PGPLOT Subroutine Descriptions

Introduction

This appendix includes a list of all the PGPLOT subroutines, and then gives detailed instructions for the use of each routine in Fortran programs. The subroutine descriptions are in alphabetical order.

Arguments

The subroutine descriptions indicate the data type of each argument. When arguments are described as ``input'', they may be replaced with constants or expressions in the `CALL` statement, but make sure that the constant or expression has the correct data type.

`INTEGER` arguments:
: these should be declared `INTEGER` or `INTEGER*4` in the calling program, not `INTEGER*2`.

`REAL` arguments:
: these should be declared `REAL` or `REAL*4` in the calling program, not `REAL*8` or `DOUBLE PRECISION`.

`LOGICAL` arguments:
: these should be declared `LOGICAL` or `LOGICAL*4` in the calling program.

`CHARACTER` arguments:
: any valid Fortran `CHARACTER` variable may be used (declared `CHARACTER*n` for some integer `n`).

Index of Routines

Version 5.02

- GRGRAY -- gray-scale map of a 2D data array
- GRGTC0 -- obtain character digitization
- GRIMG0 -- color image of a 2D data array
- GRIMG1 -- image of a 2D data array (image-primitive devices)
- GRIMG2 -- image of a 2D data array (pixel-primitive devices)
- GRIMG3 -- gray-scale map of a 2D data array, using dither
- GRINQFONT -- inquire current font [obsolete]
- GRINQLI -- *obsolete routine*
- GRINQPEN -- *obsolete routine*
- GRITOC - convert integer to character string
- GRLDEV -- list supported device types
- GRLEN -- inquire plotted length of character string
- GRLIN0 -- draw a line
- GRLIN1 -- draw a dashed line
- GRLIN2 -- draw a normal line
- GRLIN3 -- draw a thick line (multiple strokes)
- GRLINA -- draw a line (absolute, world coordinates)
- GRLINR -- draw a line (relative, world coordinates)
- GRMARK -- mark points with specified symbol
- GRMKER -- draw graph markers
- GRMOVA -- move pen (absolute, world coordinates)
- GRMOVR -- move pen (relative, world coordinates)
- GRMSG -- issue message to user
- GROPEN -- open device for graphics
- GRPAGE -- end picture
- GRPARS -- parse device specification string
- GRPIXL -- solid-fill multiple rectangular areas
- GRPOCL -- polygon clip
- GRPROM -- prompt user before clearing screen
- GRPXPO -- Emulate pixel operations using points
- GRPXPS -- pixel dump for color or grey PostScript.
- GRPXPX -- Perform pixel operations using pixel primitive
- GRPXRE -- Emulate pixel operations using rectangles
- GRQCAP -- inquire device capabilities
- GRQCI -- inquire current color index
- GRQCOL -- inquire color capability
- GRQCR -- inquire color representation
- GRQDEV -- inquire current device
- GRQDT -- inquire current device and type
- GRQFNT -- inquire current font
- GRQLS -- inquire current line-style
- GRQLW -- inquire current line width
- GRQPOS -- return current pen position (absolute, world coordinates)
- GRQTXT -- get text bounding box
- GRQTYP -- inquire current device type
- GRQUIT -- report a fatal error and abort execution
- GRREC0 -- fill a rectangle (device coordinates)
- GRRECT -- fill a rectangle

- GRSCI -- set color index
- GRSCR -- set color representation
- GRSETC -- set character size
- GRSETFONT -- set text font [obsolete]
- GRSETLI -- *obsolete routine*
- GRSETPEN -- *obsolete routine*
- GRSETS -- change size of view surface
- GRSFNT -- set text font
- GRSIZE -- inquire device size and resolution
- GRSKPB -- skip blanks in character string
- GRSLCT -- select active output device
- GRSLS -- set line style
- GRSLW -- set line width
- GRSYDS -- decode character string into list of symbol numbers
- GRSYMK -- convert character number into symbol number
- GRSYXD -- obtain the polyline representation of a given symbol
- GRTERM -- flush buffer to output device
- GRTEXT -- draw text
- GRTOUP -- convert character string to upper case
- GRTRAN -- define scaling transformation
- GRTRIM -- length of string excluding trailing blanks
- GRTRN0 -- define scaling transformation
- GRTXY0 -- convert world coordinates to device coordinates
- GRVCT0 -- draw line segments or dots
- GRVECT -- draw line segments or dots
- GRWARN -- issue warning message to user
- GRXHLS -- convert RGB color to HLS color
- GRXRGB -- convert HLS color to RGB color
- PGADVANCE -- non-standard alias for PGPAGE
- PGARRO -- draw an arrow
- PGASK -- control new page prompting
- PGBAND -- read cursor position, with anchor
- PGBBUF -- begin batch of output (buffer)
- PGBEG -- begin PGPLOT, open output device
- PGBEGIN -- non-standard alias for PGBEG
- PGBIN -- histogram of binned data
- PGBOX -- draw labeled frame around viewport
- PGCIRC -- draw a filled or outline circle
- PGCONB -- contour map of a 2D data array, with blanking
- PGCONL -- label contour map of a 2D data array
- PGCONS -- contour map of a 2D data array (fast algorithm)
- PGCONT -- contour map of a 2D data array (contour-following)
- PGCONX -- contour map of a 2D data array (non-rectangular)
- PGCTAB -- install the color table to be used by PGIMAG
- PGCURS -- read cursor position
- PGCURSE -- non-standard alias for PGCURS
- PGDRAW -- draw a line from the current pen position to a point
- PGEBUF -- end batch of output (buffer)

- PGEND -- terminate PGPLOT
- PGENV -- set window and viewport and draw labeled frame
- PGERAS -- erase all graphics from current page
- PGERRB -- horizontal or vertical error bar
- PGERRX -- horizontal error bar
- PGERRY -- vertical error bar
- PGETXT -- erase text from graphics display
- PGFUNT -- function defined by X = F(T), Y = G(T)
- PGFUNX -- function defined by Y = F(X)
- PGFUNY -- function defined by X = F(Y)
- PGGRAY -- gray-scale map of a 2D data array
- PGHI2D -- cross-sections through a 2D data array
- PGHIST -- histogram of unbinned data
- PGIDEN -- write username, date, and time at bottom of plot
- PGIMAG -- color image from a 2D data array
- PGLAB -- write labels for x-axis, y-axis, and top of plot
- PGLABEL -- non-standard alias for PGLAB
- PGLCUR -- draw a line using the cursor
- PGLDEV -- list available device types
- PGLEN -- find length of a string in a variety of units
- PGLINE -- draw a polyline (curve defined by line-segments)
- PGMOVE -- move pen (change current pen position)
- PGMTEXT -- non-standard alias for PGMTXT
- PGMTXT -- write text at position relative to viewport
- PGNCUR -- mark a set of points using the cursor
- PGNCURSE -- non-standard alias for PGNCUR
- PGNUMB -- convert a number into a plottable character string
- PGOLIN -- mark a set of points using the cursor
- PGPAGE -- advance to new page
- PGPANL -- switch to a different panel on the view surface
- PGPAP -- change the size of the view surface
- PGPAPER -- non-standard alias for PGPAP
- PGPIXL -- draw pixels
- PGPNTS -- draw one or more graph markers, not all the same
- PGPOINT -- non-standard alias for PGPT
- PGPOLY -- fill a polygonal area with shading
- PGPT -- draw one or more graph markers
- PGPTEXT -- non-standard alias for PGPTXT
- PGPTXT -- write text at arbitrary position and angle
- PGQAH -- inquire arrow-head style
- PGQCF -- inquire character font
- PGQCH -- inquire character height
- PGQCI -- inquire color index
- PGQCIR -- inquire color index range
- PGQCOL -- inquire color capability
- PGQCR -- inquire color representation
- PGQCS -- inquire character height in a variety of units
- PGQFS -- inquire fill-area style

- PGQHS -- inquire hatching style
- PGQINF -- inquire PGPLOT general information
- PGQITF -- inquire image transfer function
- PGQLS -- inquire line style
- PGQLW -- inquire line width
- PGQPOS -- inquire current pen position
- PGQTBG -- inquire text background color index
- PGQTXT -- find bounding box of text string
- PGQVP -- inquire viewport size and position
- PGQVSZ -- find the window defined by the full view surface
- PGQWIN -- inquire window boundary coordinates
- PGRECT -- draw a rectangle, using fill-area attributes
- PGRND -- find the smallest 'round' number greater than x
- PGRNGE -- choose axis limits
- PGSAH -- set arrow-head style
- PGSAVE -- save PGPLOT attributes
- PGUNSA -- restore PGPLOT attributes
- PGSCF -- set character font
- PGSCH -- set character height
- PGSCI -- set color index
- PGSCIR -- set color index range
- PGSCR -- set color representation
- PGSCRN -- set color representation by name
- PGSFS -- set fill-area style
- PGSHLS -- set color representation using HLS system
- PGSHS -- set hatching style
- PGSITF -- set image transfer function
- PGSLS -- set line style
- PGSLW -- set line width
- PGSTBG -- set text background color index
- PGSUBP -- subdivide view surface into panels
- PGSVP -- set viewport (normalized device coordinates)
- PGSWIN -- set window
- PGTBOX -- draw frame and write (DD) HH MM SS.S labelling
- PGTEXT -- write text (horizontal, left-justified)
- PGUPDT -- update display
- PGVECT -- vector map of a 2D data array, with blanking
- PGVPORT -- non-standard alias for PGSVP
- PGVSIZ -- set viewport (inches)
- PGVSIZE -- non-standard alias for PGVSIZ
- PGVSTAND -- non-standard alias for PGVSTD
- PGVSTD -- set standard (default) viewport
- PGWEDG -- annotate an image plot with a wedge
- PGWINDOW -- non-standard alias for PGSWIN
- PGWNAD -- set window and adjust viewport to same aspect ratio

---

GRAREA -- define a clipping window

```
      SUBROUTINE GRAREA (IDENT,X0,Y0,XSIZE,YSIZE)
```

```
GRPCKG: Define a rectangular window in the current plotting area. All
graphics (except characters written with GRCHAR) will be blanked
outside this window.  The default window is the full plotting area
defined by default or by GRSETS.

Arguments:

IDENT (input, integer): the plot identifier, returned by GROPEN.
X0, Y0 (input, real): the lower left corner of the window, in absolute
      device coordinates.
XSIZE, YSIZE (input, real): width and height of the window in absolute
      coordinates; if either is negative, the window will be reset to
      the full plotting area.
```

GRBPIC -- begin picture

```
      SUBROUTINE GRBPIC
```

```
GRPCKG (internal routine). Send a "begin picture" command to the
device driver, and send commands to set deferred attributes (color,
line width, etc.)

***********************************************************************
*                                                                     *
*  PGPLOT Fortran Graphics Subroutine Library                         *
*                                                                     *
*  T. J. Pearson, California Institute of Technology,                 *
*  Pasadena, California 91125.                                        *
*                                                                     *
*  Routines for handling the obsolete character set                   *
*  ------------------------------------------------                   *
*  These routines are not called by PGPLOT but are called by some     *
*  old user-written programs.                                         *
***********************************************************************

******* Index of Modules **********************************************

* GRCHAR -- draw a string of characters
* GRCHR0 -- support routine for GRCHAR and GRMARK
* GRDAT2 -- character set definition (block data)
* GRGTC0 -- obtain character digitization
* GRMARK -- mark points with specified symbol

***********************************************************************
```

GRCHAR -- draw a string of characters

```
      SUBROUTINE GRCHAR (IDENT,CENTER,ORIENT,ABSXY,X0,Y0,STRING)
```

```
GRPCKG: Draw a string of characters. The plot is not windowed
in the current subarea, but in the full plotting area.

Arguments:

IDENT (input, integer): plot identifier, as returned by GROPEN.
CENTER (input, logical): if .TRUE., the first character of the string
     is centered at (X0,Y0); otherwise the bottom left corner of the
     first character is placed at (X0,Y0).
ORIENT (input, real): the angle in degrees that the string is to make
     with the horizontal, increasing anticlockwise.
ABSXY (input, logical): if .TRUE., (X0,Y0) are absolute device
     coordinates; otherwise they are world coordinates (the scaling
     transformation is applied).
X0, Y0 (input, real): position of first character (see CENTER).
STRING (input, character): the string of ASCII characters; control
     characters 0-20 have special representations; all other
     non-graphic characters are plotted as blank spaces.

(1-Feb-1983)
```

GRCHR0 -- support routine for GRCHAR and GRMARK

```
      SUBROUTINE GRCHR0 (WINDOW,CENTER,ORIENT,ABSXY,X0,Y0,STRING)

GRPCKG (internal routine): Support routine for GRCHAR and GRMARK.
Draw a string of characters.

Arguments:

WINDOW (input, logical): if .TRUE., the plot is windowed in the
     current window.
CENTER (input, logical): if .TRUE., the first character of the string
     is centered at (X0,Y0); otherwise the bottom left corner of the
     first character is placed at (X0,Y0).
ORIENT (input, real): the angle in degrees that the string is to make
     with the horizontal, increasing anticlockwise.
ABSXY (input, logical): if .TRUE., (X0,Y0) are absolute device
     coordinates; otherwise they are world coordinates (the scaling
     transformation is applied).
X0, Y0 (input, real): position of first character (see CENTER).
STRING (input, character): the string of ASCII characters; control
     characters 0-20 have special representations; all other
     non-graphic characters are plotted as blank spaces.

(1-Mar-1983)
```

GRCHSZ -- inquire default character attributes

```
      SUBROUTINE GRCHSZ (IDENT,XSIZE,YSIZE,XSPACE,YSPACE)

GRPCKG: Obtain the default character attributes.

Arguments:
```

```
IDENT (input, integer): the plot identifier, returned by GROPEN.
XSIZE, YSIZE (output, real): the default character size
     (absolute device units).
XSPACE, YSPACE (output, real): the default character spacing
     (absolute units); XSPACE is the distance between the lower left
     corners of adjacent characters in a plotted string; YSPACE
     is the corresponding vertical spacing.
```

GRCLIP -- clip a point against clipping rectangle

```
      SUBROUTINE GRCLIP (X,Y,XMIN,XMAX,YMIN,YMAX,C)
      REAL X,Y
      REAL XMIN,XMAX,YMIN,YMAX
      INTEGER C
```

GRPCKG (internal routine): support routine for the clipping algorithm; called from GRLIN0 only. C is a 4 bit code indicating the relationship between point (X,Y) and the window boundaries; 0 implies the point is within the window.

Arguments:

GRCLOS -- close graphics device

```
      SUBROUTINE GRCLOS
```

GRPCKG: Close the open plot on the current device. Any pending output is sent to the device, the device is released for other users or the disk file is closed, and no further plotting is allowed on the device without a new call to GROPEN.

Arguments: none.

GRCLPL -- clip line against clipping rectangle

```
      SUBROUTINE GRCLPL (X0,Y0,X1,Y1,VIS)
```

GRPCKG (internal routine): Change the end-points of the line (X0,Y0) (X1,Y1) to clip the line at the window boundary. The algorithm is that of Cohen and Sutherland (ref: Newman & Sproull).

Arguments:

```
X0, Y0 (input/output, real): device coordinates of starting point
      of line.
X1, Y1 (input/output, real): device coordinates of end point of line.
VIS (output, logical): .TRUE. if line lies wholly or partially
      within the clipping rectangle; .FALSE. if it lies entirely
      outside the rectangle.
```

GRCTOI -- convert character string to integer

```
      INTEGER FUNCTION GRCTOI (S, I)
      CHARACTER*(*) S
      INTEGER I
```

```
GRCTOI: attempt to read an integer from a character string, and return
the result. No attempt is made to avoid integer overflow. A valid
integer is any sequence of decimal digits.

Returns:
 GRCTOI           : the value of the integer; if the first character
                    read is not a decimal digit, the value returned
                    is zero.
Arguments:
 S      (input)   : character string to be parsed.
 I      (in/out)  : on input, I is the index of the first character
                    in S to be examined; on output, either it points
                    to the next character after a valid integer, or
                    it is equal to LEN(S)+1.
```

GRCURS -- read cursor position

```
      INTEGER FUNCTION GRCURS (IDENT,IX,IY,IXREF,IYREF,MODE,POSN,CH)
      INTEGER IDENT, IX, IY, IXREF, IYREF, MODE, POSN
      CHARACTER*(*) CH
```

```
GRPCKG: Read the cursor position and a character typed by the user.
The position is returned in absolute device coordinates (pixels).
GRCURS positions the cursor at the position specified, and
allows the user to move the cursor using the joystick or
arrow keys or whatever is available on the device. When he has
positioned the cursor, the user types a single character on his
keyboard; GRCURS then returns this character and the new cursor
position.

"Rubber band" feedback of cursor movement can be requested (although
it may not be supported on some devices). If MODE=1, a line from
the anchor point to the current cursor position is displayed as
the cursor is moved. If MODE=2, a rectangle with vertical and
horizontal sides and one vertex at the anchor point and the opposite
vertex at the current cursor position is displayed as the cursor is
moved.

Returns:

GRCURS (integer): 1 if the call was successful; 0 if the device
     has no cursor or some other error occurs.

Arguments:

IDENT (integer, input):  GRPCKG plot identifier (from GROPEN).
```

```
IX    (integer, in/out): the device x-coordinate of the cursor.
IY    (integer, in/out): the device y-coordinate of the cursor.
IXREF (integer, input):  x-coordinate of anchor point.
IYREF (integer, input):  y-coordinate of anchor point.
MODE  (integer, input):  type of rubber-band feedback.
CH    (char,    output): the character typed by the user; if the device
     has no cursor or if some other error occurs, the value CHAR(0)
     [ASCII NUL character] is returned.
```

GRDAT2 -- character set definition (block data)

```
      BLOCK DATA GRDAT2
```

GRPCKG (internal routine): Block data for to define the character set.

Arguments: none.

(1-Feb-1983)

GRDOT0 -- draw a dot

```
      SUBROUTINE GRDOT0 (X,Y)
```

GRPCKG (internal routine): Draw a single dot (pixel) at a specified location.

Arguments:

X, Y (real, input): absolute device coordinates of the dot (these are rounded to the nearest integer by GRDOT0).

GRDTYP -- decode graphics device type string

```
      INTEGER FUNCTION GRDTYP (TEXT)
```

GRPCKG (internal routine): determine graphics device type code from type name. It compares the argument with the table of known device types in common.

Argument:

TEXT (input, character): device type name, eg 'PRINTRONIX'; the name may be abbreviated to uniqueness.

Returns:

GRDTYP (integer): the device type code, in the range 1 to GRTMAX, or zero if the type name is not recognised or ambiguous.

GREPIC -- end picture

```
      SUBROUTINE GREPIC
```

GRPCKG: End the current picture.

Arguments: none.

GRESC -- escape routine

```
      SUBROUTINE GRESC (TEXT)
```

GRPCKG: "Escape" routine. The specified text is sent directly to the selected graphics device, with no interpretation by GRPCKG. This routine must be used with care; e.g., the programmer needs to know the device type of the currently selected device, and the instructions that that device can accept.

Arguments: none.
 TEXT (input, character*(*)): text to be sent to the device.

15-May-1985 - new routine [TJP].
26-May-1987 - add GREXEC support [TJP].
19-Dec-1988 - start new page if necessary [TJP].

GRETXT -- erase text from graphics screen

```
      SUBROUTINE GRETXT
```

GRPCKG: Erase the text screen. Some graphics devices have two superimposed view surfaces, of which one is used for graphics and the other for alphanumeric text. This routine erases the text view surface without affecting the graphics view surface. It does nothing if there is no text view surface associated with the device.

Arguments: none.

