data point.

X2,Y2    X,Y coordinates of second data point.

IDSHPN   Dashed pattern of the line used to connect the two data points. IDSHPN is specified as the number of times the dashed pattern (solid line/blank space) is repeated per inch.

THK      Thickness of the dashes in the connecting line. If THK = 0., the current line width is used.

Example: DSHELIN

Sample code:
call dshlin(1., 1., 2., 2., 6, .02)
call dshlin(3., .5, 3.5, 3., 8, .01)
FACTOR

PURPOSE  FACTOR enlarges or reduces the size of a plot by a specified ratio.

SYNTAX   CALL FACTOR (FACT)

ARGUMENTS

FACT  The ratio of the desired plot size to the actual size. FACT must be a positive floating point number.

For example, to enlarge a plot to twice its normal size:

    CALL FACTOR (2.0)
FAROHED

PURPOSE  FAROHED draws a "fancy" arrowhead at a specified location.

SYNTAX   CALL FAROHED (XPP, YPP, DIR, AROLNP, SPRANG, LOCXY, FILL)

ARGUMENTS

XPP, YPP  X,Y coordinates of the tip of the arrowhead.

DIR       Direction of arrowhead, measured east from north.

AROLNP    Length of arrowhead sides.

SPRANG    Half the angular spread of arrowhead.

LOCXY     Location of arrowhead point
           LOCXY=1  XPP,YPP at arrowhead point  (most commonly used)
           LOCXY=2  XPP, YPP at center of arrowhead.
           LOCXY=3  XPP,YPP at tail of arrowhead.

FILL      Logical variable.  If FILL= .TRUE., the arrowhead is filled; otherwise it is outlined.

Example: FAROHED

Sample code:
    call setlw(.01)
    call plot(.5, .5, 3)
    call plot(1., 1., 2)
    call farohed (1., 1., 45., .1, 20., 0, .true.)
    call setlw(0.)
    call plot(3., 2.5, 3)
    call plot(2., 1.5, 2)
    call farohed (2., 1.5, 225., .15, 20., 0, .false.)

(0., 0.)
FILLBOX

PURPOSE  FILLBOX fills a shape with a specified gray level.

SYNTAX   CALL FILLBOX (XARR, YARR, NPTS, GRY)

ARGUMENTS

XARR    Array containing the x-coordinates of the points forming the perimeter of the shape to fill.

YARR    Array containing the y-coordinates of the points forming the perimeter of the shape to fill.

NPTS    Number of points comprising the shape perimeter.

GRY     Gray level of filled shape.  GRY must lie between 0. (black) and 1. (white), inclusive.

Note: The current gray level remains at value GRY after exiting FILLBOX.

Example: FILLBOX

Sample code:

dimension xa(5), ya(5)
data xa / 1., 2.5, 3., 4.25, 3. /
data ya / 1., 2., 3.5, 3.25, .25 /
call fillbox(xa, ya, 5, .7)
call setgry ( 0.)
call setlw (.01)
c Outline shape perimeter
call drwcrv( xa, ya, 5, .01, .true.)

(0., 0.)
FILRGN

PURPOSE  FILRGN is the same as FILLBOX, except it restores the current gray level after filling.

SYNTAX   CALL FILRGN (XARR, YARR, NPTS, GRY)

ARGUMENTS

XARR     Array containing the X coordinates of the points forming the perimeter of the shape to fill.

YARR     Array containing the Y coordinates of the points forming the perimeter of the shape to fill.

NPTS     Number of points comprising the shape perimeter.

GRY      Gray level of filled shape. GRY must lie between 0.(black) and 1. (white), inclusive.

Example: FILRGN

Sample code:

dimension xa(5), ya(5)
data xa / 1., 2.5, 3., 4.25, 3. /
data ya / 1., 2., 3.5, 3.25, .25 /
call filrgn(xa, ya, 5, .7 )
call setlw (.01)
c Outline shape perimeter
call drwcrv(xa, ya, 5, .01, .true. )

Note: call setgry(0.) before drwcrv is not needed as in FILLBOX
FILRGNC

PURPOSE   FILRGNC fills a region with the current gray level or color.

SYNTAX    CALL FILRGNC (XARR, YARR, NPTS)

ARGUMENTS

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XARR</td>
<td>Array containing the X coordinates of the points forming the perimeter of the shape to fill.</td>
</tr>
<tr>
<td>YARR</td>
<td>Array containing the Y coordinates of the points forming the perimeter of the shape to fill.</td>
</tr>
<tr>
<td>NPTS</td>
<td>Number of points comprising the shape perimeter.</td>
</tr>
</tbody>
</table>
GREST

PURPOSE  GREST calls the PostScript operator `grestore`. It is useful after establishing and using a clipping region when it is no longer needed.

SYNTAX  CALL GREST

ARGUMENTS

NONE

GREST is used to restore the graphics state of a PostScript page. It is called after a clipping region has been established and is no longer needed. GREST is used in conjunction with GSAV.
GRKSYM

PURPOSE  GRKSYM plots a Greek symbol.

SYNTAX   CALL GRKSYM (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS)

ARGUMENTS

XPP,YPP   X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT   Height of Greek symbol to be plotted.

ICH      Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below).

ANG      Angle, measured counterclockwise from the X-axis, at which the character is to be plotted.

NCHAR    Number of characters to be plotted (should be set to 1).

MJUS     Controls the justification of the character to be plotted.

  If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted character.

  If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted character.

  If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted character.

PostScript Greek Symbols

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Α Β Γ Ω Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ ψ ω θ φ
GSAV

PURPOSE  GSAV  calls the PostScript operator *gsave*. It is useful before establishing and using a clipping region.

