

# Appendix A

## Subroutine Descriptions

### A.1 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.

### A.2 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** should be declared `INTEGER` or `INTEGER*4` in the calling program, not `INTEGER*2`.

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

### A.3 Index of Routines

**PGARRO** – draw an arrow

**PGASK** – control new page prompting

**PGAXIS** – draw an axis

**PGBAND** – read cursor position, with anchor

**PGBBUF** – begin batch of output (buffer)

**PGBEG** – open a graphics device

**PGBIN** – histogram of binned data

**PGBOX** – draw labeled frame around viewport

**PGCIRC** – draw a circle, using fill-area attributes

**PGCLOS** – close the selected graphics device

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

**PGCONF** – fill between two contours

**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

**PGDRAW** – draw a line from the current pen position to a point

**PGEBUF** – end batch of output (buffer)

**PGEND** – close all open graphics devices

**PGENV** – set window and viewport and draw labeled frame

**PGERAS** – erase all graphics from current page

**PGERR1** – horizontal or vertical error bar

**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  
**PGLCUR** – draw a line using the cursor  
**PGLDEV** – list available device types on standard output  
**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)  
**PGMTXT** – write text at position relative to viewport  
**PGNCUR** – mark a set of points using the cursor  
**PGNUMB** – convert a number into a plottable character string  
**PGOLIN** – mark a set of points using the cursor  
**PGOPEN** – open a graphics device  
**PGPAGE** – advance to new page  
**PGPANL** – switch to a different panel on the view surface  
**PGPAP** – change the size of the view surface  
**PGPIXL** – draw pixels  
**PGPNTS** – draw several graph markers, not all the same  
**PGPOLY** – draw a polygon, using fill-area attributes  
**PGPT** – draw several graph markers  
**PGPT1** – draw one graph marker

**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

**PGQCLP** – inquire clipping status

**PGQCOL** – inquire color capability

**PGQCR** – inquire color representation

**PGQCS** – inquire character height in a variety of units

**PGQDT** – inquire name of nth available device type

**PGQFS** – inquire fill-area style

**PGQHS** – inquire hatching style

**PGQID** – inquire current device identifier

**PGQINF** – inquire PGPLOT general information

**PGQITF** – inquire image transfer function

**PGQLS** – inquire line style

**PGQLW** – inquire line width

**PGQNDT** – inquire number of available device types

**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** – inquire size of 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

**PGSCLP** – enable or disable clipping at edge of viewport

**PGSCR** – set color representation

**PGSCRL** – scroll window

**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

**PGSLCT** – select an open graphics device

**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)

**PGTICK** – draw a single tick mark on an axis

**PGUPDT** – update display

**PGVECT** – vector map of a 2D data array, with blanking  
**PGVSIZ** – set viewport (inches)  
**PGVSTD** – set standard (default) viewport  
**PGWEDG** – annotate an image plot with a wedge  
**PGWNAD** – set window and adjust viewport to same aspect ratio  
**PGADVANCE** – non-standard alias for PGPAGE  
**PGBEGIN** – non-standard alias for PGBEG  
**PGCURSE** – non-standard alias for PGCURS  
**PGLABEL** – non-standard alias for PGLAB  
**PGMTEXT** – non-standard alias for PGMTXT  
**PGNCURSE** – non-standard alias for PGNCUR  
**PGPAPER** – non-standard alias for PGPAP  
**PGPOINT** – non-standard alias for PGPT  
**PGPTEXT** – non-standard alias for PGPTXT  
**PGVPORT** – non-standard alias for PGSVP  
**PGVSIZE** – non-standard alias for PGVSIZ  
**PGVSTAND** – non-standard alias for PGVSTD  
**PGWINDOW** – non-standard alias for PGSWIN

---

### **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 the device has been opened) 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.

```

---

**PGAXIS – draw an axis**