GRFA -- fill area (polygon)

```
      SUBROUTINE GRFA (N,PX,PY)
      INTEGER N
      REAL PX(*), PY(*)
```

GRPCKG: FILL AREA: fill a polygon with solid color. The polygon is defined by the (x,y) world coordinates of its N vertices. If this is not a function supported by the device, shading is accomplished by drawing horizontal lines spaced by 1 pixel. By

```
selecting color index 0, the interior of the polygon can be erased
on devices which permit it.  The polygon need not be convex, but if
it is re-entrant (i.e., edges intersect other than at the vertices),
it may not be obvious which regions are "inside" the polygon.  The
following rule is applied: for a given point, create a straight line
starting at the point and going to infinity. If the number of
intersections between the straight line and the polygon is odd, the
point is within the polygon; otherwise it is outside. If the
straight line passes a polygon vertex tangentially, the
intersection  count is not affected. The only attribute which applies
to FILL AREA is color index: line-width and line-style are ignored.
There is a limitation on the complexity of the polygon: GFA will
fail if any horizontal line intersects more than 32 edges of the
polygon.

Arguments:

N (input, integer): the number of vertices of the polygon (at least
      3).
PX, PY (input, real arrays, dimension at least N): world coordinates
      of the N vertices of the polygon.
```

GRFAO - format character string containing integers

```
      SUBROUTINE GRFAO (FORMAT, L, STR, V1, V2, V3, V4)
      CHARACTER*(*) FORMAT
      INTEGER L
      CHARACTER*(*) STR
      INTEGER V1, V2, V3, V4

The input string FORMAT is copied to the output string STR with
the first occurrence of '#' replaced by the value of V1, the second
by the value of V2, etc.  The length of the resulting string is
returned in L.
```

GRGFIL -- find data file

```
      SUBROUTINE GRGFIL(TYPE, NAME)
      CHARACTER*(*) TYPE, NAME

This routine encsapsulates the algorithm for finding the PGPLOT
run-time data files.

1. The binary font file: try the following in order:
    file specified by PGPLOT_FONT
    file "grfont.dat" in directory specified by PGPLOT_DIR
                      (with or without '/' appended)
    file "grfont.dat" in directory /usr/local/pgplot/

2. The color-name database: try the following in order:
    file specified by PGPLOT_RGB
    file "rgb.txt" in directory specified by PGPLOT_DIR
                      (with or without '/' appended)
    file "rgb.txt" in directory /usr/local/pgplot/
```

```
Arguments:
 TYPE (input)  : either 'FONT' or 'RGB' to request the corresponding
                 file.
 NAME (output) : receives the file name.
```

GRGRAY -- gray-scale map of a 2D data array

```
      SUBROUTINE GRGRAY (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   FG, BG, PA, MININD, MAXIND, MODE)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, MININD, MAXIND, MODE
      REAL    A(IDIM,JDIM)
      REAL    FG, BG
      REAL    PA(6)
```

```
This is a device-dependent support routine for PGGRAY.

Draw gray-scale map of an array in current window. Array
values between FG and BG are shaded in gray levels determined
by linear interpolation. FG may be either less than or greater
than BG.  Array values outside the range FG to BG are
shaded black or white as appropriate.

GRGRAY uses GRIMG0 on devices with enough color indices available.
Note that it changes the color table to gray-scale.
Otherwise in does a random dither with GRIMG3.

Arguments:
 A      (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 FG     (input)  : the array value which is to appear in
                   foreground color.
 BG     (input)  : the array value which is to appear in
                   background color.
 PA     (input)  : transformation matrix between array grid and
                   device coordinates (see GRCONT).
 MODE   (input)  : transfer function.
```

GRGTC0 -- obtain character digitization

```
      SUBROUTINE GRGTC0 (CHAR,CENTER,POINTS,X,Y,MORE)
```

```
GRPCKG (internal routine): obtain character digitization.

(10-Feb-1983)
```

GRIMG0 -- color image of a 2D data array

```
      SUBROUTINE GRIMG0 (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   A1, A2, PA, MININD, MAXIND, MODE)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, MININD, MAXIND, MODE
      REAL    A(IDIM,JDIM), A1, A2, PA(6)
```

```
This is a support routine for PGIMAG.

Arguments:
 A       (input)  : the array to be plotted.
 IDIM    (input)  : the first dimension of array A.
 JDIM    (input)  : the second dimension of array A.
 I1, I2  (input)  : the inclusive range of the first index
                    (I) to be plotted.
 J1, J2  (input)  : the inclusive range of the second
                    index (J) to be plotted.
 A1      (input)  : the array value which is to appear in color
                    index MININD.
 A2      (input)  : the array value which is to appear in color
                    index MAXIND.
 PA      (input)  : transformation matrix between array grid and
                    device coordinates.
 MININD  (input)  : minimum color index to use.
 MAXIND  (input)  : maximum color index to use.
 MODE    (input)  : =0 for linear, =1 for logarithmic, =2 for
                    square-root mapping of array values to color
                    indices.
```

GRIMG1 -- image of a 2D data array (image-primitive devices)

```
      SUBROUTINE GRIMG1 (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   A1, A2, PA, MININD, MAXIND, MODE)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, MININD, MAXIND, MODE
      REAL    A(IDIM,JDIM), A1, A2, PA(6)
```

```
(This routine is called by GRIMG0.)
```

GRIMG2 -- image of a 2D data array (pixel-primitive devices)

```
      SUBROUTINE GRIMG2 (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   A1, A2, PA, MININD, MAXIND, MODE)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, MININD, MAXIND, MODE
      REAL    A(IDIM,JDIM)
      REAL    A1, A2
      REAL    PA(6)
```

```
(This routine is called by GRIMG0.)
```

GRIMG3 -- gray-scale map of a 2D data array, using dither

```
      SUBROUTINE GRIMG3 (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   BLACK, WHITE, PA, MODE)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, MODE
      REAL    A(IDIM,JDIM)
      REAL    BLACK, WHITE
      REAL    PA(6)
```

GRINQFONT -- inquire current font [obsolete]

GRINQLI -- *obsolete routine*

```
      SUBROUTINE GRINQLI (INTEN)
```

```
GRPCKG: obtain the line intensity of the current graphics device.
Obsolete routine.
Argument:

INTEN (integer, output): always returns 1.
```

GRINQPEN -- *obsolete routine*

```
      SUBROUTINE GRINQPEN (IP)
```

```
GRPCKG: obtain the pen number of the current graphics device.
Obsolete routine.
Argument:

IP (integer, output): always receives 1.
```

GRITOC - convert integer to character string

```
      INTEGER FUNCTION GRITOC(INT, STR)
      INTEGER INT
      CHARACTER*(*) STR
```

```
Convert integer INT into (decimal) character string in STR.
```

GRLDEV -- list supported device types

```
      SUBROUTINE GRLDEV
```

Support routine for PGLDEV.

Arguments: none

GRLEN -- inquire plotted length of character string

```
      SUBROUTINE GRLEN (STRING, D)
```

GRPCKG: length of text string (absolute units)

GRLIN0 -- draw a line

```
      SUBROUTINE GRLIN0 (XP,YP)
```

```
GRPCKG (internal routine): draw a line from the current position to a
specified position, which becomes the new current position. This
routine takes care of clipping at the viewport boundary, dashed and
thick lines.

Arguments:

XP, YP (input, real): absolute device coordinates of the end-point of
      the line.
```

GRLIN1 -- draw a dashed line

```
      SUBROUTINE GRLIN1 (X0,Y0,X1,Y1,RESET)
```

```
GRPCKG : dashed line. Generate a visible dashed line between points
(X0,Y0) and (X1,Y1) according to the dash pattern stored in common.
If RESET = .TRUE., the pattern will start from the beginning.
Otherwise, it will continue from its last position.
    DASHED LINE PATTERN ARRAY CONTAINING LENGTHS OF
         MARKS AND SPACES IN UNIT CUBE: GRPATN(*)
    OFFSET IN CURRENT PATTERN SEGMENT: GRPOFF
    CURRENT PATTERN SEGMENT NUMBER: GRIPAT
    NUMBER OF PATTERN SEGMENTS: 8
```

GRLIN2 -- draw a normal line

```
      SUBROUTINE GRLIN2 (X0,Y0,X1,Y1)
```

```
GRPCKG : plot a visible line segment in absolute coords from
(X0,Y0) to (X1,Y1).  The endpoints of the line segment are rounded
```

to the nearest integer and passed to the appropriate device-specific
routine. It is assumed that the entire line-segment lies within the
view surface, and that the physical device coordinates are
non-negative.

GRLIN3 -- draw a thick line (multiple strokes)

```
      SUBROUTINE GRLIN3 (X0,Y0,X1,Y1)
```

GRPCKG: draw a heavy line from (X0,Y0) to (X1,Y1) by making multiple
strokes. In order to simulate a thick pen, the line drawn has
circular, rather than square, end points. If this is not done,
thick letters and other figures have an abnormal and unpleasant
appearance.

Vocabulary:

```
LINEWT: the number of strokes required to draw the line; if
      this is odd, one stroke will lie along the requested vector.
      The nominal line thickness is (LINEWT-1)*0.005 in.
RSQURD: the square of the semi-line thickness.
(DX,DY): the vector length of the line.
(VX,VY): a vector of length 1 pixel in the direction of the line.
(VY,-VX): a vector of length 1 pixel perpendicular to (VX,VY).
OFF: the offset parallel to (VY,-VX) of the K'th stroke.
(VXK,VYK): the vector increment of the K'th stroke to allow for the
      semi-circular terminal on the line.
(PXK,PYK): the vector offset of the K'th stroke perpendicular to the
      line vector.
```

GRLINA -- draw a line (absolute, world coordinates)

```
      SUBROUTINE GRLINA (X,Y)
```

GRPCKG: draw line from current position to a specified position.

Arguments:

X, Y (real, input): world coordinates of the end-point of the line.

GRLINR -- draw a line (relative, world coordinates)

```
      SUBROUTINE GRLINR (DX,DY)
```

GRPCKG: draw a line from the current position by a specified
relative displacement.

Arguments:

DX, DY (real, input): the displacement in world coordinates: the pen position is incremented by DX in x and DY in y.

GRMARK -- mark points with specified symbol

```
SUBROUTINE GRMARK (IDENT,CENTER,SYMBOL,ABSXY,POINTS,X,Y)
```

GRPCKG: mark a sequence of points with a specified symbol. The plot is windowed in the current subarea.

Arguments:

IDENT (integer, input): plot identifier from GROPEN.
CENTER (input, logical): if .TRUE. the symbol is centered on the point, otherwise the bottom left corner is placed at the point.
SYMBOL (byte or integer, input): code number of symbol in range 0-127 (ASCII character or special symbol); if SYMBOL is outside this range, nothing is plotted.
ABSXY (logical, input): if .TRUE. (X,Y) are absolute (device) coordinates; otherwise they are world coordinates and the scaling transformation is applied.
POINTS (integer, input): the number of points; if POINTS is less than or equal to 0, nothing is plotted.
X,Y (real arrays, dimension at least POINTS, input): the coordinate pairs; if POINTS=1, these may be scalars instead of arrays.

(9-Mar-1983)

GRMKER -- draw graph markers

```
SUBROUTINE GRMKER (SYMBOL,ABSXY,N,X,Y)
```

GRPCKG: Draw a graph marker at a set of points in the current window. Line attributes (color, intensity, and thickness) apply to markers, but line-style is ignored. After the call to GRMKER, the current pen position will be the center of the last marker plotted.

Arguments:

SYMBOL (input, integer): the marker number to be drawn. Numbers 0-31 are special marker symbols; numbers 32-127 are the corresponding ASCII characters (in the current font). If the number is >127, it is taken to be a Hershey symbol number. If -ve, a regular polygon is drawn.
ABSXY (input, logical): if .TRUE., the input corrdinates (X,Y) are taken to be absolute device coordinates; if .FALSE., they are taken to be world coordinates.
N (input, integer): the number of points to be plotted.
X, Y (input, real arrays, dimensioned at least N): the (X,Y) coordinates of the points to be plotted.

GRMOVA -- move pen (absolute, world coordinates)

```
      SUBROUTINE GRMOVA (X,Y)
```

GRPCKG: move the pen to a specified location.

Arguments:

X, Y (real, input): world coordinates of the new pen position.

GRMOVR -- move pen (relative, world coordinates)

```
      SUBROUTINE GRMOVR (DX,DY)
```

GRPCKG: move the pen through a specified displacement.

Arguments:

DX, DY (real, input): the displacement in world coordinates: the pen
      position is incremented by DX in x and DY in y.

GRMSG -- issue message to user

```
      SUBROUTINE GRMSG (TEXT)
      CHARACTER*(*) TEXT
```

Display a message on standard output.

Argument:
 TEXT (input): text of message to be printed (the string
     may not be blank).

GROPEN -- open device for graphics

```
      INTEGER FUNCTION GROPEN (TYPE,DUMMY,FILE,IDENT)
      INTEGER   TYPE, DUMMY, IDENT
      CHARACTER*(*) FILE
```

GRPCKG: assign a device and prepare for plotting. GROPEN must be
called before all other calls to GRPCKG routines.

Returns:

GROPEN (output, integer): 1 => success, any other value
      indicates a failure (usually the value returned will
      be a VMS error code). In the event of an error, a
      message will be sent to the standard error unit.

Arguments:

```
TYPE (input, integer): default device type (integer code).
DUMMY (input, integer): not used at present.
FILE (input, character): plot specifier, of form 'device/type'.
IDENT (output, integer): plot identifier to be used in later
      calls to GRPCKG.

 1-Jun-1984 - [TJP].
 2-Jul-1984 - change to call GRSLCT [TJP].
13-Jul-1984 - add device initialization [TJP].
23-Jul-1984 - add /APPEND qualifier.
19-Oct-1984 - add VV device [TJP].
26-Dec-1984 - obtain default file name from common [TJP].
29-Jan-1985 - add HP2648 device [KS/TJP].
 5-Aug-1986 - add GREXEC support [AFT].
12-Oct-1986 - fix bug causing GREXEC to erase screen [AFT].
 3-Jun-1987 - remove declaration of exit handler [TJP].
15-Dec-1988 - standardize [TJP].
25-Jun-1989 - remove code that removes spaces from the device name
              [TJP].
26-Nov-1990 - [TJP].
 5-Jan-1993 - [TJP].
 1-Sep-1994 - store device capabilities in common for later use [TJP].
17-Apr-1995 - zero-length string fix [TJP].
 6-Jun-1995 - explicitly initialize GRSTAT [TJP].
```

GRPAGE -- end picture

```
      SUBROUTINE GRPAGE

GRPCKG: Advance the plotting area to a new page. For video devices,
this amounts to erasing the screen; for hardcopy devices, the plot
buffer is written to the output file followed by a form-feed to
advance the paper to the start of the next page.

Arguments: none.
```

GRPARS -- parse device specification string

```
      INTEGER FUNCTION GRPARS (SPEC,DEV,TYPE,APPEND)
      CHARACTER*(*) SPEC, DEV
      INTEGER  TYPE
      LOGICAL  APPEND

GRPCKG: decode a device-specification; called by GROPEN.

Returns:
 GRPARS (output): 1 if the device-specification is
      acceptable; any other value indicates an error.

Arguments:
 SPEC (input): the device specification.
 DEV  (output):  device name or file spec.
 TYPE (output): device type (integer code); 0 if no device
```

```
      type is specified.
 APPEND (output): .TRUE. if /APPEND specified, .FALSE. otherwise.
```

GRPIXL -- solid-fill multiple rectangular areas

```
      SUBROUTINE GRPIXL (IA, IDIM, JDIM, I1, I2, J1, J2,
     1                   X1, X2, Y1, Y2)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL    X1, X2, Y1, Y2
```

```
Determine the size of each rectangular element. If it is equal
to the device pen width and the device supports pixel primitives,
use pixel primitives. Otherwise, if the size is smaller than the
device pen width emulate pixel output by plotting points. If the
size is larger than the device pen width, emulate by outputting
solid-filled rectangles.

Arguments:
 IA     (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 X1, Y1 (input)  : world coordinates of one corner of the output
                   region
 X2, Y2 (input)  : world coordinates of the opposite corner of the
                   output region
```

GRPOCL -- polygon clip

```
      SUBROUTINE GRPOCL (N,PX,PY, EDGE, VAL, MAXOUT, NOUT, QX, QY)
      INTEGER N, NOUT, EDGE, MAXOUT
      REAL    PX(*), PY(*), QX(*), QY(*)
      REAL    VAL
```

```
Clip a polygon against a rectangle: Sutherland-Hodgman algorithm.
this routine must be called four times to clip against each of the
edges of the rectangle in turn.

Arguments:

N (input, integer): the number of vertices of the polygon (at least
      3).
PX, PY (input, real arrays, dimension at least N): world coordinates
      of the N vertices of the input polygon.
EDGE (input, integer):
    1: clip against left edge,   X > XMIN=VAL
    2: clip against right edge,  X < XMAX=VAL
    3: clip against bottom edge, Y > YMIN=VAL
    4: clip against top edge,    Y < YMIN=VAL
VAL  (input, real): coordinate value of current edge.
```

```
MAXOUT (input, integer): maximum number of vertices allowed in
    output polygon (dimension of QX, QY).
NOUT (output, integer): the number of vertices in the clipped polygon.
QX, QY (output, real arrays, dimension at least MAXOUT): world
      coordinates of the NOUT vertices of the output polygon.
```

GRPROM -- prompt user before clearing screen

```
      SUBROUTINE GRPROM
```

If the program is running under control of a terminal, display message and wait for the user to type <CR> before proceeding.

```
Arguments:
 none
```

GRPXPO -- Emulate pixel operations using points

```
      SUBROUTINE GRPXPO (IA, IDIM, JDIM, I1, I2, J1, J2,
     1                   X1, X2, Y1, Y2)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL    X1, X2, Y1, Y2

Arguments:
 IA     (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 X1, X2 (input)  : the horizontal range of the output region
 Y1, Y2 (input)  : the vertical range of the output region
```

GRPXPS -- pixel dump for color or grey PostScript.

```
      SUBROUTINE GRPXPS (IA, IDIM, JDIM, I1, I2, J1, J2,
     :                   XMIN, XMAX, YMIN, YMAX)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL XMIN, XMAX, YMIN, YMAX
```

This routine is called by GRPIXL.

GRPXPX -- Perform pixel operations using pixel primitive

```
      SUBROUTINE GRPXPX (IA, IDIM, JDIM, I1, I2, J1, J2, X, Y)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL    X, Y

Arguments:
 IA      (input)  : the array to be plotted.
 IDIM    (input)  : the first dimension of array A.
 JDIM    (input)  : the second dimension of array A.
 I1, I2  (input)  : the inclusive range of the first index
                    (I) to be plotted.
 J1, J2  (input)  : the inclusive range of the second
                    index (J) to be plotted.
 X, Y    (input)  : the lower left corner of the output region
                    (device coordinates)
```

GRPXRE -- Emulate pixel operations using rectangles

```
      SUBROUTINE GRPXRE (IA, IDIM, JDIM, I1, I2, J1, J2,
     1                   X1, X2, Y1, Y2)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL    X1, X2, Y1, Y2

Arguments:
 IA      (input)  : the array to be plotted.
 IDIM    (input)  : the first dimension of array A.
 JDIM    (input)  : the second dimension of array A.
 I1, I2  (input)  : the inclusive range of the first index
                    (I) to be plotted.
 J1, J2  (input)  : the inclusive range of the second
                    index (J) to be plotted.
 X1, X2  (input)  : the horizontal range of the output region
 Y1, Y2  (input)  : the vertical range of the output region
```

GRQCAP -- inquire device capabilities

```
      SUBROUTINE GRQCAP (STRING)
      CHARACTER*(*) STRING

GRPCKG: obtain the "device capabilities" string from the device
driver for the current device.

Arguments:

STRING (output, CHARACTER*(*)): receives the device capabilities
      string.
```

GRQCI -- inquire current color index

```
      SUBROUTINE GRQCI (C)
```

GRPCKG: obtain the color index of the current graphics device.

Argument:

C (integer, output): receives the current color index (0-255).

GRQCOL -- inquire color capability

```
      SUBROUTINE GRQCOL (CI1, CI2)
      INTEGER  CI1, CI2
```

Query the range of color indices available on the current device.

```
Argument:
 CI1    (output) : the minimum available color index. This will be
                   either 0 if the device can write in the
                   background color, or 1 if not.
 CI2    (output) : the maximum available color index. This will be
                   1 if the device has no color capability, or a
                   larger number (e.g., 3, 7, 15, 255).
```

GRQCR -- inquire color representation

```
      SUBROUTINE GRQCR (CI, CR, CG, CB)
      INTEGER  CI
      REAL     CR, CG, CB
```

Return the color representation (red, green, blue intensities) currently associated with the specified color index. This may be different from that requested on some devices.

Arguments:

CI (integer, input): color index.
CR, CG, CB (real, output): red, green, and blue intensities,
      in range 0.0 to 1.0.

GRQDEV -- inquire current device

```
      SUBROUTINE GRQDEV (DEVICE, L)
      CHARACTER*(*) DEVICE
      INTEGER L
```

Obtain the name of the current graphics device or file.

```
Argument:
 DEVICE (output): receives the device name of the
      currently active device.
```

```
 L (output): number of characters in DEVICE, excluding trailing
      blanks.
```

GRQDT -- inquire current device and type

```
      SUBROUTINE GRQDT (DEVICE)
```

GRPCKG: obtain the name and type of the current graphics device.

Argument:

```
DEVICE (output, character): receives the device name and type of the
      currently active device in the form 'device/type'; this is a
      valid string for input to GROPEN.
```

GRQFNT -- inquire current font

```
      SUBROUTINE GRQFNT (IF)
```

GRPCKG: obtain the font number of the current graphics device.

Argument:

IF (integer, output): receives the current font number (1-3).

GRQLS -- inquire current line-style

```
      SUBROUTINE GRQLS (ISTYLE)
      INTEGER  ISTYLE
```

GRPCKG: obtain the line-style of the current graphics device.

Argument:
 ISTYLE (output): receives the current line-style code.

GRQLW -- inquire current line width

```
      SUBROUTINE GRQLW (IWIDTH)
      INTEGER  IWIDTH
```

GRPCKG: obtain the line-width of the current graphics device.

Argument:
 IWIDTH (output): receives the current line-width.

GRQPOS -- return current pen position (absolute, world coordinates)

```
      SUBROUTINE GRQPOS(X,Y)
```

GRQPOS: returns the current pen position in absolute, world coordinates.

Arguments:

X, Y (real, output): world coordinates of the pen position.

GRQTXT -- get text bounding box

```
      SUBROUTINE GRQTXT (ORIENT,X0,Y0,STRING, XBOX, YBOX)
```

GRPCKG: get the bounding box of a string drawn by GRTEXT.

GRQTYP -- inquire current device type

```
      SUBROUTINE GRQTYP (TYPE,INTER)
      CHARACTER*(*) TYPE
      LOGICAL INTER
```

GRPCKG: obtain the device type of the currently selected graphics device, and determine whether or not it is an interactive device.

Arguments:

```
TYPE (output, CHARACTER*(*)): receives the device type, as a
      character string, eg 'PRINTRONIX', 'TRILOG', 'VERSATEC',
      'TEK4010', 'TEK4014', 'GRINNELL', or 'VT125'.  The character
      string should have a length of at least 8 to ensure that the
      type is unique.
INTER (output, LOGICAL): receives the value .TRUE. if the device is
      interactive, .FALSE. otherwise.
```

GRQUIT -- report a fatal error and abort execution

```
      SUBROUTINE GRQUIT (TEXT)
      CHARACTER*(*) TEXT
```

Report a fatal error (via GRWARN) and exit program.
This routine should be called in the event of an unrecoverable
PGPLOT error.

Argument:

TEXT (input): text of message to be sent to GRWARN.