SYNTAX   CALL GSAV

ARGUMENTS

NONE

GSAV is used to save the graphics state of the PostScript page prior to defining a clipping region. After the clipping region is used, GREST is called to restore the original graphics state.
HILITEC

PURPOSE   HILITEC prints text surrounded by a rectangular box (color).

SYNTAX    CALL HILITEC (XPP, YPP, HEIGHT, CSTR, ANG, EDG, JUSX, JUSY,
                       FRED, FGREEN, FBLUE, BRED, BGREEN, BBLUE)

ARGUMENTS

XPP, YPP X,Y coordinates of the box surrounding the text.

HEIGHT Height of text to be plotted.

CSTR Character string text to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the text
       is to be plotted.

EDG Fraction of the text height to use as an edge border.

JUSX Justification in the x-direction.
       JUSX=0 is left justification.
       JUSX=1 is centered.
       JUSX=2 is right justification.

JUSY Similar to JUSX, but in the y-direction.
       JUSY=0 is bottom justification.
       JUSY=1 is centered.
       JUSY=2 is top justification.

FRED, FGREEN, FBLUE Foreground text RGB values. Must lie between 0. and
                      1.0, inclusive.

BRED, BGREEN, BBLUE Background box RGB values. Must lie between 0. and
                      1.0, inclusive.

See code example for HILITEG.
HILITEG

PURPOSE  HILITEG prints text surrounded by a rectangular box (grayscale).

SYNTAX   CALL HILITEG (XPP, YPP, HEIGHT, CSTR, ANG, EDG, JUSX, JUSY, FGRY, BGRY)

ARGUMENTS

XPP, YPP  X,Y coordinates of the box surrounding the text.

HEIGHT   Height of text to be plotted.

CSTR     Character string text to be plotted.

ANG      Angle, measured counterclockwise from the X-axis, at which the text is to be plotted.

EDG      Fraction of the text height to use as an edge border.

JUSX     Justification in the x-direction.
          JUSX=0 is left justification.
          JUSX=1 is centered.
          JUSX=2 is right justification.

JUSY     Similar to JUSX, but in the y-direction.
          JUSY=0 is bottom justification.
          JUSY=1 is centered.
          JUSY=2 is top justification.

FGRY     Graylevel of the (foreground) text. FGRY must lie between 0. and 1.0, inclusive.

BGRY     Graylevel of the (background) surrounding box. BGRY must lie between 0. and 1.0, inclusive.
Example: HILITEG

Sample code:
call hiliteg(1.,1.,15,'Text string 1',0.,2,0,0,1.,0.)
call hiliteg(4.,3.,2,'Text string 2',45.,2,1,0.,8.,4)
INTEGRAL

PURPOSE   INTEGRAL plots an integral with upper and lower limits.

SYNTAX    CALL INTEGRAL (XPP, YPP, HEIGHT, ANG, LOWER, NL, LUPPER, NU)

ARGUMENTS

XPP, YPP    X,Y coordinates of the lower left edge of the integral.

HEIGHT     Height of the integral.

ANG        Angle, measured counterclockwise from the X-axis, at which the integral is plotted.

LOWER      Character string of the lower limit (Hollerith).

NL         Number of characters in the lower limit text.

LUPPER     Character string of the upper limit (Hollerith).

NU         Number of characters in the upper limit text.

Example: INTEGRAL

Sample code:
call integral(1.,1.,15.,0.,1ha,1,1hb,1)
call integral(2.,2.,245.,1hx,1,1hy,1)

\[ \int_{a}^{b} \]
\[ 0, 0 \]
KEKEXP

PURPOSE  KEKEXP plots a floating point number in exponential format.

SYNTAX   CALL KEKEXP (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

ARGUMENTS

XPP, YPP  X,Y coordinates of the number to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT   Height of number to be plotted.

FNUM     Floating point number to be plotted.

ANG      Angle, measured counterclockwise from the X-axis, at which the number is to be plotted.

NDEC     Controls the plotted precision of FNUM.

If NDEC > 0, it specifies the number of digits to the right of the decimal point, after rounding.
If NDEC = 0, only the number’s integer portion and a decimal point are plotted, after rounding.
If NDEC = -1, only the number’s integer portion is plotted, after rounding.

MJUS     Controls the justification of the number to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted number.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted number.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted number.

Example: KEKEXP

Sample code:
   call kekexp(1., .25, .1, 12., 45., 3, 0)
   call kekexp(2., .5, .1, 1234., 0., 4, 0)

(0., 0.)

1.200E+01

1.2340E+03
KEKFLT

PURPOSE  KEKFLT is the same as KEKNUM.

SYNTAX  CALL KEKFLT (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

ARGUMENTS

   Same as KEKNUM.
KEKNUM

PURPOSE  KEKNUM plots a floating point number in floating point format.

SYNTAX   CALL KEKNUM (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

ARGUMENTS

XPP,YPP  X,Y coordinates of the number to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of number to be plotted.

FNUM  Floating point number to be plotted.

ANG  Angle, measured counterclockwise from the X-axis, at which the number is to be plotted.

NDEC  Controls the plotted precision of FNUM.

If NDEC > 0, it specifies the number of digits to the right of the decimal point, after rounding.
If NDEC = 0, only the number's integer portion and a decimal point are plotted, after rounding.
If NDEC = -1, only the number's integer portion is plotted, after rounding.

MJUS  Controls the justification of the number to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted number.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted number.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted number.

Example: KEKNUM

Sample code:
call keknum(1., .25, .1, 12.5, 45., 2, 0 )
call keknum(2., .5, .1, 12.34, 0., 1, 0 )
KEKSYM

PURPOSE KEKSYM plots a character string.