```

SUBROUTINE PGAXIS (OPT, X1, Y1, X2, Y2, V1, V2, STEP, NSUB,
:                 DMAJL, DMAJR, FMIN, DISP, ORIENT)
CHARACTER*(*) OPT
REAL X1, Y1, X2, Y2, V1, V2, STEP, DMAJL, DMAJR, FMIN, DISP
REAL ORIENT
INTEGER NSUB

```

Draw a labelled graph axis from world-coordinate position (X1,Y1) to (X2,Y2).

Normally, this routine draws a standard LINEAR axis with equal subdivisions. The quantity described by the axis runs from V1 to V2; this may be, but need not be, the same as X or Y.

If the 'L' option is specified, the routine draws a LOGARITHMIC axis. In this case, the quantity described by the axis runs from  $10^{V1}$  to  $10^{V2}$ . A logarithmic axis always has major, labeled, tick marks spaced by one or more decades. If the major tick marks are spaced by one decade (as specified by the STEP argument), then minor tick marks are placed at 2, 3, .., 9 times each power of 10; otherwise minor tick marks are spaced by one decade. If the axis spans less than two decades, numeric labels are placed at 1, 2, and 5 times each power of ten.

If the axis spans less than one decade, or if it spans many decades, it is preferable to use a linear axis labeled with the logarithm of the quantity of interest.

**Arguments:**

OPT (input) : a string containing single-letter codes for various options. The options currently recognized are:  
     L : draw a logarithmic axis  
     N : write numeric labels  
     1 : force decimal labelling, instead of automatic choice (see PGNUMB).  
     2 : force exponential labelling, instead of automatic.  
 X1, Y1 (input) : world coordinates of one endpoint of the axis.  
 X2, Y2 (input) : world coordinates of the other endpoint of the axis.  
 V1 (input) : axis value at first endpoint.  
 V2 (input) : axis value at second endpoint.  
 STEP (input) : major tick marks are drawn at axis value 0.0 plus or minus integer multiples of STEP. If STEP=0.0, a value is chosen automatically.  
 NSUB (input) : minor tick marks are drawn to divide the major divisions into NSUB equal subdivisions (ignored if STEP=0.0). If NSUB <= 1, no minor tick marks are drawn. NSUB is ignored for a logarithmic axis.  
 DMAJL (input) : length of major tick marks drawn to left of axis (as seen looking from first endpoint to second), in units of the character height.  
 DMAJR (input) : length of major tick marks drawn to right of axis, in units of the character height.  
 FMIN (input) : length of minor tick marks, as fraction of major.  
 DISP (input) : displacement of baseline of tick labels to right of axis, in units of the character height.  
 ORIENT (input) : orientation of label text, in degrees; angle between baseline of text and direction of axis (0-360).

---

### 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 – open a graphics device

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

Note: new programs should use PGOPEN rather than PGBEG. PGOPEN is retained for compatibility with existing programs. Unlike PGOPEN, PGBEG closes any graphics devices that are already open, so it cannot be used to open devices to be used in parallel.

PGBEG opens a graphical device or file and prepares it for subsequent plotting. A device must be opened with PGBEG or PGOPEN before any other calls to PGPLOT subroutines for the device.

If any device is already open for PGPLOT output, it is closed before the new device 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. Some Fortran compilers allow  
 you to use CALL PGBEG(...) and discard the  
 return value, but this is not standard Fortran.

Arguments:

UNIT (input) : this argument is ignored by PGBEG (use zero).  
 FILE (input) : the "device specification" for the plot device.  
 (For explanation, see description of PGOPEN.)  
 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.

### 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 circle, using fill-area attributes

```
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).
```

### PGCLOS – close the selected graphics device

```
SUBROUTINE PGCLOS
```

Close the currently selected graphics device. After the device has been closed, either another open device must be selected with PGSLCT or another device must be opened with PGOOPEN before any further plotting can be done. If the call to PGCLOS is omitted, some or all of the plot may be lost.

[This routine was added to PGLOT in Version 5.1.0. Older programs use PGEND instead.]

Arguments: none

### 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).

### PGCONF – fill between two contours