GRREC0 -- fill a rectangle (device coordinates)

```
      SUBROUTINE GRREC0 (X0,Y0,X1,Y1)
      REAL X0, Y0, X1, Y1
```

GRPCKG: Fill a rectangle with solid color. The rectangle
is defined by the (x,y) device coordinates of its lower left and
upper right corners; the edges are parallel to the coordinate axes.
X0 is guaranteed to be <= X1 and Y0 <= Y1. The rectangle possible
extends beyond the clipping boundaries

Arguments:

X0, Y0 (input, real): device coordinates of one corner of the
      rectangle.
X1, Y1 (input, real): device coordinates of the opposite corner of
      the rectangle.

GRRECT -- fill a rectangle

```
      SUBROUTINE GRRECT (X0,Y0,X1,Y1)
      REAL X0, Y0, X1, Y1
```

GRPCKG: Fill a rectangle with solid color. The rectangle
is defined by the (x,y) world coordinates of its lower left and upper
right corners; the edges are parallel to the coordinate axes.

Arguments:

X0, Y0 (input, real): world coordinates of one corner of the
      rectangle.
X1, Y1 (input, real): world coordinates of the opposite corner of the
      rectangle.

GRSCI -- set color index

```
      SUBROUTINE GRSCI (IC)
```

GRPCKG: Set the color index for subsequent plotting. Calls to GRSCI
are ignored for monochrome devices. The default color index is 1,
usually white on a black background for video displays or black on a
white background for printer plots. The color index is an integer in
the range 0 to a device-dependent maximum. Color index 0 corresponds
to the background color; lines may be "erased" by overwriting them
with color index 0.

Color indices 0-7 are predefined as follows: 0 = black (background
color), 1 = white (default), 2 = red, 3 = green, 4 = blue, 5 = cyan

(blue + green), 6 = magenta (red + blue), 7 = yellow (red + green).
The assignment of colors to color indices can be changed with
subroutine GRSCR (set color representation).

Argument:

IC (integer, input): the color index to be used for subsequent
plotting on the current device (in range 0-255). If the
index exceeds the device-dependent maximum, the result is
device-dependent.

GRSCR -- set color representation

```
      SUBROUTINE GRSCR (CI, CR, CG, CB)
      INTEGER  CI
      REAL     CR, CG, CB
```

GRPCKG: SET COLOUR REPRESENTATION -- define the colour to be
associated with a colour index. Ignored for devices which do not
support variable colour or intensity. On monochrome output
devices (e.g. VT125 terminals with monochrome monitors), the
monochrome intensity is computed from the specified Red, Green, Blue
intensities as 0.30*R + 0.59*G + 0.11*B, as in US color television
systems, NTSC encoding. Note that most devices do not have an
infinite range of colours or monochrome intensities available;
the nearest available colour is used.

Arguments:

CI (integer, input): colour index. If the colour index is outside the
range available on the device, the call is ignored. Colour
index 0 applies to the background colour.
CR, CG, CB (real, input): red, green, and blue intensities,
in range 0.0 to 1.0.

GRSETC -- set character size

```
      SUBROUTINE GRSETC (IDENT,XSIZE)
```

GRPCKG : change the character size (user-callable routine).

```
Input:   IDENT : plot identifier
         XSIZE : the new character width. The character height
                 and spacing will be scaled by the same factor.
                 If XSIZE is negative or zero, the character size
                 will be set to the default size.
```

GRSETFONT -- set text font [obsolete]

GRSETLI -- *obsolete routine*

```
      SUBROUTINE GRSETLI (IN)
```

```
GRPCKG: Set the line intensity for subsequent plotting on the current
device. *** OBSOLETE ROUTINE *** Intensity is now set with GRSCI
and GRSCR. For compatibility, GRSETLI now sets color zero if its
argument is 0, and resets the previous color if its argument is
non-zero.

Argument:

IN (integer, input): the intensity to be used for subsequent
      plotting on the current device (in range 0-3).
```

GRSETPEN -- *obsolete routine*

```
      SUBROUTINE GRSETPEN
```

```
GRPCKG: Set the pen number for subsequent plotting.  Obsolete
routine: ignored.
```

GRSETS -- change size of view surface

```
      SUBROUTINE GRSETS (IDENT,XSIZE,YSIZE)
```

```
GRPCKG : change size of plotting area. The requested dimensions
will be reduced to the absolute maximum of the plot device if
necessary.

Arguments:

IDENT (input, integer): plot identifier from GROPEN.
XSIZE (input, real): new x dimension of plot area (absolute
              units); if less than zero, the default dimension
              will be used.
YSIZE (input, real): new y dimension of plot area (absolute
              units); if less than zero, the default dimension
              will be used.
```

GRSFNT -- set text font

```
      SUBROUTINE GRSFNT (IF)
      INTEGER IF
```

```
GRPCKG: Set the font for subsequent text plotting.
The default font is 1 ("Normal" font); others available are 2
("Roman"), 3 ("Italic"), and 4 ("Script").
```

```
Argument:
 IF (input): the font number to be used for subsequent
      text plotting on the current device (in range 1-4).
```

GRSIZE -- inquire device size and resolution

```
      SUBROUTINE GRSIZE (IDENT,XSZDEF,YSZDEF,XSZMAX,YSZMAX,
     1                   XPERIN,YPERIN)

GRPCKG : obtain device parameters (user-callable routine).
```

GRSKPB -- skip blanks in character string

```
      SUBROUTINE GRSKPB (S, I)
      CHARACTER*(*) S
      INTEGER I

GRSKPB: increment I so that it points to the next non-blank
character in string S.  'Blank' characters are space and tab (ASCII
character value 9).

Arguments:
 S      (input)  : character string to be parsed.
 I      (in/out) : on input, I is the index of the first character
                   in S to be examined; on output, either it points
                   to the next non-blank character, or it is equal
                   to LEN(S)+1 (if all the rest of the string is
                   blank).
```

GRSLCT -- select active output device

```
      SUBROUTINE GRSLCT (IDENT)

GRPCKG: Check that IDENT is a valid plot identifier, and select the
corresponding plot as the current plot. All subsequent plotting will
be directed to this device until the assignment is changed by another
call to GRSLCT.

Argument:

IDENT (input, integer): the identifier of the plot to be selected, as
      returned by GROPEN.
```

GRSLS -- set line style

```
      SUBROUTINE GRSLS (IS)
      INTEGER IS
```

GRPCKG: Set the line style for subsequent plotting on the current device. The different line styles are generated in hardware on some devices and by GRPCKG software for the other devices. Five different line styles are available, with the following codes: 1 (full line), 2 (dashed), 3 (dot-dash-dot-dash), 4 (dotted), 5 (dash-dot-dot-dot). The default is 1 (normal full line). Line style is ignored when drawing characters, which are always drawn with a full line.

Argument:

IS (input, integer): the line-style code for subsequent plotting on
      the current device (in range 1-5).

GRSLW -- set line width

```
      SUBROUTINE GRSLW (IW)
      INTEGER IW
```

GRPCKG: Set the line width for subsequent plotting on the current device. If the hardware does not support thick lines, they are simulated by tracing each line with multiple strokes offset in the direction perpendicular to the line. The line width is specified by the number of strokes to be used, which must be in the range 1-201. The actual line width obtained depends on the device resolution. If the hardware does support thick lines, the width of the line is approximately 0.005 inches times the value of argument IW.

Argument:

IW (integer, input): the number of strokes to be used for subsequent
      plotting on the current device (in range 1-201).

GRSYDS -- decode character string into list of symbol numbers

```
      SUBROUTINE GRSYDS (SYMBOL, NSYMBS, TEXT, FONT)
      INTEGER SYMBOL(*), NSYMBS, FONT
      CHARACTER*(*) TEXT
```

Given a character string, this routine returns a list of symbol numbers to be used to plot it. It is responsible for interpreting all escape sequences. Negative ‘symbol numbers’ are inserted in the list to represent pen movement. The following escape sequences are defined (the letter following the \ may be either upper or lower case):

```
\u       :      up one level (returns -1)
\d       :      down one level (returns -2)
\b       :      backspace (returns -3)
\A       :      (upper case only) Angstrom symbol, roman font
\x       :      multiplication sign
```

```
\\       :      \, returns the code for backslash
\gx      :      greek letter corresponding to roman letter x
\fn      :      switch to Normal font
\fr      :      switch to Roman font
\fi      :      switch to Italic font
\fs      :      switch to Script font
\mn or \mnn :   graph marker number n or nn (1 or 2 digits)
\(nnn)   :      Hershey symbol number nnn (any number of digits)
```

```
Arguments:
 SYMBOL (output) : receives the list of symbol numers.
 NSYMBS (output) : receives the actual number of symbols specified
                   by the string; it is assumed that the dimension of
                   SYMBOL is big enough (not less than LEN(TEXT)).
 TEXT   (input)  : the text string to be decoded.
 FONT   (input)  : the font number (1..4) to be used for decoding the
                   string (this can be overridden by an escape
                   sequence within the string).
```

GRSYMK -- convert character number into symbol number

```
      SUBROUTINE GRSYMK (CODE, FONT, SYMBOL)
      INTEGER CODE, FONT, SYMBOL
```

This routine returns the Hershey symbol number (SYMBOL) corresponding to ASCII code CODE in font FONT.

Characters 0-31 are the same in all fonts, and are the standard graph markers. Characters 32-127 are standard representations of the ASCII codes. Characters >127 are used for special characters (greek alphabet, etc.).

```
Arguments:
 CODE   (input)  : the ASCII code number (range 0-255).
 FONT   (input)  : the font to be used 31 (range 1-4).
 SYMBOL (output) : the number of the symbol to be plotted.
```

GRSYXD -- obtain the polyline representation of a given symbol

```
      SUBROUTINE GRSYXD (SYMBOL, XYGRID, UNUSED)
      INTEGER SYMBOL
      INTEGER XYGRID(300)
      LOGICAL UNUSED
```

Return the digitization coordinates of a character. Each character is defined on a grid with X and Y coordinates in the range (-49,49), with the origin (0,0) at the center of the character. The coordinate system is right-handed, with X positive to the right, and Y positive upward.

```
Arguments:
 SYMBOL (input)  : symbol number in range (1..3000).
 XYGRID (output) : height range, width range, and pairs of (x,y)
                   coordinates returned.  Height range = (XYGRID(1),
```

```
                  XYGRID(3)).  Width range = (XYGRID(4),XYGRID(5)).
                  (X,Y) = (XYGRID(K),XYGRID(K+1)) (K=6,8,...).
 UNUSED (output) : receives .TRUE. if SYMBOL is an unused symbol
                  number. A character of normal height and zero width
                  is returned. Receives .FALSE. if SYMBOL is a
                  valid symbol number.
```

The height range consists of 3 values: (minimum Y, baseline Y, maximum Y). The first is reached by descenders on lower-case g, p, q, and y. The second is the bottom of upper-case letters. The third is the top of upper-case letters. A coordinate pair (-64,0) requests a pen raise, and a pair (-64,-64) terminates the coordinate list. It is assumed that movement to the first coordinate position will be done with the pen raised - no raise command is explicitly included to do this.

GRTERM -- flush buffer to output device

```
      SUBROUTINE GRTERM
```

GRPCKG: flush the buffer associated with the current plot. GRTERM should be called only when it is necessary to make sure that all the graphics created up to this point in the program are visible on the device, e.g., before beginning a dialog with the user. GRTERM has no effect on hardcopy devices.

Arguments: none.

GRTEXT -- draw text

```
      SUBROUTINE GRTEXT (CENTER,ORIENT,ABSXY,X0,Y0,STRING)
```

GRPCKG: Write a text string using the high-quality character set. The text is NOT windowed in the current viewport, but may extend over the whole view surface. Line attributes (color, intensity thickness) apply to text, but line-style is ignored. The current pen position after a call to GRTEXT is undefined.

Arguments:

STRING (input, character): the character string to be plotted. This may include standard escape-sequences to represent non-ASCII characters and special commands. The number of characters in STRING (i.e., LEN(STRING)) should not exceed 256.

GRTOUP -- convert character string to upper case

```
      SUBROUTINE GRTOUP (DST, SRC)
      CHARACTER*(*) DST, SRC
```

```
GRPCKG (internal routine): convert character string to upper case.

Arguments:
 DST    (output) : output string (upper case).
 SRC    (input)  : input string to be converted.
```

GRTRAN -- define scaling transformation

```
      SUBROUTINE GRTRAN (IDENT,XORG,YORG,XSCALE,YSCALE)

GRPCKG (internal routine): Define scaling transformation.

Arguments:

IDENT (input, integer): plot identifier, as returned by GROPEN.
XORG, YORG, XSCALE, YSCALE (input, real): parameters of the scaling
      transformation. This is defined by:
              XABS = XORG + XWORLD * XSCALE,
              YABS = YORG + YWORLD * YSCALE,
      where (XABS, YABS) are the absolute device coordinates
      corresponding to world coordinates (XWORLD, YWORLD).
```

GRTRIM -- length of string excluding trailing blanks

```
      INTEGER FUNCTION GRTRIM(S)
      CHARACTER*(*) S

Find the length of a character string excluding trailing blanks.
A blank string returns a value of 0.

Argument:
 S      (input)  : character string.

Returns:
 GRTRIM          : number of characters in S, excluding trailing
                   blanks, in range 0...LEN(S). A blank string
                   returns a value of 0.

Subroutines required:
 None

Fortran 77 extensions:
 None

History:
 1987 Nov 12 - TJP.
```

GRTRN0 -- define scaling transformation

```
      SUBROUTINE GRTRN0 (XORG,YORG,XSCALE,YSCALE)
```

GRPCKG (internal routine): Define scaling transformation for current device (equivalent to GRTRAN without device selection).

Arguments:

```
XORG, YORG, XSCALE, YSCALE (input, real): parameters of the scaling
      transformation. This is defined by:
              XABS = XORG + XWORLD * XSCALE,
              YABS = YORG + YWORLD * YSCALE,
      where (XABS, YABS) are the absolute device coordinates
      corresponding to world coordinates (XWORLD, YWORLD).
```

GRTXY0 -- convert world coordinates to device coordinates

```
      SUBROUTINE GRTXY0 (ABSXY,X,Y,XT,YT)
```

GRPCKG (internal routine): Convert scaled position to absolute position.

Arguments:

```
ABSXY (input, logical): if FALSE, convert world coordinates to
      absolute device coordinates; if TRUE, return the input
      coordinates unchanged.
X, Y (input, real): input coordinates (absolute or world, depending
      on setting of ABSXY).
XT, YT (output, real): output absolute device coordinates.
```

GRVCT0 -- draw line segments or dots

```
      SUBROUTINE GRVCT0 (MODE,ABSXY,POINTS,X,Y)
```

GRPCKG (internal routine): Draw a line or a set of dots. This is the same as GRVECT, but without device selection. It can be used to draw a single line-segment, a continuous series of line segments, or one or more single dots (pixels).

Arguments:

```
MODE (input, integer): if MODE=1, a series of line segments is drawn,
      starting at the current position, moving to X(1),Y(1), ... and
      ending at X(POINTS),Y(POINTS).
      If MODE=2, the first vector is blanked, so the line starts at
      X(1),Y(1).
      If MODE=3, a single dot is placed at each coordinate pair, with
      no connecting lines.
ABSXY (input, logical): if TRUE, the coordinates are absolute device
      coordinates; if FALSE, they are world coordinates and the
      scaling transformation is applied.
POINTS (input, integer): the number of coordinate pairs.
X, Y (input, real arrays, dimensioned POINTS or greater): the
      X and Y coordinates of the points.
```

GRVECT -- draw line segments or dots

```
      SUBROUTINE GRVECT (IDENT,MODE,ABSXY,POINTS,X,Y)

GRPCKG: Draw a line or a set of dots. This routine can be used to
draw a single line-segment, a continuous series of line segments, or
one or more single dots (pixels).

Arguments:

IDENT (input, integer): the plot identifier, as returned by GROPEN.
MODE (input, integer): if MODE=1, a series of line segments is drawn,
      starting at the current position, moving to X(1),Y(1), ... and
      ending at X(POINTS),Y(POINTS).
      If MODE=2, the first vector is blanked, so the line starts at
      X(1),Y(1).
      If MODE=3, a single dot is placed at each coordinate pair, with
      no connecting lines.
ABSXY (input, logical): if TRUE, the coordinates are absolute device
      coordinates; if FALSE, they are world coordinates and the
      scaling transformation is applied.
POINTS (input, integer): the number of coordinate pairs.
X, Y (input, real arrays, dimensioned POINTS or greater): the
      X and Y coordinates of the points.
```

GRWARN -- issue warning message to user

```
      SUBROUTINE GRWARN (TEXT)
      CHARACTER*(*) TEXT

Report a warning message on standard output, with prefix "%PGPLOT, ".

Argument:
 TEXT (input): text of message to be printed (the string
     may not be blank).
```

GRXHLS -- convert RGB color to HLS color

```
      SUBROUTINE GRXHLS (R,G,B,H,L,S)

GRPCKG: Convert a color specified in the RGB color model to one in
the HLS model.  This is a support routine: no graphics I/O occurs.
The inverse transformation is accomplished with routine GRXRGB.
Reference: SIGGRAPH Status Report of the Graphic Standards Planning
Committee, Computer Graphics, Vol.13, No.3, Association for
Computing Machinery, New York, NY, 1979.

Arguments:

R,G,B (real, input): red, green, blue color coordinates, each in the
```

```
      range 0.0 to 1.0. Input outside this range causes HLS = (0,1,0)
      [white] to be returned.
H,L,S (real, output): hue (0 to 360), lightness (0 to 1.0), and
      saturation (0 to 1.0).
```

GRXRGB -- convert HLS color to RGB color

```
      SUBROUTINE GRXRGB (H,L,S,R,G,B)
```

GRPCKG: Convert a color specified in the HLS color model to one in the RGB model. This is a support routine: no graphics I/O occurs. The inverse transformation is accomplished with routine GRXHLS. Reference: SIGGRAPH Status Report of the Graphic Standards Planning Committee, Computer Graphics, Vol.13, No.3, Association for Computing Machinery, New York, NY, 1979.

Arguments:

```
H,L,S (real, input): hue (0 to 360), lightness (0 to 1.0), and
      saturation (0 to 1.0).
R,G,B (real, output): red, green, blue color coordinates, each in the
      range 0.0 to 1.0.
```

PGADVANCE -- non-standard alias for PGPAGE

```
      SUBROUTINE PGADVANCE
```

PGARRO -- draw an arrow

```
      SUBROUTINE PGARRO (X1, Y1, X2, Y2)
      REAL X1, Y1, X2, Y2
```

Draw an arrow from the point with world-coordinates (X1,Y1) to (X2,Y2). The size of the arrowhead at (X2,Y2) is determined by the current character size set by routine PGSCH. The default size is 1/40th of the smaller of the width or height of the view surface. The appearance of the arrowhead (shape and solid or open) is controlled by routine PGSAH.

```
Arguments:
 X1, Y1 (input)  : world coordinates of the tail of the arrow.
 X2, Y2 (input)  : world coordinates of the head of the arrow.
```

PGASK -- control new page prompting

```
      SUBROUTINE PGASK (FLAG)
```

```
      LOGICAL FLAG
```

Change the ``prompt state'' of PGPLOT. If the prompt state is ON, PGPAGE will type ``Type RETURN for next page:'' and will wait for the user to type a carriage-return before starting a new page. The initial prompt state (after a call to PGBEG) is ON for interactive devices. Prompt state is always OFF for non-interactive devices.

```
Arguments:
 FLAG   (input)  : if .TRUE., and if the device is an interactive
                   device, the prompt state will be set to ON. If
                   .FALSE., the prompt state will be set to OFF.
```

PGBAND -- read cursor position, with anchor

```
      INTEGER FUNCTION PGBAND (MODE, POSN, XREF, YREF, X, Y, CH)
      INTEGER MODE, POSN
      REAL XREF, YREF, X, Y
      CHARACTER*(*) CH
```

Read the cursor position and a character typed by the user. The position is returned in world coordinates. PGBAND positions the cursor at the position specified (if POSN=1), allows the user to move the cursor using the mouse or arrow keys or whatever is available on the device. When he has positioned the cursor, the user types a single character on the keyboard; PGBAND then returns this character and the new cursor position (in world coordinates).

Some interactive devices offer a selection of cursor types, implemented as thin lines that move with the cursor, but without erasing underlying graphics. Of these types, some extend between a stationary anchor-point at XREF,YREF, and the position of the cursor, while others simply follow the cursor without changing shape or size. The cursor type is specified with one of the following MODE values. Cursor types that are not supported by a given device, are treated as MODE=0.

