Simpleplot 2-15

# SIMPLEPLOT
## Reference Manual

BUSS Ltd.

**SIMPLEPLOT Reference Manual (version 2-15)**

BUSS Ltd.
Business and Innovation Centre
Angel Way
Bradford
West Yorkshire
BD7 1BX
United Kingdom

Telephone 01274 841396
+44 1274 841396
Fax 01274 841388
+44 1274 841388

Web Site `http://www.buss.co.uk/buss`
Email info@buss.co.uk

Ninth Edition, October 2000

# Contents

# Preface

SIMPLEPLOT is a library of FORTRAN subroutines for plotting graphs. A wide variety of graphs can be drawn as well as more general pictures and diagrams. Facilities are biased towards the graphical representation of data; in particular, scientific data.

SIMPLEPLOT was originally designed for programmers who wanted to draw pictures of their data with minimum programming effort. Although it still achieves this goal, SIMPLEPLOT has developed into a much more powerful tool for professional software engineers.

Six separate sections constitute the complete SIMPLEPLOT Mark 2:

- The basic package for conventional graph plotting – $x$-$y$ plots and polar plots.
- Additional subroutines for 3-dimensional plotting – contour maps and surface pictures of 3-D data.
- Additional subroutines for presentation graphics – bar charts, histograms and pie charts.
- *SIMPLEPLOT Volumes* – perspective pictures of 4-dimensional data.
- *SIMPLEPLOT Maps* – for representing data based on geographical coordinate systems.
- *SIMPLEPLOT ViSualization* – for full colour modelling of functions of two, three and four variables.

SIMPLEPLOT-PLUS refers to a SIMPLEPLOT library which is made up from the first three sections and contains many additional facilities.

This manual covers all subroutines supported in the 2–14 release of the library. Further utilities have also been relesed and are specified in the SIMPLEPLOT Supplement.

## Graphics device interface

The SIMPLEPLOT library is independent of any single graphics system but, rather than describe SIMPLEPLOT as either device *independent* or device *dependent*, it is more appropriate to describe it as device *sensitive* with a device independent interface for the user. This means that the user is protected from having to know about the features of the target output device, but SIMPLEPLOT makes as much use of these features as possible.

The SIMPLEPLOT library is usually supplied with the BUSS *Single Entry Point* (S.E.P.) device driver system; this includes interfaces to a large number of graphics devices, and the range of validated device drivers is constantly being extended. Moreover, SIMPLEPLOT can address graphics devices directly, or through a separate low-level graphics systems (*eg.* GKS, CGM) or graphics languages (*eg.* PostScript).

# Introduction

The *SIMPLEPLOT Reference manual* ($8^{th}$ edition) is a comprehensive reference manual for the Simpleplot library of graph drawing subroutines, and covers sections 1, 2, 4, 7 and Simpleplot-Plus. It can be used as a manual in its own right but, for an introduction to Simpleplot, please refer to the *SIMPLEPLOT Primer*.

## Overview

The *SIMPLEPLOT Reference manual* is organized as an alphabetical listing of subroutine specifications, with an introduction, technical appendix and index:

- Introduction to the *SIMPLEPLOT Reference manual*
- Subroutine specifications
- Index

Other appendices, which were listed in previous additions of the *SIMPLEPLOT Reference manual* can now be found in the *SIMPLEPLOT 2-14 Supplement*.

## Software version

This manual is based on Simpleplot Mark 2, version 2-14.

## Target audience

The *SIMPLEPLOT Reference manual* has been written with the following readers in mind:

- All users of Simpleplot who need details of subroutine arguments and behaviour.
- Existing users of Simpleplot who wish to update their knowledge (and programs) to include the new facilities of Simpleplot version 2-14.

All readers should be familiar with programming in Fortran on their host computer system.

## Related documents

Related documents include

- The *SIMPLEPLOT 2-14 Supplement*, which gives appendices and details of the facilities introduced in Simpleplot version 2-14.
- The *SIMPLEPLOT 2-13 Supplement*, which gives details of the facilities introduced in Simpleplot version 2-13.
- The *SIMPLEPLOT Primer*, which gives an introduction to Simpleplot, especially those facilities available for plotting 2-D data.
- *SIMPLEPLOT 3-D*, which describes how to draw representations of 3-D data including the new waterfall charts, stored curve coordinates and labelled curves.
- *SIMPLEPLOT Volumes*, which describes subroutines for drawing perspective volumes of 3-D `REAL` arrays.

- *SIMPLEPLOT Maps*, which describes subroutines for plotting geographical data.
- *SIMPLEPLOT ViSualization*, which describes subroutines for perspective drawing of functions of two, three and four variables.

## How to report problems

If you have any problems with SIMPLEPLOT software or its associated products and services please notify us on one of our Software Performance Report (SPR) forms. One of these should be sent out with every software kit – please photocopy it or contact us if you would like extra copies.

## Conventions

The following conventions are used in this manual for subroutine specifications. All specifications are given in a similar format whether they are classified as graphics, specification or auxiliary subroutines.

- Numbers in square brackets, [1], [2], [3] and [4], within subroutine specifications relate to the use of SIMPLEPLOT with plotting devices which have program-selectable pens; these pointers may be reset as often as required (see `PEN` and `SETPNS`).
- Subroutines which take text strings as arguments are provided in two forms but both use FORTRAN 77 `CHARACTER` expressions. The only difference between them is that the routines which have the extra `INTEGER` argument interpret this number as the exact length of the text string; in this case,
  - There is no removal of trailing spaces.
  - Only the specified number of characters are used; this may lead to truncation of the string itself.
- Potential diagnostic messages are given in each subroutine specification with a number indicating the type of message and therefore at which diagnostic level it is issued. Additional information about individual messages is given in Appendix M in the *SIMPLEPLOT 2-14 Supplement*.

# Subroutine Specifications

**SUBROUTINE NAME (ARG1, ARG2)**

**Name**

NAME – brief summary line

**Availability** Section 1, 2, 4, 5, 6, 7 or +, released version 2-*n*.

**Arguments** Throughout the specifications, arguments of type INTEGER have been given names starting with I–N, and arguments of type REAL have been given names starting with the letters A–H and O–Z.

IN only arguments are identified as *expression*; a variable name, a constant value or an expression may be used for such arguments.

INOUT or OUT only arguments are identified as *variable*.

| | | |
|---|---|---|
| RVAL | REAL expression | Expression with floating point value |
| IVAL | INTEGER expression | Expression with integer value |
| TORF | LOGICAL expression | Expression with logical value (.TRUE. or .FALSE.) |
| CHAR | CHARACTER*1 | Expression with single character value |
| CHTYPE | CHARACTER*2 | Expression with two character value |
| CAP | STRING expression | Expression with any length character value |
| VARX | REAL variable | Variable of type REAL |
| STR | STRING variable | Variable of type CHARACTER*$n$ |
| LABARR | STRING array | Array of string values |
| IARR | INTEGER array | 1-dimensional array of INTEGER values |
| I2ARR | INTEGER 2-D array | 2-dimensional array of INTEGER values |
| DARR | REAL array | 1-dimensional array of REAL values |
| D2ARR | REAL 2-D array | 2-dimensional array of REAL values |
| D3ARR | REAL 3-D array | 3-dimensional array of REAL values |

Subroutine specifications are listed alphabetically regardless of whether they belong to Section 1, 2, 4, 7 or plus.

## SUBROUTINE ADDCP7 (CAP)
## SUBROUTINE ADDCAP (CAP, NCAP)

### Name

ADDCP7 – to draw a caption in a previously defined area.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

ADDCAP/ADDCP7 draws [2] the caption, CAP, in the next position in the area defined by DEFCAP (or DEFKEY).

When a key or caption area is defined, a pointer to the top space in the area is created. As key entries or captions are inserted, the pointer is automatically moved down to the next vacant space.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)$^2$
(Key/caption area full)$^2$
(No active key/caption area)$^2$

### See also

ADDJST, ADDTOP, BLNKKY, DEFCAP, QKYCAP, RESNXT and TEXTLF.

## SUBROUTINE ADDJST (JCHAR)

### Name

`ADDJST` – to specify how entries in caption areas/keys are to be justified.

**Availability** Section 4, released before version 2-5.

### Argument

`JCHAR` `CHARACTER*1` Justification of key entries/captions

| `JCHAR` | *Horizontal justification* |
|---|---|
| `'C'` | Centred |
| `'L'` | Left justified |
| `'R'` | Right justified |
| `'D'` | Default |

### Description

By default, captions drawn by `ADDCP7` and key entries drawn by `*KY`/`*K7` are left justified within the space available; composite keys drawn by `SHKEYS`, `MPK7H`/`MPKYH` and `MPK7V`/`MPKYV` have a centred caption, followed by right-justified numeric key entries (for vertical map keys) or left-justified text entries.

`ADDJST` changes the justification of key entries and captions. `ADDJST` also affects the justification of entries in segment labels on pie charts (see `PIINCL`).

### Default

`CALL ADDJST('D')` restores the default.

### See also

`ADDCP7`, `BOTHK7`, `FILLK7`, `LINEK7`, `MARKK7`, `MPK7H`, `MPK7V`, `PIINCL`, `SHDEK7` and `SHKEYS`.

## SUBROUTINE ADDTOP

### Name

`ADDTOP` – to reset the current key/caption usage pointer to the top of sequence.

**Availability** Section 4, released before version 2-5.

### Arguments

None.

### Description

When a key or sequence of captions is set up, a pointer to the top space in the rectangle is created. As key entries or captions are inserted, the pointer is automatically moved down to the next vacant space. `ADDTOP` sets the pointer back to the top.

For example, to produce two sets of captions, left justified and right justified in the same caption area, once the first sequence has filled the area from top to bottom, `ADDJST` can specify right justification, `ADDTOP` sets the pointer back to the top and then the right justified captions can be drawn.

### See also

`ADDCP7`, `ADDJST`, `DEFCAP` and `DEFKEY`.

## SUBROUTINE ARHEAD (PROP)

### Name

ARHEAD – to specify the proportional size of arrow heads.

**Availability** Section 4, released version 2-6.

### Argument

| | | |
|---|---|---|
| PROP | REAL expression | Proportion of head w.r.t. to shaft |

### Description

By default, arrows are drawn with heads of a constant size, related to the size of text. ARHEAD specifies variably sized arrow heads such that each head is proportional to the length of the shaft. The ratio of head size to shaft length is indicated by the argument PROP (0.2 gives reasonable results).

### Default

CALL ARHEAD(0.0) restores the default.

### See also

ARROW, ARSHFT and TEXTSZ.

### SUBROUTINE ARROW (XFROM, YFROM, XTO, YTO)

**Name**

ARROW – to draw an arrow between two specified points.

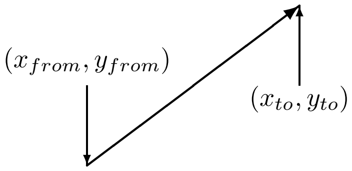
**Availability** Section 4, released version 2-6.

**Arguments**

| | | |
|---|---|---|
| XFROM,<br>YFROM | REAL expressions | Coordinates of the end of the arrow in units of plotting scales |
| XTO, YTO | REAL expressions | Coordinates of the head of the arrow in units of plotting scales |

**Description**

ARROW draws [1] an arrow from $(x_{from}, y_{from})$ to $(x_{to}, y_{to})$ with its head at $(x_{to}, y_{to})$. The position and direction of the arrow are completely defined by the four arguments which specify the coordinates of the end and the head of the arrow in terms of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine KXYXY.



If $x_{from} = x_{to}$ and $y_{from} = y_{to}$, nothing is drawn.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(**$x_1, y_1, x_2, y_2$**)$^2$

**See also**

ARHEAD, ARSHFT, CLIPLV and KXYXY.

## SUBROUTINE ARSHFT (TORF)

### Name

`ARSHFT` – to specify whether arrows are to include a shaft.

**Availability** Section 4, released version 2-6.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether shaft is to be included |

### Description

By default, arrows are drawn with both a shaft and an arrow head but it can be required to draw arrow heads on existing lines. `ARSHFT` specifies whether or not the shaft is to be drawn by `ARROW`. The arguments of `ARROW` are still needed to define the direction of the arrow head and, if variable sized arrow heads have been requested by `ARHEAD`, to control the size of arrow heads.

### Default

`CALL ARSHFT(.TRUE.)` restores the default.

### See also

`ARROW` and `ARHEAD`.

## SUBROUTINE AXCLR (CHAXIS, ILEVEL)

### Name

AXCLR – to specify the level of annotation drawn with the axis.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| ILEVEL | INTEGER expression | Level of axis annotation |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| ILEVEL | *Effect* |
|---|---|
| 0 | Tick marks and annotation labels are drawn |
| 1 | Only axis line drawn |

### Description

By default, when an axis is drawn it is subdivided and annotated at regular intervals using automatically generated labels. Calling AXCLR(CHAXIS,1) before the axis is drawn suppresses the drawing of tick marks and annotation labels; user-defined labels can then be drawn using AXLAB7 or AXTXT7.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*. AXCLR can be used for all axis types.

### Diagnostics

(Invalid axis type: *XX*)[1]

### Default

CALL AXCLR(CHAXIS,0) returns to automatic annotation of the specified axis.

### See also

AXIS7, AXLAB7 and AXTXT7.

## SUBROUTINE AXCRSS (CHAXIS, CROSS)

### Name

`AXCRSS` – to specify the intersection of one axis with its natural pair.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `CROSS` | `REAL` expression | Position of axis in units of the ‘other’ scale |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | × | × | × | × | × | √ | √ | × | × | × | × |

### Description

By default, a Cartesian or radial axis is positioned such that it intersects the other axis at zero, or the value closest to zero on the scale. An angular axis intersects the radial axis at the outer extreme of the radial scale. `AXCRSS` specifies an alternative position for an axis. `AXCRSS` must be called before an axis is drawn.

If the point of intersection lies outside the plotting scale, the axis is drawn at the edge of the picture nearest the intersection.

A fuller description of the values of `CHAXIS` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(Inappropriate axis type: XX)¹
(Invalid axis type: XX)¹
```

### Default

`CALL AXCRSS(CHAXIS,0.0)` restores the default except for an angular axis for which you must specify the outer radial value or `CALL AXLOCN('AP','D')`.

### See also

`AXIS7` and `AXLOCN`.

## SUBROUTINE AXES7 (CAPX, CAPY)
## SUBROUTINE AXES (CAPX, NCAPX, CAPY, NCAPY)

### Name

AXES7 – to start a new picture and draw axes.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| CAPX | STRING expression | Caption for horizontal axis |
| CAPY | STRING expression | Caption for vertical axis |
| NCAPX | INTEGER expression | Number of characters in CAPX |
| NCAPY | INTEGER expression | Number of characters in CAPY |

### Description

AXES7 starts a new picture and draws [1] a pair of axes which are annotated [2] with the scales which exist when the call occurs. The characteristics of these axes can be controlled or augmented by all the other axis subroutines which affect Cartesian or polar axes (see below).

If polar units have been specified by EQSCAL, AXES7 draws a polar framework with one radial axis at the initial $\theta$ value, labelled with CAPX; if the angular range is less than a cycle, another radial axis is drawn at the other end of the range, labelled with CAPY; an angular axis is drawn covering the angular range.

If scales are extended to provide similar linear scales in $x$ and $y$ (see EQSCAL), the axes annotate the requested range rather than the extended range. If a box is required to surround these axes, DRAWBX can be used to draw a box on the current picture.

### See also

AXIS7, DRAWBX, EQSCAL, NEWPIC and POLAR7.

## SUBROUTINE AXGRID (CHAXIS, ILEVEL, LTYPE)

### Name

AXGRID – to specify the style of grids drawn in association with axis subdivisions.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| ILEVEL | INTEGER expression | Level of grid |
| LTYPE | INTEGER expression | Type of broken line pattern |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | × | × | × | √ | √ | √ | √ | × | × | × | × |

| ILEVEL | *Level of grid* |
|---|---|
| 0 | No grid |
| 1 | Grid lines drawn [3] at annotated and major unannotated tick marks |
| 2 | Grid lines drawn [4] at all tick marks including minor |

### Description

By default, when an axis is drawn no grid lines are drawn. AXGRID specifies that grid lines are drawn in association with axis subdivisions.

Grids can have any of the available broken line patterns as specified by LTYPE. If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless LTYPE $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(Inappropriate axis type: XX)¹
(Invalid axis type: XX)¹
```

### Default

CALL AXGRID(CHAXIS,0,-999) restores the default.

### See also

AXIS7, AXLAB7, AXMAJ, AXMIN, AXTXT7 and BUNLPR.

## SUBROUTINE AXIS7 (CHAXIS, CAP)
## SUBROUTINE AXIS (CHAXIS, CAP, NCAP)

### Name

`AXIS7` – to draw an axis on the current picture.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `CAP` | `STRING` expression | Axis caption |
| `NCAP` | `INTEGER` expression | Number of characters in `CAP` |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | × |

### Description

`AXIS/AXIS7` draws [1] an axis over the current range, annotated [2] according to the scale which exists when the call occurs.

The axis caption, `CAP`, is drawn [2] in the centre of and parallel to the axis, on the same side of the axis as the axis annotation; the caption is ignored for polar angular axes (`CHAXIS='AP'`).

If `AXGRID` has been called to requests grids, grid lines are drawn at axis subdivisions.

A fuller description of the values of `CHAXIS` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`(Invalid axis type: XX)`[1]
`(Inappropriate axis type: XX)`[2]

### See also

`AXCLR`, `AXCRSS`, `AXGRID`, `AXLBAN`, `AXLBGP`, `AXLBJS`, `AXLBLV`, `AXLOCN`, `AXRNGE`, `AXSBDV`, `AXSBTK` and `AXSUBS`

## SUBROUTINE AXLAB7 (CHAXIS, W, CAP)
## SUBROUTINE AXLAB (CHAXIS, W, CAP, NCAP)

### Name

AXLAB7 – to draw a single axis annotation label and tick mark.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| W | REAL expression | Position along axis in units of the plotting scales |
| CAP | STRING expression | Annotation label |
| NCAP | INTEGER expression | Number of characters in CAP |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

By default, when an axis is drawn it is subdivided and annotated at regular intervals using automatically generated labels. If additional labels are required on an axis, each one can be drawn by a separate call of AXLAB/AXLAB7. A tick mark is drawn [1] at position $w$ along the axis, and labelled [2] with the caption, CAP.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Invalid axis type: *XX*)$^1$
(Inappropriate axis type: *XX*)$^2$

### See also

AXCLR, AXGRID, AXLBAN, AXLBJS, AXLBSL, AXSBTK and AXTXT7.

## SUBROUTINE AXLBAN (CHAXIS, ACHAR)

### Name

`AXLBAN` – to specify the style of annotation labels on an axis.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `ACHAR` | `CHARACTER*1` | Position of annotation |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | $\surd^U$ | $\surd^U$ | $\surd^U$ | × |

*U*: `'P'`receding, `'F'`ollowing and `'I'`nside unavailable

| `ACHAR` | *Style* |
|---|---|
| `'D'` | Default – towards the outside |
| `'F'` | Following axis (in direction of other axis) |
| `'I'` | Towards the inside of the picture |
| `'N'` | None – annotation labels omitted |
| `'O'` | Towards the outside of the picture |
| `'P'` | Preceding axis (in direction of other axis) |

### Description

By default, all axes are annotated towards the outside of the picture if there is room, or towards the inside otherwise. `AXLBAN` specifies the style of annotation labels according to the value of `ACHAR`.

If there is insufficient space for labels in the specified location, annotation is drawn elsewhere but is not omitted. If `ACHAR`=`'I'`, `'O'`, `'P'` or `'F'`, and the annotation request cannot be satisfied, a diagnostic is issued.

A fuller description of the values of `CHAXIS` and the interpretation of `'P'` and `'F'` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(AXLBAN not satisfied)²
(Invalid axis type: XX)¹
```

### Default

`CALL AXLBAN(CHAXIS,'D')` restores the default.

### See also

`AXIS7`, `AXLAB7`, `AXLBJS`, `AXSBTK` and `AXTXT7`.

## SUBROUTINE AXLBGP (CHAXIS, ILEVEL)

### Name

`AXLBGP` – to specify the level of axis annotation near an intersection.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `ILEVEL` | `INTEGER` expression | Level of annotation at intersection |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | × | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | √ |

| `ILEVEL` | *Effect* |
|---|---|
| 0 | All annotation drawn |
| 1 | Annotation omitted near intersection |

### Description

By default, annotation labels which could be ambiguous or coincident near the intersection of two axes, are omitted from both. `CALL AXLBGP(CHAXIS,0)` specifies that no gap is to be left in the annotation of the specified axis near an intersection.

`AXLBGP` must be called before an axis is drawn.

A fuller description of the values of `CHAXIS` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`(Inappropriate axis type:` *`XX`*`)`[1]
`(Invalid axis type:` *`XX`*`)`[1]

### Default

`CALL AXLBGP(CHAXIS,1)` restores the default.

### See also

`AXCRSS` and `AXIS7`.

## SUBROUTINE AXLBJS (CHAXIS, JCHAR)

### Name

AXLBJS – to specify the justification of axis annotation.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| JCHAR | CHARACTER*1 | Justification of annotation |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | × | × | × | √ | √ | × | √ | √ | × | × | × | × |

| JCHAR | *Justification of annotation* |
|---|---|
| 'B' | Between tick marks (centred) |
| 'C' | Centered on the tick mark |
| 'F' | Following the tick mark |
| 'P' | Preceding the tick mark |
| 'D' | Default position |

### Description

By default, axis annotation labels are drawn preceding the subdivision tick mark with which they are associated. AXLBJS specifies justification of annotation relative to the tick marks.

AXLBJS must be called before an annotation label is drawn (either as part of an axis) or before labels are drawn (using AXLAB7 or AXTXT7).

A fuller description of the possible values of CHAXIS and the interpretation of 'P' and 'F' is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)$^1$
(Invalid axis type: *XX*)$^1$

### Default

CALL AXLBJS(CHAXIS,'D') restores the default.

### See also

AXIS7, AXLAB7, AXLBAN, AXLBSL and AXTXT7.

## SUBROUTINE AXLBLV (CHAXIS, ILEV)

### Name

AXLBLV – to specify the level of additional annotation on some axes.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| ILEV | INTEGER expression | Level of annotation |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √$^{NL}$ | √$^{NL}$ | × | √ | × | × | × | × | × | × | × | × | × | × | × |

*NL*: non-linear scales only

| ILEV | *Non-linear Cartesian axes* | *Angular polar axis* |
|---|---|---|
| 0 | determined by scales | 30°,60°,120°,150°,210°,240°,300°,330° |
| 1 | powers of 10 only | plus 0°, 90°, 180°, 270° |
| 2 | 3 | plus 10° intervals |
| 3 | 2,5 | — |
| 4 | 1,2,3,4,5,6,7,8,9 | — |

### Description

By default, axes are subdivided and annotated at regular intervals but non-linear axes and angular polar axes may need additional axis annotation due to the small extent of scale displayed. AXLBLV specifies the level of additional labelling.

AXLBLV must be called before an non-linear axis is drawn.

SIMPLEPLOT may omit these labels due to lack of space.

The possible values of CHAXIS are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)[1]
(Invalid axis type: *XX*)[1]

### Default

CALL AXLBLV(CHAXIS,0) restores the default.

### See also

AXCLR, AXIS7, AXLAB7 and AXTXT7.

## SUBROUTINE AXLBSL (CHAXIS, SCHAR)

### Name

AXLBSL – to specify the slope of used-defined axis annotation.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| SCHAR | CHARACTER*1 | Direction of slope of labels |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | × | √ | √ | √ | √ | √ | √ | √ | × | × | × | × |

| SCHAR | *Direction of label* |
|---|---|
| 'H' | Horizontal |
| 'V' | Vertical |
| 'S' | Sloping |
| 'D' | Default (different for labels of different lengths) |

### Description

By default, user-defined axis annotation labels are drawn horizontally for short labels, but long labels are sloped by staggering the characters in an attempt to make efficient use of the limited space. Labels added to axes are considered to be long if they contain six or more characters but no escape sequences. Axis labels which contain active escape sequences are never sloped.

AXLBSL specifies their slope according to the value of SCHAR. AXLBSL must be called before user-defined annotation labels are drawn (using AXLAB7, AXTXT7 or BARCHT). Axis labels on an angular axis are always horizontal.

A fuller description of the possible values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)[1]
(Invalid axis type: *XX*)[1]

### Default

CALL AXLBSL(CHAXIS,'D') restores the default.

### See also

AXIS7, AXLAB7, AXLBAN, AXLBJS and AXTXT7.

## SUBROUTINE AXLBSP (CHAXIS, CHAR1, CHAR2)

### Name

AXLBSP – to specify separators between label components of *time-date* axis annotation.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| CHAR1 | CHARACTER*1 | Separator between first pair |
| CHAR2 | CHARACTER*1 | Separator between second pair |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

*Time-date* axis annotation labels are built from up to three components. By default, time components are separated by colons (*eg.* 12:00:00) and date components are separated by spaces (*eg.* 30 Mar 1990). AXLBSP specifies alternative single-character separators – CHAR1 between the first and second components, CHAR2 between the second and third components.

In addition to a user-specified character, there are two special codes for separators which can be specified for CHAR1 and/or CHAR2:

| *CHARi* | *Meaning* |
|---|---|
| 'N' | No separator, not even a space |
| 'D' | Default separator – ':' for Times, ' ' for Dates |

Separators should not be confused with the prefix and suffix characters which can be controlled independently by TIMFMT and TIMLDC.

AXLBSP has no effect on numerical axes; it only works after AXLBTP has requested *time-date* scales. A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)[1]
(Invalid axis type: *XX*)[1]

### Default

CALL AXLBSP(CHAXIS,'D','D') restores the default.

### See also

AX*, TIMFMT, TIMLDA and TIMLDC

## SUBROUTINE AXLBTM (CHAXIS, COMP1, COMP2, COMP3)

### Name

AXLBTM – to specify components for *time-date* axis annotation.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| COMP1 | CHARACTER*2 | Component for first label position |
| COMP2 | CHARACTER*2 | Component for second label position |
| COMP3 | CHARACTER*2 | Component for third label position |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

By default, SIMPLEPLOT selects components for *time-date* axis labels according to the range of the relevant scale: times are in descending order (*eg.* hour/minute/second) and dates in ascending order (*eg.* day/month/year). AXLBTM specifies the components to be used, in the required order.

For only two components, and set COMP3 to 'NL' or ' '. For only one component, use COMP1, set COMP2 to 'NL' or ' ', and COMP3 is also ignored.

AXLBTM has no effect on numerical axes; it only becomes active after AXLBTP has been called to request *time-date* scales.

A fuller description of the values of CHAXIS and the interpretations of COMP1, COMP2 and COMP3 is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *X*X)[1]
(Invalid axis type: *X*X)[1]
(AXLBTM cancelled: *X*X)[1]

### Default

CALL AXLBTM(CHAXIS,'DF',' ',' ') restores the default.

### See also

AX*.

## SUBROUTINE AXLBTP (CHAXIS, CHTYPE)

### Name

AXLBTP – to specify *numeric* or *time-date* axis annotation.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| CHTYPE | CHARACTER*1 | Annotation type – 'N'umeric,<br>'D'ate or 'T'ime |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

| CHTYPE | *Annotation type* |
|---|---|
| 'N' | Numeric (default) |
| 'D' | Date |
| 'T' | Time |

### Description

By default, SIMPLEPLOT labels an axis with numeric labels representing a continuous REAL scale. Encoded times and dates are similar to any other REAL values therefore, by default, axis annotation represents numerical values. AXLBTP specifies that scale values on the specified axis represent time or date based data according to the value of CHTYPE.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)[1]
(Invalid axis type: *XX*)[1]

### Default

CALL AXLBTP(CHAXIS,'N') restores the default.

### See also

AXIS7, AXLBSP and AXLBTM.

## SUBROUTINE AXLOCN (CHAXIS, LCHAR)

### Name

`AXLOCN` – to specify the location of an axis relative to the picture.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `LCHAR` | `CHARACTER*1` | Location of axis |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| `LCHAR` | *Location of axis* |
|---|---|
| `'P'` | Preceding the other axis |
| `'F'` | Following the other axis |
| `'D'` | Default |

### Description

By default, a Cartesian or radial axis is positioned such that it intersects the other axis at zero or at its lowest extreme; a $z$-axis precedes the $x$-axis and $y$-axis; similarly the numeric and labels axes on a bar chart precede one another. `AXLOCN` specifies the location of the axis with respect to the picture boundary.

`AXLOCN` must be called before an axis is drawn.

A fuller description of the values of `CHAXIS` and the interpretation of `'P'` and `'F'` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`(Invalid axis type: XX)`[1]

### Default

`CALL AXLOCN(CHAXIS,'D')` restores the default.

### See also

`AXIS7`, `AXCRSS` and `AXLBGP`.

## SUBROUTINE AXMAJ (CHAXIS, W)

### Name

AXMAJ – to draw a single major axis subdivision.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| W | REAL expression | Position along axis in units of the plotting scales |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

AXMAJ draws [1] a single major unannotated axis subdivision at position $w$ along the specified axis; the tick mark is smaller than those drawn with annotated subdivisions. If grids have been specified, a grid line [3] is drawn.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Invalid axis type: *XX*)[1]
(Inappropriate axis type: *XX*)[2]

### See also

AXGRID, AXMIN, AXSBDV, AXSBMN, AXSBTK and AXSUBS.

## SUBROUTINE AXMIN (CHAXIS, W)

### Name

AXMIN – to draw a single minor axis subdivision.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| W | REAL expression | Position along axis in units of the plotting scales |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

AXMIN draws [1] a single minor unannotated axis subdivision at position *w* along the specified axis; the tick mark is smaller than those drawn with annotated and major unannotated subdivisions. If grids have been specified, a grid line [4] is drawn.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Invalid axis type: *XX*)[1]
(Inappropriate axis type: *XX*)[2]

### See also

AXGRID, AXMAJ, AXSBDV, AXSBMN, AXSBTK and AXSUBS.

## SUBROUTINE AXRNGE (CHAXIS, START, STOP)

### Name

`AXRNGE` – to specify the subrange over which an axis is to be drawn.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `START` | `REAL` expression | Start of axis in units of scale |
| `STOP` | `REAL` expression | End of axis in units of scale |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | √ |

### Description

By default, when an axis is drawn it is drawn over the entire range of the scale. `AXRNGE` specifies an alternative subrange for the specified axis. `AXRNGE` must be called before the axis is drawn.

This subrange cannot extend the axis beyond the limits of the scale. If one end of the subrange lies beyond the limits of the scale, only part of the subrange is drawn; if the whole of the subrange is outside the limits of the scale, the default applies.

A fuller description of the values of `CHAXIS` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`(Inappropriate axis type: `*`XX`*`)`[1]
`(Invalid axis type: `*`XX`*`)`[1]

### Default

The default is restored by calling `AXRNGE` with a zero range (*ie.* `START`=`STOP`). For example, `CALL AXRNGE(CHAXIS,0.0,0.0)`.

### See also

`AXIS7` and `SCALES`.

## SUBROUTINE AXSBDV (CHAXIS, OFFSET, DELTA)

### Name

AXSBDV – to specify the interval at which a linear axis is to be subdivided and annotated.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| OFFSET | REAL expression | Offset value for subdivisions |
| DELTA | REAL expression | Interval between subdivisions of axis in units of the scale |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √$^{L}$ | √$^{L}$ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | √ |

*L*: Linear scales only

### Description

By default, linear axes are subdivided at equal intervals chosen for clarity to give between 5 and 10 subdivisions at multiples of a 'simple' number. AXSBDV specifies the annotation interval and an offset from which divisions are calculated. AXSBDV must be called before an axis is drawn.

If DELTA is less than $1.0E-4\times$ axis range, the default interval is used instead and, at diagnostic level 2, a diagnostic message is issued (see below).

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(Inappropriate axis type: XX)¹
(Invalid axis type: XX)¹
(Too many axis intervals: default used)¹
```

### Default

CALL AXSBDV(CHAXIS,0.0,0.0) restores the default.