```

SUBROUTINE PGCONF (A, IDIM, JDIM, I1, I2, J1, J2, C1, C2, TR)
  INTEGER IDIM, JDIM, I1, I2, J1, J2
  REAL    A(IDIM,JDIM), C1, C2, TR(6)

```

Shade the region between two contour levels of a function defined on the nodes of a rectangular grid. The routine uses the current fill attributes, hatching style (if appropriate), and color index.

If you want to both shade between contours and draw the contour lines, call this routine first (once for each pair of levels) and then CALL PGCONT (or PGCONS) to draw the contour lines on top of the

shading.

Note 1: This routine is not very efficient: it generates a polygon fill command for each cell of the mesh that intersects the desired area, rather than consolidating adjacent cells into a single polygon.

Note 2: If both contours intersect all four edges of a particular mesh cell, the program behaves badly and may consider some parts of the cell to lie in more than one contour range.

Note 3: If a contour crosses all four edges of a cell, this routine may not generate the same contours as PGCONT or PGCONS (these two routines may not agree either). Such cases are always ambiguous and the routines use different approaches to resolving the ambiguity.

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).  
 C1, C2 (input) : contour levels; note that C1 must be less than C2.  
 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.

---

### 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 underlying 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; to label 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  $n=0,1,\dots,10$ , and  $r=0.25m$ , for  $m=0,1,\dots,4$ . 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.

### Notes:

To reverse the sense of a color table, change the chosen contrast and brightness to  $-CONTRA$  and  $1-BRIGHT$ .

In the following, the term 'color table' refers to the input L,R,G,B arrays, whereas 'color ramp' refers to the resulting ramp of colors that would be seen with PGWEDG.

### 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).  
                   Negative values reverse the direction of the ramp.  
 BRIGHT (input) : The brightness of the color ramp. This is normally  
                   0.5, but can sensibly hold any value between 0.0  
                   and 1.0. Values at or beyond the latter two  
                   extremes, saturate the color ramp with the colors  
                   of the respective end of the color table.

---

### 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.

---

### 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 PGGBUF. PGGBUF and PGEBUF calls should always be paired. Each call to PGGBUF 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 – close all open graphics devices**

SUBROUTINE PGEND

Close and release any open graphics devices. All devices must be closed by calling either PGCLDS (for each device) or PGEND before the program terminates. If a device is not closed properly, some or all of the graphical output 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 current panel, if the view surface has been divided into panels with PGSUBP).

Arguments: none

**PGERR1 – horizontal or vertical error bar**

```

SUBROUTINE PGERR1 (DIR, X, Y, E, T)
INTEGER DIR
REAL X, Y, E
REAL T
  
```

Plot a single error bar 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.  
 To plot many error bars, use PGERRB.

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).

X (input) : world x-coordinate of the data.  
 Y (input) : world y-coordinate 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.

### 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.,

$$\text{shade} = [A(i,j) - BG] / [FG - BG]$$

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 fraction 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:

$$\begin{aligned} X &= TR(1) + TR(2)*I + TR(3)*J \\ Y &= TR(4) + TR(5)*I + TR(6)*J \end{aligned}$$

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 before PGIMAG. The default values for C1 and C2 are device-dependent; 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  $\leq A1$  being given color index C1 and values  $\geq A2$  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:

$$\begin{aligned} X &= \text{TR}(1) + \text{TR}(2)*I + \text{TR}(3)*J \\ Y &= \text{TR}(4) + \text{TR}(5)*I + \text{TR}(6)*J \end{aligned}$$

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).

---

### **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 on standard output**

```
SUBROUTINE PGLDEV
```

Writes (to standard output) a list of all device types available in the current PGPLOT installation.

Arguments: none.

---

### **PGLLEN – find length of a string in a variety of units**

```
SUBROUTINE PGLLEN (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.

## 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 FJUST = 0.5, the center of the string will be placed at COORD; if FJUST = 1.0, the right-hand end of the string will be placed at COORD. Other values between 0 and 1 give intermediate 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.

```

**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.

```

## PGOPEN – open a graphics device

```

INTEGER FUNCTION PGOPEN (DEVICE)
CHARACTER*(*) DEVICE

```

Open a graphics device for PGPLOT output. If the device is opened successfully, it becomes the selected device to which graphics output is directed until another device is selected with PGSLECT or the device is closed with PGCLOSE.

The value returned by PGOPEN should be tested to ensure that the device was opened successfully, e.g.,

```

ISTAT = PGOPEN('plot.ps/PS')
IF (ISTAT .LE. 0 ) STOP

```

Note that `PGOPEN` must be declared `INTEGER` in the calling program.

The `DEVICE` argument is a character constant or variable; its value should be one of the following:

- (1) A complete device specification of the form `'device/type'` or `'file/type'`, where `'type'` is one of the allowed `PGPLOT` device types (installation-dependent) and `'device'` or `'file'` is the name of a graphics device or disk file appropriate for this type. The `'device'` or `'file'` may contain `'/'` characters; the final `'/'` delimits the `'type'`. If necessary to avoid ambiguity, the `'device'` part of the string may be enclosed in double quotation marks.
- (2) A device specification of the form `'/type'`, where `'type'` is one of the allowed `PGPLOT` device types. `PGPLOT` supplies a default file or device name appropriate for this device type.
- (3) A device specification with `'/type'` omitted; in this case the type is taken from the environment variable `PGPLOT_TYPE`, if defined (e.g., `setenv PGPLOT_TYPE PS`). Because of possible confusion with `'/'` in file-names, omitting the device type in this way is not recommended.
- (4) A blank string (`' '`); in this case, `PGOPEN` will use the value of environment variable `PGPLOT_DEV` as the device specification, or `'/NULL'` if the environment variable is undefined.
- (5) A single question mark, with optional trailing spaces (`'?'`); in this case, `PGPLOT` will prompt the user to supply the device specification, with a prompt string of the form  
`'Graphics device/type (? to see list, default XXX):'`  
where `'XXX'` is the default (value of environment variable `PGPLOT_DEV`).
- (6) A non-blank string in which the first character is a question mark (e.g., `'?Device: '`); in this case, `PGPLOT` will prompt the user to supply the device specification, using the supplied string as the prompt (without the leading question mark but including any trailing spaces).

In cases (5) and (6), the device specification is read from the standard input. The user should respond to the prompt with a device specification of the form (1), (2), or (3). If the user types a question-mark in response to the prompt, a list of available device types is displayed and the prompt is re-issued. If the user supplies an invalid device specification, the prompt is re-issued. If the user responds with an end-of-file character, e.g., `ctrl-D` in `UNIX`, program execution is aborted; this avoids the possibility of an infinite prompting loop. A programmer should avoid use of `PGPLOT`-prompting if this behavior is not desirable.

The device type is case-insensitive (e.g., '/ps' and '/PS' are equivalent). The device or file name may be case-sensitive in some operating systems.

Examples of valid DEVICE arguments:

- (1) 'plot.ps/ps', 'dir/plot.ps/ps', '"dir/plot.ps"/ps',  
'user:[tjp.plots]plot.ps/PS'
- (2) '/ps' (PGPLOT interprets this as 'pgplot.ps/ps')
- (3) 'plot.ps' (if PGPLOT\_TYPE is defined as 'ps', PGPLOT  
interprets this as 'plot.ps/ps')
- (4) ' ' (if PGPLOT\_DEV is defined)
- (5) '? '
- (6) '?Device specification for PGPLOT: '

[This routine was added to PGPLOT in Version 5.1.0. Older programs use PGBEG instead.]

Returns:

PGOPEN : returns either a positive value, the identifier of the graphics device for use with PGSLECT, or a 0 or negative value indicating an error. In the event of error a message is written on the standard error unit.

Arguments:

DEVICE (input) : the 'device specification' for the plot device (see above).

## 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 changed 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.

```

**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 several 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.

## PGPOLY – draw a polygon, using fill-area attributes

```
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 an 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 several 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.

---

### PGPT1 – draw one graph marker

```
SUBROUTINE PGPT1 (XPT, YPT, SYMBOL)
  REAL XPT, YPT
  INTEGER SYMBOL
```

Primitive routine to draw a single Graph Marker at a specified point. The marker is 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 (XPT,YPT) in world coordinates.

To draw several markers with coordinates specified by X and Y arrays, use routine PGPT.

Arguments:

XPT (input) : world x-coordinate of the point.  
 YPT (input) : world y-coordinate of the point.  
 SYMBOL (input) : code number of the symbol to be drawn:  
   -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.

## 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, BARB)  
 INTEGER FS  
 REAL ANGLE, BARB

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.  
 BARB (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

**PGQCLP – inquire clipping status**

```
SUBROUTINE PGQCLP(STATE)
  INTEGER  STATE
```

Query the current clipping status (set by routine PGSCLP).

**Argument:**

STATE (output) : receives the clipping status (0 => disabled,  
 1 => enabled).

**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
                UNITS = 4 : world coordinates
                Other values give an error message, and are
                treated as 0.
XCH   (output) : The character height for text written with a
                vertical baseline.
YCH   (output) : The character height for text written with
                a horizontal baseline (the usual case).

```

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=4, XCH receives the height in horizontal world coordinates (as used for the x-axis), and YCH receives the height in vertical world coordinates (as used for the y-axis). Unless special care has been taken to achieve equal world-coordinate scales on both axes, the values of 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.

## PGQDT – inquire name of nth available device type