-- If MODE=0, the anchor point is ignored and the routine behaves like PGCURS.
-- If MODE=1, a straight line is drawn joining the anchor point and the cursor position.
-- If MODE=2, a hollow rectangle is extended as the cursor is moved, with one vertex at the anchor point and the opposite vertex at the current cursor position; the edges of the rectangle are horizontal and vertical.
-- If MODE=3, two horizontal lines are extended across the width of the display, one drawn through the anchor point and the other through the moving cursor position. This could be used to select a Y-axis range when one end of the range is known.
-- If MODE=4, two vertical lines are extended over the height of the display, one drawn through the anchor point and the other through the moving cursor position. This could be used to select an X-axis range when one end of the range is known.
-- If MODE=5, a horizontal line is extended through the cursor position over the width of the display. This could be used to select an X-axis value such as the start of an X-axis range. The anchor point

```
is ignored.
-- If MODE=6, a vertical line is extended through the cursor
position over the height of the display. This could be used to select
a Y-axis value such as the start of a Y-axis range. The anchor point
is ignored.
-- If MODE=7, a cross-hair, centered on the cursor, is extended over
the width and height of the display. The anchor point is ignored.

Returns:
 PGBAND          : 1 if the call was successful; 0 if the device
                   has no cursor or some other error occurs.
Arguments:
 MODE   (input)  : display mode (0, 1, ..7: see above).
 POSN   (input)  : if POSN=1, PGBAND attempts to place the cursor
                   at point (X,Y); if POSN=0, it leaves the cursor
                   at its current position. (On some devices this
                   request may be ignored.)
 XREF   (input)  : the world x-coordinate of the anchor point.
 YREF   (input)  : the world y-coordinate of the anchor point.
 X      (in/out) : the world x-coordinate of the cursor.
 Y      (in/out) : the world y-coordinate of the cursor.
 CH     (output) : the character typed by the user; if the device has
                   no cursor or if some other error occurs, the value
                   CHAR(0) [ASCII NUL character] is returned.

Note: The cursor coordinates (X,Y) may be changed by PGBAND even if
the device has no cursor or if the user does not move the cursor.
Under these circumstances, the position returned in (X,Y) is that of
the pixel nearest to the requested position.
```

PGBBUF -- begin batch of output (buffer)

```
      SUBROUTINE PGBBUF

Begin saving graphical output commands in an internal buffer; the
commands are held until a matching PGEBUF call (or until the buffer
is emptied by PGUPDT). This can greatly improve the efficiency of
PGPLOT.  PGBBUF increments an internal counter, while PGEBUF
decrements this counter and flushes the buffer to the output
device when the counter drops to zero.  PGBBUF and PGEBUF calls
should always be paired.

Arguments: none
```

PGBEG -- begin PGPLOT, open output device

```
      INTEGER FUNCTION PGBEG (UNIT, FILE, NXSUB, NYSUB)
      INTEGER       UNIT
      CHARACTER*(*) FILE
      INTEGER       NXSUB, NYSUB

Begin PGPLOT, open the plot file.  A call to PGBEG is
required before any other calls to PGPLOT subroutines.  If a plot
file is already open for PGPLOT output, it is closed before the new
```

```
file is opened.

Returns:
 PGBEG         : a status return value. A value of 1 indicates
                   successful completion, any other value indicates
                   an error. In the event of error a message is
                   written on the standard error unit.
                   To test the return value, call
                   PGBEG as a function, eg IER=PGBEG(...); note
                   that PGBEG must be declared INTEGER in the
                   calling program.
Arguments:
 UNIT  (input)   : this argument is ignored by PGBEG (use zero).
 FILE  (input)   : the "device specification" for the plot device.
                   Device specifications are installation dependent,
                   but usually have the form "device/type" or
                   "file/type". If this argument is a
                   question mark ('?'), PGBEG will prompt the user
                   to supply a string. If the argument is a blank
                   string (' '), PGBEG will use the value of
                   environment variable PGPLOT_DEV.
 NXSUB  (input)  : the number of subdivisions of the view surface in
                   X (>0 or <0).
 NYSUB  (input)  : the number of subdivisions of the view surface in
                   Y (>0).
                   PGPLOT puts NXSUB x NYSUB graphs on each plot
                   page or screen; when the view surface is sub-
                   divided in this way, PGPAGE moves to the next
                   panel, not the  next physical page. If
                   NXSUB > 0, PGPLOT uses the panels in row
                   order; if <0, PGPLOT uses them in column order.
```

PGBEGIN -- non-standard alias for PGBEG

```
      INTEGER FUNCTION PGBEGIN (UNIT, FILE, NXSUB, NYSUB)
      INTEGER       UNIT
      CHARACTER*(*) FILE
      INTEGER       NXSUB, NYSUB
```

PGBIN -- histogram of binned data

```
      SUBROUTINE PGBIN (NBIN, X, DATA, CENTER)
      INTEGER NBIN
      REAL X(*), DATA(*)
      LOGICAL CENTER

Plot a histogram of NBIN values with X(1..NBIN) values along
the ordinate, and DATA(1...NBIN) along the abscissa. Bin width is
spacing between X values.

Arguments:
 NBIN   (input)  : number of values.
 X      (input)  : abscissae of bins.
 DATA   (input)  : data values of bins.
```

```
 CENTER (input)  : if .TRUE., the X values denote the center of the
                   bin; if .FALSE., the X values denote the lower
                   edge (in X) of the bin.
```

PGBOX -- draw labeled frame around viewport

```
      SUBROUTINE PGBOX (XOPT, XTICK, NXSUB, YOPT, YTICK, NYSUB)
      CHARACTER*(*) XOPT, YOPT
      REAL XTICK, YTICK
      INTEGER NXSUB, NYSUB
```

```
Annotate the viewport with frame, axes, numeric labels, etc.
PGBOX is called by on the user's behalf by PGENV, but may also be
called explicitly.

Arguments:
 XOPT   (input)  : string of options for X (horizontal) axis of
                   plot. Options are single letters, and may be in
                   any order (see below).
 XTICK  (input)  : world coordinate interval between major tick marks
                   on X axis. If XTICK=0.0, the interval is chosen by
                   PGBOX, so that there will be at least 3 major tick
                   marks along the axis.
 NXSUB  (input)  : the number of subintervals to divide the major
                   coordinate interval into. If XTICK=0.0 or NXSUB=0,
                   the number is chosen by PGBOX.
 YOPT   (input)  : string of options for Y (vertical) axis of plot.
                   Coding is the same as for XOPT.
 YTICK  (input)  : like XTICK for the Y axis.
 NYSUB  (input)  : like NXSUB for the Y axis.

Options (for parameters XOPT and YOPT):
 A : draw Axis (X axis is horizontal line Y=0, Y axis is vertical
     line X=0).
 B : draw bottom (X) or left (Y) edge of frame.
 C : draw top (X) or right (Y) edge of frame.
 G : draw Grid of vertical (X) or horizontal (Y) lines.
 I : Invert the tick marks; ie draw them outside the viewport
     instead of inside.
 L : label axis Logarithmically (see below).
 N : write Numeric labels in the conventional location below the
     viewport (X) or to the left of the viewport (Y).
 P : extend ("Project") major tick marks outside the box (ignored if
     option I is specified).
 M : write numeric labels in the unconventional location above the
     viewport (X) or to the right of the viewport (Y).
 T : draw major Tick marks at the major coordinate interval.
 S : draw minor tick marks (Subticks).
 V : orient numeric labels Vertically. This is only applicable to Y.
     The default is to write Y-labels parallel to the axis.
 1 : force decimal labelling, instead of automatic choice (see PGNUMB).
 2 : force exponential labelling, instead of automatic.

To get a complete frame, specify BC in both XOPT and YOPT.
Tick marks, if requested, are drawn on the axes or frame
or both, depending which are requested. If none of ABC is specified,
tick marks will not be drawn. When PGENV calls PGBOX, it sets both
```

XOPT and YOPT according to the value of its parameter AXIS:
-1: 'BC', 0: 'BCNST', 1: 'ABCNST', 2: 'ABCGNST'.

For a logarithmic axis, the major tick interval is always 1.0. The numeric label is 10**(x) where x is the world coordinate at the tick mark. If subticks are requested, 8 subticks are drawn between each major tick at equal logarithmic intervals.

To label an axis with time (days, hours, minutes, seconds) or angle (degrees, arcmin, arcsec), use routine PGTBOX.

PGCIRC -- draw a filled or outline circle

```
      SUBROUTINE PGCIRC (XCENT, YCENT, RADIUS)
      REAL XCENT, YCENT, RADIUS
```

Draw a circle. The action of this routine depends on the setting of the Fill-Area Style attribute. If Fill-Area Style is SOLID (the default), the interior of the circle is solid-filled using the current Color Index. If Fill-Area Style is HOLLOW, the outline of the circle is drawn using the current line attributes (color index, line-style, and line-width).

```
Arguments:
 XCENT  (input)  : world x-coordinate of the center of the circle.
 YCENT  (input)  : world y-coordinate of the center of the circle.
 RADIUS (input)  : radius of circle (world coordinates).
```

PGCONB -- contour map of a 2D data array, with blanking

```
      SUBROUTINE PGCONB (A, IDIM, JDIM, I1, I2, J1, J2, C, NC, TR,
     1                   BLANK)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, NC
      REAL    A(IDIM,JDIM), C(*), TR(6), BLANK
```

Draw a contour map of an array. This routine is the same as PGCONS, except that array elements that have the "magic value" defined by argument BLANK are ignored, making gaps in the contour map. The routine may be useful for data measured on most but not all of the points of a grid.

```
Arguments:
 A      (input)  : data array.
 IDIM   (input)  : first dimension of A.
 JDIM   (input)  : second dimension of A.
 I1,I2  (input)  : range of first index to be contoured (inclusive).
 J1,J2  (input)  : range of second index to be contoured (inclusive).
 C      (input)  : array of contour levels (in the same units as the
                   data in array A); dimension at least NC.
 NC     (input)  : number of contour levels (less than or equal to
                   dimension of C). The absolute value of this
                   argument is used (for compatibility with PGCONT,
                   where the sign of NC is significant).
 TR     (input)  : array defining a transformation between the I,J
```

```
                 grid of the array and the world coordinates. The
                 world coordinates of the array point A(I,J) are
                 given by:
                   X = TR(1) + TR(2)*I + TR(3)*J
                   Y = TR(4) + TR(5)*I + TR(6)*J
                 Usually TR(3) and TR(5) are zero - unless the
                 coordinate transformation involves a rotation
                 or shear.
 BLANK   (input) : elements of array A that are exactly equal to
                 this value are ignored (blanked).
```

PGCONL -- label contour map of a 2D data array

```
      SUBROUTINE PGCONL (A, IDIM, JDIM, I1, I2, J1, J2, C, TR,
     1                   LABEL, INTVAL, MININT)
      INTEGER IDIM, JDIM, I1, J1, I2, J2, INTVAL, MININT
      REAL A(IDIM,JDIM), C, TR(6)
      CHARACTER*(*) LABEL
```

Label a contour map drawn with routine PGCONT. Routine PGCONT should be called first to draw the contour lines, then this routine should be called to add the labels. Labels are written at intervals along the contour lines, centered on the contour lines with lettering aligned in the up-hill direction. Labels are opaque, so a part of the under-lying contour line is obscured by the label. Labels use the current attributes (character height, line width, color index, character font).

The first 9 arguments are the same as those supplied to PGCONT, and should normally be identical to those used with PGCONT. Note that only one contour level can be specified; tolabel more contours, call PGCONL for each level.

The Label is supplied as a character string in argument LABEL.

The spacing of labels along the contour is specified by parameters INTVAL and MININT. The routine follows the contour through the array, counting the number of cells that the contour crosses. The first label will be written in the MININT'th cell, and additional labels will be written every INTVAL cells thereafter. A contour that crosses less than MININT cells will not be labelled. Some experimentation may be needed to get satisfactory results; a good place to start is INTVAL=20, MININT=10.

```
Arguments:
 A      (input) : data array.
 IDIM   (input) : first dimension of A.
 JDIM   (input) : second dimension of A.
 I1, I2 (input) : range of first index to be contoured (inclusive).
 J1, J2 (input) : range of second index to be contoured (inclusive).
 C      (input) : the level of the contour to be labelled (one of the
                  values given to PGCONT).
 TR     (input) : array defining a transformation between the I,J
                  grid of the array and the world coordinates.
                  The world coordinates of the array point A(I,J)
                  are given by:
                    X = TR(1) + TR(2)*I + TR(3)*J
```

```
                  Y = TR(4) + TR(5)*I + TR(6)*J
                Usually TR(3) and TR(5) are zero - unless the
                coordinate transformation involves a rotation or
                shear.
 LABEL  (input) : character strings to be used to label the specified
                contour. Leading and trailing blank spaces are
                ignored.
 INTVAL (input) : spacing along the contour between labels, in
                grid cells.
 MININT (input) : contours that cross less than MININT cells
                will not be labelled.
```

PGCONS -- contour map of a 2D data array (fast algorithm)

```
      SUBROUTINE PGCONS (A, IDIM, JDIM, I1, I2, J1, J2, C, NC, TR)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, NC
      REAL    A(IDIM,JDIM), C(*), TR(6)
```

```
Draw a contour map of an array. The map is truncated if
necessary at the boundaries of the viewport.  Each contour line is
drawn with the current line attributes (color index, style, and
width).  This routine, unlike PGCONT, does not draw each contour as a
continuous line, but draws the straight line segments composing each
contour in a random order.  It is thus not suitable for use on pen
plotters, and it usually gives unsatisfactory results with dashed or
dotted lines.  It is, however, faster than PGCONT, especially if
several contour levels are drawn with one call of PGCONS.

Arguments:
 A      (input)  : data array.
 IDIM   (input)  : first dimension of A.
 JDIM   (input)  : second dimension of A.
 I1,I2  (input)  : range of first index to be contoured (inclusive).
 J1,J2  (input)  : range of second index to be contoured (inclusive).
 C      (input)  : array of contour levels (in the same units as the
                   data in array A); dimension at least NC.
 NC     (input)  : number of contour levels (less than or equal to
                   dimension of C). The absolute value of this
                   argument is used (for compatibility with PGCONT,
                   where the sign of NC is significant).
 TR     (input)  : array defining a transformation between the I,J
                   grid of the array and the world coordinates. The
                   world coordinates of the array point A(I,J) are
                   given by:
                     X = TR(1) + TR(2)*I + TR(3)*J
                     Y = TR(4) + TR(5)*I + TR(6)*J
                   Usually TR(3) and TR(5) are zero - unless the
                   coordinate transformation involves a rotation
                   or shear.
```

PGCONT -- contour map of a 2D data array (contour-following)

```
      SUBROUTINE PGCONT (A, IDIM, JDIM, I1, I2, J1, J2, C, NC, TR)
      INTEGER IDIM, JDIM, I1, J1, I2, J2, NC
```

```
      REAL A(IDIM,JDIM), C(*), TR(6)
```

```
Draw a contour map of an array.  The map is truncated if
necessary at the boundaries of the viewport.  Each contour line
is drawn with the current line attributes (color index, style, and
width); except that if argument NC is positive (see below), the line
style is set by PGCONT to 1 (solid) for positive contours or 2
(dashed) for negative contours.

Arguments:
 A      (input) : data array.
 IDIM   (input) : first dimension of A.
 JDIM   (input) : second dimension of A.
 I1, I2 (input) : range of first index to be contoured (inclusive).
 J1, J2 (input) : range of second index to be contoured (inclusive).
 C      (input) : array of NC contour levels; dimension at least NC.
 NC     (input) : +/- number of contour levels (less than or equal
                  to dimension of C). If NC is positive, it is the
                  number of contour levels, and the line-style is
                  chosen automatically as described above. If NC is
                  negative, it is minus the number of contour
                  levels, and the current setting of line-style is
                  used for all the contours.
 TR     (input) : array defining a transformation between the I,J
                  grid of the array and the world coordinates.
                  The world coordinates of the array point A(I,J)
                  are given by:
                    X = TR(1) + TR(2)*I + TR(3)*J
                    Y = TR(4) + TR(5)*I + TR(6)*J
                  Usually TR(3) and TR(5) are zero - unless the
                  coordinate transformation involves a rotation or
                  shear.
```

PGCONX -- contour map of a 2D data array (non-rectangular)

```
      SUBROUTINE PGCONX (A, IDIM, JDIM, I1, I2, J1, J2, C, NC, PLOT)
      INTEGER  IDIM, JDIM, I1, J1, I2, J2, NC
      REAL     A(IDIM,JDIM), C(*)
      EXTERNAL PLOT
```

```
Draw a contour map of an array using a user-supplied plotting
routine.  This routine should be used instead of PGCONT when the
data are defined on a non-rectangular grid.  PGCONT permits only
a linear transformation between the (I,J) grid of the array
and the world coordinate system (x,y), but PGCONX permits any
transformation to be used, the transformation being defined by a
user-supplied subroutine. The nature of the contouring algorithm,
however, dictates that the transformation should maintain the
rectangular topology of the grid, although grid-points may be
allowed to coalesce.  As an example of a deformed rectangular
grid, consider data given on the polar grid theta=0.1n(pi/2),
for n=0,1,...,10, and r=0.25m, for m=0,1,..,4. This grid
contains 55 points, of which 11 are coincident at the origin.
The input array for PGCONX should be dimensioned (11,5), and
data values should be provided for all 55 elements.  PGCONX can
also be used for special applications in which the height of the
contour affects its appearance, e.g., stereoscopic views.
```

The map is truncated if necessary at the boundaries of the viewport.
Each contour line is drawn with the current line attributes (color
index, style, and width); except that if argument NC is positive
(see below), the line style is set by PGCONX to 1 (solid) for
positive contours or 2 (dashed) for negative contours. Attributes
for the contour lines can also be set in the user-supplied
subroutine, if desired.

```
Arguments:
 A      (input) : data array.
 IDIM   (input) : first dimension of A.
 JDIM   (input) : second dimension of A.
 I1, I2 (input) : range of first index to be contoured (inclusive).
 J1, J2 (input) : range of second index to be contoured (inclusive).
 C      (input) : array of NC contour levels; dimension at least NC.
 NC     (input) : +/- number of contour levels (less than or equal
                  to dimension of C). If NC is positive, it is the
                  number of contour levels, and the line-style is
                  chosen automatically as described above. If NC is
                  negative, it is minus the number of contour
                  levels, and the current setting of line-style is
                  used for all the contours.
 PLOT   (input) : the address (name) of a subroutine supplied by
                  the user, which will be called by PGCONX to do
                  the actual plotting. This must be declared
                  EXTERNAL in the program unit calling PGCONX.
```

The subroutine PLOT will be called with four arguments:
CALL PLOT(VISBLE,X,Y,Z)
where X,Y (input) are real variables corresponding to
I,J indices of the array A. If VISBLE (input, integer) is 1,
PLOT should draw a visible line from the current pen
position to the world coordinate point corresponding to (X,Y);
if it is 0, it should move the pen to (X,Y). Z is the value
of the current contour level, and may be used by PLOT if desired.
Example:

```
      SUBROUTINE PLOT (VISBLE,X,Y,Z)
      REAL X, Y, Z, XWORLD, YWORLD
      INTEGER VISBLE
      XWORLD = X*COS(Y) ! this is the user-defined
      YWORLD = X*SIN(Y) ! transformation
      IF (VISBLE.EQ.0) THEN
          CALL PGMOVE (XWORLD, YWORLD)
      ELSE
          CALL PGDRAW (XWORLD, YWORLD)
      END IF
      END
```

PGCTAB -- install the color table to be used by PGIMAG

```
      SUBROUTINE PGCTAB(L, R, G, B, NC, CONTRA, BRIGHT)
      INTEGER NC
      REAL    L(NC), R(NC), G(NC), B(NC), CONTRA, BRIGHT
```

Use the given color table to change the color representations of
all color indexes marked for use by PGIMAG. To change which
color indexes are thus marked, call PGSCIR before calling PGCTAB

or PGIMAG. On devices that can change the color representations of previously plotted graphics, PGCTAB will also change the colors of existing graphics that were plotted with the marked color indexes. This feature can then be combined with PGBAND to interactively manipulate the displayed colors of data previously plotted with PGIMAG.

```
Limitations:
 1. Some devices do not propagate color representation changes
    to previously drawn graphics.
 2. Some devices ignore requests to change color representations.
 3. The appearance of specific color representations on grey-scale
    devices is device-dependent.

Arguments:
 L      (input)  : An array of NC normalized ramp-intensity levels
                   corresponding to the RGB primary color intensities
                   in R(),G(),B(). Colors on the ramp are linearly
                   interpolated from neighbouring levels.
                   Levels must be sorted in increasing order.
                    0.0 places a color at the beginning of the ramp.
                    1.0 places a color at the end of the ramp.
                   Colors outside these limits are legal, but will
                   not be visible if CONTRA=1.0 and BRIGHT=0.5.
 R      (input)  : An array of NC normalized red intensities.
 G      (input)  : An array of NC normalized green intensities.
 B      (input)  : An array of NC normalized blue intensities.
 NC     (input)  : The number of color table entries.
 CONTRA (input)  : The contrast of the color ramp (normally 1.0).
 BRIGHT (input)  : Brightness at the center colorindex (normally 0.5).
```

PGCURS -- read cursor position

```
      INTEGER FUNCTION PGCURS (X, Y, CH)
      REAL X, Y
      CHARACTER*(*) CH
```

Read the cursor position and a character typed by the user. The position is returned in world coordinates. PGCURS positions the cursor at the position specified, allows the user to move the cursor using the joystick or arrow keys or whatever is available on the device. When he has positioned the cursor, the user types a single character on the keyboard; PGCURS then returns this character and the new cursor position (in world coordinates).

```
Returns:
 PGCURS         : 1 if the call was successful; 0 if the device
                   has no cursor or some other error occurs.
Arguments:
 X      (in/out) : the world x-coordinate of the cursor.
 Y      (in/out) : the world y-coordinate of the cursor.
 CH     (output) : the character typed by the user; if the device has
                   no cursor or if some other error occurs, the value
                   CHAR(0) [ASCII NUL character] is returned.
```

Note: The cursor coordinates (X,Y) may be changed by PGCURS even if the device has no cursor or if the user does not move the cursor.

Under these circumstances, the position returned in (X,Y) is that of
the pixel nearest to the requested position.

PGCURSE -- non-standard alias for PGCURS

```
      INTEGER FUNCTION PGCURSE (X, Y, CH)
      REAL X, Y
      CHARACTER*1 CH
```

PGDRAW -- draw a line from the current pen position to a point

```
      SUBROUTINE PGDRAW (X, Y)
      REAL X, Y
```

Draw a line from the current pen position to the point
with world-coordinates (X,Y). The line is clipped at the edge of the
current window. The new pen position is (X,Y) in world coordinates.

```
Arguments:
 X      (input)  : world x-coordinate of the end point of the line.
 Y      (input)  : world y-coordinate of the end point of the line.
```

PGEBUF -- end batch of output (buffer)

```
      SUBROUTINE PGEBUF
```

A call to PGEBUF marks the end of a batch of graphical output begun
with the last call of PGBBUF. PGBBUF and PGEBUF calls should always
be paired. Each call to PGBBUF increments a counter, while each call
to PGEBUF decrements the counter. When the counter reaches 0, the
batch of output is written on the output device.