SYNTAX CALL KEKSYM (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

ARGUMENTS

XPP, YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Special feature of KEKSYM:

You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and IBCD equal to the octal code of the character you want to plot.
Example: KEKSYM

Sample code:

```fortran
    call keksym(0., -1.1, .08,
                1 21hThis is centered text, 0., 21, 1 )
call keksym(-1.05, 0., .08,
                1 21hThis is centered text, 90., 21, 1 )
call keksym(0., -.5, .08,
                1 27hThis is left-justified text, 0., 27, 0 )
call keksym(0., .75, .08,
                1 28hThis is right-justified text, 0., 28, 2 )
    call setfnt(29)  !Sets font to Symbol
    call keksym(1., 1., .15, 266, 0., -999, 0)
    call setfnt(28)  !Sets font to Palatino-Roman
    call keksym(999., 999., .15, 1ht, 0., 1, 0)
    call setfnt(35)  !Sets font to Zapf Dingbats
    call keksym(1.5, 1.5, .15, 157, 0., -999, 0)
```

∂

t

This is right-justified text

This is centered text

(0., 0.)

This is left-justified text

This is centered text
**KEKSYMC**

**PURPOSE** KEKSYMC plots a character string. Same as KEKSYM, but text is input as a character string rather than Hollerith.

**SYNTAX** CALL KEKSYMC (XPP, YPP, HEIGHT, CSTR, ANG, NCHAR, MJUS)

**ARGUMENTS**

- **XPP, YPP** X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

- **HEIGHT** Height of character string to be plotted.

- **CSTR** Character string to be plotted (Hollerith).

- **ANG** Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

- **NCHAR** Number of characters in the string to plot.

- **MJUS** Controls the justification of the character string to be plotted.
  
  If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
  If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
  If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

**Special feature of KEKSYMC:**

You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and CSTR equal to the octal code of the character you want to plot. CSTR must be in single quotes.
KEKSYMO

PURPOSE KEKSYMO plots a character string; however, the characters are outlined rather than filled (solid).

SYNTAX CALL KEKSYMO (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

ARGUMENTS

XPP, YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Special feature of KEKSYMO:

You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and IBCD equal to the octal code of the character you want to plot.
Example: KEKSYMO

Sample code:
call keksymo(1.5, 1.5, .25, 8hVelocity, 45., 8, 0)
call keksymo(1.5, .5, .2, 12hAcceleration, 0., 12, 0)
call setfnt(29) !Sets font to Symbol
  call keksymo(-1., 1., .3, 154, 0., -999, 0)
call setfnt(35) !Sets font to Zapf Dingbats
  call keksymo(-1., -1., 4, 104, 0., -999, 0)
LENSTR

PURPOSE  LENSTR is a function whose returned value is the number of characters in the character string argument after trailing blanks have been removed.

SYNTAX   LS = LENSTR(STRING, NDIM)

ARGUMENTS

STRING   Character string variable.
NDIM     Maximum length of STRING as stated in calling program.
LS       Returned length of the character string STRING after trailing blanks have been removed.
NEWDEV

PURPOSE NEWDEV specifies a non-default filename which will contain the application program’s output PostScript plot file.

SYNTAX CALL NEWDEV (FLNAME, NCHAR)

ARGUMENTS

FLNAME Character string or FORTRAN character variable which contains the name of the output Postscript file.

NCHAR Number of characters in the filename.

For example,

CALL NEWDEV ('MYPLOT.DAT', 10)

NEWDEV is called only once per plotting session and must be called prior to PSINIT. If NEWDEV is not called, the name of the output PostScript file is pspplot.ps.
NUMBER

PURPOSE NUMBER is the same as KEKNUM with left justification.

SYNTAX CALL NUMBER (XPP, YPP, HEIGHT, FNUM, ANG, NDEC)

ARGUMENTS

    Same as KEKNUM.
ONEHLF

PURPOSE ONEHLF plots the symbol $\frac{1}{2}$.

SYNTAX CALL ONEHLF (XPP, YPP, HEIGHT, ANG, MJUS)

ARGUMENTS

XPP, YPP X,Y coordinates of the symbol to be plotted.

HEIGHT Height of symbol to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Example: ONEHLF

Sample code:

call onehlf(1., 1., .2, 0., 0 )
call onehlf(2., .5, .25, 45., 0 )

$\frac{1}{2}$
OVERBAR

PURPOSE  OVERBAR plots a character string with an overbar.

SYNTAX  CALL OVERBAR (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

ARGUMENTS

XPP, YPP  X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of number to be plotted.

IBCD  Character string to be plotted (Hollerith).

ANG  Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR  Number of characters in the string to plot.

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Example: OVERBAR

Sample code:
    call overbar(1., .5, .2, 1hQ, 90., 1, 0)
    call overbar(2., 1., .15, 3hSST, 0., 3, 0)
OVERSBSP

PURPOSE OVERSBSP plots a subscripted and superscripted character string with an overbar. The overbar extends to cover both the subscripts and superscripts.

SYNTAX CALL OVERSBSP (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS, ISUB, NSUB, ISUP, NSUP)

ARGUMENTS

XPP, YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP, YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP, YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP, YPP) denotes the position of the lower right corner of the character string.

ISUB Hollerith character(s) comprising the subscript.

NSUB Number of characters in the subscript string.

ISUP Hollerith character(s) comprising the superscript.

NSUP Number of characters in the superscript string.