```

SUBROUTINE PGQDT(N, TYPE, TLEN, DESCR, DLEN, INTER)
  INTEGER N
  CHARACTER*(*) TYPE, DESCR
  INTEGER TLEN, DLEN, INTER

```

Return the name of the Nth available device type as a character string. The number of available types can be determined by calling PGQNDT. If the value of N supplied is outside the range from 1 to the number of available types, the routine returns DLEN=TLEN=0.

### Arguments:

N (input) : the number of the device type (1..maximum).  
 TYPE (output) : receives the character device-type code of the Nth device type. The argument supplied should be large enough for at least 8 characters. The first character in the string is a '/' character.  
 TLEN (output) : receives the number of characters in TYPE, excluding trailing blanks.  
 DESCR (output) : receives a description of the device type. The argument supplied should be large enough for at least 64 characters.  
 DLEN (output) : receives the number of characters in DESCR, excluding trailing blanks.  
 INTER (output) : receives 1 if the device type is an interactive one, 0 otherwise.

**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  (output) : 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    (output) : 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   (output) : 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.
```

---

**PGQID – inquire current device identifier**

```
SUBROUTINE PGQID (ID)
  INTEGER ID
```

This subroutine returns the identifier of the currently selected device, or 0 if no device is selected. The identifier is assigned when PGOOPEN is called to open the device, and may be used as an argument to PGSLECT. Each open device has a different identifier.

[This routine was added to PGPLOT in Version 5.1.0.]

## Argument:

ID (output) : the identifier of the current device, or 0 if no device is currently selected.

**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').
'COURSOR'    * - does the current device have a graphics cursor?
                ('YES' or 'NO').
'SCROLL'     * - does current device have rectangle-scroll
                capability ('YES' or 'NO'); see PGSCRL.
```

## 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 PGSW).

Argument:

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

---

### **PGQNDT – inquire number of available device types**

```
SUBROUTINE PGQNDT(N)
INTEGER N
```

Return the number of available device types. This routine is usually used in conjunction with PGQDT to get a list of the available device types.

Arguments:

N (output) : the number of available device types.

---

### 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. The four corners are in the order: lower left, upper left, upper right, lower right (where left and right refer to the first and last characters in the string).

If the string is blank or contains no drawable 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 corresponding 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 – inquire size of view surface

```

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

```

This routine returns the dimensions of the view surface (the maximum plottable area) of the currently selected graphics device, in a variety of units. The size of the view surface is device-dependent and is established when the graphics device is opened. On some devices, it can be changed by calling PGPAP before starting a new page with PGPAGE. On some devices, the size can be changed (e.g., by a workstation window manager) outside PGLOT, and PGLOT detects the change when PGPAGE is used. Call this routine after PGPAGE to

find the current size.

Note 1: the width and the height of the view surface in normalized device coordinates are both always equal to 1.0.

Note 2: when the device is divided into panels (see PGSUBP), the view surface is a single panel.

Arguments:

UNITS (input) : 0,1,2,3 for output in normalized device coords,  
inches, mm, or device units (pixels)  
X1 (output) : always returns 0.0  
X2 (output) : width of view surface  
Y1 (output) : always returns 0.0  
Y2 (output) : height of view surface

---

### 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, BARB)
INTEGER FS
REAL ANGLE, BARB
```

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.  
**BARB** (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 PGSVAE

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, and clipping state. Color representation is not saved.

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

Note that when multiple devices are in use, PGUNSA retrieves the values saved by the last PGSVAE call, even if they were for a different device.

Arguments: none

## PGUNSA – restore PGPLOT attributes

ENTRY PGUNSA

This routine restores the PGPLOT attributes saved in the last call to PGSVAE. Usage: CALL PGUNSA (no arguments). See PGSVAE.

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

## PGSCLP – enable or disable clipping at edge of viewport

```
SUBROUTINE PGSCLP(STATE)
  INTEGER STATE
```

Normally all PGPLOT primitives except text are ‘clipped’ at the edge of the viewport: parts of the primitives that lie outside the viewport are not drawn. If clipping is disabled by calling this routine, primitives are visible wherever they lie on the view surface. The default (clipping enabled) is appropriate for almost all applications.

Argument:

STATE (input) : 0 to disable clipping, or 1 to enable clipping.

25-Feb-1997 [TJP] - new routine.

## 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)

### PGSCRL – scroll window

SUBROUTINE PGSCRL (DX, DY)  
 REAL DX, DY

This routine moves the window in world-coordinate space while leaving the viewport unchanged. On devices that have the capability, the pixels within the viewport are scrolled horizontally, vertically or both in such a way that graphics previously drawn in the window are shifted so that their world coordinates are unchanged.