Arguments: none

PGEND -- terminate PGPLOT

```
      SUBROUTINE PGEND
```

Terminate PGPLOT, close the plot file, release the graphics
device. If the call to PGEND is omitted, some or all of the plot
may be lost.

Arguments: none

PGENV -- set window and viewport and draw labeled frame

```
      SUBROUTINE PGENV (XMIN, XMAX, YMIN, YMAX, JUST, AXIS)
      REAL XMIN, XMAX, YMIN, YMAX
      INTEGER JUST, AXIS
```

```
Set PGPLOT "Plotter Environment".  PGENV establishes the scaling
for subsequent calls to PGPT, PGLINE, etc.  The plotter is
advanced to a new page or panel, clearing the screen if necessary.
If the "prompt state" is ON (see PGASK), confirmation
is requested from the user before clearing the screen.
If requested, a box, axes, labels, etc. are drawn according to
the setting of argument AXIS.

Arguments:
 XMIN   (input)  : the world x-coordinate at the bottom left corner
                   of the viewport.
 XMAX   (input)  : the world x-coordinate at the top right corner
                   of the viewport (note XMAX may be less than XMIN).
 YMIN   (input)  : the world y-coordinate at the bottom left corner
                   of the viewport.
 YMAX   (input)  : the world y-coordinate at the top right corner
                   of the viewport (note YMAX may be less than YMIN).
 JUST   (input)  : if JUST=1, the scales of the x and y axes (in
                   world coordinates per inch) will be equal,
                   otherwise they will be scaled independently.
 AXIS   (input)  : controls the plotting of axes, tick marks, etc:
     AXIS = -2 : draw no box, axes or labels;
     AXIS = -1 : draw box only;
     AXIS =  0 : draw box and label it with coordinates;
     AXIS =  1 : same as AXIS=0, but also draw the
                 coordinate axes (X=0, Y=0);
     AXIS =  2 : same as AXIS=1, but also draw grid lines
                 at major increments of the coordinates;
     AXIS = 10 : draw box and label X-axis logarithmically;
     AXIS = 20 : draw box and label Y-axis logarithmically;
     AXIS = 30 : draw box and label both axes logarithmically.

For other axis options, use routine PGBOX. PGENV can be persuaded to
call PGBOX with additional axis options by defining an environment
parameter PGPLOT_ENVOPT containing the required option codes.
Examples:
  PGPLOT_ENVOPT=P      ! draw Projecting tick marks
  PGPLOT_ENVOPT=I      ! Invert the tick marks
  PGPLOT_ENVOPT=IV     ! Invert tick marks and label y Vertically
```

PGERAS -- erase all graphics from current page

```
      SUBROUTINE PGERAS
```

```
Erase all graphics from the current page or panel.

Arguments: none
```

PGERRB -- horizontal or vertical error bar

```
      SUBROUTINE PGERRB (DIR, N, X, Y, E, T)
      INTEGER DIR, N
      REAL X(*), Y(*), E(*)
      REAL T
```

```
Plot error bars in the direction specified by DIR.
This routine draws an error bar only; to mark the data point at
the start of the error bar, an additional call to PGPT is required.

Arguments:
 DIR    (input)  : direction to plot the error bar relative to
                   the data point.
                   One-sided error bar:
                     DIR is 1 for +X (X to X+E);
                            2 for +Y (Y to Y+E);
                            3 for -X (X to X-E);
                            4 for -Y (Y to Y-E).
                   Two-sided error bar:
                     DIR is 5 for +/-X (X-E to X+E);
                            6 for +/-Y (Y-E to Y+E).
 N      (input)  : number of error bars to plot.
 X      (input)  : world x-coordinates of the data.
 Y      (input)  : world y-coordinates of the data.
 E      (input)  : value of error bar distance to be added to the
                   data position in world coordinates.
 T      (input)  : length of terminals to be drawn at the ends
                   of the error bar, as a multiple of the default
                   length; if T = 0.0, no terminals will be drawn.

Note: the dimension of arrays X, Y, and E must be greater
than or equal to N. If N is 1, X, Y, and E may be scalar
variables, or expressions.
```

PGERRX -- horizontal error bar

```
      SUBROUTINE PGERRX (N, X1, X2, Y, T)
      INTEGER N
      REAL X1(*), X2(*), Y(*)
      REAL T
```

```
Plot horizontal error bars.
This routine draws an error bar only; to mark the data point in
the middle of the error bar, an additional call to PGPT or
PGERRY is required.

Arguments:
 N      (input)  : number of error bars to plot.
 X1     (input)  : world x-coordinates of lower end of the
                   error bars.
 X2     (input)  : world x-coordinates of upper end of the
                   error bars.
 Y      (input)  : world y-coordinates of the data.
 T      (input)  : length of terminals to be drawn at the ends
                   of the error bar, as a multiple of the default
```

```
                   length; if T = 0.0, no terminals will be drawn.

Note: the dimension of arrays X1, X2, and Y must be greater
than or equal to N. If N is 1, X1, X2, and Y may be scalar
variables, or expressions, eg:
      CALL PGERRX(1,X-SIGMA,X+SIGMA,Y)
```

PGERRY -- vertical error bar

```
      SUBROUTINE PGERRY (N, X, Y1, Y2, T)
      INTEGER N
      REAL X(*), Y1(*), Y2(*)
      REAL T

Plot vertical error bars.
This routine draws an error bar only; to mark the data point in
the middle of the error bar, an additional call to PGPT or
PGERRX is required.

Arguments:
 N      (input)  : number of error bars to plot.
 X      (input)  : world x-coordinates of the data.
 Y1     (input)  : world y-coordinates of top end of the
                   error bars.
 Y2     (input)  : world y-coordinates of bottom end of the
                   error bars.
 T      (input)  : length of terminals to be drawn at the ends
                   of the error bar, as a multiple of the default
                   length; if T = 0.0, no terminals will be drawn.

Note: the dimension of arrays X, Y1, and Y2 must be greater
than or equal to N. If N is 1, X, Y1, and Y2 may be scalar
variables or expressions, eg:
      CALL PGERRY(1,X,Y+SIGMA,Y-SIGMA)
```

PGETXT -- erase text from graphics display

```
      SUBROUTINE PGETXT

Some graphics terminals display text (the normal interactive dialog)
on the same screen as graphics. This routine erases the text from the
view surface without affecting the graphics. It does nothing on
devices which do not display text on the graphics screen, and on
devices which do not have this capability.

Arguments:
 None
```

PGFUNT -- function defined by X = F(T), Y = G(T)

```
      SUBROUTINE PGFUNT (FX, FY, N, TMIN, TMAX, PGFLAG)
      REAL FX, FY
      EXTERNAL FX, FY
      INTEGER N
      REAL TMIN, TMAX
      INTEGER PGFLAG
```

Draw a curve defined by parametric equations X = FX(T), Y = FY(T).

```
Arguments:
 FX     (external real function): supplied by the user, evaluates
                   X-coordinate.
 FY     (external real function): supplied by the user, evaluates
                   Y-coordinate.
 N      (input)  : the number of points required to define the
                   curve. The functions FX and FY will each be
                   called N+1 times.
 TMIN   (input)  : the minimum value for the parameter T.
 TMAX   (input)  : the maximum value for the parameter T.
 PGFLAG (input)  : if PGFLAG = 1, the curve is plotted in the
                   current window and viewport; if PGFLAG = 0,
                   PGENV is called automatically by PGFUNT to
                   start a new plot with automatic scaling.
```

Note: The functions FX and FY must be declared EXTERNAL in the Fortran program unit that calls PGFUNT.

PGFUNX -- function defined by Y = F(X)

```
      SUBROUTINE PGFUNX (FY, N, XMIN, XMAX, PGFLAG)
      REAL FY
      EXTERNAL FY
      INTEGER N
      REAL XMIN, XMAX
      INTEGER PGFLAG
```

Draw a curve defined by the equation Y = FY(X), where FY is a user-supplied subroutine.

```
Arguments:
 FY     (external real function): supplied by the user, evaluates
                   Y value at a given X-coordinate.
 N      (input)  : the number of points required to define the
                   curve. The function FY will be called N+1 times.
                   If PGFLAG=0 and N is greater than 1000, 1000
                   will be used instead.  If N is less than 1,
                   nothing will be drawn.
 XMIN   (input)  : the minimum value of X.
 XMAX   (input)  : the maximum value of X.
 PGFLAG (input)  : if PGFLAG = 1, the curve is plotted in the
                   current window and viewport; if PGFLAG = 0,
                   PGENV is called automatically by PGFUNX to
                   start a new plot with X limits (XMIN, XMAX)
                   and automatic scaling in Y.
```

Note: The function FY must be declared EXTERNAL in the Fortran program unit that calls PGFUNX. It has one argument, the

```
x-coordinate at which the y value is required, e.g.
  REAL FUNCTION FY(X)
  REAL X
  FY = .....
  END
```

PGFUNY -- function defined by X = F(Y)

```
      SUBROUTINE PGFUNY (FX, N, YMIN, YMAX, PGFLAG)
      REAL    FX
      EXTERNAL FX
      INTEGER N
      REAL    YMIN, YMAX
      INTEGER PGFLAG
```

```
Draw a curve defined by the equation X = FX(Y), where FY is a
user-supplied subroutine.

Arguments:
 FX     (external real function): supplied by the user, evaluates
                   X value at a given Y-coordinate.
 N      (input)  : the number of points required to define the
                   curve. The function FX will be called N+1 times.
                   If PGFLAG=0 and N is greater than 1000, 1000
                   will be used instead.  If N is less than 1,
                   nothing will be drawn.
 YMIN   (input)  : the minimum value of Y.
 YMAX   (input)  : the maximum value of Y.
 PGFLAG (input)  : if PGFLAG = 1, the curve is plotted in the
                   current window and viewport; if PGFLAG = 0,
                   PGENV is called automatically by PGFUNY to
                   start a new plot with Y limits (YMIN, YMAX)
                   and automatic scaling in X.

Note: The function FX must be declared EXTERNAL in the Fortran
program unit that calls PGFUNY.  It has one argument, the
y-coordinate at which the x value is required, e.g.
  REAL FUNCTION FX(Y)
  REAL Y
  FX = .....
  END
```

PGGRAY -- gray-scale map of a 2D data array

```
      SUBROUTINE PGGRAY (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   FG, BG, TR)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      REAL    A(IDIM,JDIM), FG, BG, TR(6)
```

```
Draw gray-scale map of an array in current window. The subsection
of the array A defined by indices (I1:I2, J1:J2) is mapped onto
the view surface world-coordinate system by the transformation
matrix TR. The resulting quadrilateral region is clipped at the edge
of the window and shaded with the shade at each point determined
```

by the corresponding array value. The shade is a number in the range 0 to 1 obtained by linear interpolation between the background level (BG) and the foreground level (FG), i.e.,

```
  shade = [A(i,j) - BG] / [FG - BG]
```

The background level BG can be either less than or greater than the foreground level FG. Points in the array that are outside the range BG to FG are assigned shade 0 or 1 as appropriate.

PGGRAY uses two different algorithms, depending how many color indices are available in the color index range specified for images. (This range is set with routine PGSCIR, and the current or default range can be queried by calling routine PGQCIR).

If 16 or more color indices are available, PGGRAY first assigns color representations to these color indices to give a linear ramp between the background color (color index 0) and the foreground color (color index 1), and then calls PGIMAG to draw the image using these color indices. In this mode, the shaded region is "opaque": every pixel is assigned a color.

If less than 16 color indices are available, PGGRAY uses only color index 1, and uses a "dithering" algorithm to fill in pixels, with the shade (computed as above) determining the faction of pixels that are filled. In this mode the shaded region is "transparent" and allows previously-drawn graphics to show through.

The transformation matrix TR is used to calculate the world coordinates of the center of the "cell" that represents each array element. The world coordinates of the center of the cell corresponding to array element A(I,J) are given by:

```
         X = TR(1) + TR(2)*I + TR(3)*J
         Y = TR(4) + TR(5)*I + TR(6)*J
```

Usually TR(3) and TR(5) are zero -- unless the coordinate transformation involves a rotation or shear. The corners of the quadrilateral region that is shaded by PGGRAY are given by applying this transformation to (I1-0.5,J1-0.5), (I2+0.5, J2+0.5).

```
Arguments:
 A      (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 FG     (input)  : the array value which is to appear with the
                   foreground color (corresponding to color index 1).
 BG     (input)  : the array value which is to appear with the
                   background color (corresponding to color index 0).
 TR     (input)  : transformation matrix between array grid and
                   world coordinates.
```

PGHI2D -- cross-sections through a 2D data array

```
      SUBROUTINE PGHI2D (DATA, NXV, NYV, IX1, IX2, IY1, IY2, X, IOFF,
     1                   BIAS, CENTER, YLIMS)
      INTEGER NXV, NYV, IX1, IX2, IY1, IY2
      REAL    DATA(NXV,NYV)
      REAL    X(IX2-IX1+1), YLIMS(IX2-IX1+1)
      INTEGER IOFF
      REAL    BIAS
      LOGICAL CENTER
```

```
Plot a series of cross-sections through a 2D data array.
Each cross-section is plotted as a hidden line histogram.  The plot
can be slanted to give a pseudo-3D effect - if this is done, the
call to PGENV may have to be changed to allow for the increased X
range that will be needed.

Arguments:
 DATA   (input)  : the data array to be plotted.
 NXV    (input)  : the first dimension of DATA.
 NYV    (input)  : the second dimension of DATA.
 IX1    (input)
 IX2    (input)
 IY1    (input)
 IY2    (input)  : PGHI2D plots a subset of the input array DATA.
                   This subset is delimited in the first (x)
                   dimension by IX1 and IX2 and the 2nd (y) by IY1
                   and IY2, inclusively. Note: IY2 < IY1 is
                   permitted, resulting in a plot with the
                   cross-sections plotted in reverse Y order.
                   However, IX2 must be => IX1.
 X      (input)  : the abscissae of the bins to be plotted. That is,
                   X(1) should be the X value for DATA(IX1,IY1), and
                   X should have (IX2-IX1+1) elements.  The program
                   has to assume that the X value for DATA(x,y) is
                   the same for all y.
 IOFF   (input)  : an offset in array elements applied to successive
                   cross-sections to produce a slanted effect.  A
                   plot with IOFF > 0 slants to the right, one with
                   IOFF < 0 slants left.
 BIAS   (input)  : a bias value applied to each successive cross-
                   section in order to raise it above the previous
                   cross-section.  This is in the same units as the
                   data.
 CENTER (input)  : if .true., the X values denote the center of the
                   bins; if .false. the X values denote the lower
                   edges (in X) of the bins.
 YLIMS  (input)  : workspace.  Should be an array of at least
                   (IX2-IX1+1) elements.
```

PGHIST -- histogram of unbinned data

```
      SUBROUTINE PGHIST(N, DATA, DATMIN, DATMAX, NBIN, PGFLAG)
      INTEGER N
      REAL    DATA(*)
      REAL    DATMIN, DATMAX
      INTEGER NBIN, PGFLAG
```

```
Draw a histogram of N values of a variable in array
```

```
DATA(1...N) in the range DATMIN to DATMAX using NBIN bins.  Note
that array elements which fall exactly on the boundary between
two bins will be counted in the higher bin rather than the
lower one; and array elements whose value is less than DATMIN or
greater than or equal to DATMAX will not be counted at all.

Arguments:
 N      (input)  : the number of data values.
 DATA   (input)  : the data values. Note: the dimension of array
                   DATA must be greater than or equal to N. The
                   first N elements of the array are used.
 DATMIN (input)  : the minimum data value for the histogram.
 DATMAX (input)  : the maximum data value for the histogram.
 NBIN   (input)  : the number of bins to use: the range DATMIN to
                   DATMAX is divided into NBIN equal bins and
                   the number of DATA values in each bin is
                   determined by PGHIST.  NBIN may not exceed 200.
 PGFLAG (input)  : if PGFLAG = 1, the histogram is plotted in the
                   current window and viewport; if PGFLAG = 0,
                   PGENV is called automatically by PGHIST to start
                   a new plot (the x-limits of the window will be
                   DATMIN and DATMAX; the y-limits will be chosen
                   automatically.
                   IF PGFLAG = 2,3 the histogram will be in the same
                   window and viewport but with a filled area style.
                   If pgflag=4,5 as for pgflag = 0,1, but simple
                   line drawn as for PGBIN
```

PGIDEN -- write username, date, and time at bottom of plot

```
      SUBROUTINE PGIDEN

Write username, date, and time at bottom of plot.

Arguments: none.
```

PGIMAG -- color image from a 2D data array

```
      SUBROUTINE PGIMAG (A, IDIM, JDIM, I1, I2, J1, J2,
     1                   A1, A2, TR)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      REAL    A(IDIM,JDIM), A1, A2, TR(6)

Draw a color image of an array in current window. The subsection
of the array A defined by indices (I1:I2, J1:J2) is mapped onto
the view surface world-coordinate system by the transformation
matrix TR. The resulting quadrilateral region is clipped at the edge
of the window. Each element of the array is represented in the image
by a small quadrilateral, which is filled with a color specified by
the corresponding array value.

The subroutine uses color indices in the range C1 to C2, which can
be specified by calling PGSCIR. If PGSCIR is not called before PGIMAG
```

is called, then PGIMAG uses device-dependent values for C1 and C2; these values can be determined by calling PGQCIR. Note that color representations should be assigned to color indices C1 to C2 by calling PGSCR before calling PGIMAG. On some devices (but not all), the color representation can be changed after the call to PGIMAG by calling PGSCR again.

Array values in the range A1 to A2 are mapped on to the range of color indices C1 to C2, with array values <= A1 being given color index C1 and values >= A2 being given color index C2. The mapping function for intermediate array values can be specified by calling routine PGSITF before PGIMAG; the default is linear.

On devices which have no available color indices (C1 > C2), PGIMAG will return without doing anything. On devices with only one color index (C1=C2), all array values map to the same color which is rather uninteresting. An image is always "opaque", i.e., it obscures all graphical elements previously drawn in the region.

The transformation matrix TR is used to calculate the world coordinates of the center of the "cell" that represents each array element. The world coordinates of the center of the cell corresponding to array element A(I,J) are given by:

```
         X = TR(1) + TR(2)*I + TR(3)*J
         Y = TR(4) + TR(5)*I + TR(6)*J
```

Usually TR(3) and TR(5) are zero -- unless the coordinate transformation involves a rotation or shear. The corners of the quadrilateral region that is shaded by PGIMAG are given by applying this transformation to (I1-0.5,J1-0.5), (I2+0.5, J2+0.5).

```
Arguments:
 A      (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 A1     (input)  : the array value which is to appear with shade C1.
 A2     (input)  : the array value which is to appear with shade C2.
 TR     (input)  : transformation matrix between array grid and
                   world coordinates.
```

PGLAB -- write labels for x-axis, y-axis, and top of plot

```
      SUBROUTINE PGLAB (XLBL, YLBL, TOPLBL)
      CHARACTER*(*) XLBL, YLBL, TOPLBL
```

Write labels outside the viewport. This routine is a simple interface to PGMTXT, which should be used if PGLAB is inadequate.

```
Arguments:
 XLBL   (input) : a label for the x-axis (centered below the
                  viewport).
```

```
 YLBL   (input) : a label for the y-axis (centered to the left
                  of the viewport, drawn vertically).
 TOPLBL (input) : a label for the entire plot (centered above the
                  viewport).
```

PGLABEL -- non-standard alias for PGLAB

```
      SUBROUTINE PGLABEL (XLBL, YLBL, TOPLBL)
      CHARACTER*(*) XLBL, YLBL, TOPLBL
```

PGLCUR -- draw a line using the cursor

```
      SUBROUTINE PGLCUR (MAXPT, NPT, X, Y)
      INTEGER MAXPT, NPT
      REAL    X(*), Y(*)
```

```
Interactive routine for user to enter a polyline by use of
the cursor.  Routine allows user to Add and Delete vertices;
vertices are joined by straight-line segments.

Arguments:
 MAXPT  (input)  : maximum number of points that may be accepted.
 NPT    (in/out) : number of points entered; should be zero on
                   first call.
 X      (in/out) : array of x-coordinates (dimension at least MAXPT).
 Y      (in/out) : array of y-coordinates (dimension at least MAXPT).

Notes:

(1) On return from the program, cursor points are returned in
the order they were entered. Routine may be (re-)called with points
already defined in X,Y (# in NPT), and they will be plotted
first, before editing.

(2) User commands: the user types single-character commands
after positioning the cursor: the following are accepted:
  A (Add)    - add point at current cursor location.
  D (Delete) - delete last-entered point.
  X (eXit)   - leave subroutine.
```

PGLDEV -- list available device types

```
      SUBROUTINE PGLDEV
```

```
Writes a list to the terminal of all device types known to the
current version of PGPLOT.

Arguments: none.
```

PGLEN -- find length of a string in a variety of units

```
      SUBROUTINE PGLEN (UNITS, STRING, XL, YL)
      REAL XL, YL
      INTEGER UNITS
      CHARACTER*(*) STRING
```

```
Work out length of a string in x and y directions

Input
 UNITS    :  0 => answer in normalized device coordinates
             1 => answer in inches
             2 => answer in mm
             3 => answer in absolute device coordinates (dots)
             4 => answer in world coordinates
             5 => answer as a fraction of the current viewport size

 STRING   :  String of interest
Output
 XL       :  Length of string in x direction
 YL       :  Length of string in y direction
```