Example: OVERSBSP

Sample code:

```fortran
  call oversbsp(2., .25, .2, 1hQ, 90., 1, 0, 1he, 1, 1h2, 1 )
  call oversbsp(.5, .5, .15, 1hn, 0., 1, 0, 1h1, 1, 2h-2, 2 )
```

Figure:

![Figure](image-url)
OVERSBSPG

PURPOSE  OVERSBSPG plots a subscripted and superscripted Greek symbol with an overbar. The overbar extends to cover both the subscripts and superscripts.

SYNTAX   CALL OVERSBSPG (XPP, YPP, HEIGHT, ICH, ANG, CHAR, MJUS, ISUB, NSUB, ISUP, NSUP)

ARGUMENTS

XPP, YPP  X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT   Height of symbol to be plotted.

ICH      Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG      Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR    Number of characters in the string to plot (should be set to 1).

MJUS     Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

ISUB     Hollerith character(s) comprising the subscript.

NSUB     Number of characters in the subscript string.

ISUP     Hollerith character(s) comprising the superscript.

NSUP     Number of characters in the superscript string.

Example: OVERSBSPG

Sample code:
  call oversbspg(2., 25., 35., 90., 1, 0, 1hs, 1, 1h2, 1)
  call oversbspg(.5, 25., 43., 0., 1, 0, 1hn, 1, 1h2, 1)
OVERSUB

PURPOSE  OVERSUB plots a subscripted character string with an overbar. The overbar extends to cover the subscripts.

SYNTAX  CALL OVERSUB (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS, ISUB, NSUB)

ARGUMENTS

XPP,YPP  X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of character string to be plotted.

IBCD  Character string to be plotted (Hollerith).

ANG  Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR  Number of characters in the string to plot.

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

ISUB  Hollerith character(s) comprising the subscript.

NSUB  Number of characters in the subscript string.

Example: OVERSUB

Sample code:

call oversub(2., .25, .2, 1hQ, 90., 1, 0, 1hs, 1)
call oversub(.5, .5, .15, 1hn, 0., 1, 0, 1h2, 1)
OVERSUBG

PURPOSE  OVERSUBG plots a subscripted Greek symbol with an overbar. The overbar extends to cover the subscript.

SYNTAX  CALL OVERSUBG(XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS, ISUB, NSUB)

ARGUMENTS

XPP,YPP  X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of the Greek symbol to be plotted.

ICH  Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG  Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

NCHAR  Number of characters in the string to plot (should be set to 1).

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.

ISUB  Hollerith character(s) comprising the subscript.

NSUB  Number of characters in the subscript string.

Example: OVERSUBG

Sample code:
call oversubg(2., .25, .2, 38, 90., 1, 0, 1h2, 1)
call oversubg(.5, .5, .15, 41, 0., 1, 0, 1ho, 1)

\[ \xi^2 \rho_o \]
OVERSUP

PURPOSE  OVERSUP plots a superscripted character string with an overbar. The overbar extends to cover the subscript.

SYNTAX  CALL OVERSUP  (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS,ISUP, NSUP)

ARGUMENTS

XPP, YPP  X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of character string to be plotted.

IBCD  Character string to be plotted (Hollerith).

ANG  Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR  Number of characters in the string to plot.

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2,  (XPP,YPP) denotes the position of the lower right corner of the character string.

ISUP  Hollerith character(s) comprising the superscript.

NSUP  Number of characters in the superscript string.

Example: OVERSUP

Sample code:
call oversup(2., .25, .2, 1hA, 90., 1, 0, 1h2, 1)
call oversup(.5, .5, .15, 1hw, 0., 1, 0, 2h-1, 2)
OVERSUPG

PURPOSE
OVERSUPG plots a subscripted Greek character with an overbar. The overbar extends to cover the superscript.

SYNTAX
CALL OVERSUPG(XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS, ISUP, NSUP)

ARGUMENTS

XPP, YPP  X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of the Greek symbol to be plotted.

ICH  Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG  Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

NCHAR  Number of characters in the string to plot (should be set to 1).

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.

ISUP  Hollerith character(s) comprising the superscript.

NSUP  Number of characters in the superscript string.

Example: OVERSUPG

Sample code:
call oversupg(2., .25, .2, 45, 90., 1, 0, 1h2, 1 )
call oversupg(.5, .5, .15, 25, 0., 1, 0, 1h3, 1 )

\[ \alpha^3 \]
\[ \phi^2 \]
**PURPOSE**  **OVRGRK** plots a Greek symbol with an overbar.

**SYNTAX**  CALL OVRGRK (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS)

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPP, YPP</td>
<td>X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Height of symbol to be plotted.</td>
</tr>
<tr>
<td>ICH</td>
<td>Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).</td>
</tr>
<tr>
<td>ANG</td>
<td>Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.</td>
</tr>
<tr>
<td>NCHAR</td>
<td>Number of characters in the string to plot (should be set to 1).</td>
</tr>
<tr>
<td>MJUS</td>
<td>Controls the justification of the symbol to be plotted. If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol. If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol. If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.</td>
</tr>
</tbody>
</table>

**Example: OVRGRK**

Sample code:

```plaintext
call ovrgrk(2., .25, .2, 28, 90., 1, 0 )
call ovrgrk(.5, .5, .15, 37, 0., 1, 0 )
```

\[ \delta \]  \[ \nu \]
PLOT

PURPOSE  **PLOT** is the most fundamental user-level plotting command. It gives you direct control of pen movement (to any X,Y coordinate position) and pen status (up or down). Additionally, it allows you to re-define the current plotting origin.

SYNTAX  CALL PLOT (XPP, YPP, IPEN)

ARGUMENTS

XPP,YPP  X,Y coordinates of the position to which the pen is to be moved. An origin may be established anywhere on or off the plotting surface, as explained below for negative IPEN values.

IPEN  A signed integer which controls pen status (up/down) and origin definition.

If IPEN = 2, the pen is down during movement, thus drawing a visible line.