If the old window coordinate range was (X1, X2, Y1, Y2), the new coordinate range will be approximately (X1+DX, X2+DX, Y1+DY, Y2+DY). The size and scale of the window are unchanged.

The window can only be shifted by a whole number of pixels (device coordinates). If DX and DY do not correspond to integral numbers of pixels, the shift will be slightly different from that requested. The new window-coordinate range, and hence the exact amount of the shift, can be determined by calling PGQWIN after this routine.

Pixels that are moved out of the viewport by this operation are lost completely; they cannot be recovered by scrolling back. Pixels that are ‘‘scrolled into’’ the viewport are filled with the background color (color index 0).

If the absolute value of DX is bigger than the width of the window, or the absolute value of DY is bigger than the height of the window, the effect will be the same as zeroing all the pixels in the viewport.

Not all devices have the capability to support this routine. It is only available on some interactive devices that have discrete pixels. To determine whether the current device has scroll capability, call PGQINF.

Arguments:

DX (input) : distance (in world coordinates) to shift the window horizontally (positive shifts window to the right and scrolls to the left).  
 DY (input) : distance (in world coordinates) to shift the window vertically (positive shifts window up and scrolls down).

## 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. This should not be zero.

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

### PGSLCT – select an open graphics device

SUBROUTINE PGSLCT(ID)  
INTEGER ID

Select one of the open graphics devices and direct subsequent plotting to it. The argument is the device identifier returned by PGOOPEN when the device was opened. If the supplied argument is not a

valid identifier of an open graphics device, a warning message is issued and the current selection is unchanged.

[This routine was added to PGPLOT in Version 5.1.0.]

Arguments:

ID (input, integer): identifier of the device to be selected.

---

### 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).

---

## 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 PGLOT'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.

```



TEXT (input) : the character string to be plotted.

---

### PGTICK – draw a single tick mark on an axis

```

SUBROUTINE PGTICK (X1, Y1, X2, Y2, V, TIKL, TIKR, DISP,
:                ORIENT, STR)
REAL X1, Y1, X2, Y2, V, TIKL, TIKR, DISP, ORIENT
CHARACTER*(*) STR

```

Draw and label single tick mark on a graph axis. The tick mark is a short line perpendicular to the direction of the axis (which is not drawn by this routine). The optional text label is drawn with its baseline parallel to the axis and reading in the same direction as the axis (from point 1 to point 2). Current line and text attributes are used.

#### Arguments:

X1, Y1 (input) : world coordinates of one endpoint of the axis.  
X2, Y2 (input) : world coordinates of the other endpoint of the axis.  
V (input) : draw the tick mark at fraction V ( $0 \leq V \leq 1$ ) along the line from (X1,Y1) to (X2,Y2).  
TIKL (input) : length of tick mark drawn to left of axis (as seen looking from first endpoint to second), in units of the character height.  
TIKR (input) : length of major tick marks drawn to right of axis, in units of the character height.  
DISP (input) : displacement of label text to right of axis, in units of the character height.  
ORIENT (input) : orientation of label text, in degrees; angle between baseline of text and direction of axis (0-360).  
STR (input) : text of label (may be blank).

---

### 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).

**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.

---

### 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 ' '.

### 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).

### PGADVANCE – non-standard alias for PGPAGE

```
SUBROUTINE PGADVANCE
```

See description of PGPAGE.

**PGBEGIN – non-standard alias for PGBEG**

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

See description of PGBEG.

---

**PGCURSE – non-standard alias for PGCURS**

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

See description of PGCURS.

---

**PGLABEL – non-standard alias for PGLAB**

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

See description of PGLAB.

---

**PGMTEXT – non-standard alias for PGMTXT**

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

See description of PGMTXT.

---

**PGNCURSE – non-standard alias for PGNCUR**

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

See description of PGNCUR.

---

**PGPAPER – non-standard alias for PGPAP**

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

See description of PGPAP.

---

### **PGPOINT – non-standard alias for PGPT**

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

See description of PGPT.

---

### **PGPTEXT – non-standard alias for PGPTXT**

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

See description of PGPTXT.

---

### **PGVPORT – non-standard alias for PGSVP**

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

See description of PGSVP.

---

### **PGVSIZE – non-standard alias for PGVSIZ**

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

See description of PGVSIZ.

---

### **PGVSTAND – non-standard alias for PGVSTD**

```
SUBROUTINE PGVSTAND
```

See description of PGVSTD.

---

### **PGWINDOW – non-standard alias for PGSWIN**

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

See description of PGSWIN.

---