PGLINE -- draw a polyline (curve defined by line-segments)

```
      SUBROUTINE PGLINE (N, XPTS, YPTS)
      INTEGER  N
      REAL     XPTS(*), YPTS(*)
```

```
Primitive routine to draw a Polyline. A polyline is one or more
connected straight-line segments.  The polyline is drawn using
the current setting of attributes color-index, line-style, and
line-width. The polyline is clipped at the edge of the window.

Arguments:
 N      (input)  : number of points defining the line; the line
                   consists of (N-1) straight-line segments.
                   N should be greater than 1 (if it is 1 or less,
                   nothing will be drawn).
 XPTS   (input)  : world x-coordinates of the points.
 YPTS   (input)  : world y-coordinates of the points.

The dimension of arrays X and Y must be greater than or equal to N.
The "pen position" is changed to (X(N),Y(N)) in world coordinates
(if N > 1).
```

PGMOVE -- move pen (change current pen position)

```
      SUBROUTINE PGMOVE (X, Y)
      REAL X, Y
```

Primitive routine to move the "pen" to the point with world coordinates (X,Y). No line is drawn.

```
Arguments:
 X      (input)  : world x-coordinate of the new pen position.
 Y      (input)  : world y-coordinate of the new pen position.
```

PGMTEXT -- non-standard alias for PGMTXT

```
      SUBROUTINE PGMTEXT (SIDE, DISP, COORD, FJUST, TEXT)
      CHARACTER*(*) SIDE, TEXT
      REAL DISP, COORD, FJUST
```

PGMTXT -- write text at position relative to viewport

```
      SUBROUTINE PGMTXT (SIDE, DISP, COORD, FJUST, TEXT)
      CHARACTER*(*) SIDE, TEXT
      REAL DISP, COORD, FJUST
```

Write text at a position specified relative to the viewport (outside or inside). This routine is useful for annotating graphs. It is used by routine PGLAB. The text is written using the current values of attributes color-index, line-width, character-height, and character-font.

```
Arguments:
 SIDE   (input)  : must include one of the characters 'B', 'L', 'T',
                   or 'R' signifying the Bottom, Left, Top, or Right
                   margin of the viewport. If it includes 'LV' or
                   'RV', the string is written perpendicular to the
                   frame rather than parallel to it.
 DISP   (input)  : the displacement of the character string from the
                   specified edge of the viewport, measured outwards
                   from the viewport in units of the character
                   height. Use a negative value to write inside the
                   viewport, a positive value to write outside.
 COORD  (input)  : the location of the character string along the
                   specified edge of the viewport, as a fraction of
                   the length of the edge.
 FJUST  (input)  : controls justification of the string parallel to
                   the specified edge of the viewport. If
                   FJUST = 0.0, the left-hand end of the string will
                   be placed at COORD; if JUST = 0.5, the center of
                   the string will be placed at COORD; if JUST = 1.0,
                   the right-hand end of the string will be placed at
                   at COORD. Other values between 0 and 1 give inter-
                   mediate placing, but they are not very useful.
 TEXT   (input) :  the text string to be plotted. Trailing spaces are
                   ignored when justifying the string, but leading
                   spaces are significant.
```

PGNCUR -- mark a set of points using the cursor

```
      SUBROUTINE PGNCUR (MAXPT, NPT, X, Y, SYMBOL)
      INTEGER MAXPT, NPT
      REAL    X(*), Y(*)
      INTEGER SYMBOL
```

```
Interactive routine for user to enter data points by use of
the cursor.  Routine allows user to Add and Delete points.  The
points are returned in order of increasing x-coordinate, not in the
order they were entered.

Arguments:
 MAXPT  (input)  : maximum number of points that may be accepted.
 NPT    (in/out) : number of points entered; should be zero on
                   first call.
 X      (in/out) : array of x-coordinates.
 Y      (in/out) : array of y-coordinates.
 SYMBOL (input)  : code number of symbol to use for marking
                   entered points (see PGPT).

Note (1): The dimension of arrays X and Y must be greater than or
equal to MAXPT.

Note (2): On return from the program, cursor points are returned in
increasing order of X. Routine may be (re-)called with points
already defined in X,Y (number in NPT), and they will be plotted
first, before editing.

Note (3): User commands: the user types single-character commands
after positioning the cursor: the following are accepted:
A (Add)    - add point at current cursor location.
D (Delete) - delete nearest point to cursor.
X (eXit)   - leave subroutine.
```

PGNCURSE -- non-standard alias for PGNCUR

```
      SUBROUTINE PGNCURSE (MAXPT, NPT, X, Y, SYMBOL)
      INTEGER MAXPT, NPT
      REAL    X(*), Y(*)
      INTEGER SYMBOL
```

PGNUMB -- convert a number into a plottable character string

```
      SUBROUTINE PGNUMB (MM, PP, FORM, STRING, NC)
      INTEGER MM, PP, FORM
      CHARACTER*(*) STRING
      INTEGER NC
```

```
This routine converts a number into a decimal character
representation. To avoid problems of floating-point roundoff, the
number must be provided as an integer (MM) multiplied by a power of 10
```

```
(10**PP).  The output string retains only significant digits of MM,
and will be in either integer format (123), decimal format (0.0123),
or exponential format (1.23x10**5). Standard escape sequences \u, \d
raise the exponent and \x is used for the multiplication sign.
This routine is used by PGBOX to create numeric labels for a plot.

Formatting rules:
  (a) Decimal notation (FORM=1):
      - Trailing zeros to the right of the decimal sign are
        omitted
      - The decimal sign is omitted if there are no digits
        to the right of it
      - When the decimal sign is placed before the first digit
        of the number, a zero is placed before the decimal sign
      - The decimal sign is a period (.)
      - No spaces are placed between digits (ie digits are not
        grouped in threes as they should be)
      - A leading minus (-) is added if the number is negative
  (b) Exponential notation (FORM=2):
      - The exponent is adjusted to put just one (non-zero)
        digit before the decimal sign
      - The mantissa is formatted as in (a), unless its value is
        1 in which case it and the multiplication sign are omitted
      - If the power of 10 is not zero and the mantissa is not
        zero, an exponent of the form \x10\u[-]nnn is appended,
        where \x is a multiplication sign (cross), \u is an escape
        sequence to raise the exponent, and as many digits nnn
        are used as needed
  (c) Automatic choice (FORM=0):
        Decimal notation is used if the absolute value of the
        number is less than 10000 or greater than or equal to
        0.01. Otherwise exponential notation is used.

Arguments:
 MM     (input)
 PP     (input)  : the value to be formatted is MM*10**PP.
 FORM   (input)  : controls how the number is formatted:
                   FORM = 0 -- use either decimal or exponential
                   FORM = 1 -- use decimal notation
                   FORM = 2 -- use exponential notation
 STRING (output) : the formatted character string, left justified.
                   If the length of STRING is insufficient, a single
                   asterisk is returned, and NC=1.
 NC     (output) : the number of characters used in STRING:
                   the string to be printed is STRING(1:NC).
```

PGOLIN -- mark a set of points using the cursor

```
      SUBROUTINE PGOLIN (MAXPT, NPT, X, Y, SYMBOL)
      INTEGER MAXPT, NPT
      REAL    X(*), Y(*)
      INTEGER SYMBOL

Interactive routine for user to enter data points by use of
the cursor.  Routine allows user to Add and Delete points.  The
points are returned in the order that they were entered (unlike
PGNCUR).
```

```
Arguments:
 MAXPT  (input)  : maximum number of points that may be accepted.
 NPT    (in/out) : number of points entered; should be zero on
                   first call.
 X      (in/out) : array of x-coordinates.
 Y      (in/out) : array of y-coordinates.
 SYMBOL (input)  : code number of symbol to use for marking
                   entered points (see PGPT).

Note (1): The dimension of arrays X and Y must be greater than or
equal to MAXPT.

Note (2): On return from the program, cursor points are returned in
the order they were entered. Routine may be (re-)called with points
already defined in X,Y (number in NPT), and they will be plotted
first, before editing.

Note (3): User commands: the user types single-character commands
after positioning the cursor: the following are accepted:
A (Add)    - add point at current cursor location.
D (Delete) - delete the last point entered.
X (eXit)   - leave subroutine.
```

PGPAGE -- advance to new page

```
      SUBROUTINE PGPAGE

Advance plotter to a new page or panel, clearing the screen if
necessary. If the "prompt state" is ON (see PGASK), confirmation is
requested from the user before clearing the screen. If the view
surface has been subdivided into panels with PGBEG or PGSUBP, then
PGPAGE advances to the next panel, and if the current panel is the
last on the page, PGPAGE clears the screen or starts a new sheet of
paper.  PGPAGE does not change the PGPLOT window or the viewport
(in normalized device coordinates); but note that if the size of the
view-surface is changed externally (e.g., by a workstation window
manager) the size of the viewport is chnaged in proportion.

Arguments: none
```

PGPANL -- switch to a different panel on the view surface

```
      SUBROUTINE PGPANL(IX, IY)
      INTEGER IX, IY

Start plotting in a different panel. If the view surface has been
divided into panels by PGBEG or PGSUBP, this routine can be used to
move to a different panel. Note that PGPLOT does not remember what
viewport and window were in use in each panel; these should be reset
if necessary after calling PGPANL. Nor does PGPLOT clear the panel:
call PGERAS after calling PGPANL to do this.

Arguments:
 IX     (input)  : the horizontal index of the panel (in the range
```

```
                    1 <= IX <= number of panels in horizontal
                    direction).
 IY     (input)  : the vertical index of the panel (in the range
                    1 <= IY <= number of panels in horizontal
                    direction).
```

PGPAP -- change the size of the view surface

```
      SUBROUTINE PGPAP (WIDTH, ASPECT)
      REAL WIDTH, ASPECT
```

This routine changes the size of the view surface ("paper size") to a specified width and aspect ratio (height/width), in so far as this is possible on the specific device. It is always possible to obtain a view surface smaller than the default size; on some devices (e.g., printers that print on roll or fan-feed paper) it is possible to obtain a view surface larger than the default.

This routine should be called either immediately after PGBEG or immediately before PGPAGE. The new size applies to all subsequent images until the next call to PGPAP.

```
Arguments:
 WIDTH  (input)  : the requested width of the view surface in inches;
                   if WIDTH=0.0, PGPAP will obtain the largest view
                   surface available consistent with argument ASPECT.
                   (1 inch = 25.4 mm.)
 ASPECT (input)  : the aspect ratio (height/width) of the view
                   surface; e.g., ASPECT=1.0 gives a square view
                   surface, ASPECT=0.618 gives a horizontal
                   rectangle, ASPECT=1.618 gives a vertical rectangle.
```

PGPAPER -- non-standard alias for PGPAP

```
      SUBROUTINE PGPAPER (WIDTH, ASPECT)
      REAL WIDTH, ASPECT
```

PGPIXL -- draw pixels

```
      SUBROUTINE PGPIXL (IA, IDIM, JDIM, I1, I2, J1, J2,
     1                   X1, X2, Y1, Y2)
      INTEGER IDIM, JDIM, I1, I2, J1, J2
      INTEGER IA(IDIM,JDIM)
      REAL    X1, X2, Y1, Y2
```

Draw lots of solid-filled (tiny) rectangles aligned with the coordinate axes. Best performance is achieved when output is directed to a pixel-oriented device and the rectangles coincide with the pixels on the device. In other cases, pixel output is emulated.

The subsection of the array IA defined by indices (I1:I2, J1:J2)
is mapped onto world-coordinate rectangle defined by X1, X2, Y1
and Y2. This rectangle is divided into (I2 - I1 + 1) * (J2 - J1 + 1)
small rectangles. Each of these small rectangles is solid-filled
with the color index specified by the corresponding element of
IA.

On most devices, the output region is "opaque", i.e., it obscures
all graphical elements previously drawn in the region. But on
devices that do not have erase capability, the background shade
is "transparent" and allows previously-drawn graphics to show
through.

```
Arguments:
 IA     (input)  : the array to be plotted.
 IDIM   (input)  : the first dimension of array A.
 JDIM   (input)  : the second dimension of array A.
 I1, I2 (input)  : the inclusive range of the first index
                   (I) to be plotted.
 J1, J2 (input)  : the inclusive range of the second
                   index (J) to be plotted.
 X1, Y1 (input)  : world coordinates of one corner of the output
                   region
 X2, Y2 (input)  : world coordinates of the opposite corner of the
                   output region
```

PGPNTS -- draw one or more graph markers, not all the same

```
      SUBROUTINE PGPNTS (N, X, Y, SYMBOL, NS)
      INTEGER N, NS
      REAL X(*), Y(*)
      INTEGER SYMBOL(*)
```

Draw Graph Markers. Unlike PGPT, this routine can draw a different
symbol at each point. The markers
are drawn using the current values of attributes color-index,
line-width, and character-height (character-font applies if the symbol
number is >31). If the point to be marked lies outside the window,
no marker is drawn. The "pen position" is changed to
(XPTS(N),YPTS(N)) in world coordinates (if N > 0).

```
Arguments:
 N      (input)  : number of points to mark.
 X      (input)  : world x-coordinate of the points.
 Y      (input)  : world y-coordinate of the points.
 SYMBOL (input)  : code number of the symbol to be plotted at each
                   point (see PGPT).
 NS     (input)  : number of values in the SYMBOL array.  If NS <= N,
                   then the first NS points are drawn using the value
                   of SYMBOL(I) at (X(I), Y(I)) and SYMBOL(1) for all
                   the values of (X(I), Y(I)) where I > NS.
```

Note: the dimension of arrays X and Y must be greater than or equal
to N and the dimension of the array SYMBOL must be greater than or
equal to NS. If N is 1, X and Y may be scalars (constants or
variables). If NS is 1, then SYMBOL may be a scalar. If N is
less than 1, nothing is drawn.

PGPOINT -- non-standard alias for PGPT

```
      SUBROUTINE PGPOINT (N, XPTS, YPTS, SYMBOL)
      INTEGER N
      REAL XPTS(*), YPTS(*)
      INTEGER SYMBOL
```

PGPOLY -- fill a polygonal area with shading

```
      SUBROUTINE PGPOLY (N, XPTS, YPTS)
      INTEGER N
      REAL XPTS(*), YPTS(*)
```

```
Fill-area primitive routine: shade the interior of a closed
polygon in the current window.  The action of this routine depends
on the setting of the Fill-Area Style attribute (see PGSFS).
The polygon is clipped at the edge of the
window. The pen position is changed to (XPTS(1),YPTS(1)) in world
coordinates (if N > 1).  If the polygon is not convex, a point is
assumed to lie inside the polygon if a straight line drawn to
infinity intersects and odd number of the polygon's edges.

Arguments:
 N       (input)  : number of points defining the polygon; the
                    line consists of N straight-line segments,
                    joining points 1 to 2, 2 to 3,... N-1 to N, N to 1.
                    N should be greater than 2 (if it is 2 or less,
                    nothing will be drawn).
 XPTS    (input)  : world x-coordinates of the vertices.
 YPTS    (input)  : world y-coordinates of the vertices.
                    Note: the dimension of arrays XPTS and YPTS must be
                    greater than or equal to N.
```

PGPT -- draw one or more graph markers

```
      SUBROUTINE PGPT (N, XPTS, YPTS, SYMBOL)
      INTEGER N
      REAL XPTS(*), YPTS(*)
      INTEGER SYMBOL
```

```
Primitive routine to draw Graph Markers (polymarker). The markers
are drawn using the current values of attributes color-index,
line-width, and character-height (character-font applies if the symbol
number is >31).  If the point to be marked lies outside the window,
no marker is drawn.  The "pen position" is changed to
(XPTS(N),YPTS(N)) in world coordinates (if N > 0).

Arguments:
 N       (input)  : number of points to mark.
```

```
 XPTS   (input)  : world x-coordinates of the points.
 YPTS   (input)  : world y-coordinates of the points.
 SYMBOL (input)  : code number of the symbol to be drawn at each
                   point:
                   -1, -2  : a single dot (diameter = current
                             line width).
                   -3..-31 : a regular polygon with ABS(SYMBOL)
                             edges (style set by current fill style).
                   0..31   : standard marker symbols.
                   32..127 : ASCII characters (in current font).
                             e.g. to use letter F as a marker, let
                             SYMBOL = ICHAR('F').
                   > 127  :  a Hershey symbol number.

Note: the dimension of arrays X and Y must be greater than or equal
to N. If N is 1, X and Y may be scalars (constants or variables). If
N is less than 1, nothing is drawn.
```

PGPTEXT -- non-standard alias for PGPTXT

```
      SUBROUTINE PGPTEXT (X, Y, ANGLE, FJUST, TEXT)
      REAL X, Y, ANGLE, FJUST
      CHARACTER*(*) TEXT
```

PGPTXT -- write text at arbitrary position and angle

```
      SUBROUTINE PGPTXT (X, Y, ANGLE, FJUST, TEXT)
      REAL X, Y, ANGLE, FJUST
      CHARACTER*(*) TEXT

Primitive routine for drawing text. The text may be drawn at any
angle with the horizontal, and may be centered or left- or right-
justified at a specified position.  Routine PGTEXT provides a
simple interface to PGPTXT for horizontal strings. Text is drawn
using the current values of attributes color-index, line-width,
character-height, and character-font.  Text is NOT subject to
clipping at the edge of the window.

Arguments:
 X      (input)  : world x-coordinate.
 Y      (input)  : world y-coordinate. The string is drawn with the
                   baseline of all the characters passing through
                   point (X,Y); the positioning of the string along
                   this line is controlled by argument FJUST.
 ANGLE  (input)  : angle, in degrees, that the baseline is to make
                   with the horizontal, increasing counter-clockwise
                   (0.0 is horizontal).
 FJUST  (input)  : controls horizontal justification of the string.
                   If FJUST = 0.0, the string will be left-justified
                   at the point (X,Y); if FJUST = 0.5, it will be
                   centered, and if FJUST = 1.0, it will be right
                   justified. [Other values of FJUST give other
                   justifications.]
 TEXT   (input)  : the character string to be plotted.
```

PGQAH -- inquire arrow-head style

```
      SUBROUTINE PGQAH (FS, ANGLE, VENT)
      INTEGER  FS
      REAL ANGLE, VENT
```

```
Query the style to be used for arrowheads drawn with routine PGARRO.

Argument:
 FS     (output) : FS = 1 => filled; FS = 2 => outline.
 ANGLE  (output) : the acute angle of the arrow point, in degrees.
 VENT   (output) : the fraction of the triangular arrow-head that
                   is cut away from the back.
```

PGQCF -- inquire character font

```
      SUBROUTINE PGQCF (FONT)
      INTEGER  FONT
```

```
Query the current Character Font (set by routine PGSCF).

Argument:
 FONT   (output)   : the current font number (in range 1-4).
```

PGQCH -- inquire character height

```
      SUBROUTINE PGQCH (SIZE)
      REAL SIZE
```

```
Query the Character Size attribute (set by routine PGSCH).

Argument:
 SIZE   (output) : current character size (dimensionless multiple of
                   the default size).
```

PGQCI -- inquire color index

```
      SUBROUTINE PGQCI (CI)
      INTEGER  CI
```

```
Query the Color Index attribute (set by routine PGSCI).

Argument:
 CI     (output) : the current color index (in range 0-max). This is
                   the color index actually in use, and may differ
```

```
                          from the color index last requested by PGSCI if
                          that index is not available on the output device.
```

PGQCIR -- inquire color index range

```
      SUBROUTINE PGQCIR(ICILO, ICIHI)
      INTEGER   ICILO, ICIHI

Query the color index range to be used for producing images with
PGGRAY or PGIMAG, as set by routine PGSCIR or by device default.

Arguments:
 ICILO  (output) : the lowest color index to use for images
 ICIHI  (output) : the highest color index to use for images
```

PGQCOL -- inquire color capability

```
      SUBROUTINE PGQCOL (CI1, CI2)
      INTEGER  CI1, CI2

Query the range of color indices available on the current device.

Argument:
 CI1    (output) : the minimum available color index. This will be
                   either 0 if the device can write in the
                   background color, or 1 if not.
 CI2    (output) : the maximum available color index. This will be
                   1 if the device has no color capability, or a
                   larger number (e.g., 3, 7, 15, 255).
```

PGQCR -- inquire color representation

```
      SUBROUTINE PGQCR (CI, CR, CG, CB)
      INTEGER CI
      REAL    CR, CG, CB

Query the RGB colors associated with a color index.

Arguments:
 CI  (input)  : color index
 CR  (output) : red, green and blue intensities
 CG  (output)   in the range 0.0 to 1.0
 CB  (output)
```

PGQCS -- inquire character height in a variety of units

```
      SUBROUTINE PGQCS(UNITS, XCH, YCH)
      INTEGER UNITS
      REAL XCH, YCH
```

```
Return the current PGPLOT character height in a variety of units.
This routine provides facilities that are not available via PGQCH.
Use PGQCS if the character height is required in units other than
those used in PGSCH.

The PGPLOT "character height" is a dimension that scales with the
size of the view surface and with the scale-factor specified with
routine PGSCH. The default value is 1/40th of the height or width
of the view surface (whichever is less); this value is then
multiplied by the scale-factor supplied with PGSCH. Note that it
is a nominal height only; the actual character size depends on the
font and is usually somewhat smaller.

Arguments:
 UNITS  (input)  : Used to specify the units of the output value:
                   UNITS = 0 : normalized device coordinates
                   UNITS = 1 : inches
                   UNITS = 2 : millimeters
                   UNITS = 3 : pixels
                   Other values give an error message, and are
                   treated as 0.
 XCH    (output) : The character height for vertically written text.
 YCH    (output) : The character height for horizontally written text.
                   N.B. XCH=YCH if UNITS=1 or UNITS=2.

The character height is returned in both XCH and YCH.

If UNITS=1 or UNITS=2, XCH and YCH both receive the same value.

If UNITS=3, XCH receives the height in horizontal pixel units, and YCH
receives the height in vertical pixel units; on devices for which the
pixels are not square, XCH and YCH will be different.

If UNITS=0, XCH receives the character height as a fraction of the
horizontal dimension of the view surface, and YCH receives the
character height as a fraction of the vertical dimension of the view
surface.
```