If IPEN = 3, the pen is up during movement, thus changing the pen’s current position only.

If IPEN = -2  or -3, a new origin is defined at the position (XPP,YPP) after the movement is completed as if IPEN were positive. The logical X,Y coordinates of the new pen position are set to (0,0), so that all subsequent pen movements use this position as a reference point.

If IPEN = 999, the call to PLOT closes the output file. Thus, a call to PLOT with IPEN = 999 may be used only once in a given plotting session, and if used, must be the last plotting command in the plotting session. Calling PLOT with IPEN = 999 is identical to calling PLOTND (which see).

Example: PLOT

Sample code:
  call plot(0., 0., 3 )
  call plot(1.5, .5, 2 )
  call plot(2.5, .5, -3 ) !Resets origin
  call plot(5., 5., 3 )
  call plot(1.5, .5, 2 )
  call plot(2., 1., 2 )

(0., 0.) [New origin]
PLOTND

PURPOSE   PLOTND closes the output PostScript plot file to terminate the current plot defined by the user application program.

SYNTAX    CALL PLOTND

ARGUMENTS

None

You must call PLOTND (or PLOT (0., 0., 999) ) as the last plotting call in the application program.
PLSMIN

**PURPOSE**  PLSMIN plots the symbol ±.

**SYNTAX**  CALL PLSMIN (XPP, YPP, HEIGHT, ANG, MJUS)

**ARGUMENTS**

XPP, YPP  X,Y coordinates of the symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT  Height of symbol to be plotted.

ANG  Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

MJUS  Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character.

---

### Example: PLSMIN

Sample code:

```fortran
  call plsmin(1., 1., .2, 0., 0)  
  call plsmin(2., .5, .25, 45., 0)
```

![Diagram](attachment:image.png)
PRIME

PURPOSE PRIME plots the symbol ‘ ’.

SYNTAX CALL PRIME (XPP, YPP, HEIGHT, ANG, MJUS)

ARGUMENTS

  XPP, YPP  X,Y coordinates of the symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

  HEIGHT  Height of symbol to be plotted.

  ANG  Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

  MJUS  Controls the justification of the symbol to be plotted.

    If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character.
    If MJUS = 1, (XPP,YPP) denotes the position of the center of the character.
    If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character.

Example: PRIME

Sample code:
  call keksym(.5, .5, .14, 2hdt, 0., 2, 0 )
  call prime(999., 999., .14, 0., 0)
PSINIT

PURPOSE PSINIT is called to begin a plotting session. It must be called before any other plotting command, with the exception of NEWDEV.

SYNTAX CALL PSINIT (PRTRT)

ARGUMENTS

PRTRT Logical variable indicating the paper orientation. If PRTRT= .TRUE., the paper is oriented in portrait mode, i.e. long side of page is vertical. If PRTRT= .false., the paper is oriented in landscape mode, i.e. long side of page is horizontal.
**PURPOSE**  RECT draws a rectangle (or square).

**SYNTAX**  CALL RECT (XX1, YY1, XX2, YY2, HEIGHT)

**ARGUMENTS**

- **XX1,YY1**  X,Y coordinates of lower left corner of rectangle.
- **XX2,YY2**  X,Y coordinates of lower right corner of rectangle.
- **HEIGHT**  Height of rectangle.

**Example: RECT**

Sample code:
```fortran
    call rect( 1., 1., 2., 2., .3 )
    call rect( 2., .5, 3., 0., .2 )
```

(0, 4)
RECTFILC

**PURPOSE**  RECTFILC draws a rectangle (or square) and fills it with a specified red, green, and blue color levels.

**SYNTAX**  CALL RECTFILC( XX1, YY1, XX2, YY2, HEIGHT, RED, GREEN, BLUE)

**ARGUMENTS**

- **XX1,YY1**  X,Y coordinates of lower left corner of rectangle.
- **XX2,YY2**  X,Y coordinates of lower right corner of rectangle.
- **HEIGHT**  Height of rectangle.
- **RED, GREEN, BLUE**  RGB values of the fill color. Must lie between 0. and 1., inclusively.
RECTFILG

PURPOSE  RECTFILG draws a rectangle (or square) and fills it with a specified graylevel.

SYNTAX  CALL RECTFILG (XX1, YY1, XX2, YY2, HEIGHT, GRYLEV)

ARGUMENTS

XX1,YY1  X,Y coordinates of lower left corner of rectangle.

XX2,YY2  X,Y coordinates of lower right corner of rectangle.

HEIGHT  Height of rectangle.

GRYLEV  Graylevel to use for filling rectangle. GRYLEV must lie beween 0. (black) and 1. (white), inclusive.

Example:  RECTFILG

Sample code:

call rect( 1., 1., 2., 2., .3, 0. )
call rect( 2., .5, 3., 0., .2, .8 )
ROTATE

PURPOSE  ROTATE rotates the current coordinate system by a specified angle. Essentially, ROTATE issues the PostScript \textit{rotate} command.

SYNTAX  \texttt{CALL ROTATE ( ANG )}

ARGUMENTS

ANGLE  Angle, measured counterclockwise, to rotate the current coordinate system.
RRECT

**PURPOSE**  
RRECT draws a rectangle (or square) with rounded corners. It can then fill the rectangle with the current graylevel or RGB values.

**SYNTAX**  
CALL RECT (XX1, YY1, WIDTH, HEIGHT, RAD, ANG, FILL)

**ARGUMENTS**

- **XX1,YY1**  X,Y coordinates of lower left corner of rectangle.
- **WIDTH**  Width of rectangle.
- **HEIGHT**  Height of rectangle.
- **RAD**  Radius of rectangle corner curves.
- **ANG**  Angle of rectangle rotation.
- **FILL**  Logical value. If FILL=.true., then the rectangle is filled with the current graylevel or RGB values; otherwise, the rectangle is outlined.