### See also

AXIS7 and KSCALE.

## SUBROUTINE AXSBMN (CHAXIS, OFFSET, DELTA)

### Name

`AXSBMN` – to specify the minor subdivisions on a linear axes independently of major subdivisions.

**Availability** Section 4, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `OFFSET` | `REAL` expression | Offset value for minor subdivisions in units of the plotting scale |
| `DELTA` | `REAL` expression | Interval between minor subdivisions in units of the plotting scale |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | $\surd^L$ | $\surd^L$ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | √ |

*L*: Linear scales only

### Description

By default, linear axes are drawn with no minor subdivisions but they can be requested in one of two ways. `AXSUBS` requests minor subdivisions which divide major subdivisions into a specified number of equal parts; `AXSBMN` specifies an offset value and interval independently of the major subdivisions. Minor axis subdivisions are indicated by small tick marks and, when grids of type 2 have been requested, grid lines are also drawn.

`AXSBMN` and `AXSUBS` provide alternative methods and therefore the most recent call of either cancels any earlier call.

A fuller description of the values of `CHAXIS` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(Inappropriate axis type: XX)¹
(Invalid axis type: XX)¹
```

### Default

`CALL AXSBMN(CHAXIS,0.0,0.0)` or `CALL AXSUBS(CHAXIS,0,0)` restores the default.

### See also

`AXGRID`, `AXIS7`, `AXMAJ`, `AXMIN`, `AXSBDV`, `AXSBTK` and `AXSUBS`.

## SUBROUTINE AXSBTK (CHAXIS, TCHAR)

### Name

`AXSBTK` – to specify the style of tick marks on an axis.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHAXIS` | `CHARACTER*2` | Axis type |
| `TCHAR` | `CHARACTER*1` | Direction of tick marks |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `CHAXIS` | `XC` | `YC` | `RP` | `AP` | `XI` | `YI` | `ZI` | `NB` | `LB` | `NW` | `LW` | `X3` | `Y3` | `Z3` | `U3` |
| | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | $\surd^{U}$ | $\surd^{U}$ | $\surd^{U}$ | $\surd^{U}$ |

*U*: `'P'`receding, `'F'`ollowing and `'I'`nside unavailable

| `TCHAR` | *Meaning* |
|---|---|
| `'A'` | Straddling across the axis |
| `'D'` | Default – towards the outside |
| `'F'` | Following axis |
| `'I'` | Towards the inside of the picture |
| `'N'` | None – tick marks omitted |
| `'O'` | Towards the outside of the picture |
| `'P'` | Preceding axis |

### Description

By default, tick marks are drawn towards the outside of the picture if there is room, or towards the inside otherwise. `AXSBTK` specifies a new style, and must be called before a tick mark is drawn as part of a composite task, or using `AXMAJ`, `AXMIN`, `AXLAB7` or `AXTXT7`.

If there is insufficient space for tick marks in the specified location, they are drawn elsewhere. If `TCHAR` cannot be satisfied, a diagnostic is issued. A fuller description of the values of `CHAXIS` and the interpretation of `'P'` and `'F'` is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`(AXSBTK not satisfied)`$^{2}$
`(Invalid axis type:` *XX*`)`$^{1}$

### Default

`CALL AXSBTK(CHAXIS,'D')` restores the default.

### See also

`AXIS7`, `AXLAB7`, `AXMAJ`, `AXMIN` and `AXTXT7`.

## SUBROUTINE AXSUBS (CHAXIS, NMAJOR, NMINOR)

### Name

AXSUBS – to specify the numbers of major and minor subdivisions on linear axes.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| NMAJOR | INTEGER expression | Number of major subdivisions |
| NMINOR | INTEGER expression | Number of minor subdivisions |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √$^L$ | √$^L$ | √ | × | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

*L*: Linear scales only

### Description

By default, all linear axes are drawn with no minor subdivisions, and major subdivisions are only used when the space available is too small to fit annotation labels at all tick marks. AXSUBS specifies the intervals between annotated tick marks. These intervals are subdivided into NMAJOR equal major intervals which are each subdivided into NMINOR equal minor intervals. Major tick marks are smaller than annotated tick marks.

AXSUBS must be called before a linear axis is drawn and minor tick marks are smaller than major tick marks.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Inappropriate axis type: *XX*)[1]
(Invalid axis type: *XX*)[1]

### Default

CALL AXSUBS(CHAXIS,0,0) restores the default.

### See also

AXGRID, AXIS7, AXMAJ, AXMIN, AXSBDV and AXSBTK.

## SUBROUTINE AXTXT7 (CHAXIS, W1, STEP, LABARR, NARR)
## SUBROUTINE AXTXT (CHAXIS, W1, STEP, LABARR, NARR, NCAP)

### Name

AXTXT7 – to draw a set of axis annotation labels and tick marks.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| W1 | REAL expression | Position for first label |
| STEP | REAL expression | Interval between labels |
| LABARR | STRING array | Set of labels |
| NARR | INTEGER expression | Number of labels in LABARR |
| NCAP | INTEGER expression | Maximum number of characters in any label |

| *axis type* | Cartesian | | Polar | | Isometric | | | Bars | | Water | | ViSualization | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHAXIS | XC | YC | RP | AP | XI | YI | ZI | NB | LB | NW | LW | X3 | Y3 | Z3 | U3 |
| | √ | √ | √ | √ | √ | √ | √ | √ | × | √ | √ | √ | √ | √ | × |

### Description

By default, when an axis is drawn it is subdivided and annotated at regular intervals using automatically generated labels. To draw an axis with user-defined labels, suppress the normal annotation by calling AXCLR before the axis is drawn then call AXTXT/AXTXT7 to draw alternative annotation and tick marks. LABARR(1) is positioned at W1, LABARR(2) is positioned at W1+STEP *etc.*

LABARR must be declared with elements large enough for the longest label in the set, *eg.* CHARACTER*9 LABARR(7) for a set of 7 labels of up to 9 characters each.

A fuller description of the values of CHAXIS is given in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Invalid axis type: *XX*)[1]
(Inappropriate axis type: *XX*)[2]

### See also

AXGRID, AXCLR, AXIS7, AXLAB7, AXLBAN, AXLBJS, AXLBSL and AXSBTK.

## SUBROUTINE BARCHT (D2ARR, LABARR, NARR, NSETS, CAPN, CAPL)

### Name

BARCHT – to draw a bar chart on the current picture.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| D2ARR | REAL 2-D array | Data values, D2ARR(NARR,NSETS) |
| LABARR | STRING array | Names of items in each data set |
| NARR | INTEGER expression | Number of elements in each data set and in LABARR |
| NSETS | INTEGER expression | Number of data sets |
| CAPN | STRING expression | Numeric axis caption |
| CAPL | STRING expression | Label axis caption |

### Description

BARCHT draws a bar chart on the current picture of data held in the two-dimensional array, D2ARR(NARR,NMAX), where NMAX is greater or equal to the number of data sets to be plotted (NSETS).

BARCHT draws numeric and label axes which can be configured using appropriate AX* subroutines with CHAXIS='NB' and 'LB' respectively. By default, the bars are drawn vertically and, if NSETS > 1, bars are clustered side-by-side to represent values from each data set; bars (or clusters of bars) occupy 80% of the space available and are centred within this space; the numeric scale is calculated to accommodate the bar chart exactly and both scales run in the natural direction of the corresponding axis. These default settings can be altered by BARTYP, BARFMT, BARRNG and BARDIR respectively.

BARCHT draws a bar chart on the current 2-D picture and must therefore be preceded by the *new picture* subroutine, NEWPIC.

### Diagnostics

(BAR CHART OMITTED: INVALID DIMENSIONS)[1]

### See also

BARRNG, BARFMT, BARTYP, BARDIR, SHEDGE, SHKEYS and SQSHAD.

## SUBROUTINE BARDIR (NBTORF, LBTORF)

### Name

`BARDIR` – to specify the directions of bar chart scales.

**Availability** Section 4, released version 2-11.

### Arguments

|  |  |  |
|---|---|---|
| `NBTORF` | `LOGICAL` expression | Whether to reverse numeric scale |
| `LBTORF` | `LOGICAL` expression | Whether to reverse labels scale |

### Description

By default, the scales on a bar chart are calculated to fit the data exactly and are configured such that scales go from a small value to a larger value. `BARDIR` can set any combination of scale directions for bar charts by indicating whether to reverse the numeric scale and/or the implicit label scale; the range of the numeric scale can be specified by `BARRNG`.

### Default

`CALL BARDIR(.FALSE.,.FALSE.)` restores the default.

### See also

`BARCHT` and `BARRNG`.

## SUBROUTINE BARFMT (JCHAR, IVAL)

**Name**

`BARFMT` – to specify how bars in a bar chart occupy the space available.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `JCHAR` | `CHARACTER*1` | Justification of bars |
| `IVAL` | `INTEGER` expression | Percentage (1–100%) of space available |

| `JCHAR` | *Justification of bars* |
|---|---|
| `'P'` | Preceding the space – left for vertical bars, bottom for horizontal bars |
| `'C'` | Centred within the space |
| `'F'` | Following the space – right for vertical bars, top for horizontal bars |

**Description**

By default, bars (or clusters of bars) occupy 80% of the space available and are centred within this space. `BARFMT` specifies the size and justification of bars on a bar chart.

**Default**

`CALL BARFMT('C',0)` restores the default.

**See also**

`BARCHT`.

## SUBROUTINE BARRNG (VMIN, VMAX)

### Name

`BARRNG` – to specify the range of the numerical scale for bar charts.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `VMIN` | `REAL` expression | Minimum scale value |
| `VMAX` | `REAL` expression | Maximum scale value |

### Description

By default, the numerical scale of bar charts is calculated to accommodate the data exactly. `BARRNG` specifies alternative range of scales (the direction of scales can be specified separately by `BARDIR`.)

If the specified scale can not accommodate the entire bar chart, part of the picture cannot be drawn. This contributes to the number of incomplete picture tasks and a diagnostic is issued at the end of the picture.

### Default

Calling `BARRNG` with `VMAX` less than or equal to `VMIN` restores the default. *eg.* `CALL BARRNG(0.0,0.0)`.

### See also

`BARCHT` and `BARDIR`.

## SUBROUTINE BARTYP (OCHAR, ITYPE)

**Name**

BARTYP – to specify the type of bar chart required.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| OCHAR | CHARACTER*1 | Orientation of bars |
| ITYPE | INTEGER expression | Type of bar chart |

| OCHAR | *Orientation of bars* |
|---|---|
| 'V' | Vertical bars |
| 'H' | Horizontal bars |

| ITYPE | *Type of bar chart* |
|---|---|
| 1 | Clustered bars – data sets side-by-side |
| 2 | Stacked bars – data sets accumulated on same bars |
| 3 | Pyramid bars – data sets alternating |

**Description**

By default, bar charts drawn using BARCHT have vertical bars which, if there is more than one data set, are clustered side-by-side. The type of bar chart drawn by BARCHT can be changed by a prior call of BARTYP.

Orientation can be either horizontal or vertical. If horizontal bars are drawn, the bars and their associated labels are ordered from the top to the bottom of the picture.

**Default**

CALL BARTYP('V',1) restores the default.

**See also**

BARCHT.

## SUBROUTINE BLNKKY

### Name

`BLNKKY` – to add a blank entry in a key or caption area.

**Availability** Section 1, released before version 2-5.

### Arguments

None.

### Description

`BLNKKY` inserts a blank entry in the next empty space in the current key or caption area (defined by `DEFKEY`/`DEFCAP` or selected by `RESNXT`). All blank entries in a key/caption area are counted towards the total number of entries.

A similar effect can be achieved by increasing the interline spacing (see `TEXTLF`) or by using embedded sub-commands within text strings (see `STRLV` and `STRCHS`).

### Diagnostics

`(Key/caption area full)`$^2$
`(No active key/caption area)`$^2$

### See also

`BOTHK7`, `DEFCAP`, `DEFKEY`, `LINEK7`, `MARKK7`, `RESNXT`, `SHDEK7` and `FILLK7`.

## SUBROUTINE BOTHK7 (LTYPE, MKTYPE, CAP)
## SUBROUTINE BOTHKY (LTYPE, MKTYPE, CAP, NCAP)

### Name

BOTHK7 – to draw an annotated sample of a broken line pattern and marker symbol in a key.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| LTYPE | INTEGER expression | Type of broken line pattern |
| MKTYPE | INTEGER expression | Type of marker symbol |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

BOTHKY/BOTHK7 inserts a key to a broken line pattern (as used by the BRKN* subroutines) with a marker symbol (as used by MARK*) in the next empty space in the current key or caption area (defined by DEFKEY/DEFCAP or selected by RESNXT).

A sample of the broken line pattern, LTYPE, is drawn [1] on the left of the entry, with the marker symbol, MKTYPE, drawn [3] on the centre of the line; the caption, CAP, is drawn [2] alongside the sample. If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless

LTYPE $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)[2]
(Key/caption area full)[2]
(No active key/caption area)[2]

### See also

ADDJST, BLNKKY, BUNLPR, DEFKEY, LINEK7, LINSET, MARKK7 and MKSET.

## SUBROUTINE BOXCAP (TORF)

### Name

`BOXCAP` – to specify whether boxes are to be drawn around caption areas.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether captions are to be boxed |

### Description

By default, caption areas defined by `DEFCAP` are not boxed. `BOXCAP` specifies whether or not each caption area is to be surrounded by a box.

The box is drawn [1] when the caption area is set up; when this occurs depends on the type of caption area defined by `CALL DEFCAP(ITYPE,...)`:

| `ITYPE` | *Box drawn* |
|---|---|
| 1 | when a new caption area is defined by `DEFCAP` |
| 2 | when a new caption area is defined by `DEFCAP` |
| 3 | whenever a new picture is started |

### Default

`CALL BOXCAP(.FALSE.)` restores the default.

### See also

`DEFCAP`.

### SUBROUTINE BOXGRP (TORF)

**Name**

`BOXGRP` – to specify whether boxes are to be drawn around groups of pictures.

**Availability** Section 4, released version 2-7.

**Argument**

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether groups are to be boxed |

**Description**

By default, a group of pictures is not boxed. `BOXGRP` specifies whether or not a rectangular box is to be drawn around each group of pictures.

The area surrounded by the box is the group of pictures as defined by `GROUP` including margins but excluding peripheries. If `GROUP` has not been called (or a 1 × 1 group has been defined), the box surrounds a single picture and its margin.

The box is drawn [1] when the first picture on the SIMPLEPLOT page is started.

**Default**

`CALL BOXGRP(.FALSE.)` restores the default.

**See also**

`BOXPAG`, `BOXPIC` and `GROUP`.

## SUBROUTINE BOXKY (TORF)

### Name

`BOXKY` – to specify whether boxes are to be drawn around keys.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether keys are to be boxed |

### Description

By default, a box is drawn around all key areas. `BOXKY` specifies whether or not each key area is to be boxed.

The box is drawn [1] when the key is set up; when this occurs depends on the type of key defined by `CALL DEFKEY(ITYPE, ...)`:

| `ITYPE` | *Box drawn* |
|---|---|
| 1 | when a new key is defined by `DEFKEY` |
| 2 | when a new key is defined by `DEFKEY` |
| 3 | whenever a new picture is started |

`MPK7H`, `MPK7V` and `SHKEYS` correspond to `ITYPE`=2.

### Default

`CALL BOXKY(.TRUE.)` restores the default.

### See also

`DEFKEY`, `MPK7H`, `MPK7V` and `SHKEYS`.

### SUBROUTINE BOXPG2 (ITYPE)

**Name**

`BOXPG2` – to specify how the boundaries of pages are to be drawn.

**Availability** Section 1, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `ITYPE` | `INTEGER` expression | Type of boundary (see below) |

| `ITYPE` | *Boundary type* |
|---|---|
| 0 | No drawn boundary (Default) |
| 1 | Simple box or parallel line boundary |
| 2 | Alternative box with logo |

**Description**

By default, the boundary of the Simpleplot page is not drawn. `BOXPG2` specifies what type of page boundary is required on each Simpleplot page.

The box [1], and optionally a logo [2] (with `BOXPG2(2)`), are drawn when the Simpleplot page is started.

**Default**

`BOXPG2(0)` restores the default.

**Diagnostics**

`[Failed to read LOGO file]`$^1$

**See also**

`BOXGRP`, `BOXPIC` and `PAGE`.

## SUBROUTINE BOXPIC (TORF)

### Name

`BOXPIC` – to specify whether boxes are to be drawn around individual pictures.

**Availability** Section 4, released version 2-7.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether pictures are to be boxed |

### Description

By default, pictures are not boxed. `BOXPIC` specifies whether or not a rectangular box is to be drawn around every picture (excluding margin).

The box is drawn [1] when a new picture is started.

If scales are extended to provide similar linear scales in $x$ and $y$ (see `EQSCAL`), the box is drawn around the full extent of the scales at the picture limits. If a box is required to fit the requested range of scales, and the axes drawn to annotate such scales, `DRAWBX` can be used to draw a box on the current picture.

### Default

`CALL BOXPIC(.FALSE.)` restores the default.

### See also

`BOXGRP`, `BOXPAG`, `DRAWBX` and `PERIM`.

## SUBROUTINE BREAK

### Name

`BREAK` – to force a break between separate sequences of point-by-point plotting.

**Availability** Section 1, released before version 2-5.

### Arguments

None.

### Description

A sequence of lines drawn by `BRKNPT/JOINPT` is interrupted when any other plotting subroutine is called, and the next call of `BRKNPT/JOINPT` is treated as a first call again. `BREAK` explicitly interrupts the sequence without performing any other function. `BREAK` does no drawing.

`MARKPT` (or `CP7PT/CAPPT`) includes the equivalent of `BRKNPT/JOINPT` after drawing the symbol (and text), therefore `BREAK` should be called between `MARKPT` and a succeeding `BRKNPT/JOINPT` if a line is not required joining symbol and point.

### See also

`BRKNPT`, `CP7PT`, `JOINPT` and `MARKPT`.

**SUBROUTINE BRKNAR (X1, DX, DARR, NARR, LTYPE)**

### Name

`BRKNAR` – to draw a curve from an array of values, using a specified broken line pattern.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `X1` | `REAL` expression | Value of independent variable for plotting `DARR(1)` |
| `DX` | `REAL` expression | Independent variable interval between function values |
| `DARR` | `REAL` array | Function values for plotting |
| `NARR` | `INTEGER` expression | Number of points |
| `LTYPE` | `INTEGER` expression | Type of broken line pattern |

### Description

Using broken line pattern `LTYPE`, `BRKNAR` draws [1] the curve which connects the following points:

```
(X1,DARR(1)), (X1+DX,DARR(2)),...
(X1+(NARR-1)*DX,DARR(NARR))
```

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ and $x_i = x_1 + (i-1)dx$. The type of curve depends on `CVTYPE`.

If line style has been requested as a bundled attribute, `LTYPE` overrides the line style of the bundle (but not colour or thickness) unless `LTYPE` $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Data curve exceeds scales)`[2]

### See also

`BRKNCV`, `BUNLPR`, `CVTYPE`, `CLIPLV`, `MARKAR`, `RIDDLE` and `SHDEAR`.

## SUBROUTINE BRKNBX (X1, Y1, X2, Y2, LTYPE)

### Name

BRKNBX – to draw a box, using a specified broken line pattern.

**Availability** Section 1, released version 2-8.

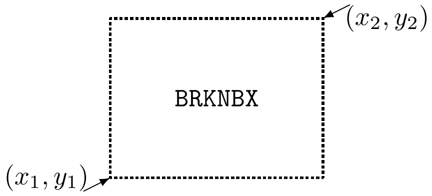
### Arguments

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of opposite corners of the box, specified in units of the plotting scales |
| LTYPE | INTEGER expression | Type of broken line pattern |

### Description

Using broken line pattern LTYPE, BRKNBX draws [1] the box extending from $x_1$ to $x_2$ by $y_1$ to $y_2$. If $x_1$=$x_2$ or $y_1$=$y_2$, then nothing is drawn.

The points $(x_1, y_1)$ and $(x_2, y_2)$ are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, KXYXY.



If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless LTYPE = −999 which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(**$x_1, y_1, x_2, y_2$**)$^2$
(Null box, nothing drawn)$^2$

### See also

BUNLPR, CLIPLV and SHDEBX.

## SUBROUTINE BRKNCL (XCENT, YCENT, RADIUS, LTYPE)

### Name

`BRKNCL` – to draw a circular function with specified centre, radius and broken line pattern.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `XCENT`,<br>`YCENT` | `REAL` expressions | Coordinates of the centre of the circle in units of the plotting scales |
| `RADIUS` | `REAL` expression | Circle radius in plotting scale units |
| `LTYPE` | `INTEGER` expression | Type of broken line pattern |

### Description

Using broken line pattern, `LTYPE`, `BRKNCL` draws [1] the circle (or the part of the circle which lies within the range of the plotting scales) centred at point $(x_{cent}, y_{cent})$, with radius `RADIUS`. The resulting shape is only geometrically circular when both scales are equal and linear.

The centre point and the radius are both specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, `KXYXY`.

If line style has been requested as a bundled attribute, `LTYPE` overrides the line style of the bundle (but not colour or thickness) unless

`LTYPE` $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if it exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Circle exceeds scales)`$^2$

### See also

`BUNLPR`, `CLIPLV`, `EQSCAL` and `SHDECL`.

## SUBROUTINE BRKNCV (XARR, YARR, NARR, LTYPE)

### Name

`BRKNCV` – to draw a curve from a set of coordinates, using a specified broken line pattern.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `XARR`, `YARR` | `REAL` arrays | Coordinates of points, in units of the plotting scales |
| `NARR` | `INTEGER` expression | Number of points |
| `LTYPE` | `INTEGER` expression | Type of broken line pattern |

### Description

Using broken line pattern `LTYPE`, `BRKNCV` draws [1] the curve corresponding to the following points:

(`XARR(1)`, `YARR(1)`), (`XARR(2)`, `YARR(2)`), ...
(`XARR(NARR)`, `YARR(NARR)`)

A closed curve can be obtained by using data in which `XARR(NARR)`=`XARR(1)` and `YARR(NARR)`= `YARR(1)`. The type of curve drawn depends on `CVTYPE`.

If line style has been requested as a bundled attribute, `LTYPE` overrides the line style of the bundle (but not colour or thickness) unless

`LTYPE` = −999 which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Data curve exceeds scales)`[2]

### See also

`BUNLPR`, `CLIPLV`, `CVTYPE`, `MARKCV`, `RIDDLE` and `SHDECV`.

### SUBROUTINE BRKNFN (FUNX, LTYPE)

**Name**

BRKNFN – to draw a curve of a user-defined function, $y = f(x)$, using the specified broken line pattern.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| FUNX | function name | REAL function with one REAL argument, also declared in an EXTERNAL statement |
| LTYPE | INTEGER expression | Type of broken line pattern |

**Description**

BRKNFN evaluates and draws [1] the curve of the specified function, FUNX, using the specified broken line pattern, LTYPE.

With Cartesian coordinates, the specified function, FUNX, must calculate $y$ given $x$, and with polar coordinates FUNX must calculate $r$ given $\theta$. The intervals of the independent variable over which BRKNFN evaluates the function, vary with the curvature of the curve.

Care must be taken in the programming of FUNX to ensure that it is continuous and cannot fail for any arguments within the plotting range. Discontinuities may cause SIMPLEPLOT to loop indefinitely.

If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless

LTYPE $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Function curve exceeds scales)[2]

**See also**

BUNLPR, CLIPLV, FNRNGE, POLY, RIDDLE and SHDEFN.

### SUBROUTINE BRKNPR (SUBR, T1, T2, LTYPE)

**Name**

BRKNPR – to draw the user-defined parametric function using a specified broken line pattern.

**Availability** Section 4, released version 2-9.

**Arguments**

| | | |
|---|---|---|
| SUBR | subroutine name | Subroutine with three REAL arguments, also declared in an EXTERNAL statement |
| T1, T2 | REAL expressions | Values of the function parameter between which curve is to be drawn |
| LTYPE | INTEGER expression | Type of broken line pattern |

**Description**

BRKNPR evaluates and draws [1] the curve of a user-defined parametric function using the specified broken line pattern, LTYPE.

The parametric function is specified in subroutine SUBR; SUBR must have three REAL arguments: T, the value of the function argument, and VARX and VARY, to receive coordinates corresponding to T. SUBR is called by BRKNPR with values of T ranging from T1 to T2.

If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless

LTYPE = −999 which indicates that all the bundled line-drawing attributes of the current pen are to be used.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Parametric function exceeds scales)$^2$
(No. of incomplete picture tasks=*n*)$^1$

**See also**

BUNLPR, CLIPLV, DRAWPR, RIDDLE and SHDEPR.

## SUBROUTINE BRKNPT (X, Y, LTYPE)

### Name

BRKNPT – to draw a broken line to the point $(x, y)$.

**Availability** Section 1, released version 2-9.

### Arguments

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point specified in units of the plotting scale |
| LTYPE | INTEGER expression | Type of broken line pattern |

### Description

The first call of BRKNPT executes a move to the point $(x, y)$ without drawing anything. Subsequent calls of BRKNPT draw [1] a straight line from the previous point to the specified point, using broken line pattern LTYPE. The line is straight even on non-linear scales and polar charts.

A sequence of lines drawn by BRKNPT/JOINPT is interrupted when any other plotting subroutine is called, and the next call of BRKNPT/JOINPT is treated as a first call again. BREAK explicitly interrupts the sequence without performing any other function. MARKPT (or CP7PT/CAPPT) draws a symbol (and a caption) and then executes the equivalent to a first call of BRKNPT/JOINPT in order to move to the centre of the symbol.

If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless LTYPE $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

The point $(x, y)$ is specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, KXYXY.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(**$x, y$**)$^2$

### See also

BREAK, BUNLPR, CLIPLV, CP7PT, JOINPT and MARKPT.

## SUBROUTINE BUNLPR (CHATTR)

### Name

BUNLPR – to specify the order of priority of bundled line-drawing attributes.

**Availability** Section 1, released version 2-13.

### Argument

CHATTR    CHARACTER*3    Attributes to be bundled

| CHATTR | *Attribute* | *Cyclic order* |
|---|---|---|
| 'C' | Colour | 1, 2, 3, . . . NPENS |
| 'S' | Style | 0, −1, −2 . . . , 1, 2 . . . |
| 'T' | Thickness | 1, 2, 3, 4 |

### Description

By default, SIMPLEPLOT associates only one attribute, colour, with pen numbers. BUNLPR may be called to specify which attributes are to be combined to distinguish pens used for line drawing; text, marker symbols and shading are not affected.

CHATTR can consist of the initials (upper- or lower-case) of one, two or three attributes in their order of priority.

The total number of attribute combinations in the current bundle is device dependent; the effect of selecting a pen number beyond the total is undefined.

Line Styles specified explicitly as arguments (*eg.* . LTYPE) override the bundled Style without affecting the other bundled attributes. The bundled Style may be specified by LTYPE=-999.

### Default

CALL BUNLPR('C') restores the default.

### See also

PEN, SETPNS, SQPEN, THCKMG and WFPNS.

### SUBROUTINE CHDESC (IHGHT, ITAIL, ITHCK, ITILT)

**Name**

`CHDESC` – to specify characteristics of adjustable software characters.

**Availability** Section 4, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `IHGHT` | `INTEGER` expression | Height of characters relative to width |
| `ITAIL` | `INTEGER` expression | Size of serifs |
| `ITHCK` | `INTEGER` expression | Character line thickness |
| `ITILT` | `INTEGER` expression | Slope of characters |

| | −1 | 0 (*default*) | 1 |
|---|---|---|---|
| `IHGHT` | Squat characters | Normal height | Tall characters |
| `ITAIL` | No serifs | Small serifs | Large serifs |
| `ITHCK` | Single line characters | Lightly thickened characters | Thick characters |
| `ITILT` | Backwards slope | Upright | Forwards slope |

**Description**

`CHDESC` specifies a preferred combination of characteristics of the adjustable software characters selected by `CHSET` with `ISET`=51 or 52. The relationships between argument values and the characteristics are summarized in the table below.

`CHDESC` has no effect on hardware or other software characters (*eg.* Hershey fonts).

**Default**

`CALL CHDESC(0,0,0,0)` restores the default.

**See also**

`CHSET`.

## SUBROUTINE CHMASK (TORF)

**Name**

`CHMASK` – to specify whether text is to be masked.

**Availability** Section 4, released version 2-9.

**Argument**

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether text is to be masked |

**Description**

By default, SIMPLEPLOT draws anywhere within the page unless specific areas have been reserved for specific activities. `CHMASK` specifies that the area immediately around text is to be masked from any further drawing.

After a call of `CHMASK(.TRUE.)` text is masked; this means that the text itself looks no different from unmasked text, but any subsequent drawing over the text is omitted as it crosses the text.

Masking of text is currently limited to horizontal and vertical text only.

The limit to the number of masked areas which can be used for individual symbols and text strings is 50. When this limit is reached a diagnostic is issued and subsequent masked areas take the place of the oldest areas and overdrawing of these areas may occur.

**Diagnostics**

`(Maximum no. of masked areas reached)`[1]

**Default**

`CALL CHMASK(.FALSE.)` restores the default.

**See also**

`MKMASK`.

## SUBROUTINE CHSET (ISET)

### Name

`CHSET` – to specify an alternative source of character sets or fonts.

**Availability** Section 1, released version 2-8.

### Argument

`ISET` `INTEGER` expression Character set source

| `ISET` | *Font source* |
|---|---|
| $\leq -1$ | Alternative hardware/firmware sets (if available) |
| 0 | Hardware/firmware font (if available) |
| 1 | Simple device independent software font |
| 2–19 | Hershey fonts (Simpleplot-Plus) |
| 20 | Solid font (Simpleplot-Plus) |
| 21 | Outline font (Simpleplot-Plus) |
| 22 | Mathematical symbols (Simpleplot-Plus) |
| 23 | Large mathematical symbols (Simpleplot-Plus) |
| 51 | Adjustable ANSI characters (Section 4) |
| 52 | Adjustable UK characters (Section 4) |

### Description

For efficiency and clarity, Simpleplot is set up to draw text on pictures using characters from device hardware or firmware wherever possible. `CHSET` selects an alternative character set (or font). The Hershey fonts are only available with Simpleplot-Plus.

### Default

`CALL CHSET(0)` restores the default.

### See also

`FFSET`, `LINSET`, `MKSET` and `SHSET`.

## SUBROUTINE CHUNDL (NLINES)

### Name

`CHUNDL` – to specify the number of times text is underlined.

**Availability** Section 4, released version 2-9.

### Argument

| | | |
|---|---|---|
| `NLINES` | `INTEGER` expression | Number of lines under text (0–3) |

### Description

After a call of `CHUNDL`, every time text is drawn, `NLINES` lines are drawn underneath. `CHUNDL` is currently limited to horizontal text.

Single underlining of text can also be achieved by using embedded sub-commands within the text string (see `STRCHS` and `STRLV`).

### Default

`CALL CHUNDL(0)` restores the default.

### See also

`STRCHS` and `STRLV`.

## SUBROUTINE CLIPLV (ILEVEL)

### Name

`CLIPLV` – to specify the clipping region for all plotting.

**Availability** Section plus, released version 2-12.

### Argument

`ILEVEL` `INTEGER` expression Level of clipping

| `ILEVEL` | *Clipping region* |
|---|---|
| 0 | Default (see above) |
| 1 | Picture limits |
| 2 | SIMPLEPLOT page limits |

### Description

By default, plotting is clipped as follows:

- Text (including axis annotation and tick marks) is clipped to limits of SIMPLEPLOT page;
- Arrows are clipped to picture limits;
- Boxes are clipped to picture limits;
- Plotting of lines and shaded areas is clipped to picture limits;
- Marker symbols are plotted only when the point on which symbol is centred is on, or inside, the picture limits; in a similar way, range bars can spill over the picture limits.

`CLIPLV` specifies the all plotting is to be clipped to the specified limits.

### Diagnostics

The following diagnostics may be produced by graphics subroutines that draw text and/or marker symbols:

`(No. of incomplete page tasks=`*n*`)`[1]
`(Symbol clipped)`[2]
`(Text clipped)`[2]

### Default

`CALL CLIPLV(0)` restores the default.

## SUBROUTINE COORDS (IUNITS)

### Name

`COORDS` – to change the interpretation of coordinates.

**Availability** Section 1, released version 2-12.

### Argument

`IUNITS` `INTEGER` expression Type of units

| `IUNITS` | *Coordinate interpretation* |
|---|---|
| 0 | Cartesian $(x, y)$ |
| 1 | polar, $z = f(r, \theta)$, $\theta$ in degrees |
| 2 | polar, $z = f(r, \theta)$, $\theta$ in radians |
| 3 | polar – user-defined angular scale (see `POLRNG`) |

### Description

`COORDS` specifies coordinate interpretation as Cartesian, polar (in degrees) or polar (in radians). This interpretation remains in place until reset by `COORDS` or `EQSCAL`.

If `EQSCAL` has not been called, `COORDS` controls the type of the next new picture started by `NEWPIC` or `AXES7`. If `EQSCAL` is active, each new picture is started with the units specified by `EQSCAL` regardless of any call of `COORDS`; once a picture has been started, `COORDS` changes coordinate interpretation.

`EQSCAL` overrides `COORDS` but `COORDS` only modifies `EQSCAL` (saved scale ranges are converted).

### Default

The default can be restored by `CALL COORDS(-1)`.

### See also

`AXES7`, `EQSCAL`, `NEWPIC`, `POLAR7` and `POLRNG`.

## SUBROUTINE CP7LB (X, Y, CAP)<br>SUBROUTINE CAPLB (X, Y, CAP, NCAP)

### Name

`CP7LB` – to draw a caption at $(x, y)$.

**Availability** Section 4, released version 2-6.

### Arguments

| | | |
|---|---|---|
| `X, Y` | `REAL` expressions | Coordinates of a point, specified in units of the plotting scales |
| `CAP` | `STRING` expression | Caption |
| `NCAP` | `INTEGER` expression | Number of characters in `CAP` |

### Description

`CP7LB/CAPLB` draws [2] the caption, `CAP`, at the point $(x, y)$.

The point $(x, y)$ is specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine `KXYXY`. `REAL` and `INTEGER` values can be converted to text strings using the conversion subroutines `KREAL` and `KNUMB` respectively.

By default, if the caption is too long to fit within the Simpleplot page, it is clipped such that only whole characters appear; if coordinates exceed the limits of the Simpleplot page, the caption is omitted. Both these occurrences contribute to the number of incomplete page tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Text clipped)`$^2$

### See also

`CLIPLV`, `CP7PT`, `CP7XC`, `CP7YC`, `LABJST`, `LABANG`, `KNUMB` and `KREAL`.

## SUBROUTINE CP7PT (X, Y, MKTYPE, CAP)
## SUBROUTINE CAPPT (X, Y, MKTYPE, CAP, NCAP)

### Name

CP7PT – to draw a marker symbol at $(x, y)$ with a caption.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point specified in units of the plotting scales |
| MKTYPE | INTEGER expression | Type of marker symbol |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

CP7PT/CAPPT draws [3] the symbol, MKTYPE, centred at point $(x, y)$ and draws [2] the caption, CAP, to the right of the symbol.

After drawing the specified symbol and text, the pen is left at the centre of the symbol to allow a line to be drawn from the symbol to another point by a succeeding call of BRKNPT/JOINPT. BREAK should be called if a line is not required to join symbol and point.

The point $(x, y)$ is specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine KXYXY. REAL and INTEGER values can be converted to text strings using the conversion subroutines KREAL and KNUMB respectively.

By default, if the caption is too long to fit within the SIMPLEPLOT page, it is clipped such that only whole characters appear; if coordinates exceed the limits of the SIMPLEPLOT page, the caption is omitted. Both these occurrences contribute to the number of incomplete page tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Symbol clipped)$^2$
(Symbol spills over boundary)$^2$
(Text clipped)$^2$

### See also

BREAK, CLIPLV, CP7LB, MARKPT, MKMASK, MKSIZE, KREAL and KNUMB.

## SUBROUTINE CP7XC (X1, X2, Y, CAP)<br>SUBROUTINE CAPXC (X1, X2, Y, CAP, NCAP)

### Name

CP7XC – to draw a horizontally centred caption.

**Availability** Section 4, released version 2-6.

### Arguments

| | | |
|---|---|---|
| X1, X2 | REAL expressions | $x$ coordinates between which the label is centred |
| Y | REAL expression | Vertical position of the caption |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

CP7XC/CAPXC draws [2] the caption, CAP, centred between $x_1$ and $x_2$ at vertical position $y$.

All coordinates are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine KXYXY. REAL and INTEGER values can be converted to text strings using the conversion subroutines KREAL and KNUMB respectively.

By, default, if the caption is too long to fit within the SIMPLEPLOT page, it is clipped such that only whole characters appear; if coordinates exceed the limits of the SIMPLEPLOT page, the caption is omitted. Both these occurrences contribute to the number of incomplete page tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Text clipped)$^2$

### See also

CLIPLV, CP7LB, CP7PT, CP7YC, LABJST, LABANG, KNUMB and KREAL.

## SUBROUTINE CP7YC (Y1, Y2, X, CAP)<br>SUBROUTINE CAPYC (Y1, Y2, X, CAP, NCAP)

### Name

CP7YC – to draw a vertically centred caption.

**Availability** Section 4, released version 2-6.

### Arguments

| | | |
|---|---|---|
| Y1, Y2 | REAL expressions | $y$ coordinates, between which the label is centred |
| X | REAL expression | Horizontal position of the label |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

CP7YC/CAPYC draws [2] the caption, CAP, centred between $y_1$ and $y_2$ at horizontal position $x$.

All coordinates are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine KXYXY. REAL and INTEGER values can be converted to text strings using the conversion subroutines KREAL and KNUMB respectively.

By default, if the caption is too long to fit within the SIMPLEPLOT page, it is clipped such that only whole characters appear; if coordinates exceed the limits of the SIMPLEPLOT page, the caption is omitted. Both these occurrences contribute to the number of incomplete page tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Text clipped)$^2$

### See also

CLIPLV, CP7LB, CP7PT, CP7XC, LABJST, LABANG, KNUMB and KREAL.

## SUBROUTINE CTBRKN (LTYPE)

### Name

CTBRKN – to specify the broken line pattern used for contours and surface sections.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| LTYPE | INTEGER expression | Type of broken line pattern |

### Description

Individual contour curves and surface sections are drawn with the bundled attributes of the current pen (by default, this is normally a solid line); CTBRKN selects an alternative broken line pattern, LTYPE.

If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless

LTYPE = −999 which indicates that all the bundled line-drawing attributes of the current pen are to be used.

CTBRKN controls the line pattern used by *CNTS, *CONT, *CUT, ZEDGE, ZZEDGE, ZELEM, ZELEMS and ZZELMS. CTBRKN has no effect on the subroutines for 2-D charts and graphs.

### Default

CALL CTBRKN(-999) restores the default.

### See also

BUNLPR, LINSET and SQBRKN.

## SUBROUTINE CTCURV (ITYPE)

### Name

CTCURV – to specify the curve type used for drawing contours.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Method of curve drawing |

| ITYPE | *Curve type* |
|---|---|
| 1 | Tight fitting smooth curve |
| 2 | Looser fitting smooth curve |
| 3 | Straight lines from point to point |

### Description

Contour curves are drawn by joining interpolated points with some sort of curve; CTCURV specifies the method which is to be used for drawing contour curves.

The three types correspond exactly to the first three options of CVTYPE (used to control 2-D curve drawing).

Contours which change rapidly may not be suitable for loose fitting curves (ITYPE=2) because the curvature assigned by the curve drawing may produce intersections of contour curves.

CTCURV controls the curve type used for all subroutines related to contour pictures (*CNTS, *CONT and *CUT). CTCURV has no effect on shaded contours nor on the subroutines for charts and graphs.

### Default

CALL CTCURV(1) restores the default.

### See also

CVTYPE and SFMESH.

## SUBROUTINE CTHOLD (ICODE)

### Name

CTHOLD – to specify whether coordinates of internally generated curves are to be saved.

**Availability** Section plus, released version 2-13.

### Argument

| | | |
|---|---|---|
| ICODE | INTEGER expression | Behaviour of subsequent curve drawing |

| ICODE | *Graphics output* | *Coordinates of contour* |
|---|---|---|
| 1 | Yes | Forgotten (default) |
| 2 | No | Stored |
| 3 | Yes | Stored |

### Description

By default, the contour drawing subroutines (*CONT) plot the curves corresponding to a single contour level and the coordinates of these curves are forgotten. CTHOLD specifies that the coordinates are remembered and stored within SIMPLEPLOT for subsequent inquiry by QCURVE.

The coordinates of other curves are also remembered:

- Cross-sectional curves drawn by *CUT,
- Element boundaries drawn by ZZELMS and ZELEMS, and
- Ungridded data boundaries drawn by ZZEDGE and ZEDGE.

### Default

CALL CTHOLD(1) restores the default.

### See also

*CONT, *CUT, QCURVE, ZEDGE, ZELEMS, ZZEDGE and ZZELMS.

## SUBROUTINE CTLABS (IFREQ)

### Name

`CTLABS` – to specify the frequency of labels included with contour curves.

**Availability** Section 2, released version 2-6.

### Argument

| | | |
|---|---|---|
| `IFREQ` | `INTEGER` expression | Frequency of labels |

### Description

By default, contour curves are not labelled. `CTLABS` specifies the frequency at which labels are to be included.

After a call of `CTLABS(IFREQ)`, the $z$ value associated with each contour curve is drawn near the curve when a sequence of `IFREQ` or more points is joined in one operation.

For more frequent labelling, reduce `IFREQ` and for less frequent labelling, increase `IFREQ`; the exception to this rule is that a frequency of 0 results in no labels at all.

A call of `CTLABS(10)` is equivalent to a call of `CTNUMB(.TRUE.)` and a call of `CTLABS(0)` is equivalent to `CALL CTNUMB(.FALSE.)`.

### Default

`CALL CTLABS(0)` restores the default.

### See also

`CTNUMB` and `SQZLAB`.

## SUBROUTINE CTNUMB (TORF)

### Name

`CTNUMB` – to specify whether contour curves are to be labelled.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether labelling is required |

### Description

By default, contour curves are not labelled. `CTNUMB` specifies whether or not contours are to be labelled.

After a call of `CTNUMB(.TRUE.)`, the $z$ value associated with each contour curve is drawn near the curve when a sequence of 10 or more points is joined in one operation.

A call of `CTNUMB(.TRUE.)` is equivalent to a call of `CTLABS(10)` and a call of `CTNUMB(.FALSE.)` is equivalent to `CALL CTLABS(0)`.

### Default

`CALL CTNUMB(.FALSE.)` restores the default.

### See also

`CTLABS` and `SQZLAB`.

## SUBROUTINE CVLBJS (VJUST, HJUST)

**Name**

`CVLBJS` – to specify the justification of the label drawn on a labelled curve.

**Availability** Section 4, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `VJUST` | `CHARACTER*1` | Vertical justification of label |
| `HJUST` | `CHARACTER*1` | Horizontal justification of label |

| `VJUST` | *Vertical justification* |
|---|---|
| `'D'` | Default |
| `'B'` | Bottom of the letters level with the curve |
| `'C'` | Centre of the letters level with the curve (default) |
| `'T'` | Top of the letters level with the curve |

| `HJUST` | *Horizontal justification* |
|---|---|
| `'D'` | Default |
| `'C'` | Label is centred on the reference point |
| `'L'` | Label starts at the reference point (default) |
| `'R'` | Label ends at the reference point |

**Description**

By default, curves drawn by `LABCV`/`LABCV7` are vertically centred on the curve, and left-justified relative to the reference point for the curve (see `CVLBPS`). `CVLBJS` specifies an alternative justification of the label.

`VJUST` and `HJUST` are similar to the arguments of `LABJST`.

If either ordinate of the reference point is equal to the current no-data value, the label is justified against the next valid point. If horizontal justification is `'L'`eft or `'C'`entre, the next point is to the right of the reference point; if justification is `'R'`ight, the next point is to the left of the reference point.

**Default**

`CALL CVLBJS('D','D')` restores the default.

**See also**

`CVLBPS`, `LABCV7` and `LIMIDX`.

## SUBROUTINE CVLBPS (IPOS)

### Name

CVLBPS – to specify the reference point for the label drawn on a labelled curve.

**Availability** Section 4, released version 2-13.

### Argument

| | | |
|---|---|---|
| IPOS | INTEGER expression | Position of label, given as an index of data arrays |

### Description

By default, curves drawn by LABCV/LABCV7 are positioned relative to the mid-point of the curve. CVLBPS specifies an alternative reference point for the label.

Labels are drawn relative to the point (XARR(IPOS),YARR(IPOS)), where XARR and YARR are the parallel arrays of coordinates which are passed as arguments to LABCV7. Labels can be drawn relative to the highest or lowest point on the curve by setting IPOS to a value evaluated using LIMIDX.

If either XARR(IPOS) or YARR(IPOS) is equal to the current no-data value, the label is justified against the next valid point. If horizontal justification is 'L'eft or 'C'entre, the next point is IPOS+1; if justification is 'R'ight, the next point is IPOS − 1.

If there are no valid points between IPOS and the end of the curve, the label is omitted.

If IPOS is greater than NARR (the number of points in XARR and YARR) or less than 1, the default is restored.

### Default

CALL CVLBPS(0) restores the default.

### See also

CVLBJS, LABCV7 and LIMIDX.

## SUBROUTINE CVTYPE (ITYPE)

### Name

`CVTYPE` – to specify the type of curve to be drawn on 2-D graphs.

**Availability** Section 1, released before version 2-5.

### Argument

`ITYPE` `INTEGER` expression Method for curve construction

| `ITYPE` | *Curve type* |
|---|---|
| 1 | A tight fitting smooth curve through every data point, which is confined between the same limits as the data. Sudden changes in the data result in sharp corners in the curve. A Cartesian curve drawn by this method is not invariant under rotation |
| 2 | A looser fitting smooth curve through every data point, which will probably extend higher or lower than the data at turning points, but should not have sharp corners; a Cartesian curve drawn by this method is invariant under rotation |
| 3 | Straight lines from point to point through the data, even on non-linear scales and polar charts |
| 4 | Linear regression * |
| 5 | Quadratic regression * |

* If `ITYPE`=4 or 5, the curve drawn covers the range of the $x$ scale defined by `FNRNGE`.

### Description

By default, the 2-D composite graph drawing subroutines (*eg.* `BRKNCV`, `DRAWAR`, `SHDEC2`) draw curves using a tight fitting curve to interpolate between the data points. `CVTYPE` specifies an alternative methods of curve fitting.

`CVTYPE` has no effect on surface pictures, contour maps or cross-sections (see `CTCURV`).

### Default

`CALL CVTYPE(1)` restores the default.

### See also

`CTCURV`, `FNRNGE`, `POLOUT` and `POLY`.

## SUBROUTINE DDDATA (IPTR)

**Name**

`DDDATA` – to pass device driver specific information from the application to a device driver.

**Availability** Section plus, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `IPTR` | `INTEGER` array | array with 8 elements holding required value. All elements of the array must be defined |

**Description**

Some applications have a requirement to pass data to a device driver. The information to be passed is usually of non-standard type and may be many bytes in length. Driver-specific data include widgets, lists of pixel values, points to bitmaps and pasteboard IDs.

Rather than provide a subroutine which could directly handle every possible type and quantity of data, a simple value is passed. It is expected that in the majority of cases this will be a pointer to the actual data.

An array of eight integers is used to allow for systems where `INTEGER`s are shorter than addresses. The exact conversion between a pointer and the `INTEGER` array is system dependent. Typically it will be the union (equivalence) of a pointer and the `INTEGER` array.

This subroutine may be called at any time. Data passed to this subroutine are stored until a device driver reads it or the device driver sub-system goes through a final closedown.

It is the responsibility of the calling code to ensure that the routine is called at an appropriate time and for an appropriate device driver.

**See also**

`IOCHAN` and device driver specific information.

### SUBROUTINE DEFCAP (ITYPE, VCHAR, HCHAR, NROWS, NCOLMS)

**Name**

DEFCAP – to define an area for captions.

**Availability** Section 1, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Type of reservation |
| VCHAR | CHARACTER*1 | Vertical position of area |
| HCHAR | CHARACTER*1 | Horizontal position of area |
| NROWS | INTEGER expression | Number of lines in area |
| NCOLMS | INTEGER expression | Maximum number of characters in any line of a caption |

**Description**

DEFCAP defines an area for captions which is large enough for NROWS lines of text containing NCOLMS characters; the size is calculated using the nominal text width of the current font.

VCHAR and HCHAR are single characters representing the initial letters of vertical and horizontal positions. These determine the position of the area and also whether it is associated with a picture or the SIMPLEPLOT page; this in turn effects the limits of the area, the size of the text and therefore the size of the area.

The possible interpretations of VCHAR, HCHAR and ITYPE are listed in Appendix T of the *SIMPLEPLOT Supplement*.

ITYPE determines the behaviour of the caption area.

If BOXCAP has been used, a box is drawn around the caption area.

**Diagnostics**