PGQFS -- inquire fill-area style

```
      SUBROUTINE PGQFS (FS)
      INTEGER  FS
```

```
Query the current Fill-Area Style attribute (set by routine
PGSFS).

Argument:
 FS     (output) : the current fill-area style:
                     FS = 1 => solid (default)
                     FS = 2 => outline
                     FS = 3 => hatched
                     FS = 4 => cross-hatched
```

PGQHS -- inquire hatching style

```
      SUBROUTINE PGQHS (ANGLE, SEPN, PHASE)
      REAL ANGLE, SEPN, PHASE
```

```
Query the style to be used hatching (fill area with fill-style 3).

Arguments:
 ANGLE  (input)  : the angle the hatch lines make with the
                   horizontal, in degrees, increasing
                   counterclockwise (this is an angle on the
                   view surface, not in world-coordinate space).
 SEPN   (input)  : the spacing of the hatch lines. The unit spacing
                   is 1 percent of the smaller of the height or
                   width of the view surface.
 PHASE  (input)  : a real number between 0 and 1; the hatch lines
                   are displaced by this fraction of SEPN from a
                   fixed reference.  Adjacent regions hatched with the
                   same PHASE have contiguous hatch lines.
```

PGQINF -- inquire PGPLOT general information

```
      SUBROUTINE PGQINF (ITEM, VALUE, LENGTH)
      CHARACTER*(*) ITEM, VALUE
      INTEGER LENGTH
```

```
This routine can be used to obtain miscellaneous information about
the PGPLOT environment. Input is a character string defining the
information required, and output is a character string containing the
requested information.

The following item codes are accepted (note that the strings must
match exactly, except for case, but only the first 8 characters are
significant). For items marked *, PGPLOT must be in the OPEN state
for the inquiry to succeed. If the inquiry is unsuccessful, either
because the item code is not recognized or because the information
is not available, a question mark ('?') is returned.

  'VERSION'     - version of PGPLOT software in use.
  'STATE'       - status of PGPLOT ('OPEN' if a graphics device
                  is open for output, 'CLOSED' otherwise).
  'USER'        - the username associated with the calling program.
  'NOW'         - current date and time (e.g., '17-FEB-1986 10:04').
  'DEVICE'    * - current PGPLOT device or file.
  'FILE'      * - current PGPLOT device or file.
  'TYPE'      * - device-type of the current PGPLOT device.
  'DEV/TYPE'  * - current PGPLOT device and type, in a form which
                  is acceptable as an argument for PGBEG.
  'HARDCOPY'  * - is the current device a hardcopy device? ('YES' or
                  'NO').
  'TERMINAL'  * - is the current device the user's interactive
                  terminal? ('YES' or 'NO').
  'CURSOR'    * - does the current device have a graphics cursor?
                  ('YES' or 'NO').
```

```
Arguments:
 ITEM  (input)  : character string defining the information to
                  be returned; see above for a list of possible
                  values.
 VALUE (output) : returns a character-string containing the
                  requested information, truncated to the length
                  of the supplied string or padded on the right with
                  spaces if necessary.
 LENGTH (output): the number of characters returned in VALUE
                  (excluding trailing blanks).
```

PGQITF -- inquire image transfer function

```
      SUBROUTINE PGQITF (ITF)
      INTEGER  ITF
```

```
Return the Image Transfer Function as set by default or by a previous
call to PGSITF. The Image Transfer Function is used by routines
PGIMAG, PGGRAY, and PGWEDG.

Argument:
 ITF    (output) : type of transfer function (see PGSITF)
```

PGQLS -- inquire line style

```
      SUBROUTINE PGQLS (LS)
      INTEGER  LS
```

```
Query the current Line Style attribute (set by routine PGSLS).

Argument:
 LS     (output) : the current line-style attribute (in range 1-5).
```

PGQLW -- inquire line width

```
      SUBROUTINE PGQLW (LW)
      INTEGER  LW
```

```
Query the current Line-Width attribute (set by routine PGSLW).

Argument:
 LW     (output)  : the line-width (in range 1-201).
```

PGQPOS -- inquire current pen position

```
      SUBROUTINE PGQPOS (X, Y)
```

```
      REAL X, Y
```

```
Query the current "pen" position in world C coordinates (X,Y).

Arguments:
 X      (output)  : world x-coordinate of the pen position.
 Y      (output)  : world y-coordinate of the pen position.
```

PGQTBG -- inquire text background color index

```
      SUBROUTINE PGQTBG (TBCI)
      INTEGER  TBCI
```

```
Query the current Text Background Color Index (set by routine
PGSTBG).

Argument:
 TBCI   (output) : receives the current text background color index.
```

PGQTXT -- find bounding box of text string

```
      SUBROUTINE PGQTXT (X, Y, ANGLE, FJUST, TEXT, XBOX, YBOX)
      REAL X, Y, ANGLE, FJUST
      CHARACTER*(*) TEXT
      REAL XBOX(4), YBOX(4)
```

```
This routine returns a bounding box for a text string. Instead
of drawing the string as routine PGPTXT does, it returns in XBOX
and YBOX the coordinates of the corners of a rectangle parallel
to the string baseline that just encloses the string.

If the string is blank or contains no drwable characters, all
four elements of XBOX and YBOX are assigned the starting point
of the string, (X,Y).

Arguments:
 X, Y, ANGLE, FJUST, TEXT (input) : these arguments are the same as
                  the corrresponding arguments in PGPTXT.
 XBOX, YBOX (output) : arrays of dimension 4; on output, they
                  contain the world coordinates of the bounding
                  box in (XBOX(1), YBOX(1)), ..., (XBOX(4), YBOX(4)).
```

PGQVP -- inquire viewport size and position

```
      SUBROUTINE PGQVP (UNITS, X1, X2, Y1, Y2)
      INTEGER UNITS
      REAL    X1, X2, Y1, Y2
```

```
Inquiry routine to determine the current viewport setting.
The values returned may be normalized device coordinates, inches, mm,
```

```
or pixels, depending on the value of the input parameter CFLAG.

Arguments:
 UNITS  (input)  : used to specify the units of the output parameters:
                   UNITS = 0 : normalized device coordinates
                   UNITS = 1 : inches
                   UNITS = 2 : millimeters
                   UNITS = 3 : pixels
                   Other values give an error message, and are
                   treated as 0.
 X1     (output) : the x-coordinate of the bottom left corner of the
                   viewport.
 X2     (output) : the x-coordinate of the top right corner of the
                   viewport.
 Y1     (output) : the y-coordinate of the bottom left corner of the
                   viewport.
 Y2     (output) : the y-coordinate of the top right corner of the
                   viewport.
```

PGQVSZ -- find the window defined by the full view surface

```
      SUBROUTINE PGQVSZ (UNITS, X1, X2, Y1, Y2)
      INTEGER UNITS
      REAL X1, X2, Y1, Y2

Return the window, in a variety of units, defined by the full
device view surface (0 -> 1 in normalized device coordinates).

Input:
  UNITS    0,1,2,3 for output in normalized device coords,
           inches, mm, or absolute device units (dots)
Output
  X1,X2    X window
  Y1,Y2    Y window
```

PGQWIN -- inquire window boundary coordinates

```
      SUBROUTINE PGQWIN (X1, X2, Y1, Y2)
      REAL X1, X2, Y1, Y2

Inquiry routine to determine the current window setting.
The values returned are world coordinates.

Arguments:
 X1     (output) : the x-coordinate of the bottom left corner
                   of the window.
 X2     (output) : the x-coordinate of the top right corner
                   of the window.
 Y1     (output) : the y-coordinate of the bottom left corner
                   of the window.
 Y2     (output) : the y-coordinate of the top right corner
                   of the window.
```

PGRECT -- draw a rectangle, using fill-area attributes

```
      SUBROUTINE PGRECT (X1, X2, Y1, Y2)
      REAL X1, X2, Y1, Y2
```

```
This routine can be used instead of PGPOLY for the special case of
drawing a rectangle aligned with the coordinate axes; only two
vertices need be specified instead of four.  On most devices, it is
faster to use PGRECT than PGPOLY for drawing rectangles.  The
rectangle has vertices at (X1,Y1), (X1,Y2), (X2,Y2), and (X2,Y1).

Arguments:
 X1, X2 (input) : the horizontal range of the rectangle.
 Y1, Y2 (input) : the vertical range of the rectangle.
```

PGRND -- find the smallest 'round' number greater than x

```
      REAL FUNCTION PGRND (X, NSUB)
      REAL X
      INTEGER NSUB
```

```
Routine to find the smallest "round" number larger than x, a
"round" number being 1, 2 or 5 times a power of 10. If X is negative,
PGRND(X) = -PGRND(ABS(X)). eg PGRND(8.7) = 10.0,
PGRND(-0.4) = -0.5.  If X is zero, the value returned is zero.
This routine is used by PGBOX for choosing  tick intervals.

Returns:
 PGRND         : the "round" number.
Arguments:
 X      (input)  : the number to be rounded.
 NSUB   (output) : a suitable number of subdivisions for
                   subdividing the "nice" number: 2 or 5.
```

PGRNGE -- choose axis limits

```
      SUBROUTINE PGRNGE (X1, X2, XLO, XHI)
      REAL X1, X2, XLO, XHI
```

```
Choose plotting limits XLO and XHI which encompass the data
range X1 to X2.

Arguments:
 X1, X2 (input)   : the data range (X1<X2), ie, the min and max values
                   to be plotted.
 XLO, XHI (output) : suitable values to use as the extremes of a graph
                   axis (XLO <= X1, XHI >= X2).
```

PGSAH -- set arrow-head style

```
      SUBROUTINE PGSAH (FS, ANGLE, VENT)
      INTEGER  FS
      REAL ANGLE, VENT
```

Set the style to be used for arrowheads drawn with routine PGARRO.

```
Argument:
 FS     (input)  : FS = 1 => filled; FS = 2 => outline.
                   Other values are treated as 2. Default 1.
 ANGLE  (input)  : the acute angle of the arrow point, in degrees;
                   angles in the range 20.0 to 90.0 give reasonable
                   results. Default 45.0.
 VENT   (input)  : the fraction of the triangular arrow-head that
                   is cut away from the back. 0.0 gives a triangular
                   wedge arrow-head; 1.0 gives an open >. Values 0.3
                   to 0.7 give reasonable results. Default 0.3.
```

PGSAVE -- save PGPLOT attributes

```
      SUBROUTINE PGSAVE
```

This routine saves the current PGPLOT attributes in a private storage area. They can be restored by calling PGUNSA (unsave). Attributes saved are: character font, character height, color index, fill-area style, line style, line width, pen position, arrow-head style, hatching style. Color representation is not saved.

Calls to PGSAVE and PGUNSA should always be paired. Up to 20 copies of the attributes may be saved. PGUNSA always retrieves the last-saved values (first-in first-out stack).

Arguments: none

PGUNSA -- restore PGPLOT attributes

```
      ENTRY PGUNSA
```

This routine restores the PGPLOT attributes saved in the last call to PGSAVE. See PGSAVE.

Arguments: none

PGSCF -- set character font

```
      SUBROUTINE PGSCF (FONT)
      INTEGER  FONT
```

Set the Character Font for subsequent text plotting. Four different
fonts are available:
  1: (default) a simple single-stroke font ("normal" font)
  2: roman font
  3: italic font
  4: script font
This call determines which font is in effect at the beginning of
each text string. The font can be changed (temporarily) within a text
string by using the escape sequences \fn, \fr, \fi, and \fs for fonts
1, 2, 3, and 4, respectively.

```
Argument:
 FONT   (input)  : the font number to be used for subsequent text
                   plotting (in range 1-4).
```

PGSCH -- set character height

```
      SUBROUTINE PGSCH (SIZE)
      REAL SIZE
```

Set the character size attribute. The size affects all text and graph
markers drawn later in the program. The default character size is
1.0, corresponding to a character height about 1/40 the height of
the view surface. Changing the character size also scales the length
of tick marks drawn by PGBOX and terminals drawn by PGERRX and PGERRY.

```
Argument:
 SIZE   (input)  : new character size (dimensionless multiple of
                   the default size).
```

PGSCI -- set color index

```
      SUBROUTINE PGSCI (CI)
      INTEGER  CI
```

Set the Color Index for subsequent plotting, if the output device
permits this. The default color index is 1, usually white on a black
background for video displays or black on a white background for
printer plots. The color index is an integer in the range 0 to a
device-dependent maximum. Color index 0 corresponds to the background
color; lines may be "erased" by overwriting them with color index 0
(if the device permits this).

If the requested color index is not available on the selected device,
color index 1 will be substituted.

The assignment of colors to color indices can be changed with
subroutine PGSCR (set color representation). Color indices 0-15
have predefined color representations (see the PGPLOT manual), but
these may be changed with PGSCR. Color indices above 15 have no
predefined representations: if these indices are used, PGSCR must
be called to define the representation.

Argument:

```
 CI     (input)  : the color index to be used for subsequent plotting
                   on the current device (in range 0-max). If the
                   index exceeds the device-dependent maximum, the
                   default color index (1) is used.
```

PGSCIR -- set color index range

```
      SUBROUTINE PGSCIR(ICILO, ICIHI)
      INTEGER   ICILO, ICIHI
```

```
Set the color index range to be used for producing images with
PGGRAY or PGIMAG. If the range is not all within the range supported
by the device, a smaller range will be used. The number of
different colors available for images is ICIHI-ICILO+1.

Arguments:
 ICILO  (input)  : the lowest color index to use for images
 ICIHI  (input)  : the highest color index to use for images
```

PGSCR -- set color representation

```
      SUBROUTINE PGSCR (CI, CR, CG, CB)
      INTEGER CI
      REAL    CR, CG, CB
```

```
Set color representation: i.e., define the color to be
associated with a color index.  Ignored for devices which do not
support variable color or intensity.  Color indices 0-15
have predefined color representations (see the PGPLOT manual), but
these may be changed with PGSCR.  Color indices 16-maximum have no
predefined representations: if these indices are used, PGSCR must
be called to define the representation. On monochrome output
devices (e.g. VT125 terminals with monochrome monitors), the
monochrome intensity is computed from the specified Red, Green, Blue
intensities as 0.30*R + 0.59*G + 0.11*B, as in US color television
systems, NTSC encoding.  Note that most devices do not have an
infinite range of colors or monochrome intensities available;
the nearest available color is used.  Examples: for black,
set CR=CG=CB=0.0; for white, set CR=CG=CB=1.0; for medium gray,
set CR=CG=CB=0.5; for medium yellow, set CR=CG=0.5, CB=0.0.

Argument:
 CI     (input)  : the color index to be defined, in the range 0-max.
                   If the color index greater than the device
                   maximum is specified, the call is ignored. Color
                   index 0 applies to the background color.
 CR     (input)  : red, green, and blue intensities,
 CG     (input)    in range 0.0 to 1.0.
 CB     (input)
```

PGSCRN -- set color representation by name

```
      SUBROUTINE PGSCRN(CI, NAME, IER)
      INTEGER CI
      CHARACTER*(*) NAME
      INTEGER IER
```

```
Set color representation: i.e., define the color to be
associated with a color index.  Ignored for devices which do not
support variable color or intensity.  This is an alternative to
routine PGSCR. The color representation is defined by name instead
of (R,G,B) components.

Color names are defined in an external file which is read the first
time that PGSCRN is called. The name of the external file is
found as follows:
1. if environment variable (logical name) PGPLOT_RGB is defined,
   its value is used as the file name;
2. otherwise, if environment variable PGPLOT_DIR is defined, a
   file "rgb.txt" in the directory named by this environment
   variable is used;
3. otherwise, file "rgb.txt" in the current directory is used.
If all of these fail to find a file, an error is reported and
the routine does nothing.

Each line of the file
defines one color, with four blank- or tab-separated fields per
line. The first three fields are the R, G, B components, which
are integers in the range 0 (zero intensity) to 255 (maximum
intensity). The fourth field is the color name. The color name
may include embedded blanks. Example:

255   0   0 red
255 105 180 hot pink
255 255 255 white
  0   0   0 black

Arguments:
 CI     (input)  : the color index to be defined, in the range 0-max.
                   If the color index greater than the device
                   maximum is specified, the call is ignored. Color
                   index 0 applies to the background color.
 NAME   (input)  : the name of the color to be associated with
                   this color index. This name must be in the
                   external file. The names are not case-sensitive.
                   If the color is not listed in the file, the
                   color representation is not changed.
 IER    (output) : returns 0 if the routine was successful, 1
                   if an error occurred (either the external file
                   could not be read, or the requested color was
                   not defined in the file).
```

PGSFS -- set fill-area style

```
      SUBROUTINE PGSFS (FS)
      INTEGER  FS
```

```
Set the Fill-Area Style attribute for subsequent area-fill by
PGPOLY, PGRECT, or PGCIRC.  Four different styles are available:
solid (fill polygon with solid color of the current color-index),
outline (draw outline of polygon only, using current line attributes),
hatched (shade interior of polygon with parallel lines, using
current line attributes), or cross-hatched. The orientation and
spacing of hatch lines can be specified with routine PGSHS (set
hatch style).

Argument:
 FS     (input)  : the fill-area style to be used for subsequent
                   plotting:
                     FS = 1 => solid (default)
                     FS = 2 => outline
                     FS = 3 => hatched
                     FS = 4 => cross-hatched
                   Other values give an error message and are
                   treated as 2.
```

PGSHLS -- set color representation using HLS system

```
      SUBROUTINE PGSHLS (CI, CH, CL, CS)
      INTEGER CI
      REAL    CH, CL, CS

Set color representation: i.e., define the color to be
associated with a color index.  This routine is equivalent to
PGSCR, but the color is defined in the Hue-Lightness-Saturation
model instead of the Red-Green-Blue model. Hue is represented
by an angle in degrees, with red at 120, green at 240,
and blue at 0 (or 360). Lightness ranges from 0.0 to 1.0, with black
at lightness 0.0 and white at lightness 1.0. Saturation ranges from
0.0 (gray) to 1.0 (pure color). Hue is irrelevant when saturation
is 0.0.

Examples:           H     L     S        R     G     B
    black          any   0.0   0.0      0.0   0.0   0.0
    white          any   1.0   0.0      1.0   1.0   1.0
    medium gray    any   0.5   0.0      0.5   0.5   0.5
    red            120   0.5   1.0      1.0   0.0   0.0
    yellow         180   0.5   1.0      1.0   1.0   0.0
    pink           120   0.7   0.8      0.94  0.46  0.46

Reference: SIGGRAPH Status Report of the Graphic Standards Planning
Committee, Computer Graphics, Vol.13, No.3, Association for
Computing Machinery, New York, NY, 1979. See also: J. D. Foley et al,
``Computer Graphics: Principles and Practice'', second edition,
Addison-Wesley, 1990, section 13.3.5.

Argument:
 CI     (input)  : the color index to be defined, in the range 0-max.
                   If the color index greater than the device
                   maximum is specified, the call is ignored. Color
                   index 0 applies to the background color.
 CH     (input)  : hue, in range 0.0 to 360.0.
 CL     (input)  : lightness, in range 0.0 to 1.0.
 CS     (input)  : saturation, in range 0.0 to 1.0.
```

PGSHS -- set hatching style

```
      SUBROUTINE PGSHS (ANGLE, SEPN, PHASE)
      REAL ANGLE, SEPN, PHASE
```

```
Set the style to be used for hatching (fill area with fill-style 3).
The default style is ANGLE=45.0, SEPN=1.0, PHASE=0.0.

Arguments:
 ANGLE  (input)  : the angle the hatch lines make with the
                   horizontal, in degrees, increasing
                   counterclockwise (this is an angle on the
                   view surface, not in world-coordinate space).
 SEPN   (input)  : the spacing of the hatch lines. The unit spacing
                   is 1 percent of the smaller of the height or
                   width of the view surface.
 PHASE  (input)  : a real number between 0 and 1; the hatch lines
                   are displaced by this fraction of SEPN from a
                   fixed reference.  Adjacent regions hatched with the
                   same PHASE have contiguous hatch lines. To hatch
                   a region with alternating lines of two colors,
                   fill the area twice, with PHASE=0.0 for one color
                   and PHASE=0.5 for the other color.
```

PGSITF -- set image transfer function

```
      SUBROUTINE PGSITF (ITF)
      INTEGER  ITF
```

```
Set the Image Transfer Function for subsequent images drawn by
PGIMAG, PGGRAY, or PGWEDG. The Image Transfer Function is used
to map array values into the available range of color indices
specified with routine PGSCIR or (for PGGRAY on some devices)
into dot density.

Argument:
 ITF    (input)  : type of transfer function:
                     ITF = 0 : linear
                     ITF = 1 : logarithmic
                     ITF = 2 : square-root
```

PGSLS -- set line style

```
      SUBROUTINE PGSLS (LS)
      INTEGER  LS
```

```
Set the line style attribute for subsequent plotting. This
attribute affects line primitives only; it does not affect graph
markers, text, or area fill.
Five different line styles are available, with the following codes:
```

```
1 (full line), 2 (dashed), 3 (dot-dash-dot-dash), 4 (dotted),
5 (dash-dot-dot-dot). The default is 1 (normal full line).

Argument:
 LS     (input)  : the line-style code for subsequent plotting
                   (in range 1-5).
```