**Example: RRECT**

Sample code:
```
call setlw(.02)
call rrect(1.,-.5, 2., 1., .2, 0., .false.)
call setgry(.8)
call rrect(1.5, 1., 1., .5, .1, 45., .true.)
```

(0, 0)
SETCOLR

PURPOSE  SETCOLR sets the current red, green, and blue color levels.

SYNTAX    CALL SETCOLR (RED, GREEN, BLUE)

ARGUMENTS

    RED    Red color saturation level, between 0. and 1.0.
    GREEN  Green color saturation level, between 0. and 1.0.
    BLUE   Blue color saturation level, between 0. and 1.0.
**PURPOSE**

SETFNT sets the current font.

**SYNTAX**

CALL SETFNT (NFONT)

**ARGUMENTS**

NFONT Number of desired font between 1 and 35 (see table below).

The fonts sets 29 (Symbol) and 35 (ZapfDingbats) require the octal code of the character to be plotted. See the tables on the following pages for the character sets and corresponding octal code values for these two fonts.

## Fonts for Subroutine SETFNT

| 1  | AvantGarde-Book           | 19 | Helvetica-Oblique            |
| 2  | AvantGarde-BookOblique    | 20 | Helvetica                   |
| 3  | AvantGarde-Demi           | 21 | NewCenturySchlbk-Bold       |
| 4  | AvantGarde-DemiOblique    | 22 | NewCenturySchlbk-BoldItalic |
| 5  | Bookman-Demi              | 23 | NewCenturySchlbk-Italic     |
| 6  | Bookman-DemiItalic        | 24 | NewCenturySchlbk-Roman      |
| 7  | Bookman-Light             | 25 | Palatino-Bold               |
| 8  | Bookman-LightItalic       | 26 | Palatino-BoldItalic         |
| 9  | Courier-Bold              | 27 | Palatino-Italic             |
| 10 | Courier-BoldOblique       | 28 | Palatino-Roman              |
| 11 | Courier-Oblique           | 29 | Symbol                      |
| 12 | Courier                  | 30 | Times-Bold                  |
| 13 | Helvetica-Bold            | 31 | Times-BoldItalic            |
| 14 | Helvetica-BoldOblique     | 32 | Times-Italic                |
| 15 | Helvetica-Narrow-Bold     | 33 | Times-Roman                 |
| 16 | Helvetica-Narrow-BoldOblique | 34 | ZapfChancery-MediumItalic   |
| 17 | Helvetica-Narrow-Oblique  | 35 | ZapfDingbats                |
### Characters and octal codes for Font Symbol

![Characters and octal codes for Font Symbol](image)
### Characters and octal codes for Font ZapfDingbats

<table>
<thead>
<tr>
<th>Characters</th>
<th>Octal Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-40</td>
<td>040-047</td>
</tr>
<tr>
<td>41-57</td>
<td>050-057</td>
</tr>
<tr>
<td>58-63</td>
<td>060-065</td>
</tr>
<tr>
<td>64-70</td>
<td>070-077</td>
</tr>
<tr>
<td>71-100</td>
<td>100-117</td>
</tr>
<tr>
<td>101-122</td>
<td>120-157</td>
</tr>
<tr>
<td>123-156</td>
<td>160-177</td>
</tr>
<tr>
<td>157-240</td>
<td>240-257</td>
</tr>
<tr>
<td>241-247</td>
<td>250-256</td>
</tr>
<tr>
<td>248-254</td>
<td>257-257</td>
</tr>
<tr>
<td>255-260</td>
<td>260-267</td>
</tr>
<tr>
<td>261-270</td>
<td>270-277</td>
</tr>
<tr>
<td>271-300</td>
<td>300-317</td>
</tr>
<tr>
<td>301-320</td>
<td>320-337</td>
</tr>
<tr>
<td>321-340</td>
<td>340-357</td>
</tr>
<tr>
<td>341-360</td>
<td>360-376</td>
</tr>
</tbody>
</table>
SETGRY

PURPOSE SETGRY sets the current gray level.

SYNTAX CALL SETGRY (GRYLVL)

ARGUMENTS

GRYLVL Gray level between 0. (black) and 1. (white).

EXAMPLE: SETGRY

Sample code:

call setgry (.7)
call keksym(1., .5, .15, 22HThis is setgray of 0.7, 0., 22, 0)
call setgry (.0) !Sets gray level to black for outline
call keksymo(1., .5, .15, 22HThis is setgray of 0.7, 0., 22, 0)

0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

This is setgray of 0.7
SETLW

PURPOSE  SETLW sets the current linewidth.

SYNTAX  CALL SETLW (RLW)

ARGUMENTS

RLW  Line width.

After calling SETLW, RLW remains the current linewidth until SETLW is called again.

Example: SETLW

Sample code:

call setlw( 0. )
call plot( 0., 0., 3 )
call plot( 1., 1., 2 )
call setlw( .02 )
call plot( 1.5, .5, 3 )
call plot( 3., 2., 2 )
SIGMA

PURPOSE  SIGMA draws the symbol Σ, with upper and lower limits.

SYNTAX   CALL SIGMA ( XPP, YPP, HEIGHT, ANG, LOWER, NL, LUPPER, NU)

ARGUMENTS

  XPP, YPP   X, Y coordinates of the lower left edge of the symbol.

  HEIGHT    Height of the symbol.

  ANG       Angle, measured counterclockwise from the X-axis, at which the
             symbol is to be plotted.