PGSLW -- set line width

```
      SUBROUTINE PGSLW (LW)
      INTEGER  LW

Set the line-width attribute. This attribute affects lines, graph
markers, and text. The line width is specified in units of 1/200
(0.005) inch (about 0.13 mm) and must be an integer in the range
1-201. On some devices, thick lines are generated by tracing each
line with multiple strokes offset in the direction perpendicular to
the line.

Argument:
 LW     (input)  : width of line, in units of 0.005 inch (0.13 mm)
                   in range 1-201.
```

PGSTBG -- set text background color index

```
      SUBROUTINE PGSTBG (TBCI)
      INTEGER  TBCI

Set the Text Background Color Index for subsequent text. By default
text does not obscure underlying graphics. If the text background
color index is positive, however, text is opaque: the bounding box
of the text is filled with the color specified by PGSTBG before
drawing the text characters in the current color index set by PGSCI.
Use color index 0 to erase underlying graphics before drawing text.

Argument:
 TBCI   (input)  : the color index to be used for the background
                   for subsequent text plotting:
                     TBCI < 0  => transparent (default)
                     TBCI >= 0 => text will be drawn on an opaque
                   background with color index TBCI.
```

PGSUBP -- subdivide view surface into panels

```
      SUBROUTINE PGSUBP (NXSUB, NYSUB)
      INTEGER NXSUB, NYSUB

PGPLOT divides the physical surface of the plotting device (screen,
window, or sheet of paper) into NXSUB x NYSUB 'panels'. When the
view surface is sub-divided in this way, PGPAGE moves to the next
```

```
panel, not the next physical page. The initial subdivision of the
view surface is set in the call to PGBEG. When PGSUBP is called,
it forces the next call to PGPAGE to start a new physical page,
subdivided in the manner indicated. No plotting should be done
between a call of PGSUBP and a call of PGPAGE (or PGENV, which calls
PGPAGE).

If NXSUB > 0, PGPLOT uses the panels in row order; if <0,
PGPLOT uses them in column order, e.g.,

 NXSUB=3, NYSUB=2            NXSUB=-3, NYSUB=2

+-----+-----+-----+         +-----+-----+-----+
|  1  |  2  |  3  |         |  1  |  3  |  5  |
+-----+-----+-----+         +-----+-----+-----+
|  4  |  5  |  6  |         |  2  |  4  |  6  |
+-----+-----+-----+         +-----+-----+-----+

PGPLOT advances from one panels to the next when PGPAGE is called,
clearing the screen or starting a new page when the last panel has
been used. It is also possible to jump from one panel to another
in random order by calling PGPANL.

Arguments:
 NXSUB  (input)   : the number of subdivisions of the view surface in
                    X (>0 or <0).
 NYSUB  (input)   : the number of subdivisions of the view surface in
                    Y (>0).
```

PGSVP -- set viewport (normalized device coordinates)

```
      SUBROUTINE PGSVP (XLEFT, XRIGHT, YBOT, YTOP)
      REAL XLEFT, XRIGHT, YBOT, YTOP

Change the size and position of the viewport, specifying
the viewport in normalized device coordinates.  Normalized
device coordinates run from 0 to 1 in each dimension. The
viewport is the rectangle on the view surface "through"
which one views the graph.  All the PG routines which plot lines
etc. plot them within the viewport, and lines are truncated at
the edge of the viewport (except for axes, labels etc drawn with
PGBOX or PGLAB).  The region of world space (the coordinate
space of the graph) which is visible through the viewport is
specified by a call to PGSWIN.  It is legal to request a
viewport larger than the view surface; only the part which
appears on the view surface will be plotted.

Arguments:
 XLEFT  (input)   : x-coordinate of left hand edge of viewport, in NDC.
 XRIGHT (input)   : x-coordinate of right hand edge of viewport,
                    in NDC.
 YBOT   (input)   : y-coordinate of bottom edge of viewport, in NDC.
 YTOP   (input)   : y-coordinate of top  edge of viewport, in NDC.
```

PGSWIN -- set window

```
      SUBROUTINE PGSWIN (X1, X2, Y1, Y2)
      REAL X1, X2, Y1, Y2
```

Change the window in world coordinate space that is to be mapped on to the viewport. Usually PGSWIN is called automatically by PGENV, but it may be called directly by the user.

```
Arguments:
 X1     (input)  : the x-coordinate of the bottom left corner
                   of the viewport.
 X2     (input)  : the x-coordinate of the top right corner
                   of the viewport (note X2 may be less than X1).
 Y1     (input)  : the y-coordinate of the bottom left corner
                   of the viewport.
 Y2     (input)  : the y-coordinate of the top right corner
                   of the viewport (note Y2 may be less than Y1).
```

```
      SUBROUTINE PGTBOX (XOPT, XTICK, NXSUB, YOPT, YTICK, NYSUB)

      REAL XTICK, YTICK
      INTEGER NXSUB, NYSUB
      CHARACTER XOPT*(*), YOPT*(*)
```

Draw a box and optionally label one or both axes with (DD) HH MM SS style numeric labels (useful for time or RA - DEC plots). If this style of labelling is desired, then PGSWIN should have been called previously with the extrema in SECONDS of time.

In the seconds field, you can have at most 3 places after the decimal point, so that 1 ms is the smallest time interval you can time label.

Large numbers are coped with by fields of 6 characters long. Thus you could have times with days or hours as big as 999999. However, in practice, you might have trouble with labels overwriting themselves with such large numbers unless you a) use a small time INTERVAL, b) use a small character size or c) choose your own sparse ticks in the call to PGTBOX.

PGTBOX will attempt, when choosing its own ticks, not to overwrite the labels, but this algorithm is not very bright and may fail.

Note that small intervals but large absolute times such as TMIN = 200000.0 s and TMAX=200000.1 s will cause the algorithm to fail. This is inherent in PGPLOT's use of single precision and cannot be avoided. In such cases, you should use relative times if possible.

PGTBOX's labelling philosophy is that the left-most or bottom tick of the axis contains a full label. Thereafter, only changing fields are labelled. Negative fields are given a '-' label, positive fields have none. Axes that have the DD (or HH if the day field is not used) field on each major tick carry the sign on each field. If the axis crosses zero, the zero tick will carry a full label and sign.

This labelling style can cause a little confusion with some special cases, but as long as you know its philosophy, the truth can be divined. Consider an axis with TMIN=20s, TMAX=-20s. The labels will look like

```
       +----------+----------+----------+----------+
    0h0m20s      10s      -0h0m0s      10s        20s

Knowing that the left field always has a full label and that
positive fields are unsigned, informs that time is decreasing
from left to right, not vice versa.   This can become very
unclear if you have used the 'F' option, but that is your problem !

Exceptions to this labelling philosophy are when the finest time
increment being displayed is hours (with option 'Y') or days.
Then all fields carry a label.  For example,

       +----------+----------+----------+----------+
     -10h        -8h        -6h        -4h        -2h


PGTBOX can be used in place of PGBOX; it calls PGBOX and only invokes
time labelling if requested. Other options are passed intact to PGBOX.

Inputs:
 XOPT   :  X-options for PGTBOX.  Same as for PGBOX plus

            'Z' for (DD) HH MM SS.S time labelling
            'Y' means don't include the day field so that labels
                are HH MM SS.S rather than DD HH MM SS.S   The hours
                will accumulate beyond 24 if necessary in this case.
            'X' label the HH field as modulo 24.  Thus, a label
                such as 25h 10m would come out as 1h 10m
            'H' means superscript numbers with d, h, m, & s  symbols
            'D' means superscript numbers with    o, ', & '' symbols
            'F' causes the first label (left- or bottom-most) to
                be omitted. Useful for sub-panels that abut each other.
                Care is needed because first label carries sign as well.
            'O' means omit leading zeros in numbers < 10
                E.g.  3h 3m 1.2s rather than 03h 03m 01.2s  Useful
                to help save space on X-axes. The day field does not
                use this facility.


 YOPT   :  Y-options for PGTBOX.  See above.
 XTICK  :  X-axis major tick increment.  0.0 for default.
 YTICK  :  Y-axis major tick increment.  0.0 for default.
           If the 'Z' option is used then XTICK and/or YTICK must
           be in seconds.
 NXSUB  :  Number of intervals for minor ticks on X-axis. 0 for default
 NYSUB  :  Number of intervals for minor ticks on Y-axis. 0 for default


 The regular XOPT and YOPT axis options for PGBOX are

 A : draw Axis (X axis is horizontal line Y=0, Y axis is vertical
     line X=0).
 B : draw bottom (X) or left (Y) edge of frame.
 C : draw top (X) or right (Y) edge of frame.
 G : draw Grid of vertical (X) or horizontal (Y) lines.
 I : Invert the tick marks; ie draw them outside the viewport
     instead of inside.
 L : label axis Logarithmically.
 N : write Numeric labels in the conventional location below the
     viewport (X) or to the left of the viewport (Y).
 P : extend ("Project") major tick marks outside the box (ignored if
```

```
     option I is specified).
 M : write numeric labels in the unconventional location above the
     viewport (X) or to the right of the viewport (Y).
 T : draw major Tick marks at the major coordinate interval.
 S : draw minor tick marks (Subticks).
 V : orient numeric labels Vertically. This is only applicable to Y.
     The default is to write Y-labels parallel to the axis


       ******************        EXCEPTIONS       *******************

       Note that
         1) PGBOX option 'L' (log labels) is ignored with option 'Z'
         2) The 'O' option will be ignored for the 'V' option as it
            makes it impossible to align the labels nicely
         3) Option 'Y' is forced with option 'D'

       ***************************************************************
```

PGTBOX -- draw frame and write (DD) HH MM SS.S labelling

```
      SUBROUTINE PGTBOX (XOPT, XTICK, NXSUB, YOPT, YTICK, NYSUB)

      REAL XTICK, YTICK
      INTEGER NXSUB, NYSUB
      CHARACTER XOPT*(*), YOPT*(*)
```

```
Draw a box and optionally label one or both axes with (DD) HH MM SS
style numeric labels (useful for time or RA - DEC plots).   If this
style of labelling is desired, then PGSWIN should have been called
previously with the extrema in SECONDS of time.

In the seconds field, you can have at most 3 places after the decimal
point, so that 1 ms is the smallest time interval you can time label.

Large numbers are coped with by fields of 6 characters long.  Thus
you could have times with days or hours as big as 999999.  However,
in practice, you might have trouble with labels overwriting  themselves
with such large numbers unless you a) use a small time INTERVAL,
b) use a small character size or c) choose your own sparse ticks in
the call to PGTBOX.

PGTBOX will attempt, when choosing its own ticks, not to overwrite
the labels, but this algorithm is not very bright and may fail.

Note that small intervals but large absolute times such as
TMIN = 200000.0 s and TMAX=200000.1 s will cause the algorithm
to fail.  This is inherent in PGPLOT's use of single precision
and cannot be avoided.  In such cases, you should use relative
times if possible.

PGTBOX's labelling philosophy is that the left-most or bottom tick of
the axis contains a full label.  Thereafter, only changing fields are
labelled.  Negative fields are given a '-' label, positive fields
have none.   Axes that have the DD (or HH if the day field is not
```

used) field on each major tick carry the sign on each field. If the axis crosses zero, the zero tick will carry a full label and sign.

This labelling style can cause a little confusion with some special cases, but as long as you know its philosophy, the truth can be divined. Consider an axis with TMIN=20s, TMAX=-20s. The labels will look like

```
       +----------+----------+----------+----------+
    0h0m20s      10s      -0h0m0s      10s        20s
```

Knowing that the left field always has a full label and that positive fields are unsigned, informs that time is decreasing from left to right, not vice versa. This can become very unclear if you have used the 'F' option, but that is your problem !

Exceptions to this labelling philosophy are when the finest time increment being displayed is hours (with option 'Y') or days. Then all fields carry a label. For example,

```
       +----------+----------+----------+----------+
     -10h        -8h        -6h        -4h        -2h
```

PGTBOX can be used in place of PGBOX; it calls PGBOX and only invokes time labelling if requested. Other options are passed intact to PGBOX.

```
Inputs:
 XOPT   :  X-options for PGTBOX.  Same as for PGBOX plus

            'Z' for (DD) HH MM SS.S time labelling
            'Y' means don't include the day field so that labels
                are HH MM SS.S rather than DD HH MM SS.S   The hours
                will accumulate beyond 24 if necessary in this case.
            'H' means superscript numbers with d, h, m, & s  symbols
            'D' means superscript numbers with    o, ', & '' symbols
            'F' causes the first label (left- or bottom-most) to
                be omitted. Useful for sub-panels that abut each other.
                Care is needed because first label carries sign as well.
            'O' means omit leading zeros in numbers < 10
                E.g.  3h 3m 1.2s rather than 03h 03m 01.2s  Useful
                to help save space on X-axes. The day field does not
                use this facility.

 YOPT   :  Y-options for PGTBOX.  See above.
 XTICK  :  X-axis major tick increment.  0.0 for default.
 YTICK  :  Y-axis major tick increment.  0.0 for default.
           If the 'Z' option is used then XTICK and/or YTICK must
           be in seconds.
 NXSUB  :  Number of intervals for minor ticks on X-axis. 0 for default
 NYSUB  :  Number of intervals for minor ticks on Y-axis. 0 for default


 The regular XOPT and YOPT axis options for PGBOX are

 A : draw Axis (X axis is horizontal line Y=0, Y axis is vertical
     line X=0).
 B : draw bottom (X) or left (Y) edge of frame.
 C : draw top (X) or right (Y) edge of frame.
 G : draw Grid of vertical (X) or horizontal (Y) lines.
 I : Invert the tick marks; ie draw them outside the viewport
     instead of inside.
```

```
 L : label axis Logarithmically.
 N : write Numeric labels in the conventional location below the
     viewport (X) or to the left of the viewport (Y).
 P : extend ("Project") major tick marks outside the box (ignored if
     option I is specified).
 M : write numeric labels in the unconventional location above the
     viewport (X) or to the right of the viewport (Y).
 T : draw major Tick marks at the major coordinate interval.
 S : draw minor tick marks (Subticks).
 V : orient numeric labels Vertically. This is only applicable to Y.
     The default is to write Y-labels parallel to the axis


       ******************        EXCEPTIONS       *******************

       Note that
         1) PGBOX option 'L' (log labels) is ignored with option 'Z'
         2) The 'O' option will be ignored for the 'V' option as it
            makes it impossible to align the labels nicely
         3) Option 'Y' is forced with option 'D'

       ***************************************************************
```

PGTEXT -- write text (horizontal, left-justified)

```
      SUBROUTINE PGTEXT (X, Y, TEXT)
      REAL X, Y
      CHARACTER*(*) TEXT
```

```
Write text. The bottom left corner of the first character is placed
at the specified position, and the text is written horizontally.
This is a simplified interface to the primitive routine PGPTXT.
For non-horizontal text, use PGPTXT.

Arguments:
 X      (input)  : world x-coordinate of start of string.
 Y      (input)  : world y-coordinate of start of string.
 TEXT   (input)  : the character string to be plotted.
```

PGUPDT -- update display

```
      SUBROUTINE PGUPDT
```

```
Update the graphics display: flush any pending commands to the
output device. This routine empties the buffer created by PGBBUF,
but it does not alter the PGBBUF/PGEBUF counter. The routine should
be called when it is essential that the display be completely up to
date (before interaction with the user, for example) but it is not
known if output is being buffered.

Arguments: none
```

PGVECT -- vector map of a 2D data array, with blanking

```
      SUBROUTINE PGVECT (A, B, IDIM, JDIM, I1, I2, J1, J2, C, NC, TR,
     1                   BLANK)
      INTEGER IDIM, JDIM, I1, I2, J1, J2, NC
      REAL    A(IDIM,JDIM), B(IDIM, JDIM), TR(6), BLANK, C
```

```
Draw a vector map of two arrays.  This routine is similar to
PGCONB in that array elements that have the "magic value" defined by
the argument BLANK are ignored, making gaps in the vector map.  The
routine may be useful for data measured on most but not all of the
points of a grid. Vectors are displayed as arrows; the style of the
arrowhead can be set with routine PGSAH, and the the size of the
arrowhead is determined by the current character size, set by PGSCH.

Arguments:
 A      (input)  : horizontal component data array.
 B      (input)  : vertical component data array.
 IDIM   (input)  : first dimension of A and B.
 JDIM   (input)  : second dimension of A and B.
 I1,I2  (input)  : range of first index to be mapped (inclusive).
 J1,J2  (input)  : range of second index to be mapped (inclusive).
 C      (input)  : scale factor for vector lengths, if 0.0, C will be
                   set so that the longest vector is equal to the
                   smaller of TR(2)+TR(3) and TR(5)+TR(6).
 NC     (input)  : vector positioning code.
                   <0 vector head positioned on coordinates
                   >0 vector base positioned on coordinates
                   =0 vector centered on the coordinates
 TR     (input)  : array defining a transformation between the I,J
                   grid of the array and the world coordinates. The
                   world coordinates of the array point A(I,J) are
                   given by:
                     X = TR(1) + TR(2)*I + TR(3)*J
                     Y = TR(4) + TR(5)*I + TR(6)*J
                   Usually TR(3) and TR(5) are zero - unless the
                   coordinate transformation involves a rotation
                   or shear.
 BLANK   (input) : elements of arrays A or B that are exactly equal to
                   this value are ignored (blanked).
```

PGVPORT -- non-standard alias for PGSVP

```
      SUBROUTINE PGVPORT (XLEFT, XRIGHT, YBOT, YTOP)
      REAL XLEFT, XRIGHT, YBOT, YTOP
```

PGVSIZ -- set viewport (inches)

```
      SUBROUTINE PGVSIZ (XLEFT, XRIGHT, YBOT, YTOP)
      REAL XLEFT, XRIGHT, YBOT, YTOP
```

```
Change the size and position of the viewport, specifying
the viewport in physical device coordinates (inches).  The
viewport is the rectangle on the view surface "through"
which one views the graph.  All the PG routines which plot lines
etc. plot them within the viewport, and lines are truncated at
the edge of the viewport (except for axes, labels etc drawn with
PGBOX or PGLAB).  The region of world space (the coordinate
space of the graph) which is visible through the viewport is
specified by a call to PGSWIN.  It is legal to request a
viewport larger than the view surface; only the part which
appears on the view surface will be plotted.

Arguments:
 XLEFT  (input)  : x-coordinate of left hand edge of viewport, in
                   inches from left edge of view surface.
 XRIGHT (input)  : x-coordinate of right hand edge of viewport, in
                   inches from left edge of view surface.
 YBOT   (input)  : y-coordinate of bottom edge of viewport, in
                   inches from bottom of view surface.
 YTOP   (input)  : y-coordinate of top  edge of viewport, in inches
                   from bottom of view surface.
```

PGVSIZE -- non-standard alias for PGVSIZ

```
      SUBROUTINE PGVSIZE (XLEFT, XRIGHT, YBOT, YTOP)
      REAL XLEFT, XRIGHT, YBOT, YTOP
```

PGVSTAND -- non-standard alias for PGVSTD

```
      SUBROUTINE PGVSTAND
```

PGVSTD -- set standard (default) viewport

```
      SUBROUTINE PGVSTD

Define the viewport to be the standard viewport.  The standard
viewport is the full area of the view surface (or panel),
less a margin of 4 character heights all round for labelling.
It thus depends on the current character size, set by PGSCH.

Arguments: none.
```

PGWEDG -- annotate an image plot with a wedge

```
      SUBROUTINE PGWEDG(SIDE, DISP, WIDTH, FG, BG, LABEL)
      CHARACTER *(*) SIDE,LABEL
```

```
      REAL DISP, WIDTH, FG, BG
```

```
Plot an annotated grey-scale or color wedge parallel to a given axis
of the the current viewport. This routine is designed to provide a
brightness/color scale for an image drawn with PGIMAG or PGGRAY.
The wedge will be drawn with the transfer function set by PGSITF
and using the color index range set by PGSCIR.

Arguments:
 SIDE   (input)  : The first character must be one of the characters
                   'B', 'L', 'T', or 'R' signifying the Bottom, Left,
                   Top, or Right edge of the viewport.
                   The second character should be 'I' to use PGIMAG
                   to draw the wedge, or 'G' to use PGGRAY.
 DISP   (input)  : the displacement of the wedge from the specified
                   edge of the viewport, measured outwards from the
                   viewport in units of the character height. Use a
                   negative value to write inside the viewport, a
                   positive value to write outside.
 WIDTH  (input)  : The total width of the wedge including annotation,
                   in units of the character height.
 FG     (input)  : The value which is to appear with shade
                   1 ("foreground"). Use the values of FG and BG
                   that were supplied to PGGRAY or PGIMAG.
 BG     (input)  : the value which is to appear with shade
                   0 ("background").
 LABEL  (input)  : Optional units label. If no label is required
                   use ' '.
```

PGWINDOW -- non-standard alias for PGSWIN

```
      SUBROUTINE PGWINDOW (X1, X2, Y1, Y2)
      REAL X1, X2, Y1, Y2
```

PGWNAD -- set window and adjust viewport to same aspect ratio

```
      SUBROUTINE PGWNAD (X1, X2, Y1, Y2)
      REAL X1, X2, Y1, Y2
```

```
Change the window in world coordinate space that is to be mapped on
to the viewport, and simultaneously adjust the viewport so that the
world-coordinate scales are equal in x and y. The new viewport is
the largest one that can fit within the previously set viewport
while retaining the required aspect ratio.

Arguments:
 X1     (input)  : the x-coordinate of the bottom left corner
                   of the viewport.
 X2     (input)  : the x-coordinate of the top right corner
                   of the viewport (note X2 may be less than X1).
 Y1     (input)  : the y-coordinate of the bottom left corner
                   of the viewport.
 Y2     (input)  : the y-coordinate of the top right corner of the
                   viewport (note Y2 may be less than Y1).
```