  LOWER     Character string specifying the lower limit (Hollerith).

  NL        Number of characters in the lower limit character string.

  LUPPER    Character string specifying the upper limit (Hollerith).

  NU        Number of characters in the upper limit character string.

If NL (NU) is set to -999, then LOWER (LUPPER) is the octal code of the character
in the font SYMBOL.

Example: SIGMA

Sample code:
call sigma( 1.,1,.,2,0.,1h0,1,1hN,1)
call sigma( 3.,2.,2,45.,1h1,1,245,-999)
**SLDCRV**

**PURPOSE**  
SLDCRV connects points in two data arrays using a line with a specified thickness.

**SYNTAX**  
CALL SLDCRV (XARR, YARR, NPTS, THK)

**ARGUMENTS**

- **XARR**  
  Array containing the X coordinates of the data points to be plotted.

- **YARR**  
  Array containing the Y coordinates of the data points to be plotted.

- **NPTS**  
  Number of points in the curve.

- **THK**  
  Specifies the thickness of the curve. If THK = 0., the current linewidth is used.

**Example: SLDCRV**

Sample code:
```plaintext
dimension xp1(6), yp1(6)
dimension xp2(5), yp2(5)
data xp1/ .5, 1., 1.5, 2., 2.5, 3. /
data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
data xp2/ 3.5, 4., 4.5, 5., 5.5 /
data yp2/ .5, 1., 2., .5, 1. /
call sldcrv(xp1, yp1, 6, .02)
call sldcrv(xp2, yp2, 5, .01)
```

Code to draw axes not shown.
SLDLIN

PURPOSE  SLDLIN connects two points using a line with a specified thickness.

SYNTAX   CALL SLDLIN (X1, Y1, X2, Y2, THK)

ARGUMENTS

X1,Y1  X,Y coordinates of first data point.

X2,Y2  X,Y coordinates of second data point.

THK    Specifies the thickness of the connecting line. If THK=0., the current lineweight is used.

Example: SLDLIN

Sample code:
call sldlin(1., 1., 2., 2., .02 )
call sldlin(3., .5, 4., 1.5, .01 )
SQRSGN

PURPOSE  SQRSGN draws a radical (square root) sign.

SYNTAX   CALL SQRSGN (XPP, YPP, HEIGHT, RLEN)

ARGUMENTS

XPP,YPP  X,Y coordinates of lower left corner of the radical.

HEIGHT  Height of the radical.

RLEN    Length of the top of the radical.

Example: SQRSGN

Sample code:
    call sqrsgn(3., 0., .2, .25 )
    call sqrsgn(-1.5, .5, .4, 3. )
SQUARE

PURPOSE   SQUARE draws a centered square.

SYNTAX    CALL SQUARE (XC, YC, SIDE)

ARGUMENTS

XC,YC     X,Y coordinates of the center of the square.

SIDE      Length of the sides of the square.

Example: SQUARE

Sample code:

    call square( 1., 1., .25 )
    call setlw(.02 )
    call square(-1, 2, .5 )


(0., 0.)
STROKE

PURPOSE  STROKE calls the PostScript operator stroke (which paints or draws the current path).

SYNTAX  CALL STROKE

ARGUMENTS

None
**SUBBER**

**PURPOSE**  
SUBBER draws a subscript.

**SYNTAX**  
CALL SUBBER (ISUB, NSUB, SIZE, ANG)

**ARGUMENTS**

- **ISUB**  
  Hollerith variable specifying the subscript character string.

- **NSUB**  
  Number of characters in the subscript.

- **SIZE**  
  Height of the variable to be subscripted. It is **not** the height of the subscript characters themselves.

- **ANG**  
  Angle of the subscripted variable.

**Note:**  
SUBBER must be called immediately after the call to create the subscripted variable.

**Example: SUBBER**

Sample code:

```plaintext
  call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
  call subber(1hs, 1 , .15, 0. )
  call grksym(1., 1., .2, 43, 0., 1, 0 )
  call subber(1hx, 1, .2, 0. )
  call keksym(-1., .5, .15, 1hW, 45., 1, 0 )
  call subber(1he, 1, .15, 45. )
```

---

The diagram illustrates the usage of SUBBER, with subscripted variables `Q_s`, `W_e`, and `\tau_x`.
SUBBERSP

PURPOSE  SUBBERSP draws a subscript of “special” characters.

SYNTAX  CALL SUBBERSP (NSET, NFNT, ITITLE, NCHR, HEIGHT, ANG

ARGUMENTS

NSET  Number of different font sets needed.

NFNT  Array holding the font numbers 1 to NSET.

ITITLE  ITITLE holds the octal codes for characters of fonts 29 or 35; otherwise, it hold the text characters themselves.

NCHR  Number of characters of a given font. Usually one, but can be greater than one for fonts other than 29 and 35.

HEIGHT  Height of the variable to be subscripted. It is not the height of the subscript characters themselves.

ANG  Angle of the subscripted variable.

Note:  SUBBERSP must be called immediately after the call to create the subscripted variable.

Example:  SUBBERSP

Sample code:

dimension nfnt(2),ititle(20,2),nchr(2)
nfnt(1)=20
nfnt(2)=29
ititle(1,1)=1h2
ititle(1,2)=142
nchr(1)=1
nchr(2)=1
call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
call subbersp(1, 29, 141, 1, .15, 0.)
call keksym(2., 1., .2, 1hA, 0., 1, 0 )
call subbersp(2, nfnt, ititle, nchr, 2, 0.)
call keksym(-1., .5, .15, 1hW, 45, 1, 0 )
call subbersp(1, 35, 153, 1, .15, 45.)

Q*\beta

A_{2\beta}

W

Q_{4\beta}

(0, 0)
**SUBSUP**

**PURPOSE**  
**SUBSUP** draws a subscript and superscript.

**SYNTAX**  
CALL SUBSUP (ISUB, NSUB, ISUP, NSUP, SIZE, ANG)

**ARGUMENTS**

- **ISUB**  Hollerith variable specifying the subscript character string.
- **NSUB**  Number of characters in the subscript.
- **ISUP**  Hollerith variable specifying the superscript character string.
- **NSUP**  Number of characters in the superscript.
- **SIZE**  Height of the variable to be subscripted and superscripted. It is not the height of the subscript and superscript characters themselves.
- **ANG**  Angle of the subscripted and superscripted variable.

Note: **SUBSUP** must be called immediately after the call to create the subscripted and superscripted variable.

---

**Example: SUBSUP**

Sample code:

```plaintext
call keksym(.5, .5, .15, 1hU, 0., 1, 0 )
call subsup(1h1, 1, 1h2, 1, .15, 0. )
call grksym(1., 1., .2, 41, 0., 1, 0 )
call subsup(1hx, 1, 2h-1, 2, .2, 0. )
call keksym(-1., .5, .15, 1hW, 45., 1, 0 )
call subsup(1he, 1, 1h2, 1, .15, 45. )
```

---

\[ \rho_x^{-1} \]

\[ W_e \]

\[ U_t^2 \]
SUBSUPSP

PURPOSE  SUBSUPSP draws subscripts and superscripts of “special” characters.

SYNTAX   CALL SUBSUPSP (NSUB, NFNTSB, ITITLESB, NCHRSB, NSUP, 
                      NFNTSP, ITITLESP, NCHRSP, HEIGHT, ANG)

ARGUMENTS

NSUB, NSUP   Number of different font sets needed for subscripts and 
             superscripts, respectively.

NFNTSB, NFNTSP  Arrays holding the font numbers 1 to NSET for subscripts 
                and superscripts, respectively.

ITITLESB, ITITLESP  Arrays holding the octal codes for characters of fonts 29 or 
                     35; otherwise, holding text characters themselves.

NCHRSB, NCHRSP  Number of characters of a given font for subscripts and 
                superscripts, respectively. Usually one, but can be greater 
                than one for fonts other than 29 and 35.

HEIGHT  Height of the variable to be subscripted. It is not the height 
        of the subscript characters themselves.

ANG   Angle of the subscripted variable.

Note:  SUBSUPSP must be called immediately after the call to create the 
       subscripted variable.

Example: SUBSUPSP

Sample code:
  call keksym(0.5, .5, .15, 'Q', 0., 1, 0 )
  call subsupsp(1, 29, 141, 1, 1, 29, 142, 1, .15, 0.)

\( Q \)
\( \alpha \)
\( \beta \)
SUPER

PURPOSE   SUPER draws a superscript.

SYNTAX    CALL SUPER (ISUP, NSUP, SIZE, ANG)

ARGUMENTS

ISUP    Hollerith variable specifying the superscript character string.

NSUP    Number of characters in the superscript.

SIZE    Height of the \textbf{variable to be superscripted}. It is \textbf{not} the height of the subscript characters themselves.

ANG     Angle of the superscripted variable.

Note: SUPER must be called immediately after the call to create the superscripted variable.

Example: SUPER

Sample code:

\begin{verbatim}
call keksym(.5, .5, .15, 1hu, 0., 1, 0 )
call super(1h2, 1, .15, 0. )
call grksym(1., 1., 2, 41, 0., 1, 0 )
call super(2h-1, 2, 2, 0. )
call keksym(-1., .5, .15, 1hW, 45., 1, 0 )
call super(1h2, 1, .15, 45. )
\end{verbatim}
SUPERSP

PURPOSE  SUPERSP draws a superscript of “special” characters.

SYNTAX   CALL SUPERSP (NSET, NFNT, ITITLE, NCHR, HEIGHT, ANG)

ARGUMENTS

NSET    Number of different font sets needed.

NFNT    Array holding the font numbers 1 to NSET.

ITITLE  ITITLE holds the octal codes for characters of fonts 29 or 35; otherwise, it hold the text characters themselves.

NCHR    Number of characters of a given font. Usually one, but can be greater than one for fonts other than 29 and 35.

HEIGHT  Height of the variable to be superscripted. It is not the height of the superscript characters themselves.

ANG     Angle of the superscripted variable.

Note: SUPERSP must be called immediately after the call to create the superscripted variable.

Example: SUPERSP

Sample code:

dimension nfnt(2), ititle(20,2), nchr(2)
nfnt(1)=20
nfnt(2)=29
ititle(1,1)=1h2
ititle(1,2)=142
nchr(1)=1
nchr(2)=1
call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
call supersp(1, 29, 141, 1, .15, 0.)
call keksym(2, .1, .2, 1hA, 0., 1, 0 )
call supersp(2, nfnt, ititle, nchr, .2, 0.)
call keksym(-1., .25, .15, 1hW, 45., 1, 0 )
call supersp(1, 35, 153, 1, .15, 45.)

\[ A^{2\beta} \]
SYMBOL

PURPOSE SYMBOL plots a character string. It is similar to subroutine KEKSYM, except that justification and continuation are not supported.

SYNTAX CALL SYMBOL (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR)

ARGUMENTS

XPP, YPP X,Y coordinates of the lower left corner of the first character of the string to be plotted.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string.

Example: SYMBOL

Sample code:

```fortran
  call symbol(-1.5, .5, .15, '8hVelocity', 0., 8)
  call symbol(1.5, .5, .15, '8hDistance', 45., 8)
```

Velocity Distance

(0, 0)