```
(Key/caption attempted with no page)²
(Key/caption attempted with no picture)²
(Key/caption height reduced)²
(Key/caption width reduced)²
(Maximum no. of keys/captions reached)²
(No room for key/caption area)²
```

**See also**

ADDCP7, BLNKKY, BOXCAP, DEFKEY, DEFPOS, KPOSXY, PUTCP7, QKYCAP and TITLE7.

### SUBROUTINE DEFKEY (ITYPE, VCHAR, HCHAR, NROWS, NCOLMS)

**Name**

DEFKEY – to define an area for a key box.

**Availability** Section 1, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Type of reservation |
| VCHAR | CHARACTER*1 | Vertical position of area |
| HCHAR | CHARACTER*1 | Horizontal position of area |
| NROWS | INTEGER expression | Number of lines in area |
| NCOLMS | INTEGER expression | Maximum number of characters in any key caption |

**Description**

DEFKEY defines an area for a key which is large enough for NROWS lines of text containing NCOLMS characters when written alongside a key sample. The size is calculated using the nominal text width of the current font.

A box is drawn [1] around the key unless cancelled by BOXKY. If ITYPE=1 or 2, the key is associated with the current picture, and the box is drawn immediately. If ITYPE=3, a key box is drawn with each subsequent new picture.

VCHAR and HCHAR are single characters representing the initial letters of vertical and horizontal positions. ITYPE is an integer between 1 and 3. The possible interpretations of VCHAR, HCHAR and ITYPE are listed in Appendix T of the *SIMPLEPLOT Supplement*.

**Diagnostics**

(Key/caption attempted with no page)[2]
(Key/caption attempted with no picture)[2]
(Key/caption height reduced)[2]
(Key/caption width reduced)[2]
(Maximum no. of keys/captions reached)[2]
(No room for key/caption area)[2]

**See also**

BLNKKY, BOTHK7, BOXKY, DEFCAP, DEFKYW, DEFPOS, FILLK7, KPOSXY, LINEK7, MARKK7, QKYCAP and SHDEK7.

## SUBROUTINE DEFKYW (NCHARS)

### Name

`DEFKYW` – to specify the width of samples in a key box.

**Availability** Section 4, released version 2-12.

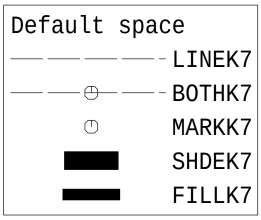
### Argument

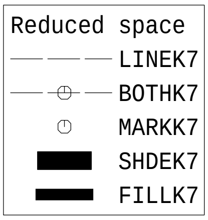
| | | |
|---|---|---|
| `NCHARS` | `INTEGER` expression | Width of sample in equivalent number of characters (or columns) |

### Description

By default, the space allocated for a key sample is 12 characters wide. This is sufficient to illustrate uniquely the thirteen standard SIMPLEPLOT software broken line patterns (−6..6), but can appear too wide for samples of shading pattern and marker symbols.

`DEFKYW` specifies the width of the space in terms of the equivalent number of characters (or columns).





`DEFKYW` takes effect when the key is set up. This depends on the type of key defined by `CALL DEFKEY(ITYPE, ...)`:

| `ITYPE` | *Box drawn* |
|---|---|
| 1 | when a new key is defined by `DEFKEY` |
| 2 | when a new key is defined by `DEFKEY` |
| 3 | whenever a new picture is started |

`MPK7H`, `MPK7V` and `SHKEYS` correspond to `ITYPE`=2.

### Default

`CALL DEFKYW(0)` restores the default.

### See also

`DEFKEY` and `SHKEYS`.

## SUBROUTINE DEFPOS (VCMS, VJST, HCMS, HJST, ITYPE)

### Name

DEFPOS – to specify page coordinates of the movable descriptor, 'M'.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| VCMS | REAL expression | $y$ (or vertical) coordinate |
| VJST | CHARACTER*1 | Vertical justification of area |
| HCMS | REAL expression | $x$ (or horizontal) coordinate |
| HJST | CHARACTER*1 | Horizontal justification of area |
| ITYPE | INTEGER expression | Intended use of text |

| ITYPE | *Text usage* |
|---|---|
| 0 | Default – determined by SIMPLEPLOT* |
| 1 | Titles, keys or captions relating to pictures |
| 2 | Titles, keys or captions relating to groups/pages |

* Page usage is assumed, unless 'M' is used in conjunction with a descriptor specifically relating to pictures (*eg.* 'C'entre or 'T'op).

### Description

The movable picture descriptor, 'M', is an additional descriptive position for titles, keys and caption areas. DEFPOS specifies a position for annotation by defining a point (HCMS,VCMS), and justification with respect to this point (VJST,HJST)

VCMS and HCMS are specified in centimetres relative to the bottom left-hand corner of the current SIMPLEPLOT page. The possible interpretations of VJST and HJST are listed in Appendix T if the *SIMPLEPLOT Supplement*.

Please note that a call to DEFPOS terminates a multi-line sequence of titles whether or not the defined position is used for a title.

The conversion subroutine, KXYXY, can be used to convert user coordinates (ICODE1=5), normalized page coordinates (ICODE1=4) or device coordinates (ICODE1=0, 1, 2) into physical page coordinates in centimetres (ICODE2=3).

### Default

CALL DEFPOS(0.0,'D',0.0,'D',0) restores the default.

### See also

DEFCAP, DEFKEY, KPOSXY, KXYXY, MPK7H, MPK7V, SHKEYS and TITLE7.

## SUBROUTINE DEVNAM (STR)

**Name**

`DEVNAM` – to specify a graphics device by name.

**Availability** Section plus, released version 2-11.

**Argument**

| | | |
|---|---|---|
| `STR` | `STRING` expression | Site or absolute name of device |

**Description**

If SIMPLEPLOT is used with a multiple device driver system (*eg.* the Single Entry Point system), a graphics output device can be specified by a number of alternative methods. `DEVNAM` specifies a graphics device when a program is executed.

If the program has not yet started plotting, `DEVNAM` nominates the device which will be opened when plotting starts. Only one device can be open at any time therefore, if a device is already open, the current page is terminated (graphics buffers are flushed) and the device is closed. The device identified by `STR` is nominated as the device which is to be opened when plotting resumes.

`DEVNAM` is only applicable at installations which offer SIMPLEPLOT with a multiple device driver system. With the S.E.P. system, `STR` represents either the *site name* or the *absolute name* of a device driver option; `STR` is assumed to be a site name unless it is preceded by a `'/'` which identifies an absolute name.

**Diagnostics**

`(DEVICE CLOSED)`[1]

**Default**

`CALL DEVNAM(' ')` restores the device selection menu.

**See also**

`DEVNO` and `QDEV`.

## SUBROUTINE DEVNO (IDEV)

### Name

DEVNO – to specify a graphics device by number.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| IDEV | INTEGER expression | Number to identify device |

### Description

If Simpleplot is used with a multiple device driver system (*eg.* the Single Entry Point system), a graphics output device can be specified by a number of alternative methods. DEVNO specifies a graphics device when a program is executed.

If the program has not yet started plotting, DEVNO nominates the device which will be opened when plotting starts. Only one device can be open at any time; therefore, if a device is already open, the current Simpleplot page is terminated (graphics buffers are flushed) and the device is closed. The device identified by IDEV is nominated as the device which is to be opened when plotting resumes.

DEVNO is only applicable at installations which offer Simpleplot with a multiple device driver system. With the S.E.P. system, IDEV is interpreted either as a *site number* or an *absolute number* of a device driver option; site numbers are in the range 1–999 and absolute numbers are in the range 1000–32767.

### Diagnostics

(DEVICE CLOSED)[1]

### Default

CALL DEVNO(0) restores the device selection menu.

### See also

DEVNAM, QDEV and REDRAW.

## SUBROUTINE DIAGDD (ILEVEL)

**Name**

`DIAGDD` – to specify the level of device driver diagnostics.

**Availability** Section 1, released version 2-13.

**Argument**

`ILEVEL` `INTEGER` expression Level of device driver diagnostics

| `ILEVEL` | *Diagnostics level* |
|---|---|
| 0 | No device driver diagnostics |
| 1 | Device driver error messages (default) |
| 2 | Level 1 + device driver warning messages |
| 3 | Level 2 + device driver informative messages |

**Description**

The S.E.P. device driver system issues diagnostic messages which are classified as either error messages, warning messages or informative messages, and, by default, only error messages are issued. `DIAGDD` specifies an alternative level of device driver diagnostics, independently of the Simpleplot library messages.

For example,

`CALL DIAGDD(0)` switches off all device driver diagnostics.
`CALL DIAGDD(3)` switches on all device driver diagnostics.

Please refer to host specific information about how to control the destination of diagnostic messages.

**Default**

`CALL DIAGDD(1)` restores the default.

**See also**

`DIAGLV` and `DIAGOP`.

## SUBROUTINE DIAGLV (ILEVEL)

### Name

`DIAGLV` – to specify the level of diagnostics.

**Availability** Section plus, released version 2-9.

### Argument

`ILEVEL` `INTEGER` expression Level of diagnostics

| `ILEVEL` | *Description* | *Messages* |
|---|---|---|
| 0 | No messages output at all | None |
| 1 | Brief messages (default) | Type 1 |
| 2 | Level 1 messages plus fuller details | Type 1+2 |
| 3 | Level 1 messages plus subroutine trace | Type 1+3 |
| 4 | Level 2 messages plus subroutine trace | Type 1+2+3 |

### Description

By default, SIMPLEPLOT issues diagnostic messages to provide information about progress, to warn of omissions, and for fatal errors. `DIAGLV` specifies an alternative level of diagnostic information. Three types of diagnostic output can be generated:

1. Brief messages about key events during program execution
2. Fuller details whenever any omission has been necessary
3. Trace of each SIMPLEPLOT subroutine called

Five diagnostic levels are available, according to the value of `ILEVEL`, to output combinations of these.

Please refer to Appendix M of the *SIMPLEPLOT Supplement* for further information about diagnostic messages, and to host specific information about how to control the destination of diagnostic messages.

### Default

`CALL DIAGLV(1)` restores the default.

### See also

`DIAGDD`, `DIAGOP` and `QSTAT`.

### SUBROUTINE DIAGOP (STR, ITYPE)

**Name**

`DIAGOP` – to output a message to the SIMPLEPLOT diagnostics channel.

**Availability** Section 1, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `STR` | `STRING` expression | Message |
| `ITYPE` | `INTEGER` expression | Type of message |

| `ITYPE` | *Suggested type of message* |
|---|---|
| 1 | Brief messages about key events during the execution of the program |
| 2 | Fuller details whenever any omission has been necessary |
| 3 | Trace of each subroutine called |

**Description**

`DIAGOP` outputs the specified message, `STR`, on the SIMPLEPLOT diagnostic I/O channel according to the value of `ITYPE` and the current diagnostic level.

It is not necessary to follow these descriptions exactly but, if you do, `DIAGOP` interacts with `DIAGLV` such that user-defined messages behave in the same way as SIMPLEPLOT diagnostic messages. For example, after `CALL DIAGLV(3)` user-defined messages of type 1 and 3 are output, but type 2 messages are omitted (see `DIAGLV` for further details).

Please refer to Appendix M of the *SIMPLEPLOT Supplement* for further information about diagnostic messages, and to host specific information about how to control the destination of diagnostic messages.

**See also**

`DIAGDD` and `DIAGLV`.

## SUBROUTINE DRAWAR (X1, DX, DARR, NARR)

### Name

DRAWAR – to draw a curve from an array of values.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| X1 | REAL expression | Value of independent variable for plotting DARR(1) |
| DX | REAL expression | Independent variable interval between function values |
| DARR | REAL array | Function values for plotting |
| NARR | INTEGER expression | Number of points |

### Description

Using the bundled line-drawing attributes of the current pen, DRAWAR draws [1] the curve which connects the following points:

(X1,DARR(1)), (X1+DX,DARR(2)),...
(X1+(NARR-1)*DX,DARR(NARR))

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ and $x_i = x_1 + (i-1)dx$. The type of curve depends on CVTYPE.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Data curve exceeds scales)$^2$

### See also

BRKNAR, BUNLPR, CLIPLV, CVTYPE, MARKAR, RIDDLE and SHDEAR.

## SUBROUTINE DRAWBX (X1, Y1, X2, Y2)

### Name

DRAWBX – to draw a box.

**Availability** Section 1, released version 2-8.

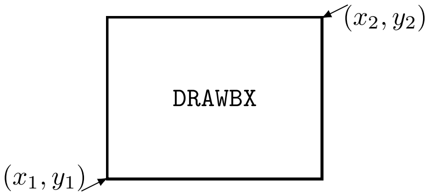
### Arguments

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of opposite corners of the box, in units of the plotting scales |

### Description

Using the bundled line-drawing attributes of the current pen, DRAWBX draws [1] the box extending from $x_1$ to $x_2$ by $y_1$ to $y_2$. If $x_1$=$x_2$ or $y_1$=$y_2$, then nothing is drawn.

The points $(x_1, y_1)$ and $(x_2, y_2)$ are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, KXYXY.



By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(**$x_1, y_1, x_2, y_2$**)$^2$
(Null box, nothing drawn)$^2$

### See also

BRKNBX, BUNLPR, CLIPLV and SHDEBX.

## SUBROUTINE DRAWCL (XCENT, YCENT, RADIUS)

### Name

`DRAWCL` – to draw a circular function with specified centre and radius.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `XCENT`,<br>`YCENT` | `REAL` expressions | Coordinates of the centre of the circle in units of the plotting scales |
| `RADIUS` | `REAL` expression | Circle radius in plotting scale units |

### Description

Using the bundled line-drawing attributes of the current pen, `DRAWCL` draws [1] the circle (or the part of the circle which lies within the range of the plotting scales) centred at point $(x_{cent}, y_{cent})$, with radius `RADIUS`. The resulting shape is only geometrically circular when both scales are equal and linear.

The centre point and the radius are both specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, `KXYXY`.

By default, all drawing is restricted by the limits of the picture scales; if it exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Circle exceeds scales)`$^2$

### See also

`BRKNCL`, `BUNLPR`, `CLIPLV`, `KXYXY`, `SCALES` and `SHDECL`.

### SUBROUTINE DRAWCV (XARR, YARR, NARR)

**Name**

DRAWCV – to draw a curve from a set of coordinates in parallel arrays.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | Coordinates of points, in units of the plotting scales |
| NARR | INTEGER expression | Number of points |

**Description**

Using the bundled line-drawing attributes of the current pen, DRAWCV draws [1] the curve corresponding to the following points:

(XARR(1), YARR(1)), (XARR(2), YARR(2)), . . .
(XARR(NARR), YARR(NARR))

A closed curve can be obtained by using data in which XARR(NARR)=XARR(1) and YARR(NARR)= YARR(1). The type of curve drawn depends on CVTYPE.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Data curve exceeds scales)[2]

**See also**

BRKNCV, BUNLPR, CLIPLV, CVTYPE, MARKCV, RIDDLE and SHDECV.

## SUBROUTINE DRAWFN (FUNX)

### Name

`DRAWFN` – to draw a curve of a user-defined function, $y = f(x)$ (or $r = f(\theta)$).

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `FUNX` | function name | `REAL` function with one `REAL` argument, also declared in an `EXTERNAL` statement |

### Description

`DRAWFN` evaluates and draws [1] the curve of the specified function, `FUNX` using the bundled line-drawing attributes of the current pen.

With Cartesian coordinates, the specified function, `FUNX`, must calculate $y$ given $x$, and with polar coordinates `FUNX` must calculate $r$ given $\theta$. The intervals of the independent variable over which `DRAWFN` evaluates the function, vary with the curvature of the curve.

Care must be taken in the programming of `FUNX` to ensure that it is continuous and cannot fail for any arguments within the plotting range. Discontinuities may cause SIMPLEPLOT to loop indefinitely.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Function curve exceeds scales)`[2]

### See also

`BRKNFN`, `BUNLPR`, `CLIPLV`, `FNRNGE`, `POLY`, `RIDDLE` and `SHDEFN`.

## SUBROUTINE DRAWLN (X, Y)

### Name

`DRAWLN` – to draw a straight line between a point and zero.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `X`, `Y` | `REAL` expressions | Coordinates of a point specified in units of the plotting scales |

### Description

Using the bundled line-drawing attributes of the current pen, `DRAWLN` draws [1] a straight line between the point $(x, y)$ and $(x, 0.0)$ in Cartesian plotting, or between $z = f(r, \theta)$ and $(0.0, \theta)$ in polar plotting. Nothing is drawn when `DRAWLN` is called with a Cartesian picture whose vertical scale excludes zero.

The point $(x, y)$ is specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, `KXYXY`.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(**`$x, y$`**)` $^{2}$

### See also

`BRKNPT`, `BUNLPR`, `CLIPLV` and `RANGE`.

## SUBROUTINE DRAWPR (SUBR, T1, T2)

### Name

DRAWPR – to draw the user-defined parametric function.

**Availability** Section 4, released version 2-9.

### Arguments

| | | |
|---|---|---|
| SUBR | subroutine name | Subroutine with three REAL arguments, also declared in an EXTERNAL statement |
| T1, T2 | REAL expressions | Values of the function parameter between which curve is to be drawn |

### Description

DRAWPR evaluates and draws [1] the curve of a user-defined parametric function using the bundled line-drawing attributes of the current pen.

The parametric function is specified in subroutine SUBR. SUBR must have three REAL arguments: T, the value of the function argument, and VARX and VARY, to receive coordinates corresponding o T. SUBR is called by DRAWPR with values of T ranging from T1 to T2.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

`(Parametric function exceeds scales)`[2]

### See also

BRKNPR, BUNLPR, CLIPLV, RIDDLE and SHDEPR.

## SUBROUTINE ENDPAG

### Name

`ENDPAG` – to finish a picture and group separately from starting another picture.

**Availability** Section 4, released version 2-11.

### Arguments

None.

### Description

A SIMPLEPLOT page consists of a group of pictures (if `GROUP` has been called) or an individual picture otherwise. By default, when a SIMPLEPLOT program produces more than one page, each page is automatically finished by the start of another or the close down of graphics. If the graphics device is a remote device, an individual page may not be accessible until the page is terminated; in this case, it may be necessary to finish a page before the program starts another.

`ENDPAG` finishes the current page by flushing graphics buffers, performing the device page termination procedures, and outputting any diagnostic messages associated with the page. If `GROUP` has not been called, `ENDPAG` has a similar effect to `ENDPIC`; if `GROUP` has been called and `ENDPAG` is called before a requested group is full, the group is terminated and the next new picture starts a new group on a new page.

After `ENDPAG`, all attempts to output graphics are ignored until the start of another page.

### Diagnostics

`(END OF GROUP)`[1]
`(END OF PICTURE)`[1]
`(No. of incomplete page tasks=`*n*`)`[1]

### See also

`ENDPIC`, `ENDPLT`, `NEWPAG`, `OUTBUF`, `OWNIO` and `OWNNEW`.

## SUBROUTINE ENDPIC

### Name

`ENDPIC` – to finish a picture separately from starting another picture.

**Availability** Section 4, released version 2-7.

### Arguments

None.

### Description

By default, when a SIMPLEPLOT program produces more than one picture, each picture is automatically finished by the start of another or the close down of graphics. `ENDPIC` finishes the current picture by flushing any graphics buffers, and outputting any diagnostic messages associated with the picture.

If `GROUP` has been called, and the latest picture does not complete the group, `ENDPIC` leaves it in a state to continue the group when the next picture is started.

If the group is completed, the next picture starts a new SIMPLEPLOT page but any titles, keys or captions positioned relative to the page or group will be drawn on the current page.

### Diagnostics

`(END OF PICTURE)`[1]
`(No. of incomplete picture tasks=`*n*`)`[1]

### See also

`ENDPAG`, `ENDPLT`, `MOVEON`, `OUTBUF`, `OWNIO` and `OWNNEW`.

## SUBROUTINE ENDPLT

### Name

`ENDPLT` – to close the plotting device and SIMPLEPLOT at the end of plotting.

**Availability** Section 1, released before version 2-5.

### Arguments

None.

### Description

`ENDPLT` flushes any graphics output buffers, closes the current graphics device (if open) and closes any permanent and temporary files opened by SIMPLEPLOT.

Every SIMPLEPLOT program should call `ENDPLT` when all plotting is complete. Failure to close the device may result in loss of some or all of the last picture, and the associated diagnostics. Failure to close down SIMPLEPLOT may result in temporary and/or permanent files being left open. `ENDPLT` should be called even if a SIMPLEPLOT program does no plotting, *eg.* if it only manipulates data.

If plotting is attempted when SIMPLEPLOT has been closed, SIMPLEPLOT is automatically reopened.

### Diagnostics

`(DEVICE CLOSED)`[1]
`(SIMPLEPLOT CLOSED)`[1]

### See also

`DEVNO` and `DEVNAM`.

## SUBROUTINE EQSCAL (XSTART, XSTOP, YSTART, YSTOP, IUNITS)

### Name

EQSCAL – to specify similar linear scales for Cartesian or polar plotting.

**Availability** Section 1, released version 2-10.

### Arguments

| | | |
|---|---|---|
| XSTART | REAL expression | Value at start of $x$ (or $r$) scale |
| XSTOP | REAL expression | Value at end $x$ (or $r$) scale |
| YSTART | REAL expression | Value at start of $y$ (or $\theta$) scale |
| YSTOP | REAL expression | Value at end of $y$ (or $\theta$) scale |
| IUNITS | INTEGER expression | Type of units |

| IUNITS | *Units* | XSTART, XSTOP | YSTART, YSTOP |
|---|---|---|---|
| −1 | Cartesian | *Values ignored,* | *centimetres used* |
| 0 | Cartesian | Horizontal units | Vertical units |
| | | RSTART, RSTOP | THSTRT, THSTOP |
| 1 | Polar | Radial units | Angles in degrees |
| 2 | Polar | Radial units | Angles in radians |
| 3 | Polar | Radial units | User-defined scale |

### Description

EQSCAL specifies the limits and units of the current plotting scales, ensuring that 1 unit in $x$ is the same physical length as 1 unit in $y$. The ranges of the scales, XSTART to XSTOP and YSTART to YSTOP can be expressed in different units, indicated by the value of IUNITS.

The plotting units can be switched by COORDS within the plotting of a picture, without cancelling EQSCAL.

When scaling is controlled by EQSCAL default axes are drawn from XSTART to XSTOP and YSTART to YSTOP. over the requested ranges rather than the extended ranges used to fit the picture size.

### Default

Calling EQSCAL with XSTART=XSTOP or YSTART=YSTOP restores the default or IUNITS=−1 can be used to set dissimilar linear scales.

### See also

AXES7, COORDS, NEWPIC, POLAR7, SCALES, XSCALE and YSCALE.

## SUBROUTINE FFSET (ISET)

### Name

FFSET – to specify an alternative source of fast area-fill characters.

**Availability** Section plus, released version 2-11.

### Argument

| | | |
|---|---|---|
| ISET | INTEGER expression | Fast area-fill character source |

| ISET | *Fast area-fill character source* |
|---|---|
| $\leq -1$ | Alternative hardware/firmware sets (if available) |
| 0 | Default (see above) |
| 1 | Shaded rectangles using currently selected area-fill shading method (see SHSET) |
| 2 | Currently selected text characters (see CHSET) |
| 3 | Currently selected broken line patterns (see LINSET) |

### Description

For efficient operation and clarity, SIMPLEPLOT is set up to use the most appropriate source of fast area-fill characters for each device. FFSET selects an alternative source of fast area-fill characters to be used for fast area-fill shaded contour maps (MPTYPE(2) or MPTYPE(3)) and shaded surfaces (RASTER(1)).

By default, hardware (or firmware) fast area-fill characters are used if they are available; when no hardware/firmware characters are available, the most efficient alternative on the current device is used for ISET=0.

If the number of fast area-fill characters required exceeds the number available, shaded rectangles are used for additional characters; these rectangles are drawn using the currently selected area-fill shading method (see SHSET) and correspond in size to the current fast-area fill characters (see FFSIZE).

### Default

CALL FFSET(0) restores the default.

### See also

CHSET, FFSIZE, LINSET, MPTYPE, RASTER and SHSET.

## SUBROUTINE FFSIZE (WCMS, HCMS)

### Name

`FFSIZE` – to specify the size of fast area-fill characters.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `WCMS` | `REAL` expression | Character width (in cms) |
| `HCMS` | `REAL` expression | Character height (in cms) |

### Description

By default, the size of fast area-fill characters depends on the type of picture being drawn:

**Shaded contour maps:** the target height and width of fast area-fill characters are equal to the default target width of text characters which is related to picture size (see `TEXTSZ`).

**Shaded pictures using raster process:** (*eg.* shaded surfaces) the target height and width of fast area-fill characters are 75 dots per inch.

If different sizes of fast area-fill characters are available, `FFSIZE` can be used to specify a target size; the size nearest `WCMS` × `HCMS` (in centimetres) is used.

If ordinary text characters are used as fast area-fill characters, SIMPLEPLOT attempts to scale them to achieve the size specified by `FFSIZE`; if `FFSIZE` has not been called, their size is affected by the `TEXT*` subroutines in the same way as other text.

On colour (or grey-scale) pixel devices, the minimum fast area-fill character size is one pixel. The physical size of a single pixel can be calculated (see `QPAGE` and `QPIXLS`) and used to specify the finest resolution.

### Default

Call `FFSIZE` with `WCMS`=0.0 or `HCMS`=0.0 restores the default width or height respectively.

### See also

`FFSET`, `MPTYPE`, `QFFSIZ`, `RASTER` and `TEXTSZ`.

## SUBROUTINE FIGFMT (CHFORM, NUMINT, NDEC)

**Name**

FIGFMT – to specify the format of REAL numbers drawn on pictures.

**Availability** Section 1, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| CHFORM | CHARACTER*1 | Type of format |
| NUMINT | INTEGER expression | Number of digits before point |
| NDEC | INTEGER expression | Number of decimal places |

| CHFORM | *Effect* |
|---|---|
| 'F' | for fixed-point representation of all values |
| 'E' | for floating-point representation of all values |
| 'G' | 'E' for very small or large values, 'F' otherwise |

**Description**

By default, the format of REAL numbers (either drawn on pictures or converted to text strings by KREAL) is chosen to produce the smallest number of characters while still representing the value of the number. FIGFMT specifies an alternative format.

CHFORM corresponds to FORTRAN format specifiers. If CHFORM='G', the number of characters taken to represent the value in both 'E' and 'F' format is calculated and the format which takes up least space is used; in the case of these being equal, 'F' format is used.

NUMINT specifies how many digits are required before the decimal point in the mantissa of floating-point representations but has no effect on fixed-point representations. NUMINT=0 forces zero mantissa, but this is not omitted from the written form (*eg.* 0.123E4).

NDEC specifies how many decimal places are to be included – by default this depends on the value of the number. NDEC=0 indicates no decimal places, NDEC=−1 indicates no decimal places and no decimal point, and NDEC=−2 restores the default.

FIGFMT can be used to control the format of axis annotation: CHFORM controls whether a fixed-point or floating-point format is used for numerical axis labels but values of NUMINT and NDEC are ignored.

**Default**

CALL FIGFMT('G',1,-2) restores the default.

**See also**

FIGSGN, KREAL and PIINCL.

## SUBROUTINE FIGSGN (SIGN, ESIGN)

### Name

`FIGSGN` – to specify the sign conventions for positive numbers and exponents.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `SIGN` | `CHARACTER*1` | Representation of sign of positive number |
| `ESIGN` | `CHARACTER*1` | Representation of sign of positive exponent |

| `SIGN`<br>or `ESIGN` | *Description* | *Examples* | |
|---|---|---|---|
| `'+'` | Always include sign | `+12.4` | `+0.123E+4` |
| `' '` | Space before +ve values | ` 12.4` | ` 0.123E 4` |
| `'D'` | Default (no + sign) | `12.4` | `0.124E4` |

### Description

By default, `REAL` numbers (either drawn on pictures or converted to text strings by `KREAL`) are preceded by a sign only if the value is negative or it is part of an exponent; `FIGSGN` specifies alternative conventions for the signs of positive numbers and/or exponents.

### Default

`CALL FIGSGN('D','D')` restores the default.

### See also

`FIGFMT`, `KREAL` and `PIINCL`.

## SUBROUTINE FILLK7 (ISHADE, CAP)<br>SUBROUTINE FILLKY (ISHADE, CAP, NCAP)

### Name

FILLK7 – to draw an annotated sample of fast area-fill characters in a key.

**Availability** Section plus, released version 2-10.

### Arguments

| | | |
|---|---|---|
| ISHADE | INTEGER expression | Shading pattern number |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

FILLK7/FILLKY inserts a key to fast area-fill characters (as used on the shaded contour maps or raster pictures) in the next empty space in the current key or caption area (defined by DEFKEY/DEFCAP or selected by RESNXT).

A rectangular sample of the fast-area fill characters, corresponding to shading pattern ISHADE, is drawn on the left of the entry; the caption, CAP, is drawn [2] alongside the sample.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)$^2$<br>
(Key/caption area full)$^2$<br>
(No active key/caption area)$^2$

### See also

ADDJST, BLNKKY, DEFKEY, FFSET, MPK7H, MPK7V, MPTYPE, RASTER, SHDEK7, SHPATT and SQSHAD.

## SUBROUTINE FITPAG (ITYPE)

**Name**

FITPAG – to specify the interpretation of PAGE on fixed picture devices.

**Availability** Section 4, released version 2-8.

**Argument**

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Interpretation of PAGE |

| ITYPE | *Device usage* |
|---|---|
| 0 | Output constrained to fit device, device size used as page size. The arguments of PAGE are only used to determine the orientation (portrait or landscape) of an area which can be rotated. |
| 1 | Page proportions as in specified by PAGE. The arguments of PAGE are observed on all devices; if the requested page size does not fit within a fixed picture device, the page is shrunk, but its proportions are retained, ensuring comparability of layout between output on different graphics devices. |
| 2 | Device independent layout, text, *ie.* WYSIWYG (what you see is what you get) output. All attempts to accommodate device differences are bypassed so that output looks similar on different devices. For example, software text is used if the exact size cannot be achieved with hardware text. |
| 3 | WYSIWYG output which may be rotated on all devices to honour page size. |

**Description**

FITPAG specifies the interpretation of the arguments to PAGE.

**Default**

CALL FITPAG(0) restores the default.

**See also**

PAGE and PAGVW.

## SUBROUTINE FNAREA (XMIN, XMAX, YMIN, YMAX)

### Name

FNAREA – to specify the $x$-$y$ plotting ranges for 3-D functions, $z = f(x, y)$.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| XMIN | REAL expression | Minimum $x$ value included |
| XMAX | REAL expression | Maximum $x$ value included |
| YMIN | REAL expression | Minimum $y$ value included |
| YMAX | REAL expression | Maximum $y$ value included |

### Description

By default, contours of functions drawn by FN* are plotted over the ranges of the data of the current picture, and surface pictures are plotted over ranges 0.0 to 10.0 in both $x$ and $y$. FNAREA specifies the ranges XMIN to XMAX and YMIN to YMAX over which 3-D functions $z = f(x, y)$ are plotted.

FNAREA can also be used to specify the $x$ and $y$ scales to be used on a null isometric picture (see ISNEW) and subsequent plotting (*via* KISXY); SFZSCL specifies $z$ scale. This is because SIMPLEPLOT assumes that if there is no data to be drawn, a function is to be drawn.

### Default

Calling FNAREA with XMIN=XMAX and/or YMIN=YMAX restores the default (if equal, the values are otherwise ignored).

### See also

ISNEW, KISXY and SFZSCL.

## SUBROUTINE FNCNTS (FUNXY)

### Name

FNCNTS – to draw a contour map from a 3-D function $z = f(x, y)$.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |

### Description

FNCNTS draws a contour map on the current picture representing values of the specified 3-D function. Contours are drawn for equally-spaced $z$ values within the range of values of FUNXY. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

When plotting the function FUNXY with parameters X and Y, the value of FUNXY(X,Y) is interpreted as the surface height, $z$, at the point $(x, y)$.

The contours drawn by FNCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set

[2] highest level when more than one level drawn

[3] lowest level of set

[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

### Diagnostics

`(Contour map not all in range)`$^2$

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, FNAREA, FNCONT, LIMSFN, RIDDLE, SFMESH, SQBRKN, SQPEN, SQZLAB and SQZVAL.

## SUBROUTINE FNCONT (ZLEV, FUNXY)

### Name

FNCONT – to draw a contour curve from a 3-D function $z = f(x, y)$.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| ZLEV | REAL expression | Contour level |
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |

### Description

FNCONT draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=ZLEV from the specified function FUNXY.

When plotting the function FUNXY with parameters X and Y, the value of FUNXY(X,Y) is interpreted as the surface height, $z$, at the point $(x, y)$.

### Diagnostics

(Contour curve not all in range)[2]

### See also

CTBRKN, CTCURV, CTHOLD, CTLABS, CTNUMB, FNAREA, FNCNTS, LIMSFN, QCURVE, RIDDLE and SFMESH.

### SUBROUTINE FNCUT (X1, Y1, X2, Y2, FUNXY)

**Name**

FNCUT – to draw a 2-D curve of a surface section from a function $z = f(x, y)$.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of two points |
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |

**Description**

If the surface represented by the function FUNXY were cut along the straight line from $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the function along that line.

FNCUT draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable scales must be set up before attempting to draw a section.

When plotting the function FUNXY with parameters X and Y, the value of FUNXY(X,Y) is interpreted as the surface height, $z$, at the point $(x, y)$.

**Diagnostics**

(Surface section not all in range)[2]

**See also**

CTBRKN, CTCURV, CTHOLD, FNAREA, LIMSFN and QCURVE.

## SUBROUTINE FNRNGE (START, STOP)

### Name

FNRNGE – to specify a restricted $x$ range over which to plot 2-D functions $y = f(x)$.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| START,<br>STOP | REAL expressions | Values of the independent variable<br>between which function to be drawn |

### Description

By default, DRAWFN, BRKNFN and SHDEFN plot the curve of a function over the complete range of the scale of the independent variable. FNRNGE specifies an alternative range. On polar pictures, FNRNGE specifies the $\theta$ range over which to plot the function $r = f(\theta)$. By default, the first cycle (0–360°) of a complete polar graph is used or the range of scale of the independent variable.

FNRNGE is essential for functions which would become infinite within the default range, or which would produce multi-cycle polar function curves.

If the 2-D composite graph drawing subroutines(*eg.* BRKNCV, DRAWAR, SHDEC2, *etc.*) linear or quadratic regression curves (see CVTYPE), the curve drawn covers the range of the $x$ scale defined by FNRNGE.

### Default

Calling FNRNGE with START=STOP restores the default.

### See also

BRKNFN, CVTYPE, DRAWFN, EQSCAL, SCALES and SHDEFN.

## SUBROUTINE FNSHAD (ZLEV1, ZLEV2, ISHADE, FUNXY)

### Name

FNSHAD – to draw the shaded area between two contour levels from a 3-D function $z = f(x, y)$.

**Availability** Section plus, released version 2-5.

### Arguments

| | | |
|---|---|---|
| ZLEV1,<br>ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |

### Description

FNSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 for the function FUNXY; the area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

When plotting the function FUNXY with parameters X and Y, the value of FUNXY(X,Y) is interpreted as the surface height, $z$, at the point $(x, y)$. If function values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min} = \min(\texttt{ZLEV1}, \texttt{ZLEV2})$ and $z_{max} = \max(\texttt{ZLEV1}, \texttt{ZLEV2})$.

### Diagnostics

(Shaded contour not all in range)$^2$

### See also

CTLABS, CTNUMB, FNAREA, FNSHDS, LIMSFN, MPTYPE, SFMESH and SHEDGE.

## SUBROUTINE FNSHDS (FUNXY)

### Name

`FNSHDS` – to draw a shaded contour map from a 3-D function $z = f(x, y)$.

**Availability** Section plus, released version 2-5.

### Argument

| | | |
|---|---|---|
| `FUNXY` | function name | `REAL` function with two `REAL` arguments, also declared in an `EXTERNAL` statement |

### Description

`FNSHDS` draws a shaded contour map on the current picture from the specified function `FUNXY`. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of values of `FUNXY`; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see `SHEDGE`).

When plotting the function `FUNXY` with parameters `X` and `Y`, the value of `FUNXY(X,Y)` is interpreted as the surface height, $z$, at the point $(x, y)$.

### Diagnostics

`(Shaded contours not all in range)`$^2$

### See also

`CTLABS`, `CTNUMB`, `FNAREA`, `FNCNTS`, `FNSHAD`, `LIMSFN`, `MPTYPE`, `SFMESH`, `SHEDGE`, `SHPATT` and `SQ*`.

## SUBROUTINE FNSURF (FUNXY)

### Name

FNSURF – to start a new picture and draw a surface from a 3-D function $z = f(x, y)$.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |

### Description

FNSURF starts a new picture and draws a surface of the specified function over the ranges of $x$ and $y$ specified by the latest call of FNAREA. If FNAREA has not been called, or has been used to indicate that the default ranges are required, the defaults ($x$ from 0.0 to 10.0, $y$ from 0.0 to 10.0) are used by FNSURF.

When plotting the function FUNXY with parameters X and Y, the value of FUNXY(X,Y) is interpreted as the surface height, $z$, at the point $(x, y)$.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with COORDS before calling FNSURF.

FNSURF uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.

[2] $z < 0$, on underside of surface.

[3] $z \geq 0$, top side of surface.

[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of ISSHAD.

### Diagnostics

(Surface not all within picture)$^2$
(SURFACE OMITTED: CONSTANT DATA)$^1$

### See also

FNAREA, IS*, ISSHAD, LIMSFN, RIDDLE, SFEQZ, SFEQZD, SFLIMS, SFMESH and SFZSCL.

## SUBROUTINE GETXY (VARX, VARY, ICODE)

### Name

GETXY – to input the coordinate of a data point from an interactive device.

**Availability** Section 4, released version 2-7.

### Arguments

| | | |
|---|---|---|
| VARX, VARY | REAL variables | To receive coordinates of a point |
| ICODE | INTEGER variable | To receive value identifying method of input |

### Description

Some graphics devices, when used interactively, are capable of allowing graphics input, *ie.* a device operation can be initiated to allow the user to select a point on the display area, and the coordinates of the point be transmitted to the computer. GETXY inputs the coordinates, $(x, y)$, of a point on an interactive device.

VARX and VARY must be names of REAL variables to receive the user coordinates of a point, and ICODE must be the name of an INTEGER variable to receive the code number of the key used for input. A command is sent to the terminal to prepare to receive graphics input, then a reply is requested in whatever form the device allows.

When the input is received by Simpleplot, the code of the key pressed (*eg.* which keyboard key was used, or which button was pressed) is placed in ICODE, and the coordinates are converted to $(x, y)$ user coordinates and placed in VARX and VARY. The conversion subroutine KXYXY can be used to convert user coordinates into page coordinates or device coordinates.

Please note that GETXY only operates on devices with input capability, and which have suitable device drivers and the necessary physical connections.

### Diagnostics

(Graphics input failed)[1]
(Graphics input not available)[1]

### See also

KXYXY.

## SUBROUTINE GROUP (NHORIZ, NVERT)

### Name

`GROUP` – to specify how pictures are to be grouped on the Simpleplot page.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `NHORIZ` | `INTEGER` expression | Number of pictures to be placed horizontally |
| `NVERT` | `INTEGER` expression | Number of pictures to be placed vertically |

### Description

By default, Simpleplot positions pictures one per page. `GROUP` specifies that pictures are to be grouped together on a page in a configuration of `NHORIZ` pictures across by `NVERT` down.

After `GROUP` has been called, the next *new picture* subroutine positions the picture in the top left of a new Simpleplot page; subsequent new pictures fill the group left to right, top to bottom. When enough pictures have been drawn to fill the group, the next new picture automatically starts a new group (on a new Simpleplot page).

If either the margins or the picture size is changed after a group has been started, subsequent pictures are drawn using the changed sizes but the total group area remains fixed. Such changes may therefore affect the number of pictures which can be accommodated in the group.

`GROUP` finishes the plotting on the current Simpleplot page if a picture has been started on it; this is to avoid any confusion due to the resetting of page layout.

There are no explicit limits to the number of pictures in a group but there may be practical constraints due to other elements of the layout.

### Diagnostics

`(Grouping discontinued)`[1]

### Default

Calling `GROUP` with either argument set to zero or both arguments set to 1 restores the default.

### See also

`BOXGRP`, `ENDPAG`, `NEWPIC`, `NEWPAG` and `PAGVW`.

## SUBROUTINE HSTCLC (DARR, NDARR, X1, DX, FARR, NFARR)

### Name

HSTCLC – to calculate data frequencies from raw data.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| DARR | REAL array | Raw data |
| NDARR | INTEGER expression | Number of elements in DARR |
| X1 | REAL variable | Offset value for subdivisions |
| DX | REAL variable | Frequency interval |
| FARR | REAL array | To receive frequency table |
| NFARR | INTEGER expression | Number of elements in FARR |

### Description

HSTCLC scans the array DARR and builds a frequency table in the array FARR. The value in DX when HSTCLC is called is used as the frequency interval unless it is found that the number of intervals would exceed NFARR, in which case a suitable frequency interval is chosen and the value of DX is reset. Interval subdivisions are placed at X1 plus integer multiples of DX, and X1 is reset to the DARR value at the interval subdivision at the start of FARR(1). DX and X1 must be names of variables which have already been given values before HSTCLC is called because their values are used, and may be changed.

### See also

HSTCUM and HSTPLT.

## SUBROUTINE HSTCUM (X1, DX, FARR, NARR)

### Name

HSTCUM – to draw a cumulative histogram from frequencies.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X1 | REAL expression | Offset value for subdivisions |
| DX | REAL expression | Frequency interval |
| FARR | REAL array | Frequency table |
| NARR | INTEGER expression | Number of elements in FARR |

### Description

HSTCUM draws a cumulative histogram on the current picture from frequency values in FARR. Frequencies can be calculated from raw data using HSTCLC.

### Diagnostics

(HISTOGRAM OMITTED: ZERO INTERVAL)[1]

### See also

HSTSHD, HSTCLC and HSTPLT.

## SUBROUTINE HSTDIR (XTORF, FTORF)

### Name

`HSTDIR` – to specify the directions of scales for histograms drawn by `HSTGRM`.

**Availability** Section 4, released version 2-11.

### Arguments

|  |  |  |
|---|---|---|
| `XTORF` | `LOGICAL` expression | Whether to reverse direction of $x$ scale |
| `FTORF` | `LOGICAL` expression | Whether to reverse direction of frequency scale |

### Description

By default, the scales on a histogram drawn by `HSTGRM` are calculated to fit the data exactly and are configured such that scales go from a small value to a larger value. `HSTDIR` specifies any combination of scale directions, by indicating for each whether to reverse the $x$ scale and/or the frequency (or $y$) scale; the range of the scales can be specified by `HSTRNG`.

### Default

`CALL HSTDIR(.FALSE.,.FALSE.)` restores the default.

### See also

`HSTRNG` and `HSTGRM`.

## SUBROUTINE HSTGRM (DARR, NARR, X1, DX, CAPX, CAPY)

### Name

HSTGRM – to draw a histogram from raw data on the current picture.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| DARR | REAL array | Raw data |
| NARR | INTEGER expression | Number of elements in DARR |
| X1 | REAL expression | Offset value for subdivision |
| DX | REAL expression | Frequency interval |
| CAPX | STRING expression | $x$-axis caption |
| CAPY | STRING expression | $y$-axis caption |

### Description

HSTGRM builds a frequency table from the values in the array DARR, and draws a histogram with annotated axes on the current picture.

DX specifies the required frequency interval; if DX=0.0, a suitable frequency interval is chosen internally. Interval subdivisions are placed at all values of the form $x_1 + n, dx$ (where $n$ is any INTEGER and $dx$ is the frequency interval) within the range of values in the array DARR.

HSTGRM draws an $x$-axis and a $y$-axis which can be configured using appropriate AX* subroutines with CHAXIS='XC' and 'YC' respectively. The $x$-axis is labelled with CAPX and the $y$-axis labelled with the caption CAPY. By default, the number of $x$-axis subdivisions relates directly to the frequency interval, DX, therefore AXSBDV should be used with care.

By default, a histogram displaying frequencies is drawn and the boxes are unshaded; the scales are calculated to fit the data exactly and both scales run in the natural direction of the corresponding axis. These default settings can be altered by HSTTYP, HSTSHD, HSTRNG and HSTDIR respectively.

Current plotting scales are ignored by HSTGRM but not reset.

### See also

HSTDIR, HSTRNG, HSTTYP and HSTSHD.

## SUBROUTINE HSTPLT (X1, DX, FARR, NARR)

### Name

`HSTPLT` – to draw a histogram from frequencies.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `X1` | `REAL` expression | Offset value for subdivisions |
| `DX` | `REAL` expression | Frequency interval |
| `FARR` | `REAL` array | Frequency table |
| `NARR` | `INTEGER` expression | Number of elements in `FARR` |

### Description

`HSTPLT` draws a histogram on the current picture from frequency values in `FARR`. Frequencies can be calculated from raw data using `HSTCLC`.

### Diagnostics

`(HISTOGRAM OMITTED: ZERO INTERVAL)`[1]

### See also

`HSTSHD`, `HSTCLC` and `HSTCUM`.

## SUBROUTINE HSTRNG (XMIN, XMAX, FMIN, FMAX)

### Name

HSTRNG – to specify the range of scales for histograms drawn by HSTGRM.

**Availability** Section 4, released version 2-11.

### Arguments

| | | |
|---|---|---|
| XMIN | REAL expression | Minimum horizontal scale value |
| XMAX | REAL expression | Maximum horizontal scale value |
| FMIN | REAL expression | Minimum vertical scale value |
| FMAX | REAL expression | Maximum vertical scale value |

### Description

By default, the scales of a histogram drawn by HSTGRM are calculated to accommodate the data exactly. HSTRNG specifies alternative range of scales (the direction of scales can be specified separately by HSTDIR).

If the specified scales can not accommodate the entire histogram, part of the picture cannot be drawn. This contributes to the number of incomplete picture tasks and a diagnostic is issued at the end of the picture.

Current plotting scales are ignored by HSTGRM but not reset.

### Default

Calling HSTRNG with XMAX$\leq$XMIN and/or FMAX$\leq$FMIN restores the default. *eg.* CALL HSTRNG(0.0,0.0,0.0,0.0).

### See also

HSTGRM and HSTDIR.

## SUBROUTINE HSTSHD (ISHADE)

**Name**

HSTSHD – to specify a shading pattern for the boxes used to draw a histogram.

**Availability** Section 4, released version 2-11.

**Argument**

| | | |
|---|---|---|
| ISHADE | INTEGER expression | Shading pattern number |

**Description**

By default, histograms are drawn with unshaded boxes. To draw a shaded histogram (using HSTGRM, HSTPLT or HSTCUM) a single shading pattern can be selected by HSTSHD.

**Default**

CALL HSTSHD(-1) restores the default.

**See also**

SHSET and SHEDGE.

## SUBROUTINE HSTTYP (ITYPE)

### Name

`HSTTYP` – to specify the type of histogram to be plotted by `HSTGRM`.

**Availability** Section 4, released version 2-11.

### Argument

`ITYPE` `INTEGER` expression Type of histogram

| `ITYPE` | *Type of histogram* |
|---|---|
| 1 | Standard histogram |
| 2 | Cumulative histogram |
| 3 | Complementary (or inverse) cumulative histogram |

### Description

By default, `HSTGRM` draws a standard histogram in which the boxes represent the frequencies calculated from the raw data. `HSTTYP` changes the type of histogram drawn.

`HSTTYP` must be called before `HSTGRM`.

### Default

`CALL HSTTYP(1)` restores the default.

### See also

`HSTGRM`.

## SUBROUTINE INITBL

**Name**

INITBL – to cancel all current blanked areas.

**Availability** Section 1, released version 2-14.

**Arguments**

None.

**Description**

INITBL cancels all areas blanked by CHMASK, DEFCAP and MKMASK. Subsequent drawing is only masked against newly defined blanked areas.

The use of masked areas affects the speed of a plot, so INITBL results in faster output.

**See also**

CHMASK, DEFCAP and MKMASK.

## SUBROUTINE INITSP

### Name

`INITSP` – to reset all SIMPLEPLOT defaults.

**Availability** Section 1, released version 2-5.

### Arguments

None.

### Description

All *specification* or *reset* subroutines change SIMPLEPLOT's default behaviour.

The documentation for each specification subroutine includes details of how to restore this default behaviour individually.

`INITSP` restores the default behaviour of all subroutines simultaneously. There are a few exceptions to this rule: the specification subroutines which control characteristics of the I/O interface such as the diagnostic level and the current device are not reset by `INITSP`. The subroutines in this category include `DIAGLV`, `DEVNAM`, `DEVNO`, `IOCHAN`, `OWNNEW`, `OWNFIL`. Similarly, `INITSP` does not reset the device palette (defined by `PENRGB` or `PENHLS`) nor does it affect metafile I/O.

`INITSP` finishes the plotting on the current SIMPLEPLOT page regardless of whether it contains a picture. This is to avoid any confusion due to the resetting of page layout.

### See also

`ENDPLT`.

## SUBROUTINE IOCHAN (ITYPE, ICHAN)

### Name

`IOCHAN` – to specify a value to be associated with a SIMPLEPLOT internal I/O channel.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `ITYPE` | `INTEGER` expression | Type of channel usage |
| `ICHAN` | `INTEGER` expression | Value to be associated with that channel |

| `ITYPE` | *Channel usage* |
|---|---|
| 1 | Diagnostics |
| 2 | Prompt output |
| 3 | Prompt input |
| 4 | Graphics output |
| 5 | Graphics input |
| 6 | Configuration |
| 8 | Scratch file used for shading |
| 9 | Hershey font file |
| 10 | Scratch file used for metafiles |

### Description

By default, SIMPLEPLOT allocates all input and output channels internally. `IOCHAN` specifies an alternative value (*eg.* FORTRAN logical unit number, GKS connection identifier *etc.*) for the type of usage described by `ITYPE`.

Valid values of `ICHAN` and their effect may depend on the host processor, device driver system and (where `ITYPE`=4, 5 or 6) the individual device driver in use.

In order to allow `IOCHAN` to be used repeatedly before SIMPLEPLOT is fully initialized, it does not trigger the initial diagnostic message:

```
(SIMPLEPLOT Mark 2-15(nnn)X)
```

### Default

`CALL IOCHAN(ITYPE,-1)` restores the default.

The default is not restored by `INITSP`.

### See also

`MTFILE`, `OWNFIL` and Appendix H, *Host Specific Information.*

## SUBROUTINE ISANG (ANGLE)

### Name

`ISANG` – to specify the angle at which surface pictures are drawn.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `ANGLE` | `REAL` expression | Angle in degrees |

### Description

By default, surfaces are drawn such that the angle between the horizontal plane and the base plane is 15.0°. `ISANG` specifies an alternative angle.

The viewing corner of the data (see `ISVIEW`) is positioned at the front of the picture before rotation is performed, and therefore will not be at the front of the picture for angles between 90.0° and 270.0°.

### Default

`CALL ISANG(15.0)` restores the default.

### See also

`ISVIEW`.

## SUBROUTINE ISAXD7 (CAPX, CAPY, CAPZ)
## SUBROUTINE ISAXDR (CAPX, NCAPX, CAPY, NCAPY, CAPZ, NCAPZ)

### Name

`ISAXD7` – to draw 3-D axes on an isometric picture.

**Availability** Section plus, released version 2-9.

### Arguments

| | | |
|---|---|---|
| `CAPX` | `STRING` expression | $x$-axis caption |
| `CAPY` | `STRING` expression | $y$-axis caption |
| `CAPZ` | `STRING` expression | $z$-axis caption |
| `NCAPX` | `INTEGER` expression | Number of characters in `CAPX` |
| `NCAPY` | `INTEGER` expression | Number of characters in `CAPY` |
| `NCAPZ` | `INTEGER` expression | Number of characters in `CAPZ` |

### Description

By default, the surface pictures drawn by the `*SURF` subroutines do not include axes. `ISAXDR`/`ISAXD7` draws $x$-$y$-$z$ axes on the current surface picture.

The $x$-$y$-$z$ diagram indicating the directions of change of $x$, $y$ and $z$, can be omitted by calling `ISDIAG`.

### Diagnostics

`(Isometric axes inappropriate)`[2]

### See also

`AXIS7`, `ISAXES` and `ISDIAG`.

## SUBROUTINE ISAXES (TORF)

### Name

`ISAXES` – to specify whether surface pictures are to include 3-D axes.

**Availability** Section plus, released version 2-9.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether axes required |

### Description

By default, surface pictures drawn by `FNSURF`, `RGSURF`, `XSURF`, `XYSURF`, `YSURF`, `ZSURF` and `ZZSURF` do not have axes on them. `ISAXES` specifies whether or not $x$-$y$-$z$ axes are to be drawn as part of a surface picture.

Captions `'X'`, `'Y'` and `'Z'` are automatically given to the $x$-axis, $y$-axis and $z$-axis respectively.

If `ISAXES` is active, the $x$-$y$-$z$ diagram normally controlled by `ISDIAG` is omitted.

### Default

`CALL ISAXES(.FALSE.)` restores the default.

### See also

`AXIS7`, `ISAXD7` and `ISDIAG`.

## SUBROUTINE ISAXFC (IXFIX, IYFIX, IZFIX)

**Name**

ISAXFC – to specify the colours for filled rear planes of *SIMPLEPLOT Isometric* pictures.

**Availability** Section plus, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| IXFIX | INTEGER expression | Fill index for rear $x$ plane |
| IYFIX | INTEGER expression | Fill index for rear $y$ plane |
| IZFIX | INTEGER expression | Fill index for rear $z$ plane |

**Description**

ISAXFC specifies colour indices for filling the rear planes of *SIMPLEPLOT Isometric* pictures. If a non-negative value is specified for any plane, it is filled whenever a new image is started. ISAXFC must therefore be called before ISNEW.

IXFIX specifies the colour index for the rear plane(s) with $x$ constant, IYFIX specifies the colour index for the rear plane(s) with $y$ constant, and IZFIX specifies the colour index for the rear plane(s) with $z$ constant.

If colour index $-1$ is specified, the associated plane is not filled. This is the default.

**Default**

CALL ISAXFC(-1,-1,-1) restores the default.

**Diagnostics**

[INTEGER out of range][1]

**See also**

ISAXGC.

## SUBROUTINE ISAXGC (IXCIX, IYCIX, IZCIX)

### Name

ISAXGC – to specify the colour indices for grid lines on the rear planes of *SIMPLEPLOT Isometric* pictures.

**Availability** Section plus, released version 2-14.

### Arguments

| | | |
|---|---|---|
| IXCIX | INTEGER expression | Colour index for $x$ |
| IYCIX | INTEGER expression | Colour index for $y$ |
| IZCIX | INTEGER expression | Colour index for $z$ |

### Description

ISAXGC specifies colour indices for grid lines on the rear planes of the *SIMPLEPLOT Isometric* picture. When grids have been requested, they are drawn in the raster image whenever a new image is started. ISAXGC must therefore be called before ISNEW.

IXCIX specifies the colour index for grid lines on the rear plane(s) with $x$ constant, IYCIX specifies the colour index for grid lines on the rear plane(s) with $y$ constant, and IZCIX specifies the colour index for grid lines on the rear plane(s) with $z$ constant.

If index $-1$ is specified, the associated grids are omitted. This is the default.

### Default

CALL ISAXGC(-1,-1,-1) restores the default.

### Diagnostics

[ISAXGC: Colour index *I* out of range][1]

### See also

ISAXFC.

### SUBROUTINE ISBRBZ (ZVAL)

**Name**

`ISBRBZ` – to specify the base $z$ value for bars in *SIMPLEPLOT Isometric* 3-D barcharts.

**Availability** Section plus, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `ZVAL` | `REAL` expression | Required base $z$ value |

**Description**

By default, *SIMPLEPLOT Isometric* 3-D barcharts show each bar extending from zero $z$, to a $z$ value from an array. `ISBRBZ` can be called before `ISBRFL` to set an alternative base z value.

**Default**

`CALL ISBRBZ(0.0)` restores the default.

**See also**

`ISBRFL`.

## SUBROUTINE ISBRFC (ICODE)

### Name

ISBRFC – to specify how colour sequences should be used on subsequent *SIMPLEPLOT Isometric* 3-D barcharts.

**Availability** Section plus, released version 2-14.

### Arguments

ICODE INTEGER expression Usage (see below)

| ICODE | *Different colours* |
|---|---|
| 0 | Faces of bars distinguished by pen pointers (Default) |
| 1 | Data Rows distinguished by colour sequence |
| 2 | Data Columns distinguished by colour sequence |
| 3 | Data Sets distinguished by colour sequence |
| 4 | Data magnitudes distinguished by colour sequence |

### Description

ISBRFL fills a *SIMPLEPLOT ViSualization* 3-D barchart. ISBRFC can be called before ISBRFL, to specify how colours are used for the bars.

When ICODE $= 0$, three colour indices distinguish the three planes of each bar: the $x$ plane uses the first colour index, the $y$ plane uses the second, and the $z$ plane uses the third.

When ICODE $> 0$, all planes of a bar are shaded with the same colour index. SQSHAD may be called to specify the sequence of colour indices. When SQSHAD has not been called, the colour indices are used in the natural sequence 1, 2, 3, ...

The edges of each plane are drawn using pen pointer [4]. Pen pointer values are controlled by SETPNS.

### Default

CALL ISBRFC(0) restores the default.

### See also

SQSHAD.

## SUBROUTINE ISBRFL (D3ARR, NX, NY, NSETS)

### Name

ISBRFL – to fill a *SIMPLEPLOT Isometric* 3-D barchart.

**Availability** Section plus, released version 2-14.

### Arguments

| | | |
|---|---|---|
| D3ARR | REAL 3-D array | NX × NY × NSETS array of user ($z$) values |
| NX, NY, NSETS | INTEGER expressions | Dimensions of 3-D data array |

### Description

ISBRFL starts a new picture and draws a *SIMPLEPLOT Isometric* 3-D barchart. The barchart represents NSETS sets of values in array D3ARR, stacked on top of each other. The sets may be stacked downward with the first set at the top, or upward with the first set at the bottom, as specified by ISBRUP.

Different colours may be used to distinguish between data columns, data rows, or data sets, or data magnitudes, as specified by ISBRFC.

If all the values in D3ARR are equal to the base $z$ value set by ISBRBZ, a diagnostic is issued and nothing is drawn.

### Diagnostics

```
(INVALID ARGUMENT: ARRAY SIZE)[1]
(BAR CHART OMITTED: NULL RANGE)[1]
```

### See also

ISBRBZ, ISBRFC and ISBRSZ.

## SUBROUTINE ISBRSZ (X1, X2, Y1, Y2)

### Name

`ISBRSZ` – to specify the widths of bars in *SIMPLEPLOT Isometric* 3-D barcharts.

**Availability** Section plus, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `X1`, `X2` | `REAL` expressions | Limits of bars in $x$ direction |
| `Y1`, `Y2` | `REAL` expressions | Limits of bars in $y$ direction |

### Description

By default, each bar in *SIMPLEPLOT Isometric* 3-D barcharts extends from 0.1 to 0.9 of the space available in both $x$ and $y$. `ISBRSZ` can be called to specify alternative limits.

When `X2` is not greater than `X1`, or either lies outside the range 0.0 to 1.0, the default in $x$ is restored; when `Y2` is not greater than `Y1`, or either lies outside the range 0.0 to 1.0, the default in $y$ is restored.

### Default

`CALL ISBRSZ(0.0, 0.0, 0.0, 0.0)` restores the default.

### See also

`ISBRFL`.

## SUBROUTINE ISBRUP (IUP)

### Name

`ISBRUP` – to specify the stacking direction for *SIMPLEPLOT Isometric* 3-D barcharts.

**Availability** Section plus, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `IUP` | `INTEGER` expression | Whether bars stack upwards |

| `IUP` | *Direction* |
|---|---|
| 0 | Downward (Default) |
| 1 | Upward |

### Description

`ISBRUP` is called before `ISBRFL` to specify whether multiple data sets should be stacked downward with the first set at the top, or upward with the first set at the bottom.

### Diagnostics

`[ISBRUP: Integer` *I* `out of range]`[1]

### Default

`CALL ISBRUP(0)` restores the default.

### See also

`ISBRFL`.

## SUBROUTINE ISCURV (NSTEPS)

### Name

ISCURV – to specify the smoothness of curves on crosshatched and cascade surface pictures.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| NSTEPS | INTEGER expression | Number of steps |

### Description

Outline surfaces are drawn as rapidly as possible using straight lines between points on an underlying grid; however, the curves used to draw the mesh lines on crosshatched and cascaded pictures (see ISTYPE) are interpolated between mesh intersections to improve the picture. By default, the number of steps used depends on the picture size; fewer steps are used on smaller pictures. ISCURV specifies the number of steps when interpolating between grid intersections.

After ISCURV has been called, each piece of curve between intersections is drawn in NSTEPS steps, interpolated on a smooth curve through the intersections. Small values of NSTEPS lead to faster processing, but less smooth drawings.

The fineness of the underlying mesh can be controlled by ISMESH or SFMESH.

### Default

Calling ISCURV with NSTEPS less than 2 (the minimum value possible) restores the default.

### See also

ISMESH and SFMESH.

## SUBROUTINE ISDIAG (TORF)

### Name

`ISDIAG` – to specify whether surface pictures are to include an *x*-*y*-*z* diagram.

**Availability** Section 2, released version 2-6.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether a diagram is required |

### Description

By default, a diagram is drawn alongside surfaces with arrows indicating the directions of change of the 3 variables, $x$, $y$ and $z$. `ISDIAG` specifies whether or not an *x*-*y*-*z* diagram is to be drawn as part of a surface picture. `ISDIAG` must be called before a surface is drawn.

If `ISAXES` is active, the *x*-*y*-*z* diagram is omitted regardless of `ISDIAG`.

### Default

`CALL ISDIAG(.TRUE.)` restores the default.

### See also

`ISAXES`.

### SUBROUTINE ISFULL (TORF)

**Name**

`ISFULL` – to specify whether surface pictures fill the space available.

**Availability** Section 2, released version 2-7.

**Argument**

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether to fill the space |

**Description**

By default, surface pictures are drawn such that all rotations of the surface can be drawn to the same scale within the picture size. This allows comparisons of rotated pictures but can result in a very small picture with a large empty area around it. `ISFULL` specifies whether or not surfaces are to be drawn to reach the picture limits.

After a call of `ISFULL(.TRUE.)`, each surface picture is drawn on a scale to make the surface as big as possible within the picture limits but without changing the aspect ratio.

**Default**

`CALL ISFULL(.FALSE.)` restores the default.

**See also**

`ISRISE`.

## SUBROUTINE ISMESH (MXY)

### Name

`ISMESH` – to specify the fineness of detail in surface pictures.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `MXY` | `INTEGER` expression | Total number of mesh lines used in $x$ and $y$ |

### Description

The processing of all surface pictures is based on the same mesh of lines at equally-spaced $x$ values and equally-spaced $y$ values as are drawn on crosshatched (and cascaded) surfaces. By default, for data on a regular grid the data grid is used, and for all other types of data `MXY`=20 is used. `ISMESH` can be to specify the total number of $x$ lines and $y$ lines ($\texttt{MXY} = \texttt{MX} + \texttt{MY}$).

The relative values of `MX` and `MY` determine the relative lengths of the $x$ scale and $y$ scale in the picture. The distribution of `MXY` between `MX` and `MY` depends on the data type and the $x$-$y$ ranges of the data. For equally-spaced gridded data, if either of the $x$-$y$ scales has not been specified, `MX` and `MY` are given values proportional to the numbers of data values in the $x$ and $y$ directions; if both of the $x$-$y$ scales are known, `MX` and `MY` are set in a similar ratio to $x$-range:$y$-range, unless that ratio would produce either `MX` or `MY` less than 5, in which case `MX`=`MY`=`MXY`/2.

Larger values of `MX` and `MY` produce surfaces with finer detail, and cross-hatched pictures with a closer mesh, but the processing is necessarily slower.

`ISMESH` has no effect while `SFMESH` is active.

### Default

Calling `ISMESH` with `MXY` less than 4 restores the default.

### See also

`ISCURV` and `SFMESH`.

## SUBROUTINE ISNEW

### Name

`ISNEW` – to start a null isometric picture.

**Availability** Section 2, released before version 2-5.

### Arguments

None.

### Description

`ISNEW` starts a new picture and sets up scales for three-dimensional plotting. The scales are the same as would be set for surface pictures – the $x$-$y$ scales cover the ranges specified by a previous call of `FNAREA` (or default to 0.0 to 10.0 for both $x$ and $y$); the $z$ scale is as specified by a previous call of `SFZSCL` (or the same default, 0.0 to 10.0).

`ISNEW` is a *graphics* subroutine but, by default, does no drawing. If one of the `BOX*` subroutines has been called to generate boxes around pictures, groups or pages, these are drawn [1] when a new picture is started.

Any point $(x, y, z)$ on this new isometric picture can be addressed as a Cartesian pair $(x, y)$ using the conversion subroutine `KISXY`. Hence, any of the graph drawing subroutines (*eg.* `CP7PT`, `BRKNPT`, `MARKPT`) can be used on an isometric picture.

### See also

`FNAREA`, `SFZSCL` and `KISXY`.

## SUBROUTINE ISRISE (FACTOR)

### Name

`ISRISE` – to specify the proportions of surface pictures.

**Availability** Section 2, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `FACTOR` | `REAL` expression | Specifies the maximum height of the surface above the base plane |

### Description

By default, the $z$ scale of surface pictures is set by assigning some length on the picture to cover the range of surface heights to be represented. `ISRISE` specifies that the length to cover the range of surface heights on surface pictures is to be `FACTOR` × *picture width* × cos(`ANGLE`) where `ANGLE` is the angle (in degrees) between the horizontal plane and the base plane as specified by `ISANG`.

### Default

`CALL ISRISE(0.4)` restores the default.

### See also

`ISANG` and `ISFULL`.

### SUBROUTINE ISSBAS (IBASE)

**Name**

ISSBAS – to specify the type of base on shaded surface pictures.

**Availability** Section plus, released version 2-13.

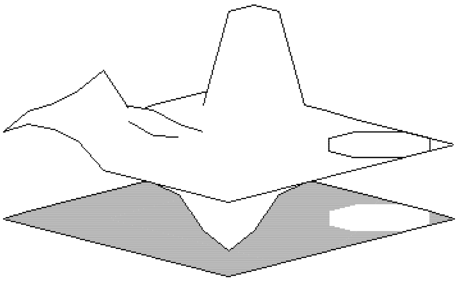
**Argument**

| | | |
|---|---|---|
| IBASE | INTEGER expression | Type of shading used for base |

| IBASE | *Type of shading used* |
|---|---|
| 0 | None (default) |
| 1 | Solid shading but only when skirts are also drawn |
| 2 | Solid shading (see ISSPAT) |

**Description**

By default, shaded surfaces are drawn with neither a skirt, a backdrop nor a base. ISSBAS specifies the method used for drawing a base.



IBASE specifies the type of shading to be used for drawing the base at the foot of the skirts.

By default, shading pattern 0 (rub-out) is used for solid shading but an alternative pattern can be selected using ISSPAT.

**Default**

CALL ISSBAS(0) restores the default.

**See also**

ISSBCK, ISSHAD, ISSKRT, ISSPAT, ISTYPE, RASTER and SQSHAD.

## SUBROUTINE ISSBCK (IBACK)

**Name**

`ISSBCK` – to specify the type of backdrops on shaded surface pictures.

**Availability** Section plus, released version 2-13.

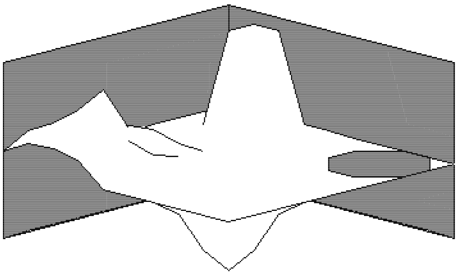
**Argument**

`IBACK` `INTEGER` expression Type of shading used for backdrop

| `IBACK` | *Type of shading used* |
|---|---|
| 0 | None (default) |
| 1 | Solid shading; default pattern 0 (see `ISSPAT`) |
| 2 | Vertical lines |
| 3 | Horizontal lines parallel to the $x$-$y$ plane |
| 4 | Shaded horizontal bands, parallel to the $x$-$y$ plane |

**Description**

`IBACK` specifies the type of shading used for the *backdrop*, the area behind the surface which covers the full range of the $z$-axis.



By default, shading pattern 0 (rub-out) is used for solid shading but an alternative pattern can be selected using `ISSPAT`.

**Default**

`CALL ISSBCK(0)` restores the default.

**See also**

`ISSBAS`, `ISSHAD`, `ISSKRT`, `ISSPAT`, `ISTYPE`, `RASTER` and `SQSHAD`.

## SUBROUTINE ISSHAD (IMETHD)

### Name

ISSHAD – to specify the method of shading used on shaded surface pictures.

**Availability** Section plus, released version 2-13.

### Argument

IMETHD INTEGER expression Method of shading

| IMETHD | *Shading method* |
|---|---|
| 0 | No shading (default) |
| 1 | Shading in whole quadrilaterals (fastest) |
| 2 | Quadrilaterals split into 2 |
| 3 | Quadrilaterals split into 4 |
| 4 | Shading corresponding to contours (slowest but most accurate) |

### Description

By default, the *SURF subroutines draw a surface represented by lines but with no shading. ISSHAD specifies the method for shading subsequent surfaces according to the value of IMETHD.

By default, the shaded surface is drawn as a raster picture (see RASTER) using the currently selected fast area-fill characters. Shaded surfaces may includelines according to a prior call of ISTYPE (*eg.* outlines, crosshatching). Axes, text and keys are not drawn as part of this raster picture. The fineness of detail of the raster image is controlled by FFSIZE.

### Default

CALL ISSHAD(0) restores the default.

### See also

ISSBAS, ISSBCK, ISSKRT, ISSPAT, ISTYPE and RASTER.

## SUBROUTINE ISSKRT (ISKIRT)

### Name

`ISSKRT` – to specify the type of skirts on shaded surface pictures.

**Availability** Section plus, released version 2-13.

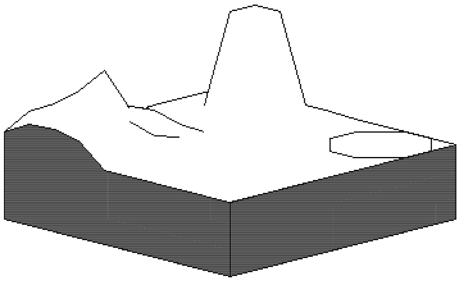
### Argument

`ISKIRT` `INTEGER` expression Type of shading used for skirts

| `ISKIRT` | *Type of shading used* |
|---|---|
| 0 | None (default) |
| 1 | Solid shading; default pattern 9 (see `ISSPAT`) |
| 2 | Vertical lines superimposed over solid shading |
| 3 | Horizontal lines parallel to the $x$-$y$ plane |
| 4 | Shaded horizontal bands, parallel to the $x$-$y$ plane |

### Description

`ISKIRT` specifies the type of shading used for *skirts*, the area on the picture between the edge of the surface and the plane of the $x$ and $y$ axes.



### Default

`CALL ISSKRT(0)` restores the default.

### See also

`ISSBAS`, `ISSBCK`, `ISSHAD`, `ISSPAT`, `ISTYPE`, `RASTER` and `SQSHAD`.

## SUBROUTINE ISSPAT (CHFACE, ISHADE)

### Name

`ISSPAT` – to specify the shading patterns to be used on shaded surfaces.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `CHFACE` | `CHARACTER*2` | Face of skirt, backdrop or base |
| `ISHADE` | `INTEGER` expression | Shading pattern of the face |

### Description

By default, shaded surfaces are drawn without skirts, backdrops or a base. If `ISSBAS`, `ISSBCK` or `ISSKRT` has been called to specify shaded skirts, backdrops and/or base, the default shading pattern used is pattern 0 (rub-out). `ISSPAT` selects an alternative.

Individual faces can be selected for shading, with the 2 character string `CHFACE`. The first character of `CHFACE` refers to the face (`'A'` to `'D'` for skirts/backdrops, `'G'` for bases). The second character refers to the position (`'I'`nner, `'O'`uter or `'B'`ackdrop). The possible interpretations of `CHFACE` are listed in Appendix T of the *SIMPLEPLOT Supplement*.

For ungridded data, or rectangular data with no-data values, skirts do not necessarily occur at the rectangular faces; skirts dropped from holes and from surfaces where the boundary is not along a face, are identified by `CHFACE='ND'`.

A wildcard character, `'*'`, can be used in place of either character of `CHFACE` to denote a set of faces. For example, `CHFACE='**'` for all faces of skirts, backdrops and the base; `'*B'` for all backdrops; `'A*'` for the inner skirt, outer skirt and backdrop associated with face A.

If no skirt, backdrop nor base is drawn, or shaded contour bands are drawn, `ISSPAT` has no effect.

### Diagnostics

`(Invalid shading area: `*`XX`*`)`[1]

### Default

`CALL ISSPAT('**',0)` restores the default.

### See also

`ISSBAS`, `ISSBCK`, `ISSKRT`, `ISSHAD`, `RASTER`, `SHSET` and `SQSHAD`.

## SUBROUTINE ISTYPE (ITYPE)

**Name**

ISTYPE – to specify the type of surface picture.

**Availability** Section 2, released before version 2-5.

**Argument**

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Type of picture |

| ITYPE | *Type of surface picture* |
|---|---|
| 0 | No lines drawn |
| 1 | Surface outline (default) |
| 2 | Surface outline and a set of contours |
| 3 | Crosshatched picture |
| 4 | Set of curves of $z$ against $x$<br>for a set of equally-spaced $y$ values |
| 5 | Set of curves of $z$ against $y$<br>for a set of equally-spaced $x$ values |

**Description**

By default, the *SURF subroutines draw a surface outline with no shading. ISTYPE specifies an alternative type of surface picture. The number of curves drawn in crosshatched and cascaded pictures (ITYPE=3, 4 or 5) can be controlled by ISMESH or SFMESH.

**Default**

CALL ISTYPE(1) restores the default.

**See also**

ISMESH, ISSHAD and SFMESH.

## SUBROUTINE ISVIEW (ICORNR)

### Name

ISVIEW – to specify the view point for surface pictures.

**Availability** Section 2, released before version 2-5.

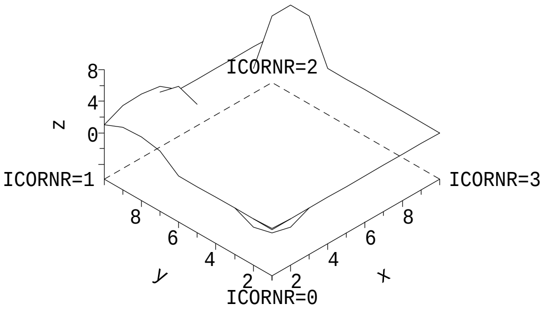
### Argument

| | | |
|---|---|---|
| ICORNR | INTEGER expression | Viewing corner |

### Description

Each surface picture covers a rectangular region of the $x$-$y$ plane, even when the data structure plotted by *SURF is non-rectangular. By default, surface pictures represent the surface viewed with one of the corners of the rectangle at the front, and the opposite corner at the back.

By default, the minimum $x$ and minimum $y$ are at the front. ISVIEW specifies an alternative corner to be at the front. ICORNR specifies the viewing corner; corners are numbered 0, 1, 2, 3 in a clockwise direction, starting from 0 at the first data point.



The viewing corner of the data is positioned at the front of the picture *b*efore any rotation is performed (see ISANG), and therefore will not be at the front of the picture for angles between 90.0° and 270.0°.

### Default

CALL ISVIEW(0) restores the default.

### See also

ISANG and ISYUP.

### SUBROUTINE ISYUP (TORF)

**Name**

ISYUP – to specify the direction of change of $y$ on surface pictures.

**Availability** Section 2, released before version 2-5.

**Argument**

| | | |
|---|---|---|
| TORF | LOGICAL expression | Whether $y$ increases upwards |

**Description**

On Cartesian pictures, $x$ is conventionally represented as increasing from left to right, and $y$ as increasing from the bottom upwards. Tabulated data is also presented with $x$ increasing from left to right, but $y$ usually increases downwards.

By default, on surface pictures, data is treated in a conventional Cartesian sense with $y$ values increasing upwards. ISYUP specifies the direction of change of $y$ on surface pictures. If TORF=.FALSE., then $y$ values are represented as increasing downwards, as is conventional for tables.

**Default**

CALL ISYUP(.TRUE.) restores the default.

**See also**

ISVIEW.

## SUBROUTINE JOINPT (X, Y)

### Name

JOINPT – to draw a straight line to the point $(x, y)$ (or $(R, \theta)$)).

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point specified in units of the plotting scales |

### Description

JOINPT draws a straight line to the point $(x, y)$, or $(R, \theta)$ if the polar coordinates have been specified by COORDS, EQSCAL or POLAR7. The line uses using the bundled line-drawing attributes of the current pen.

A line is started by the first call to JOINPT, or BRKNPT. The first call of JOINPT/BRKNPT moves to $(x, y)$ without drawing anything; subsequent calls draw a straight line from the previous point to the specified point. The line is straight even on non-linear scales and polar charts.

A line is interrupted explicitly by BREAK, or if any other plotting subroutineis called. The next call of JOINPT/BRKNPT moves to the first point of a new line. MARKPT and CP7PT draw a symbol/caption at a specified $(x, y)$, followed by the equivalent to the first call to JOINPT/BRKNPT. A sequence of symbols joined by straight lines can be drawn by repeating the following calls for a sequence of points:

```
CALL JOINPT(X,Y)
CALL MARKPT(X,Y,MKTYPE)
```

The point $(x, y)$ is specified in units of the current plotting scales (user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, KXYXY.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(**$x, y$**)$^2$

### See also

BREAK, BRKNPT, BUNLPR, CLIPLV, CP7PT and MARKPT.

## SUBROUTINE KISXY (X3, Y3, Z3, X2, Y2)

### Name

KISXY – to convert a point $(x, y, z)$ on a surface picture to its 2-D equivalent.

**Availability** Section 2, released version 2-11.

### Arguments

| | | |
|---|---|---|
| X3, Y3, Z3 | REAL expressions | Coordinates of a point, specified in 3-D picture units |
| X2, Y2 | REAL variables | To receive the coordinates of a point specified in internal 2-D picture units |

### Description

On surface pictures, user coordinates are described by 3 variables $(x, y, z)$ but the underlying coordinate system remains Cartesian and can be addressed as such. Any of the general plotting subroutines can be used by first converting a 3-D user coordinate into a 2-D picture coordinate using KISXY.

If any of the isometric coordinates, X3, Y3 or Z3, is equal to the current no-data value, then both X2 and Y2 are set to this no-data value. All plotting subroutines ignore coordinates for which either ordinate corresponds to the no-data value.

### Diagnostics

(Isometric coordinates inappropriate)[1]

### See also

BRKNPT, CP7LB, CP7PT, CP7XC, CP7YC, JOINPT, MARKPT and NODATA.

## SUBROUTINE KNUMB (IVAL, STR)

### Name

KNUMB – to convert an INTEGER value to the equivalent text string.

**Availability** Section 1, released version 2-11.

### Arguments

| | | |
|---|---|---|
| IVAL | INTEGER expression | Value to be converted |
| STR | STRING variable | To receive text string |

### Description

KNUMB converts an INTEGER value into the equivalent CHARACTER representation.

STR should be declared as a CHARACTER variable long enough to represent IVAL. CHARACTER*10 should be sufficient for most INTEGERs. If STR is not long enough to contain the converted INTEGER, SIMPLEPLOT places as much of the number as possible into STR and issues a diagnostic.

### Diagnostics

(String too short for INTEGER)[1]

### See also

CP7LB, CP7PT, CP7XC, CP7YC and KREAL.

## SUBROUTINE KPOSXY (ITYPE, VCHAR, HCHAR, VCMS, VJST, HCMS, HJST)

### Name

KPOSXY – to convert a descriptive position into page coordinates.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| ITYPE | INTEGER expression | Intended use of position |
| VCHAR | CHARACTER*1 | Vertical position of area |
| HCHAR | CHARACTER*1 | Horizontal position of area |
| VCMS | REAL variable | To receive vertical coordinate |
| VJST | CHARACTER*1 variable | To receive vertical justification |
| HCMS | REAL variable | To receive horizontal coordinate |
| HJST | CHARACTER*1 variable | To receive horizontal justification |

| ITYPE | *Text usage* |
|---|---|
| 1 | Titles |
| 2 | Key or captions areas |

### Description

KPOSXY returns coordinates (HCMS,VCMS) and justification (VJST,HJST) corresponding to a descriptive position (VCHAR,HCHAR) on the current page.

HCMS and VCMS describe the horizontal and vertical page coordinates of a title, key or caption area, specified in centimetres relative to the bottom left-hand corner of the current SIMPLEPLOT page HCHAR, VCHAR, HJST and VJST are all single characters, whose possible interpretations are listed in Appendix T of the *SIMPLEPLOT Supplement*.

KPOSXY can only return sensible values for a picture or page which has already been started. If no picture/page has been started, or if ITYPE is not equal to 1 or 2, a diagnostic is issued and VCMS, HCMS, VJST, and HJST are returned with values 0.0, 0.0, 'B', and 'L' respectively.

### Diagnostics

(KPOSXY argument invalid)[1]<br>
(KPOSXY called in wrong state)[1]

### See also

DEFPOS and KXYXY.

### SUBROUTINE KREAL (RVAL, STR)

**Name**

KREAL – to convert a REAL value to the equivalent text string.

**Availability** Section 1, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| RVAL | REAL expression | Value to be converted |
| STR | STRING variable | To receive text string |

**Description**

KREAL converts any REAL value into the equivalent CHARACTER representation.

STR should be declared as a CHARACTER variable long enough to represent RVAL. Most REAL numbers can be represented by a relatively small number of characters but, if STR is declared as a CHARACTER*30 variable, it should be able to accommodate even exceptionally large or small values. If STR is not long enough to contain the converted REAL, Simpleplot places as much of the number as possible into STR and issues a diagnostic.

**Diagnostics**

(String too short for REAL)[1]

**See also**

CP7LB, CP7PT, CP7XC, CP7YC, FIGFMT and KNUMB.

### SUBROUTINE KSCALE (START, STOP, DIV, VSTART, VSTOP, VDIV)

**Name**

KSCALE – to convert scale limits such that they span whole subdivisions.

**Availability** Section 1, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| START | REAL expression | Value required near beginning of scale |
| STOP | REAL expression | Value required near end of scale |
| DIV | REAL expression | Subdivision required |
| VSTART | REAL variable | To receive value to use at beginning of scale |
| VSTOP | REAL variable | To receive value to use at end of scale |
| VDIV | REAL variable | To receive subdivision value to use |

**Description**

KSCALE takes the values START, STOP and DIV which describe the scale requirements, and converts them to VSTART, VSTOP and VDIV which satisfy the requirements and, if necessary, extend the scales to coincide with whole multiples of the scale subdivision. KSCALE does not affect scales directly, but it computes values which may be used to set scales or axis subdivisions as arguments for SCALES, EQSCAL, AXSBDV.

DIV is the requested subdivision; if DIV is zero, SIMPLEPLOT allocates a subdivision size related to the range of the scale, and its value is placed in VDIV; if DIV is non-zero, VDIV is given the magnitude of DIV and the sign of STOP − START.

If the physical length of an axis is too small for annotations at all multiples of VDIV to fit, the annotation interval is a multiple of VDIV; KSCALE knows nothing about axis length.

If START = STOP, then VSTART, VSTOP and VDIV are returned with values START, STOP and DIV respectively.

**See also**

AXSBDV and SCALES.

### SUBROUTINE KTREAL (UNITS, IVAL1, IVAL2, RVAL, VALUE)

**Name**

KTREAL – to convert *time-date* value from external *time-date* triple to internal REAL form.

**Availability** Section plus, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| UNITS | CHARACTER*1 | 'T'ime, 'M'onths or 'W'eeks |
| IVAL1 | INTEGER expression | *Hours* or *years* |
| IVAL2 | INTEGER expression | *Minutes*, *months* or *week number* |
| RVAL | REAL expression | *Seconds*, *date* or *day of week* |
| VALUE | REAL variable | To receive encoded *time-date* value |

| UNITS | IVAL1 | IVAL2 | RVAL |
|---|---|---|---|
| 'T'ime | Hours | Minutes | Seconds |
| 'M'onths | Year | Month | Date |
| 'W'eeks | Year | Week no. | Day of week |

**Description**

*Time* and *date* based data can be converted from the values of the three components (hours, minutes, seconds or year, month, day) to a single REAL value which can be used by SIMPLEPLOT for plotting coordinates and for setting picture scales. KTREAL converts such values.

Years in this century can be abbreviated to two digits – any year in the range 0–99 received by KTREAL has 1900 added to it in the conversion.

Days of the week are interpreted such that RVAL=1.0 represents Sunday, 2.0 represents Monday, *etc.*

Values encoded by KTREAL should be used first to set up suitable plotting scales, and then as coordinates for plotting.

If UNITS ≠ 'T', 'M' or 'W', then VALUE is returned with value 0.0.

**Diagnostics**

(Invalid time-date units: *X*)[1]

**See also**

KTSTR and KTTRI.

## SUBROUTINE KTSTR (COMPNT, VALUE, STR, LENGTH)

### Name

KTSTR – to convert one component of encoded *time-date* value from internal REAL to text.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| COMPNT | CHARACTER*2 | *Time-date* component |
| VALUE | REAL expression | Encoded *time-date* |
| STR | STRING variable | To receive text string |
| LENGTH | INTEGER variable | To receive number of characters used in STR |

| COMPNT | *Meaning* | COMPNT | *Meaning* |
|---|---|---|---|
| 'YE' | Year | 'DE' | Days elapsed |
| 'MO' | Month | 'HO' | Hour |
| 'WE' | Week number | 'MI' | Minute |
| 'DM' | Day of the month | 'SE' | Second |
| 'DW' | Day of the week | | |

### Description

KTSTR converts the internal REAL representation of a *time-date* value into the text representation of one component of *time-date* annotations.

VALUE is interpreted as an encoded *time-date*, and STR is set to the text representation of the component of VALUE identified by COMPNT.

If COMPNT does not correspond to these values, a diagnostic is issued. In this case, or if VALUE corresponds to the current no-data value, STR is returned as ' ' and LENGTH=0.

The format of STR is affected by TIMFMT, TIMLDA and TIMLDC. LENGTH is set to the number of significant characters in STR.

If STR is not long enough to contain the converted component, SIMPLEPLOT places as much of the component as possible into STR and issues a diagnostic.

### Diagnostics

```
(Invalid component type: XX)¹
(String too short for component)¹
```

### See also

KTREAL, KTTRI, TIMFMT, TIMLDA and TIMLDC.

## SUBROUTINE KTTRI (UNITS, VALUE, IVAL1, IVAL2, RVAL)

### Name

KTTRI – to convert encoded *time-date* value from internal REAL to external triple form.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| UNITS | CHARACTER*1 | 'T'ime, 'M'onths or 'W'eeks |
| VALUE | REAL expression | Encoded *time-date* value |
| IVAL1 | INTEGER variable | To receive *hours* or *years* |
| IVAL2 | INTEGER variable | To receive *minutes*, *months* or *week number* |
| RVAL | REAL variable | To receive *seconds*, *date* or *day of week* |

| UNITS | IVAL1 | IVAL2 | RVAL |
|---|---|---|---|
| 'T'ime | Hours | Minutes | Seconds |
| 'M'onths | Year | Month | Date |
| 'W'eeks | Year | Week no. | Day of week |

### Description

KTTRI converts *time-date* data encoded into a REAL value into the external form consisting of three components.

Days of the week are interpreted such that RVAL=1.0 represents Sunday, 2.0 represents Monday, *etc.*

If UNITS ≠ 'T', 'M' or 'W', then IVAL1, IVAL2 and RVAL are returned with values 0, 0 and 0.0 respectively.

### Diagnostics

(Invalid time-date units: *X*)[1]

### See also

KTREAL and KTSTR.

### SUBROUTINE KVXYD (X3D, Y3D, Z3D, VARX2D, VARY2D, VARDEP)

**Name**

KVXYD – to convert a three-dimensional point $(x, y, z)$ to its two-dimensional equivalent plus depth.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| X3D, Y3D, Z3D | REAL expressions | Coordinates of a point, specified in 3-D units before transformation |
| VARX2D, VARY2D | REAL variables | To receive the coordinates of the projected point |
| VARDEP | REAL variable | To receive the distance of the projected point from the Viewing Plane |

**Description**

KVXYD converts a three-dimensional point (X3D,Y3D,Z3D) to (VARX2D,VARY2D), the equivalent 2-D coordinates once transformation and projection have been applied. VARX2D and VARY2D are measured in terms of the underlying SIMPLEPLOT scales.

KVXYD also calculates the value of the depth, VARDEP. By default, if a point is within the limits specified by VS3DLM, its depth lies in the range $-\sqrt{3} \leq \texttt{VARDEP} \leq \sqrt{3}$. Points between the Origin and the Viewing Position have negative depths – the greater the depth, the further away a point is from the Viewing Position.

The permissible range of depths is altered following a call to VSFULL.

**See also**

QVDPLM, QVSCAL, VS3DLM, VSFULL, VSVRTP and VSVXYZ.

## SUBROUTINE KWZVAL (ICURVE, ZVAL, RVAL)

### Name

KWZVAL – to convert a $z$ value on a waterfall chart to its equivalent in terms of the label scale.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| ICURVE | INTEGER expression | Number of relevant curve |
| ZVAL | REAL expression | Coordinate of a point specified in data units |
| RVAL | REAL variable | Equivalent value in units of the label scale |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. There is an underlying numerical scale controlling the label scale. KWZVAL delivers the value in the underlying scale corresponding to a given data value ZVAL in a particular curve ICURVE.

The returned value, RVAL, can be used as an ordinate to the general point-by-point plotting subroutines to augment a waterfall chart.

KWZVAL can not return a sensible value if no waterfall scales exist. If WFCHT, WFDRAW or WFZSCL has not been called already, a diagnostic is issued and the current no-data value returned in RVAL.

If ZVAL is equal to the current no-data value, RVAL is set to the no-data value.

### Diagnostics

`(No current waterfall Z scale)`[1]

### See also

QWZSCL, WFCHT, WFDRAW and WFZSCL.

### SUBROUTINE KXYXY (ICODE1, XIN, YIN, ICODE2, XOUT, YOUT)

**Name**

KXYXY – to convert coordinates of a point to other units.

**Availability** Section plus, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| ICODE1 | INTEGER expression | Code for units of $(x_{in}, y_{in})$ |
| XIN | REAL expression | $x$ coordinate of a point |
| YIN | REAL expression | $y$ coordinate of a point |
| ICODE2 | INTEGER expression | Code for units of $(x_{out}, y_{out})$ |
| XOUT | REAL variable | To receive $x$ coordinate |
| YOUT | REAL variable | To receive $y$ coordinate |

| ICODE | *Units* | *Origin* |
|---|---|---|
| 0 | Device units | Bottom left of device plotting area |
| 1 | Pixels | Bottom left of device plotting area |
| 2 | Centimetres | Bottom left of device plotting area |
| 3 | Centimetres | Bottom left of SIMPLEPLOT page |
| 4 | Normalized units | Bottom left of SIMPLEPLOT page |
| 5 | User coordinates | User origin of picture/graph |

**Description**

A point on the plotting area can be described by a Cartesian pair specified in various different units; KXYXY converts the coordinates of a point between any of the units specified by ICODE. *Normalized* units (code 4) are in the range 0.0 to 1.0 on the SIMPLEPLOT page.

If the code for either pair is between 0 and 2, the device must have been opened. For codes 3 and 4, a SIMPLEPLOT page must have been started, and for code 5 a new picture must have been started. If KXYXY is called when the relevant information is not available, a diagnostic is issued.

If ICODE1=5, and XIN or YIN is equal to the current no-data value, XOUT and YOUT are set to no-data, and will be ignored by plotting subroutines.

The orientation of the SIMPLEPLOT page and the device are not necessarily the same. The orientation of $(x, y)$ for codes 3–5 can be affected by calls to FITPAG, GROUP, PAGE and PAGVW.

**Diagnostics**

(KXYXY not viable)[1]

**See also**

KISXY, KPVXY and NODATA.

### SUBROUTINE KZRG (XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS, Z2ARR, NX, NY)

### Name

KZRG – to generate a regular grid of values from ungridded $(x, y, z)$ coordinates.

**Availability** Section 2, released version 2-11.

### Arguments

| | | |
|---|---|---|
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |
| Z2ARR | REAL 2-D array | To receive values over a regular grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |

### Description

KZRG converts the ungridded data held in ZARR into a regular grid of values in the array Z2ARR(NX,NY) using interpolation. Before using KZRG, the element array I2ARR(NNODES,NELEMS) must be set up; ZORDER or ZORDEN can be used for this purpose.

In this way, surface pictures of randomly positioned $(x, y, z)$ coordinates can be drawn either directly by ZSURF, or indirectly by KZRG followed by RGSURF. As ZSURF is very slow, conversion to a regular grid for surface pictures is advisable whenever there is sufficient storage available for the extra array Z2ARR.

KZZRG is more efficient than KZRG.

### Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]

### See also

KZZRG.

### SUBROUTINE KZZRG (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS, Z2ARR, NX, NY)

**Name**

`KZZRG` – to generate a regular grid of values from ungridded $(x, y, z)$ coordinates.

**Availability** Section 2, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `XARR`,<br>`YARR`, `ZARR` | `REAL` arrays | $(x, y, z)$ coordinates of data values in parallel arrays |
| `NPTS` | `INTEGER` expression | Number of data points |
| `I2ARR` | `INTEGER` 2-D array | Data element structure |
| `N2ARR` | `INTEGER` 2-D array | Element numbers of neighbours |
| `NNODES` | `INTEGER` expression | Number of nodes per element |
| `NELEMS` | `INTEGER` expression | Number of elements |
| `Z2ARR` | `REAL` 2-D array | To receive values over a regular grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |

**Description**

`KZZRG` converts the ungridded data held in `ZARR` into a regular grid of values in the array `Z2ARR(NX,NY)` using interpolation.

Before using `KZZRG`, the element array `I2ARR(NNODES,NELEMS)` and the neighbour array `N2ARR(NNODES,NELEMS)` must be set up; `ZZORDR` or `ZZORDN` can be used for this purpose.

In this way, surface pictures of randomly positioned $(x, y, z)$ coordinates can be drawn either directly by `ZZSURF`, or indirectly by `KZZRG` and `RGSURF`. As `ZZSURF` can be slow, conversion to a regular grid for surface pictures is advisable whenever there is sufficient storage available for the extra array `Z2ARR`.

`KZZRG` is more efficient than `KZRG`.

**Diagnostics**

`(INVALID ARGUMENT: ARRAY SIZE)`[1]

**See also**

`KZRG`.

## SUBROUTINE LABANG (IDEGS)

### Name

LABANG – to specify the angle of labels drawn by CP7LB, CP7XC or CP7YC.

**Availability** Section 4, released version 2-6.

### Argument

| | | |
|---|---|---|
| IDEGS | INTEGER expression | Angle in degrees |

### Description

By default, the labels drawn by CP7LB, CP7XC or CP7YC are drawn horizontally. LABANG specifies the angle at which they are to be drawn.

The effect of LABANG on hardware text varies according to capabilities of the device. For example, if text can only be drawn horizontally, each character will be staggered to emulate the effect of angled text.

### Default

CALL LABANG(0) restores the default.

### See also

LABJST, CP7LB, CP7XC and CP7YC.

## SUBROUTINE LABCV7 (XARR, YARR, NARR, LTYPE, CAP)<br>SUBROUTINE LABCV (XARR, YARR, NARR, LTYPE, CAP, NCAP)

### Name

`LABCV7` – to draw a labelled curve.

**Availability** Section 4, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `XARR`, `YARR` | `REAL` arrays | Coordinates of points, in units of the plotting scales |
| `NARR` | `INTEGER` expression | Number of points |
| `LTYPE` | `INTEGER` expression | Type of broken line pattern |
| `CAP` | `STRING` expression | Label for curve |
| `NCAP` | `INTEGER` expression | Number of characters in `CAP` |

### Description

Using broken line pattern `LTYPE`, `LABCV`/`LABCV7` draws [1] the curve corresponding to (`XARR(1)`, `YARR(1)`), (`XARR(2)`, `YARR(2)`), . . . (`XARR(NARR)`, `YARR(NARR)`). The label, `CAP`, is drawn [2] along the curve. By default, the label is vertically centred on the curve, and left justified to point (`XARR(NARR/2)`,`YARR(NARR/2)`). An alternative justification and position can be specified by `CVLBJS`/`CVLBPS`.

The base of each letter is parallel to the curve and the label is drawn in the direction of the curve. If the label is too long, it is truncated. `LABCV7` will not interpret embedded commands specified by `STRLV`.

A closed curve can be obtained by using data in which `XARR(NARR)` = `XARR(1)` and `YARR(NARR)` = `YARR(1)`. The curve is drawn through all points which are not equal to the current no-data value. If fewer than 2 points are valid, a diagnostic is issued and nothing is drawn.

If line style has been requested as a bundled attribute, `LTYPE` overrides the line style of the bundle (but not colour or thickness). If `LTYPE` = $-999$, all the bundled line-drawing attributes of the current pen are used.

### Diagnostics

```
(Data curve exceeds scales)²
(Insufficient number of valid points)¹
(Requested point unavailable)¹
(Text truncated on curve label)²
```

### See also

`BRKNCV`, `BUNLPR`, `CLIPLV`, `CVLBJS` and `CVLBPS`.

## SUBROUTINE LABJST (VJUST, HJUST)

### Name

LABJST – to specify the justification of labels drawn by CP7LB, CP7LBM, CP7XC or CP7YC.

**Availability** Section 4, released version 2-6.

### Arguments

| | | |
|---|---|---|
| VJUST | CHARACTER*1 | Vertical justification of label |
| HJUST | CHARACTER*1 | Horizontal justification of label |

| VJUST | *Vertical justification* |
|---|---|
| 'D' | Default |
| 'T' | Top of letters |
| 'C' | Halfway between 'T' and 'B' |
| 'B' | Bottom of letters (not including descenders) |

| HJUST | *Horizontal justification* |
|---|---|
| 'D' | Default |
| 'L' | At the left (beginning) of the label |
| 'C' | Halfway between 'L' and 'R' |
| 'R' | At the right (end) of the label |
| 'P' | Preceding the label |
| 'F' | Following the label |

### Description

By default, the labels drawn by CP7LB, CP7XC and CP7YC are positioned such that the bottom, left-hand corner of the label coincides with the specified $(x, y)$ coordinates. Labels drawn by CP7LBMCAPLBM are justified 'B'ottom 'P'receding by default. LABJST specifies the position of the label relative to the point.

This $(x, y)$ point is interpreted as one of nine possible positions described by combinations of values of VJUST and HJUST.

### Default

CALL LABJST('D','D') restores the default.

### See also

LABANG, CP7LB, CP7XC and CP7YC.

### SUBROUTINE LIMEXC (DARR, NARR, VARMIN, VARMAX)

**Name**

`LIMEXC` – to find the minimum and maximum values in a `REAL` array.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `DARR` | `REAL` array | Data values |
| `NARR` | `INTEGER` expression | Number of elements of `DARR` to be examined |
| `VARMIN` | `REAL` variable | To receive minimum value |
| `VARMAX` | `REAL` variable | To receive maximum value |

**Description**

`LIMEXC` (`LIM`its `EXC`lusive) searches through array `DARR` from `DARR(1)` to `DARR(NARR)` and sets `VARMIN` to the minimum value, `VARMAX` to the maximum value.

`VARMIN` and `VARMAX` must be the names of variables because `LIMEXC` assigns values to them.

`LIMEXC` ignores any values which are equal to the current no-data value.

**See also**

`LIMINC` and `NODATA`.

## SUBROUTINE LIMIDX (DARR, NARR, IMIN, IMAX)

### Name

`LIMIDX` – to find the position of each of the minimum and maximum values of a `REAL` array.

**Availability** Section 1, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `DARR` | `REAL` array | Data values |
| `NARR` | `INTEGER` expression | Number of elements in `DARR` to be examined |
| `IMIN` | `INTEGER` variable | Position of minimum value |
| `IMAX` | `INTEGER` variable | Position of maximum value |

### Description

`LIMIDX` finds the position of the minimum and maximum values in a `REAL` array. If the array is indexed from 1 to `NARR`, `DARR(IMIN)` is the lowest value in the array, and `DARR(IMAX)` is the highest value.

If `DARR` contains more than one value equal to the minimum or maximum, `IMIN` or `IMAX` refer to the position of the first occurrence.

`LIMIDX` ignores any values which are equal to the current no-data value. If all the values correspond to the current no-data value, `IMIN` and `IMAX` are set to zero.

### See also

`LIMEXC`, `LIMINC` and `NODATA`.

## SUBROUTINE LIMINC (DARR, NARR, VARMIN, VARMAX)

### Name

LIMINC – to find the minimum and maximum values in a REAL array.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| DARR | REAL array | Data values |
| NARR | INTEGER expression | Number of elements of DARR to be examined |
| VARMIN | REAL variable | To be reset to minimum value |
| VARMAX | REAL variable | To be reset to maximum value |

### Description

LIMINC (LIMits INClusive) searches through array DARR from DARR(1) to DARR(NARR). If the minimum is less than the existing value of VARMIN, VARMIN is set to the minimum value; otherwise, VARMIN is left unaltered. If the maximum is greater than the existing value of VARMAX, VARMAX is set to the maximum value; otherwise VARMAX is left unaltered.

VARMIN and VARMAX must be the names of variables which have already been given values, because LIMINC uses the existing values, and may assign new values to them.

LIMINC ignores any values which are equal to the current no-data value.

### See also

LIMEXC and NODATA.

## SUBROUTINE LIMSFN (FUNXY, VARMIN, VARMAX)

### Name

LIMSFN – to find the range of function values to be used for contours of a 3-D function $z = f(x, y)$.

**Availability** Section 2, released version 2-11.

### Arguments

| | | |
|---|---|---|
| FUNXY | function name | REAL function with two REAL arguments, also declared in an EXTERNAL statement |
| VARMIN,<br>VARMAX | REAL variables | To receive values of limits |

### Description

LIMSFN evaluates the function FUNXY, which must be a REAL function with two REAL arguments, at every point which would be used for plotting the function, and sets VARMIN and VARMAX to the minimum and maximum respectively.

The values returned by LIMSFN can be affected by the range of the independent variables (see FNAREA), and the underlying grid used to calculate the function (see SFMESH).

### See also

FNAREA, SFLIMS and SFMESH.

## SUBROUTINE LINEK7 (LTYPE, CAP)
## SUBROUTINE LINEKY (LTYPE, CAP, NCAP)

### Name

LINEK7 – to draw an annotated sample of a broken line pattern in a key.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| LTYPE | INTEGER expression | Type of broken line pattern |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

LINEKY/LINEK7 inserts a key to a broken line pattern (as used by the BRKN* subroutines) in the next empty space in the current key or caption area (defined by DEFKEY/DEFCAP or selected by RESNXT). A sample of the broken line pattern is drawn [1] on the left of the entry; the caption, CAP, is drawn [2] alongside the sample.

If line style has been requested as a bundled attribute, LTYPE overrides the line style of the bundle (but not colour or thickness) unless

LTYPE $= -999$ which indicates that all the bundled line-drawing attributes of the current pen are to be used.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)[2]
(Key/caption area full)[2]
(No active key/caption area)[2]

### See also

ADDJST, BLNKKY, BOTHK7, BUNLPR, DEFKEY and LINSET.

## SUBROUTINE LINSET (ISET)

### Name

`LINSET` – to specify an alternative source of broken line patterns.

**Availability** Section 1, released version 2-8.

### Argument

`ISET` `INTEGER` expression Broken line pattern source

| `ISET` | *Pattern source* |
|---|---|
| 0 | Hardware/firmware broken lines (if available) |
| 1 | Device independent software broken line patterns |

### Description

For efficient operation and clarity, Simpleplot is set up to draw some broken lines on pictures using device hardware or firmware if possible. `LINSET` can be used to select an alternative source of patterned lines.

Hardware broken line patterns are chosen to correspond as much as possible to the Simpleplot software line patterns.

### Default

`CALL LINSET(0)` restores the default.

### See also

`CHSET`, `FFSET`, `MKSET` and `SHSET`.

## SUBROUTINE MARGDV (PICTYP)

**Name**

MARGDV – to specify the distribution of picture margins.

**Availability** Section 1, released version 2-12.

**Argument**

PICTYP CHARACTER*1 Type of picture

| PICTYP | *Picture type* | *Margin distribution* |
|---|---|---|
| 'D' | Default | Large left-hand and bottom, small right-hand and top |
| 'C' | Cartesian ($x$-$y$ ) plot | Large on 'XC' and 'YC' axis sides, small on other |
| 'B' | Bar chart | Large on 'LB' and 'NB' axis sides, small on other |
| 'I' | Isometric picture | Large on 'XI', 'YI', 'ZI' axis sides, small on other |
| 'W' | Waterfall chart | Large on 'LW' and 'NW' axis sides, small on other |
| 'O' | Other, *eg.* polar plot, pie chart | Equal distribution on all sides |

**Description**

By default, margins are distributed such that the gap allocated to the right-hand side and the top is approximately the size of a single character, and the larger residue is allocated to the left-hand side and bottom.

MARGDV specifies that, while PICMRG is not active, subsequent new pictures should be set up with the margins distributed to suit the specified type of picture.

MARGDV must be called before the new picture to which it refers is started.

**Default**

CALL MARGDV('D') restores the default.

**See also**

MARGIN and PICMRG.

## SUBROUTINE MARGIN (CMS)

### Name

MARGIN – to specify the overall size of the margin around individual pictures.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| CMS | REAL expression | Width of margin (in cms) |

### Description

MARGIN specifies the size of the margin around individual pictures.

If MARGIN has not been called, SIMPLEPLOT calculates the following using the current minimum size of text:

$$\texttt{SPACE} = 5\times\textit{character width} + 3.3\times\textit{character height}$$

The maximum default value of SPACE is 2cm.

SIMPLEPLOT imposes minimum and maximum values on picture margins. The maximum size allowed for a margin is half the smaller dimension of the picture, and the minimum margin is 0.01cm.

The distribution of the overall margin depends on the current text size. If the total margin size is less than 2 text characters, it is distributed equally on all sides of the picture. With larger margins, the gap allocated to the top and right-hand side is approximately the size of 1 character, and the larger residue is allocated to the bottom and left-hand side. TEXTSZ, TEXTMN and CHSET change text characteristics which can indirectly change margin sizes.

MARGIN overrides PICMRG and *vice versa*.

### Diagnostics

($v_1$ requested)$^2$<br>
($v_2$ used)$^2$<br>
(Margin modified)$^1$

### Default

Calling MARGIN with a negative argument restores the default.

### See also

MARGDV and PICMRG.

### SUBROUTINE MARKAR (X1, DX, DARR, NARR, MKTYPE, NSTEP)

**Name**

MARKAR – to draw marker symbols at a set of values held in an array.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| X1 | REAL expression | Value of independent variable for plotting DARR(1) |
| DX | REAL expression | Independent variable interval between function values |
| DARR | REAL array | Function values for plotting |
| NARR | INTEGER expression | Number of points |
| MKTYPE | INTEGER expression | Type of marker symbol |
| NSTEP | INTEGER expression | Interval between points marked |

**Description**

MARKAR draws [3] a set of marker symbols, MKTYPE, centred on the following points:

> (X1,DARR(1)), (X1+(NSTEP*DX),DARR(1+NSTEP)),
> (X1+(2*NSTEP*DX),DARR(1+(2*NSTEP))),...

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ and $x_i = x_1 + (i - 1)dx$.

By default, all drawing is restricted by the limits of the picture scales; marker symbols can spill over the edge of a picture but the coordinates at the centre of the symbol must not exceed these limits. If coordinates exceed these limits, the symbol is omitted and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Symbol clipped)[2]
(Symbol spills over boundary)[2]

**See also**

BRKNAR, CLIPLV, MARKCV, MKSIZE, and SHDEAR.

### SUBROUTINE MARKCV (XARR, YARR, NARR, MKTYPE, NSTEP)

**Name**

MARKCV – to draw marker symbols at a set of coordinates.

**Availability** Section 1, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | Coordinates of points, in units of the plotting scales |
| NARR | INTEGER expression | Number of points |
| MKTYPE | INTEGER expression | Type of marker symbol |
| NSTEP | INTEGER expression | Interval between points marked |

**Description**

MARKCV draws [3] a set of marker symbols, MKTYPE, centred on the following points:

(XARR(1), YARR(1)),
(XARR(1+NSTEP), YARR(1+NSTEP)), ...
(XARR(1+(2*NSTEP)), YARR(1+(2*NSTEP))), ...

By default, all drawing is restricted by the limits of the picture scales; marker symbols can spill over the edge of a picture but the coordinates at the centre of the symbol must not exceed these limits. If coordinates exceed these limits, the symbol is omitted and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Symbol clipped)[2]
(Symbol spills over boundary)[2]

**See also**

BRKNCV, CLIPLV, MARKAR, MKSIZE and SHDECV.

## SUBROUTINE MARKK7 (MKTYPE, CAP)
## SUBROUTINE MARKKY (MKTYPE, CAP, NCAP)

### Name

MARKK7 – to draw an annotated sample of a marker symbol in a key.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| MKTYPE | INTEGER expression | Type of marker symbol |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

MARKKY/MARKK7 inserts a key to a marker symbol (as used by MARK*) in the next empty space in the current key or caption area (defined by DEFKEY/DEFCAP or selected by RESNXT).

A sample of the marker symbol, MKTYPE, is drawn [3] on the left of the entry; the caption, CAP, is drawn [2] alongside the sample.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)$^2$<br>
(Key/caption area full)$^2$<br>
(No active key/caption area)$^2$

### See also

ADDJST, BLNKKY, BOTHK7, DEFKEY, LINEK7 and MKSET.

## SUBROUTINE MARKPT (X, Y, MKTYPE)

### Name

MARKPT – to draw a marker symbol at a specified point.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point specified in units of the plotting scales |
| MKTYPE | INTEGER expression | Type of marker symbol |

### Description

MARKPT draws [3] a marker symbol, MKTYPE, centred on the point $(x, y)$. After drawing the specified symbol, MARKPT leaves the pen at the centre of the symbol to allow a line to be drawn from the symbol to another point by a succeeding call of BRKNPT/JOINPT. For example, a sequence of symbols joined by straight lines can be drawn by repeating the following calls for a sequence of points:

```
CALL BRKNPT(X,Y,LTYPE)
CALL MARKPT(X,Y,MKTYPE)
```

BREAK should be called between MARKPT and a succeeding call of BRKNPT/JOINPT if a line is not required joining symbol and point.

The point $(x, y)$ is specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, KXYXY.

By default, all drawing is restricted by the limits of the picture scales; marker symbols can spill over the edge of a picture but the coordinates at the centre of the symbol must not exceed these limits. If coordinates exceed these limits, the symbol is omitted and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Symbol clipped)[2]
(Symbol spills over boundary)[2]

### See also

BREAK, BRKNPT, CLIPLV, CP7PT, MKMASK, MKSIZE, MARKCV and MARKAR.

## SUBROUTINE MKMASK (TORF)

### Name

`MKMASK` – to specify whether marker symbols are to be masked.

**Availability** Section 4, released version 2-11.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether symbols are to be masked |

### Description

By default, Simpleplot draws anywhere within the page unless specific areas have been reserved for specific activities. `MKMASK` specifies that the area immediately around marker symbols is to be masked from any further drawing.

After a call of `MKMASK(.TRUE.)` symbols are masked; this means that the symbols themselves look no different from unmasked symbols, but any subsequent drawing over the symbols is omitted as it crosses each symbol.

The limit to the number of masked areas which can be used for individual symbols and text strings is 50. When this limit is reached a diagnostic is issued and subsequent masked areas take the place of the oldest areas and overdrawing of these areas may occur.

### Diagnostics

`(Maximum no. of masked areas reached)`[1]

### Default

`CALL MKMASK(.FALSE.)` restores the default.

### See also

`CHMASK`.

## SUBROUTINE MKSET (ISET)

### Name

`MKSET` – to specify an alternative source of marker symbols.

**Availability** Section 1, released version 2-8.

### Argument

`ISET` `INTEGER` expression Marker symbol source

| `ISET` | *Marker symbol source* |
|---|---|
| 0 | Hardware/firmware symbols (if available) |
| 1 | Device independent software symbols |
| 2 | Hershey Odd Shapes |
| 3 | Hershey Graph Markers |
| 4 | Hershey Solid Graph Markers |
| 5 | Hershey Map Symbols |
| 6 | Hershey Small Astronomy |
| 7 | Hershey Astronomy |
| 8 | Hershey Zodiac |

### Description

For efficient operation and clarity, Simpleplot is set up to mark points on pictures using symbols from device hardware or firmware wherever possible. `MKSET` can be used to select an alternative source of symbols.

Individual Hershey symbols can also be selected by `MKTYPE` greater than 16. The Hershey symbols are only available with Simpleplot-Plus.

### Default

`CALL MKSET(0)` restores the default.

### See also

`CHSET`, `FFSET`, `LINSET` and `SHSET`.

## SUBROUTINE MKSIZE (CMS)

**Name**

MKSIZE – to specify a fixed size of marker symbols independent of text size.

**Availability** Section 4, released version 2-11.

**Argument**

| | | |
|---|---|---|
| CMS | REAL expression | Size of marker symbols (in cms) |

**Description**

By default, the size of marker symbols is controlled by the same mechanism that controls the size of text. That is to say, TEXTSZ, TEXTMN and TEXTMG normally affect the size of both characters and symbols. MKSIZE specifies the size of marker symbols which is independent of text size.

**Default**

CALL MKSIZE(0.0) restores the default.

**See also**

TEXTSZ, TEXTMG.

## SUBROUTINE MOVEON (TORF)

### Name

`MOVEON` – to specify whether to superimpose pictures.

**Availability** Section 4, released version 2-8.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether next picture is to be drawn in the next clear space |

### Description

By default, each new picture is automatically positioned either in the next clear space within the group or on a new Simpleplot page. `MOVEON(.FALSE.)` suspends the automatic moving on for each new picture, therefore allowing pictures to be superimposed. `PICPOS` also suspends automatic picture layout.

`CALL MOVEON(.TRUE.)` restores the normal state, with each new picture starting on a clear space.

### Diagnostics

`(MOVEON cancelled)`[1]

### Default

`CALL MOVEON(.TRUE.)` restores the default.

### See also

`ENDPAG`, `ENDPIC` and `PICPOS`.

## SUBROUTINE MPK7H (VCHAR, HCHAR, ZLEFT, ZRIGHT, CAP)
## SUBROUTINE MPKYH (VCHAR, HCHAR, ZLEFT, ZRIGHT, CAP, NCAP)

### Name

MPK7H – to draw a complete horizontal key to a shaded contour map.

**Availability** Section plus, released version 2-10.

### Arguments

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of key |
| HCHAR | CHARACTER*1 | Horizontal position of key |
| ZLEFT | REAL expression | $z$ value at left of key |
| ZRIGHT | REAL expression | $z$ value at right of key |
| CAP | STRING expression | Single caption for key |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

MPK7H/MPKYH draws a complete key to the current shading patterns that represent the data range ZLEFT to ZRIGHT. Samples of the patterns are drawn side-by-side, with the key numbers above. CAP provides a heading which is, by default, centred at the top of the key (see ADDJST). A box is drawn [1] around the key unless cancelled by BOXKY.

The key area is associated only with the current picture/page but is masked to prevent overdrawing. To avoid the key being overdrawn by other plotting, it is best to call MPK7H after NEWPIC but before the main picture is drawn. The order of key entries can be reversed by interchanging ZLEFT with ZRIGHT. The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

```
(Key/caption attempted with no page)2
(Key/caption attempted with no picture)2
(Key/caption height reduced)2
(Key/caption width reduced)2
(MAP KEY OMITTED: CONSTANT DATA)1
(MAP KEY OMITTED: NULL RANGE)1
(Maximum no. of keys/captions reached)2
(No room for key/caption area)2
```

### See also

ADDJST, BOXKY, MPK7V, MPTYPE, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SHPATT and SQ*.

## SUBROUTINE MPK7V (VCHAR, HCHAR, ZTOP, ZBOTTM, CAP)
## SUBROUTINE MPKYV (VCHAR, HCHAR, ZTOP, ZBOTTM, CAP, NCAP)

### Name

MPK7V – to draw a complete vertical key to a shaded contour map.

**Availability** Section plus, released version 2-10.

### Arguments

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of key |
| HCHAR | CHARACTER*1 | Horizontal position of key |
| ZTOP | REAL expression | $z$ value at top of key |
| ZBOTTM | REAL expression | $z$ value at bottom of key |
| CAP | STRING expression | Single caption for key |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

MPK7V/MPKYV draws a complete key to the current shading patterns that represent the data range ZTOP to ZBOTTM. Samples of the patterns are drawn in a vertical list, with the key numbers alongside. CAP provides a heading which is, by default, centred at the top of the key (see ADDJST). A box is drawn [1] around the key unless cancelled by BOXKY.

The key area is associated only with the current picture/page but is masked to prevent overdrawing. To avoid the key being overdrawn by other plotting, it is best to call MPK7V after NEWPIC but before the main picture is drawn. The order of key entries can be reversed by interchanging ZTOP with ZBOTTM. The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(Key/caption attempted with no page)[2]
(Key/caption attempted with no picture)[2]
(Key/caption height reduced)[2]
(Key/caption width reduced)[2]
(MAP KEY OMITTED: CONSTANT DATA)[1]
(MAP KEY OMITTED: NULL RANGE)[1]
(Maximum no. of keys/captions reached)[2]
(No room for key/caption area)[2]

### See also

ADDJST, BOXKY, MPK7H, MPTYPE, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SHPATT and SQPEN.

## SUBROUTINE MPTYPE (IMETHD)

**Name**

MPTYPE – to specify the shading method used for shaded contour maps and map keys.

**Availability** Section plus, released version 2-10.

**Argument**

IMETHD INTEGER expression Shading method

| IMETHD | *Shading method used* |
|---|---|
| 1 | Generalized shading between contour curves |
| 2 | Fast area-fill showing data configuration |
| 3 | Fast area-fill with interpolation |

**Description**

SIMPLEPLOT offers two separate approaches to drawing shaded contour maps from data which represents a function of two variables, $z = f(x, y)$. By default, the *SHDS subroutines perform generalized shading between contour curves but SIMPLEPLOT can also use an alternative technique known as *fast area-fill.* MPTYPE specifies which technique is used.

In fast area-fill pictures, the picture is built up by working out a data value for each part of the picture in order to allocate a fill character; method 2 uses the value of the nearest data point, and method 3 interpolates between surrounding data points.

MPTYPE also determines the shading method used for map key samples (see MPK7H and MPK7V). MPTYPE has no effect on raster pictures (see RASTER) but it may be necessary to call MPTYPE with IMETHD=3 to draw a key to the patterns used on a raster picture.

**Default**

CALL MPTYPE(1) restores the default.

**See also**

FFSET, FFSIZE, MPK7H, MPK7V, RASTER and *SHDS.

## SUBROUTINE MTCLOS

### Name

`MTCLOS` – to terminate metafile redrawing.

**Availability** Section plus, released version 2-11.

### Arguments

None.

### Description

A SIMPLEPLOT metafile can be completely redrawn by calling `REDRAW` or it can be drawn line-by-line using the following procedure:

1. Open the metafile.
2. Initiate redrawing by `CALL MTOPEN`.
3. For each line of the metafile:
   - Read line from metafile.
   - Redraw line by `CALL MTDRAW(STR)`.
4. Terminate redrawing by `CALL MTCLOS`.

If any SIMPLEPLOT subroutine other than `MTDRAW` is called after `MTOPEN` but before `MTCLOS`, redrawing is abandoned and a diagnostic is issued.

This method of redrawing may be useful when the metafile is on a remote system from which it can be read line by line under program control.

### Diagnostics

`(REDRAWING ABANDONED)`[1]

### See also

`MTDRAW` and `MTOPEN`.

## SUBROUTINE MTCOMM (STR)

### Name

`MTCOMM` – to output a comment line to a metafile.

**Availability** Section plus, released version 2-11.

### Argument

| | | |
|---|---|---|
| `STR` | `STRING` expression | Metafile comment line |

### Description

A comment line can be inserted into a metafile as it is being written by calling `MTCOMM` with the comment as argument. In the metafile, the comment is preceded by the identifying letters `COMM` and all such lines are ignored when the metafile is redrawn using `REDRAW`.

The maximum length of comments is 75 characters; if a longer string is passed to `MTCOMM`, it is truncated.

### See also

`MTFILE`.

## SUBROUTINE MTDRAW (STR)

### Name

`MTDRAW` – to redraw a single line of a SIMPLEPLOT metafile.

**Availability** Section plus, released version 2-11.

### Argument

| | | |
|---|---|---|
| `STR` | `STRING` expression | A single line from a SIMPLEPLOT metafile |

### Description

A SIMPLEPLOT metafile can be completely redrawn by calling `REDRAW` or it can be drawn line-by-line using the following procedure:

1. Open the metafile.
2. Initiate redrawing by `CALL MTOPEN`.
3. For each line of the metafile:
   - Read line from metafile.
   - Redraw line by `CALL MTDRAW(STR)`.
4. Terminate redrawing by `CALL MTCLOS`.

If any SIMPLEPLOT subroutine other than `MTDRAW` is called after `MTOPEN` but before `MTCLOS`, redrawing is abandoned and a diagnostic is issued.

This method of redrawing may be useful when the metafile is on a remote system from which it can be read line by line under program control.

### Diagnostics

`(REDRAWING ABANDONED)`[1]

### See also

`MTCLOS`, `MTOPEN` and `REDRAW`.

## SUBROUTINE MTFILE (ICHAN)

### Name

`MTFILE` – to request metafile output.

**Availability** Section plus, released version 2-8.

### Argument

| | | |
|---|---|---|
| `ICHAN` | `INTEGER` expression | FORTRAN channel number |

### Description

A Simpleplot metafile, containing a complete reproducible record of all the graphics output from a program, can be created at the same time as output to a graphics device, using the following procedure:

1. Open a FORTRAN I/O channel for formatted, serial output.
2. Call `MTFILE` specifying this I/O channel, before any plotting occurs (*ie.* before a graphics device is opened).

   :

3. Close FORTRAN I/O channel.

The metafile information is directed to a scratch file as the program runs. As each Simpleplot page is completed the whole scratch file is copied to `ICHAN` and deleted, and a diagnostic message is issued. Thus, if a program terminates prematurely for any reason, the metafile only contains details of complete pages.

The metafile produced in this way reflects the limitations of the device, such that the drawing and positioning of text may only be acceptable when reproduced on the device to which it is related, or on one which is less limited. When producing a metafile to be used on different devices, it is advisable to relate it to the device with the lowest resolution. Device *independent* metafiles can be produced using the META device driver.

If a program which uses `MTFILE` is run with the META device driver, the output file nominated by `ICHAN` takes precedence over the file usually selected by the META device driver.

A call of `MTFILE` with `ICHAN` $< 0$, terminates metafile output.

### Diagnostics

`(PAGE COPIED TO METAFILE)`[1]

### See also

`MTCOMM`, `MTMISS` and `MTONLY`.

## SUBROUTINE MTMISS

### Name

MTMISS – to omit the current SIMPLEPLOT page from a metafile.

**Availability** Section plus, released version 2-8.

### Arguments

None.

### Description

If SIMPLEPLOT metafile output has been requested, the metafile details of each SIMPLEPLOT page are first written to a scratch file, and only transferred to the metafile when the page is finished; a complete page can be omitted from the metafile by calling MTMISS anywhere in the program between the start and end of the page. When a page is finished, but omitted from an active metafile as a result of calling MTMISS, a diagnostic is issued.

Please note that the end of a page occurs when ENDPAG or ENDPLT is called, or, in the absence of such calls, when a new page is started by a *new picture* subroutine.

With MTFILE and MTMISS it is possible to adapt a program to enable it to inquire whether or not it should save pages in metafiles. Separation of pages into individual files can be achieved by using CLOSE and OPEN between page ends, to change the file addressed by ICHAN.

### Diagnostics

(PAGE OMITTED FROM METAFILE)[1]

### See also

MTFILE and MTONLY.

## SUBROUTINE MTONLY

**Name**

`MTONLY` – to specify that only metafile output is to be produced.

**Availability** Section plus, released version 2-8.

**Arguments**

None.

**Description**

By default, metafile output is produced simultaneously with device output. `MTONLY` suppresses the device output.

A Simpleplot metafile, containing a complete reproducible record of all the graphics output from a program, can be created without simultaneous output to a graphics device, using the following procedure:

1. Open a FORTRAN I/O channel for formatted, serial output.
2. Call `MTFILE` specifying this I/O channel, before any plotting occurs (*ie.* before a graphics device is opened).
3. Call `MTONLY` to suppress output to the selected graphics device.

   :

4. Call Simpleplot plotting subroutines.

   :

5. Call `ENDPLT`.
6. Close FORTRAN I/O channel.

**Default**

The default cannot be restored except by `CALL INITSP`.

**See also**

`MTFILE` and `MTMISS`.

## SUBROUTINE MTOPEN

### Name

`MTOPEN` – to start metafile redrawing.

**Availability** Section plus, released version 2-11.

### Arguments

None.

### Description

A Simpleplot metafile can be completely redrawn by calling `REDRAW` or it can be drawn line-by-line using the following procedure:

1. Open the metafile.
2. Initiate redrawing by `CALL MTOPEN`.
3. For each line of the metafile:
   - Read line from metafile.
   - Redraw line by `CALL MTDRAW(STR)`.
4. Terminate redrawing by `CALL MTCLOS`.

If any Simpleplot subroutine other than `MTDRAW` is called after `MTOPEN` but before `MTCLOS`, redrawing is abandoned and a diagnostic is issued.

This method of redrawing may be useful when the metafile is on a remote system from which it can be read line by line under program control.

### Diagnostics

`(REDRAWING ABANDONED)`[1]

### See also

`MTCLOS` and `MTDRAW`.

### SUBROUTINE NEWPAG

**Name**

`NEWPAG` – to start a new SIMPLEPLOT page without starting a new picture.

**Availability** Section plus, released version 2-12.

**Arguments**

None.

**Description**

By default, a new SIMPLEPLOT page is created when the first picture is started, but sometimes it is desirable to draw page titles *etc*before any part of a picture has been drawn. `NEWPAG` starts a new SIMPLEPLOT page without starting a picture. Descriptive positions relative to the picture and user coordinates are undefined until a new picture is started.

`NEWPAG` finishes the current page and, if `BOXPAG` has been called to specify boxed pages, the box is drawn [1] when the new page is started.

**Diagnostics**

`(DEVICE OPENED: device name)`[1]

**See also**

`ENDPAG` and `NEWPIC`.

## SUBROUTINE NEWPIC

### Name

NEWPIC – to start a new 2-D picture without drawing an axis framework.

**Availability** Section 1, released before version 2-5.

### Arguments

None.

### Description

NEWPIC is used to start a new picture when a standard axis framework is not required. NEWPIC is a graphics subroutine but, by default, does no drawing. If one of the BOX* subroutines has been called to generate boxes around pictures, groups or pages, these are drawn [1] when a new picture is started.

NEWPIC does not reset the underlying coordinate system. If EQSCAL has not been called, COORDS controls the type of new picture. If EQSCAL is active, each new picture is started with the units specified by EQSCAL regardless of any call of COORDS; once a picture has been started, COORDS can be called to change coordinate interpretation.

### See also

AXES7, COORDS, EQSCAL, ISNEW, NEWPAG and POLAR7.

### SUBROUTINE NODATA (RVAL)

**Name**

`NODATA` – to specify the `REAL` value to be used to represent no-data values.

**Availability** Section 1, released version 2-10.

**Argument**

| | | |
|---|---|---|
| `RVAL` | `REAL` expression | No-data value |

**Description**

Simpleplot interprets a single `REAL` value in plotting data as a no-data value; by default, this value is $-1.0E20$. `NODATA` specifies an alternative value to be used for this purpose.

Before Simpleplot draws anything, it checks whether one of the coordinates of the item to be drawn corresponds to the no-data value. If any ordinate is missing, the item is not drawn. As this check occurs at a fairly low level within Simpleplot, it affects everything which is drawn at user coordinates – marker symbols, text captions, curves joining points, boxes, axes, *etc*– any item which can be positioned.

It is advisable to have `RVAL` so far outside the normal range of data values that no problems arise but, awkward circumstances may arise. For example, if the no-data value is set to zero and an axis is required at either $x = 0.0$ or $y = 0.0$, no axis will be drawn if `NODATA` is called before the axis is drawn. In this case, `NODATA` should be called to set the no-data value to zero after the axes have been drawn but before the data are plotted. This solution will not resolve the case where the no-data value has been set for missing $y$ coordinates but $x$ coordinates also correspond to the missing value. In this case, points with legitimate $y$ values but missing $x$ values, will not be drawn.

**Default**

`CALL NODATA(-1.0E20)` restores the default.

**See also**

`QNODAT`.

## SUBROUTINE ONEPEN (IPEN)

### Name

ONEPEN – to draw the colours of a multi-coloured picture separately.

**Availability** Section 1, released version 2-5.

### Argument

| | | |
|---|---|---|
| IPEN | INTEGER expression | Pen number |

### Description

If a multi-coloured picture is to be plotted for a printed document, it may be necessary to plot each colour on a separate page. When a program has been developed to produce the desired effect with the colours together, it can then be adapted to plot the same picture with only the parts drawn by pen number IPEN by inserting a call of ONEPEN(IPEN) before the first plotting call; Pen 1 is then used for all plotting directed to Pen IPEN, and plotting directed to other pens is omitted.

Please note that, by default, colour is the only line-drawing attribute associated with pens, and that the distinction between pens depends on the capabilities of the current device. If BUNLPR has be called to specify other attributes to be associated with pens for drawing lines, the correspondence between colour and pens may be lost and ONEPEN can no longer produce true ‘colour separations’.

### See also

BUNLPR, PEN and SETPNS.

## SUBROUTINE OUTBUF

### Name

`OUTBUF` – to update the Simpleplot page by flushing internal graphics buffer.

**Availability** Section 4, released version 2-7.

### Arguments

None.

### Description

For some devices, the efficiency of data transfer for graphics can be maximized by accumulating sequences of drawing in a buffer, and outputting them when the buffer is full. When output is buffered in this way, at any stage of a program, the most recent part of the picture may be missing because the buffer is not yet full. Whenever it is necessary to be sure that the picture is up to date, `CALL OUTBUF` forces the output of all buffered graphics. If the buffers are already empty when `OUTBUF` is called, it does nothing.

### See also

`ENDPAG`, `ENDPIC`, `GETXY`, `OWNIO`, `OWNNEW` and `SCLEAR`.

## SUBROUTINE OWNFIL (ITYPE, ILEVEL, IPRIOR, STR)

**Name**

`OWNFIL` – to specify an alternative source/destination for SIMPLEPLOT input/output.

**Availability** Section plus, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| `ITYPE` | `INTEGER` expression | Type of channel usage |
| `ILEVEL` | `INTEGER` expression | Type of drivers to be affected |
| `IPRIOR` | `INTEGER` expression | Priority over external information |
| `STR` | `STRING` expression | File/device name |

| `ITYPE` | *Channel usage* | `IPRIOR` | *Priority* |
|---|---|---|---|
| 1 | Diagnostics | 0 | Never use |
| 2 | Prompt output | 1 | Use when not defined externally |
| 3 | Prompt input | | |
| 4 | Graphics output | 2 | Use when defined externally |
| 5 | Graphics input | | |
| 6 | Configuration | 3 | Always use |

| `ILEVEL` | *Device drivers affected* |
|---|---|
| 0 | None |
| 1 | Device drivers for interactive devices |
| 2 | Device drivers for non-interactive devices |
| 3 | All device drivers |
| 4 | Non-device driver I/O (*eg.* diagnostics file, prompt) |
| 5 | Non-device driver I/O plus interactive drivers |
| 6 | Non-device driver I/O plus non-interactive drivers |
| 7 | Non-device driver I/O plus all drivers (*ie.* everything) |

**Description**

By default, SIMPLEPLOT allocates files and/or devices according to external information (*eg.* environment variables, logical names, *etc.*) or default values. `OWNFIL` specifies an alternative file (or device) to be used for a SIMPLEPLOT I/O channel (see `ITYPE`) and under what circumstances it takes priority over external information (see `ILEVEL` and `IPRIOR`).

The effect of `OWNFIL` can be restricted to certain classes of device driver or to non-device driver I/O according to the value of `ILEVEL`.

`STR` is interpreted as a file name (or file specification) and must have at least one character up to a maximum of 255. Valid file names differ from system to system and `STR` is interpreted as anything which would be valid where a file name is valid; on some systems this may include devices, path names, file specifications, network nodes *etc.*

For example, if you want your terminal output on your screen, but your hard-copy output to be sent to the files, `MYPIC.DO`*n*, then use `CALL OWNFIL(4,2,1,'MYPIC')`.

`OWNFIL` has no effect until the specified I/O channel is opened and, if the channel is already open, may have no effect at all:

- Prompt input, prompt output and diagnostic channels are allocated when SIMPLEPLOT is initialized and cannot be reassigned while SIMPLEPLOT is open. For `ITYPE`=1, 2 and 3, it is necessary for `OWNFIL` to be called before any other SIMPLEPLOT subroutine except `IOCHAN`.
- Graphics input and output channels are used only by individual device drivers and therefore can be re-assigned before a new device is selected.
- The configuration channel is used by both individual device drivers and the device driver system and therefore should normally be re-assigned before SIMPLEPLOT is fully initialized.

In order to allow `OWNFIL` to be used repeatedly before SIMPLEPLOT is fully initialized, it does not trigger the initial diagnostic message:

```
(SIMPLEPLOT Mark 2-15(nnn)X)
```

### Diagnostics

`(OWNFIL ignored: invalid argument)`[1]

### Default

`CALL OWNFIL(ITYPE,0,0,' ')` restores the default. Please note that the default is not restored by `INITSP`.

### See also

`IOCHAN` and host specific information.

## SUBROUTINE OWNIO (TORF)

### Name

`OWNIO` – to specify whether a terminal is to be used for graphics or user I/O.

**Availability** Section 4, released version 2-7.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether terminal is to be used for user I/O |

### Description

Some types of graphics terminal need to be in one state for outputting graphics, and another state when processing ordinary `READ` and `WRITE` operations from and to the terminal. If a dialogue display obscures the graphics, Simpleplot may switch it off during graphics output. With this sort of terminal, any user program which includes terminal `READ` and `WRITE` operations among graphics output calls needs to be able to switch states itself.

`CALL OWNIO(.TRUE.)` switches the terminal into a suitable state for terminal `READ` and `WRITE` operations, and

`CALL OWNIO(.FALSE.)` switches it back into the state used for graphics output.

### Default

`CALL OWNIO(.FALSE.)` restores the default.

### See also

`ENDPAG`, `ENDPIC`, `GETXY`, `OWNNEW`, `OUTBUF` and `SCLEAR`.

## SUBROUTINE OWNNEW (TORF)

**Name**

`OWNNEW` – to specify whether to bypass the normal procedure between SIMPLEPLOT pages.

**Availability** Section 4, released version 2-7.

**Argument**

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether to use own action between pages |

**Description**

By default, SIMPLEPLOT performs certain operations between pages of output. Device drivers for graphics terminals are usually set up so that the first SIMPLEPLOT page is started without requiring user interaction; but, before clearing the screen for subsequent pages, the device driver prompts the user (*eg.* the `Press <RETURN> to continue` message) to ensure that the page on the screen has been seen before it is erased.

If the user program performs its own interactive dialogue, there may be no need for the precaution of inviting a user prompt before starting a new SIMPLEPLOT page. After a call of `OWNNEW(.TRUE.)`, all such operations are bypassed such that each new page is started in the same way as the first.

**Default**

`CALL OWNNEW(.FALSE.)` restores the default.

**See also**

`ENDPAG`, `ENDPIC`, `GETXY`, `OUTBUF`, `OWNIO` and `SCLEAR`.

## SUBROUTINE PAGE (WCMS, HCMS)

### Name

`PAGE` – to specify the size of the SIMPLEPLOT page.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `WCMS` | `REAL` expression | Width of page (in cms) |
| `HCMS` | `REAL` expression | Height of page (in cms) |

### Description

`PAGE` has a different effect on packed picture and fixed picture devices:

**Packed picture devices** By default, the size of the SIMPLEPLOT page depends on the physical dimensions of the device and user calls to `PICSIZ` and `GROUP`.

When a picture size has not been specified, the allocated page size is related to the total available on the device (approximately 20cm×20cm). When a picture size has been specified, the page size is calculated to accommodate the current grouping. After `PAGE` has been called, each picture (or group) is positioned within an area of the specified size.

**Fixed picture devices** The fixed device area is treated like a page whether or not `PAGE` has been called. The arguments of `PAGE` are only used to set the page orientation when the device page may be rotated. The magnitudes of the arguments of `PAGE` are only used when `FITPAG` has also been called.

If the requested page size is larger than the device plotting area, a shrinkage factor is applied so that the page produced has the specified proportions. This shrinkage factor is then applied to all plotting within the page.

### Diagnostics

(${x_1}$ ${y_1}$ `requested)`$^2$
(${x_2}$ ${y_2}$ `used)`$^2$
`(PAGE cancelled)`$^1$
`(PAGE ignored: invalid argument)`$^1$
`(Page Size modified)`$^1$

### Default

Calling `PAGE` with either argument set to zero restores the default.

### See also

`FITPAG`, `PAG*`, `PGFULL` and `QPAGE`.

### SUBROUTINE PAGMRG (CMS, RCMS, BCMS, TCMS)

**Name**

`PAGMRG` – to specify the size and distribution of peripheral margins.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Width of left margin (in cms) |
| `RCMS` | `REAL` expression | Width of right margin (in cms) |
| `BCMS` | `REAL` expression | Width of bottom margin (in cms) |
| `TCMS` | `REAL` expression | Width of top margin (in cms) |

**Description**

By default, the peripheral margin around a group of pictures is treated as two pairs of complementary margins (left/right and bottom/top) and each pair is distributed in a 2 : 1 ratio, *left*:*right* and *bottom*:*top*. `PAGMRG` specifies the size and distribution of these peripheral margins individually.

Three combinations of values are possible for each pair of margins, $m_1$ and $m_2$:

$m_1 \geq 0$ **and** $m_2 \geq 0$ sizes used unless shrinkage is needed when proportions used.

$m_1 \geq 0$ **and** $m_2 < 0$ (or *vice versa*) non-negative value used as the absolute margin size if page size permits; remaining space put on other side.

$m_1 < 0$ **and** $m_2 < 0$ reset default.

Simpleplot imposes minimum and maximum values on peripheries. The maximum size allowed for a periphery is half the smaller dimension of the group, The minimum periphery is 0.01cm.

`PAGMRG` overrides `PERIPH` and *vice versa*.

**Default**

`CALL PAGMRG(-1.0,-1.0,-1.0,-1.0)` restores the default size and distribution of peripheries.

**See also**

`PERIPH`, `PICMRG` and `MARGIN`.

### SUBROUTINE PAGPOS (XCMS, YCMS)

**Name**

`PAGPOS` – to specify the position of the SIMPLEPLOT page within the plotting area.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `XCMS` | `REAL` expression | $x$ coordinate of picture |
| `YCMS` | `REAL` expression | $y$ coordinate of picture |

**Description**

By default, if `PAGE` has been called to define a page size which is smaller than the device plotting area, this smaller region is positioned in the centre of the space available. `PAGPOS` overrides this position and cause the SIMPLEPLOT page to be placed at the point $(x_{cms}, y_{cms})$ centimetres relative to the bottom left hand corner of the device. The conversion subroutine, `KXYXY`, can be used to convert user coordinates or page coordinates to device coordinates in centimetres.

If the specified coordinates would position the page partly off the device plotting area, both the page dimensions and the position of its origin are scaled in the same proportion to ensure a SIMPLEPLOT page completely contained within the device plotting area.

`PAGPOS` is only applicable to fixed picture devices and is ignored until `PAGE` is called.

**Diagnostics**

`(PAGPOS cancelled)`[1]

**Default**

Calling `PAGPOS` with either argument negative restores the default page positioning.

**See also**

`PAGE` and `KXYXY`.

## SUBROUTINE PAGVW (ITYPE)

### Name

`PAGVW` – to specify the orientation of a page when page dimensions have not been specified.

**Availability** Section 4, released version 2-11.

### Argument

`ITYPE` `INTEGER` expression The orientation of the page

| `ITYPE` | *Page orientation* |
|---|---|
| 0 | Default mode, determined by device and/or `GROUP` |
| 1 | Portrait (if page can be turned) |
| 2 | Landscape (if page can be turned) |
| 3 | Portrait |
| 4 | Landscape |
| 5 | Upside-down portrait |
| 6 | Upside-down landscape |

### Description

By default, Simpleplot determines the page orientation according to the dimensions of the device and the group configuration of pictures. `PAGVW` specifies an alternative orientation of a page for *fixed picture* devices.

`ITYPE` $\geq$ 3 may result in a rotated page even on devices which would not normally turn the Simpleplot page (*eg.* a screen). `ITYPE`=1 and 3 or `ITYPE`=2 and 4 are equivalent on devices which can be used in both portrait and landscape modes (*eg.* printers).

`PAGVW` is overridden by calling of `PAGE` to define page dimensions; if the default page dimensions are subsequently restored, any previous setting of `PAGVW` becomes active again.

### Default

`CALL PAGVW(0)` restores the default.

### See also

`FITPAG`, `GROUP` and `PAGE`.

## SUBROUTINE PAPINC (CMS)

### Name

`PAPINC` – to specify the length of paper on a device with continuous paper feed.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Length of paper (in cms) |

### Description

On devices with continuous paper feed (*eg.* drum plotter) the paper length available to an individual program can be increased from the installation default, up to any value not exceeding the installation maximum, using `PAPINC`.

For further information about the default and maximum, please refer to local documentation.

`PAPINC` must be called before plotting is started for it to be effective.

### Default

`CALL PAPINC(0.0)` restores the default.

## SUBROUTINE PEN (IPEN)

### Name

PEN – to select the pen to be used for all plotting.

**Availability** Section 1, released before version 2-5.

### Argument

IPEN INTEGER expression Pen number

| IPEN | *Pen usage* |
|---|---|
| −1 | plotting is omitted on *all* devices |
| 0 | plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted |
| 1, 2, 3 . . . | pens/bundles are selected as determined by the current device, palette (see PENHLS and PENRGB) and the current bundled attributes (see BUNLPR) |

### Description

By default, SIMPLEPLOT draws all graphics (*eg.* lines, text, marker symbols, *etc.*) with a single pen (IPEN=1). After PEN has been called, all plotting is performed using the specified pen (IPEN) until another change is initiated by a further call of PEN, SETPNS or WFPNS.

IPEN can have values −1, 0, 1, 2, 3, . . . *etc.* CALL PEN(I) is equivalent to CALL SETPNS(I,I,I,I).

Please note that, by default, colour is the only line-drawing attribute associated with pens and the distinction between pens depends on the capabilities of the current device. BUNLPR specifies the attributes to be associated with pens when used for drawing lines (but not text, marker symbols nor software shading) to enable differentiation between pens even on monochrome devices.

### Default

CALL PEN(1) restores the default.

### See also

BUNLPR, PENHLS, PENRGB, SETPNS, ONEPEN, SHPEN and WFPNS.

## SUBROUTINE PENHLS (ICOLOR, HUE, BRIGHT, SAT)

**Name**

`PENHLS` – to change the device palette using HLS conventions.

**Availability** Section plus, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `ICOLOR` | `INTEGER` expression | Colour index to be changed |
| `HUE` | `REAL` expression | Hue (degrees) |
| `BRIGHT` | `REAL` expression | Lightness (0.0–1.0) |
| `SAT` | `REAL` expression | Saturation (0.0–1.0) |

**Description**

On devices which allow palette changes, `PENHLS` changes the specified colour index (or device pen) to a specified Hue-Lightness-Saturation. The HLS values required to produce specific colours are listed in Appendix T of the *SIMPLEPLOT Supplement*.

`ICOLOR`=−1 sets all colour indices (or device pens) to the specified colour.

The effect of defining the palette is affected by the capability of the device and the underlying graphics system. Possible effects include:

**No change** The device (or device driver) is unable to control the palette.

**Retrospective change** All graphics (*eg.* lines, text, filled areas, *etc.*) already drawn on the current SIMPLEPLOT page, with the specified colour index, are changed to the default palette definition.

**Prospective change** Only graphics subsequently drawn with the specified colour index use the default palette definition.

**Deferred change** The new palette definition takes effect from the start of the next SIMPLEPLOT page.

Unless any of these effects is actually required, the best time to change the palette is between finishing one SIMPLEPLOT page and starting the next (see `ENDPAG` and `NEWPAG`) or before the first page.

If `BUNLPR` has been called to change line-drawing attributes, the SIMPLEPLOT pen number will not necessarily correspond to `ICOLOR`.

**Default**

`CALL PENRST(ICOLOR)` restores the default. The default is not restored by `INITSP`.

**See also**

`BUNLPR`, `ENDPAG`, `NEWPAG`, `PENRST` and `PENRGB`.

## SUBROUTINE PENRGB (ICOLOR, RED, GREEN, BLUE)

### Name

PENRGB – to change the device palette using RGB conventions.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| ICOLOR | INTEGER expression | Colour index to be changed |
| RED | REAL expression | Proportion of RED (0.0–1.0) |
| GREEN | REAL expression | Proportion of GREEN (0.0–1.0) |
| BLUE | REAL expression | Proportion of BLUE (0.0–1.0) |

### Description

On devices which allow palette changes, PENRGB changes the specified colour index (or device pen) to a specified combination of Red-Green-Blue. The RGB values required to produce specific colours are listed in Appendix T of the *SIMPLEPLOT Supplement*.

ICOLOR=−1 sets all colour indices (or device pens) to the specified colour.

The effect of defining the palette is affected by the capability of the device and the underlying graphics system. Possible effects include:

**No change** The device (or device driver) is unable to control the palette.

**Retrospective change** All graphics (*eg.* lines, text, filled areas, *etc.*) already drawn on the current SIMPLEPLOT page, with the specified colour index, are changed to the default palette definition.

**Prospective change** Only graphics subsequently drawn with the specified colour index use the default palette definition.

**Deferred change** The new palette definition takes effect from the start of the next SIMPLEPLOT page.

Unless any of these effects is actually required, the best time to change the palette is between finishing one SIMPLEPLOT page and starting the next (see ENDPAG and NEWPAG) or before the first page.

If BUNLPR has been called to change line-drawing attributes, the SIMPLEPLOT pen number will not necessarily correspond to ICOLOR.

### Default

CALL PENRST(ICOLOR) restores the default. The default is not restored by INITSP.

### See also

BUNLPR, ENDPAG, NEWPAG, PENRST and PENHLS.

### SUBROUTINE PENRST (ICOLOR)

**Name**

`PENRST` – to reset the the device palette.

**Availability** Section plus, released version 2-11.

**Argument**

| | | |
|---|---|---|
| `ICOLOR` | `INTEGER` expression | Colour index to be changed |

**Description**

On devices which allow palette changes, `PENRGB` or `PENHLS` can be called to change the colour indices; `PENRST` resets the colour index, `ICOLOR`, to the value defined in the device driver to be the default colour for that index. This can only take account of user-defined settings if the device driver can interrogate the device.

The effect of resetting the palette may cause different visual effects depending on the capability of the device or the underlying graphics system.

Possible effects include **No change** – the device (or device driver) is not able to control the palette therefore resetting has no effect; **Retrospective change** – all graphics (*eg.* lines, text, filled areas, *etc.*) already drawn on the current SIMPLEPLOT page, with the specified colour index, are changed to the default palette definition; **Prospective change** – only graphics subsequently drawn with the specified colour index use the default palette definition; **Deferred change** – the default palette definition takes effect from the start of the next SIMPLEPLOT page.

Unless any of these effects is actually required, the best time to change the palette is when there is no active SIMPLEPLOT page (*ie.* after finishing one SIMPLEPLOT page but before starting the next (see `ENDPAG` and `NEWPAG`) or before the first page).

Please note that, by default, colour is the only line-drawing attribute associated with pens and the distinction between pens depends on the capabilities of the current device. If `BUNLPR` has been called to change line-drawing attributes, the SIMPLEPLOT pen number (`IPEN`) does not necessarily correspond to the colour index (`ICOLOR`).

**Default**

`CALL PENRST(-1)` restores all colour indices.

**See also**

`BUNLPR`, `ENDPAG`, `NEWPAG`, `PENRGB` and `PENHLS`.

## SUBROUTINE PERIM

### Name

PERIM – to draw a rectangular box around the current picture.

**Availability** Section 1, released version 2-5.

### Arguments

None.

### Description

PERIM draws [1] a rectangular box around the perimeter of the picture which has already been started.

If scales have been extended to provide similar linear scales in $x$ and $y$ (see EQSCAL), the box is drawn around the full extent of the scales at the picture limits. If a box is required to fit the requested range of scales, and the axes drawn to annotate such scales, DRAWBX can be used.

### See also

BOXPIC and DRAWBX.

## SUBROUTINE PERIPH (CMS)

### Name

PERIPH – to specify the overall size of the periphery around a group of pictures.

**Availability** Section 4, released version 2-7.

### Argument

| | | |
|---|---|---|
| CMS | REAL expression | Size of periphery (in cms) |

### Description

If neither PERIPH nor PAGMRG has been called, SIMPLEPLOT controls both the size and distribution of the periphery; PERI is calculated using the current minimum size of text:

PERI = 14 × *character width* + 1.5 × *margin size*

If the calculated default value of PERI is less than 3cm, a 3cm periphery is allocated, otherwise, PERI is used as the periphery size.

SIMPLEPLOT imposes minimum and maximum values on peripheries. The maximum size allowed for a periphery is half the smaller dimension of the group. The a minimum periphery is 0.01cm.

The overall peripheral margin is distributed in a 2 : 1 ratio, *left*:*right* and *bottom*:*top*.

PERIPH overrides PAGMRG and *vice versa*.

### Diagnostics

($v_1$ requested)$^2$<br>
($v_2$ used)$^2$<br>
(Periphery modified)$^1$

### Default

Calling PERIPH with a negative argument restores the default.

### See also

PGFULL and PAGMRG.

## SUBROUTINE PGFULL (TORF)

### Name

`PGFULL` – to specify whether picture space is to be maximized.

**Availability** Section 1, released version 2-9.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether page is to be filled |

### Description

By default, the periphery around a group of pictures surround the margins on the picture(s). After a call of `PGFULL(.TRUE.)`, the periphery and the picture margins coincide, allowing bigger pictures to be included in the page.

### Default

`CALL PGFULL(.FALSE.)` restores the default.

### See also

`MARGIN`, `PAGMRG`, `PERIPH` and `PICMRG`.

## SUBROUTINE PIBOXL (TORF)

**Name**

`PIBOXL` – to specify whether a box is to be drawn around pie chart labels.

**Availability** Section 4, released before version 2-5.

**Argument**

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether a box is to be drawn |

**Description**

By default, labels for pie chart segments are enclosed in a rectangular box. `PIBOXL` changes to no boxes, or back to drawing boxes. `PIBOXL` must be called before `PIECHT`.

**Default**

`CALL PIBOXL(.TRUE.)` restores the default.

**See also**

`PIECHT`.

### SUBROUTINE PICMRG (CMS, RCMS, BCMS, TCMS)

**Name**

`PICMRG` – to specify the size of individual picture margins.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Width of left margin (in cms) |
| `RCMS` | `REAL` expression | Width of right margin (in cms) |
| `BCMS` | `REAL` expression | Width of bottom margin (in cms) |
| `TCMS` | `REAL` expression | Width of top margin (in cms) |

**Description**

By default, the picture margin is treated as two pairs of complementary margins (left/right and bottom/top) and each pair is distributed such that there is enough clearance round pictures for normal plotting requirements (*ie.* with more space to the left and bottom of the picture). `PICMRG` changes the size and distribution of these picture margins.

Three combinations of values are possible for each pair of margins, $m_1$ and $m_2$:

$m_1 \geq 0$ **and** $m_2 \geq 0$ sizes used unless shrinkage is needed when proportions used.

$m_1 \geq 0$ **and** $m_2 < 0$ (or *vice versa*) non-negative value used as the absolute margin size if page size permits; remaining space put on other side.

$m_1 < 0$ **and** $m_2 < 0$ reset default.

Simpleplot imposes minimum and maximum values on picture margins. The maximum size allowed for a margin is half the smaller dimension of the picture, and the minimum margin is 0.01cm.

`PICMRG` overrides `MARGIN` and *vice versa.*

**Default**

`CALL PICMRG(-1.0,-1.0,-1.0,-1.0)` restores default margin sizes and distribution.

**See also**

`MARGDV`, `MARGIN`, `PAGMRG` and `PERIPH`.

## SUBROUTINE PICPOS (XCMS, YCMS)

### Name

PICPOS – to specify the coordinate position of each new picture.

**Availability** Section 4, released version 2-8.

### Arguments

| | | |
|---|---|---|
| XCMS | REAL expression | $x$ coordinate of picture |
| YCMS | REAL expression | $y$ coordinate of picture |

### Description

By default, each new picture is automatically positioned either in the next clear space within the group or on a new SIMPLEPLOT page. PICPOS specifies the coordinates, $(x_{cms}, y_{cms})$, at which the bottom left-hand corner of each new picture is to be positioned. XCMS and YCMS are specified in centimetres relative to the bottom left hand corner of the current SIMPLEPLOT page; if there is no current page, a page is automatically started.

PICPOS overrides the automatic layout (there is an implicit call of MOVEON(.FALSE.)) and while PICPOS is active, ENDPAG terminates the page.

If either XCMS or YCMS exceed the page limits, or if the picture is too big to fit within the page limits at the specified position, the position is moved in one or both directions, until the picture is completely within the page, and a diagnostic is issued.

The picture size can be specified with PICSIZ. If PICPOS is active but no picture size has been specified, a 12cm × 12cm picture is used.

### Diagnostics

($x_1$ $y_1$ requested)$^2$
($x_2$ $y_2$ used)$^2$
(PICPOS cancelled)$^1$
(Picture position modified)$^1$

### Default

CALL PICPOS(XCMS,YCMS) with either argument negative.

CALL MOVEON(.TRUE.) also cancels PICPOS.

### See also

ENDPAG, ENDPIC, KXYXY and MOVEON.

## SUBROUTINE PICRST (N)

### Name

`PICRST` – to reset the current picture to the size, position and scales saved as number `N`.

**Availability** Section 1, released version 2-14.

### Argument

| | | |
|---|---|---|
| `N` | `INTEGER` expression | Picture identifier (1 to 10) |

### Description

`PICRST` and `PICSAV` facilitate simultaneous drawing on separate picture. Following a call to `PICSAV(N)`, `CALL PICRST(N)` switches drawing to the picture associated with `N`.

### Diagnostics

`[PICRST: argument invalid]`$^1$

### See also

`PICSAV`.

## SUBROUTINE PICSAV (N)

**Name**

`PICSAV` – to save the size, position and scales of the current picture as number `N`.

**Availability** Section 1, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `N` | `INTEGER` expression | Picture identifier (1 to 10) |

**Description**

`PICSAV` facilitates simultaneous drawing on separate picture. Following a call to `PICSAV(N)`, `CALL PICRST(N)` switches drawing to the picture associated with `N`.

**Diagnostics**

`[PICSAV: argument invalid]`[1]

**See also**

`PICRST`.

## SUBROUTINE PICSIZ (WCMS, HCMS)

### Name

`PICSIZ` – to specify the size of pictures.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `WCMS` | `REAL` expression | Width of pictures (in cms) |
| `HCMS` | `REAL` expression | Height of pictures (in cms) |

### Description

By default, Simpleplot calculates the picture size from the page size as follows:

1. Subtract the periphery (if any),
2. Divide by the group size,
3. Subtract the margins, and
4. Apply a shrinkage factor (if necessary) to fit the space available on the device.

`PICSIZ` specifies the target size of all pictures. If the requested picture size is larger than the space available, a shrinkage factor is applied so that the proportions (or aspect ratio) of the specified dimensions are retained.

If `PICPOS` is active but no picture size has been specified, a 12cm × 12cm picture is used.

### Diagnostics

($x_1$ $y_1$ `requested)`$^2$
($x_2$ $y_2$ `used)`$^2$
`(PICSIZ cancelled)`$^1$
`(PICSIZ ignored: invalid argument)`$^1$
`(Picture Size modified)`$^1$

### Default

Calling `PICSIZ` with either argument set to zero restores the default.

### See also

`GROUP`, `PAGE` and `PICPOS`.

## SUBROUTINE PIDIAM (CMS)

### Name

`PIDIAM` – to specify a reduced diameter for pie charts.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Diameter (in cms) |

### Description

By default, the diameter of pie charts is the largest which can be accommodated with the required configuration of exploded segments and labels, within the picture size. `PIDIAM` sets the pie diameter in centimetres. `PIDIAM` must be called before `PIECHT`.

### Default

`CALL PIDIAM(0.0)` restores the default.

### See also

`PIECHT`.

### SUBROUTINE PIECHT (DARR, LABARR, NARR)

**Name**

`PIECHT` – to draw a pie chart on the current picture.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `DARR` | `REAL` array | Data values |
| `LABARR` | `STRING` array | Names for each segment |
| `NARR` | `INTEGER` expression | Number of segments and number of elements in `DARR` and `LABARR` |

**Description**

`PIECHT` draws a pie chart on the current picture showing the distribution of data among the `NARR` values in array `DARR`. The details of labelling, shading patterns, tilt of pie, positions of separate segments *etc.* can all be controlled by prior calls of `PI*` or `SG*` subroutines or, in the absence of such calls, defaults are used.

The number of segments of a pie chart is unlimited but the limit on the maximum number of segments which can be omitted or exploded is 32.

**Diagnostics**

`(PIE CHART OMITTED: LESS THAN 1 SEGMENT)`[1]
`(PIE CHART OMITTED: -VE AND +VE VALUES)`[1]

**See also**

`PIBOXL`, `PIDIAM`, `PIEXPL`, `PIINCL`, `PIOMIT`, `PIPOSL`, `PITILT`, `SGEXPL`, `SGOMIT` and `SHKEYS`.

### SUBROUTINE PIEXPL (IARR, NARR)

**Name**

`PIEXPL` – to specify the amount by which each pie chart segment is to be exploded.

**Availability** Section 4, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `IARR` | `INTEGER` array | Percentage (0–50%) explosion for each segment |
| `NARR` | `INTEGER` expression | Number of elements in `IARR` |

**Description**

By default, pie charts are drawn with all segments touching, and meeting in the centre of the pie. `PIEXPL` indicates the amount (percentage of the radius) by which each segment is to be displaced from the centre of the pie, along the radius bisecting the segments' angles. `PIEXPL` must be called before `PIECHT`.

The number of segments of a pie chart is unlimited but the limit on the maximum number of segments which can be omitted or exploded is 32.

The effect of `PIEXPL` can be overridden by `SGEXPL` and *vice versa*.

**Default**

Calling `PIEXPL` with all elements of `IARR` set to zero restores the default.

**See also**

`PIECHT` and `SGEXPL`.

## SUBROUTINE PIINCL (LTORF, VTORF, PTORF)

### Name

PIINCL – to specify what elements are to be included in pie chart segment labels.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| LTORF | LOGICAL expression | Whether segment names are to be included in labels |
| VTORF | LOGICAL expression | Whether segment values are to be included in labels |
| PTORF | LOGICAL expression | Whether segment percentages are to be included in labels |

### Description

By default, PIECHT draws each pie segment labelled with the name supplied in the argument array LABARR. PIINCL specifies any combination of segment name, segment value and segment percentage, to be used for labelling all segments. PIINCL must be called before PIECHT.

By default, percentages are specified to 2 decimal places (see FIGFMT) and labels are left-justified (see ADDJST).

### Default

CALL PIINCL(.TRUE.,.FALSE.,.FALSE.) restores the default.

### See also

ADDJST, FIGFMT, PIECHT, PIBOXL and PIPOSL.

## SUBROUTINE PIOMIT (LARR, NARR)

### Name

`PIOMIT` – to specify which segments are to be omitted from pie charts.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `LARR` | `LOGICAL` array | Specifies for each segment whether or not it is to be omitted |
| `NARR` | `INTEGER` expression | Number of elements in `LARR` |

### Description

By default, all segments of pie charts are drawn by `PIECHT`. `PIOMIT` indicates that specific segments for which the value `LARR(ISEG)` is `.TRUE.`, are to be included in the calculations, but omitted from the picture. `PIOMIT` must be called before `PIECHT`.

The number of segments of a pie chart is unlimited but the limit on the maximum number of segments which can be omitted or exploded is 32.

The effect of `PIOMIT` can be overridden by `SGOMIT` and *vice versa*.

### Default

Calling `PIOMIT` with all elements of `LARR` set to `.FALSE.` restores the default.

### See also

`PIECHT` and `SGOMIT`.

## SUBROUTINE PIPOSL (IPOS)

### Name

`PIPOSL` – to specify the position of segment labels on pie charts.

**Availability** Section 4, released before version 2-5.

### Argument

`IPOS` `INTEGER` expression Label position

| `IPOS` | *Label position* |
|---|---|
| 1 | Labels equally-spaced down sides of picture, with arrows if there is room (default) |
| 2 | Labels alongside segments, down sides of picture, with arrows if there is room |
| 3 | Labels next to segments (no arrows) |
| 4 | Labels inside segments, or next to segment when there is not enough space inside (no arrows) |

### Description

By default, `PIECHT` draws segment labels equally spaced down the sides of the picture. `PIPOSL` selects any of the four options for label positions.

### Default

`CALL PIPOSL(1)` restores the default.

### See also

`PIECHT`, `PIBOXL` and `PIINCL`.

## SUBROUTINE PITILT (IWDTH, IHGHT)

**Name**

`PITILT` – to specify the tilt of pie charts.

**Availability** Section 4, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `IWDTH`,<br>`IHGHT` | `INTEGER` expressions | Width to height ratio |

**Description**

By default, circular pie charts are drawn by `PIECHT`. Tilted pie charts are drawn by adjusting the *w*idth:height ratio to produce elliptical pies. `PITILT` indicates the required ratio; both `IWDTH` and `IHGHT` must be positive integers, and only values with `IWDTH` > `IHGHT` are accepted. `PITILT` must be called before `PIECHT`.

**Default**

`CALL PITILT(0,0)` restores the default.

**See also**

`PIECHT`.

## SUBROUTINE POLAR7 (RADIUS, CAP)
## SUBROUTINE POLAR (RADIUS, CAP, NCAP)

### Name

`POLAR7` – to start a new polar picture and draw axis framework.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `RADIUS` | `REAL` expression | Maximum value of radial scale |
| `CAP` | `STRING` expression | Caption for radial axis |
| `NCAP` | `INTEGER` expression | Number of characters in `CAP` |

### Description

`POLAR`/`POLAR7` starts a new picture, sets linear scales from −`RADIUS` to +`RADIUS` in both horizontal and vertical directions, draws [1] a radial axis annotated [2] with the scale, and draws [1] an angular axis annotated [2] with the angular scale of degrees.

`POLAR7` sets the underlying coordinate system to be polar (in degrees). Even if Cartesian units have been specified by `EQSCAL`, when a new picture is started by `POLAR7`, a polar axis framework is drawn over complete quadrants and the plotting units are switched to degrees.

### Diagnostics

`(POLAR OMITTED: ZERO RADIUS)`[1]

### See also

`AXES7`, `AXIS7`, `COORDS`, `EQSCAL`, `NEWPIC`, `POLRNG` and `POLZER`.

## SUBROUTINE POLIN (PARR, NP)

### Name

`POLIN` – to transfer polynomial coefficients before plotting curves of polynomial functions.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `PARR` | `REAL` array | Polynomial coefficients |
| `NP` | `INTEGER` expression | Number of polynomial coefficients (1–20) |

### Description

`POLIN` copies the values of `PARR(1)` to `PARR(NP)` into storage within SIMPLEPLOT for later use in calculating the polynomial function:

$$\texttt{PARR(1)} + \texttt{PARR(2)}x + \texttt{PARR(3)}x^2 + \ldots + \texttt{PARR(NP)}x^{\texttt{NP}-1}$$

using `POLY`, normally with `DRAWFN`/`BRKNFN`/`SHDEFN`.

### See also

`POLOUT` and `POLY`.

## SUBROUTINE POLOUT (PARR)

### Name

`POLOUT` – to transfer polynomial coefficients after plotting regression curves.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `PARR` | `REAL` array | To receive polynomial coefficients |

### Description

`POLOUT` copies values of polynomial coefficients from within SIMPLEPLOT into `PARR`. If `CVTYPE` has been called with argument 4 or 5 to cause `BRKNAR/DRAWAR/SHDEAR` or `BRKNCV/DRAWCV/SHDECV` to draw regression curves, the program can obtain the regression coefficients by calling `POLOUT` afterwards. `PARR(I)` is used as the coefficient of $x^{i-1}$.

### See also

`CVTYPE`, `POLIN` and `POLY`.

## SUBROUTINE POLRNG (THSTRT, THSTOP)

### Name

`POLRNG` – to specify the user-defined polar coordinate system.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `THSTRT` | `REAL` expression | Value of scale at 0° |
| `THSTOP` | `REAL` expression | Value of scale at 360° |

### Description

By default, the angular scale used when `COORDS` has been called with `IUNITS`=3, is in radians. `POLRNG` specifies an alternative coordinate system.

`POLRNG` defines the range of a complete cycle of the angular axis from `THSTRT` to `THSTOP`. `CALL EQSCAL(..., 3)` specifies the plotting scales for polar plotting in with respect to this cyclical range.

### Default

Calling `POLRNG` with `THSTRT`=`THSTOP` restores the default.

### See also

`COORDS`, `EQSCAL` and `POLZER`.

## REAL FUNCTION POLY (RVAL)

### Name

POLY – to evaluate a polynomial function.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| RVAL | REAL expression | Argument of polynomial function in units of plotting scales |

### Description

POLY calculates the value, at the specified argument value, of the polynomial function whose coefficients have already been set up using POLIN. Its primary use is as an argument of DRAWFN, BRKNFN or SHDEFN. For example,

```
CALL DRAWFN(POLY)
CALL BRKNFN(POLY,LTYPE)
```

Please note that any program segment using POLY in this way, must declare POLY as an external subroutine. For example, EXTERNAL POLY.

This mechanism is also used internally within DRAWCV, BRKNCV and SHDEFN when CVTYPE has been called to specify linear or quadratic regressions.

### See also

POLIN and POLOUT.

## SUBROUTINE POLZER (RZERO, IRDIR, THZERO, ITHDIR)

### Name

`POLZER` – to specify the convention used for polar coordinates.

**Availability** Section 1, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `RZERO` | `REAL` expression | Radial coordinate at centre of chart |
| `IRDIR` | `INTEGER` expression | Direction of radial scale |
| `THZERO` | `REAL` expression | Angle on chart of zero angular coordinate |
| `ITHDIR` | `INTEGER` expression | Direction of angular scale |

| `IRDIR` or `ITHDIR` | *Direction of radial increase* | *Direction of angular increase* |
|---|---|---|
| -ve | Decreasing outwards | Clockwise |
| 0 | Increasing outwards (default) | Either way depending on directions of change of $x$ & $y$ (default) |
| +ve | Increasing outwards | Anticlockwise |

### Description

Polar coordinates describe a point $z = f(r, \theta)$ where $r$ is the distance of the point from an origin, and $\theta$ is the angle between a fixed line through the origin and a line joining the point to the origin. By default, Simpleplot assumes that radial units increase from zero at the Cartesian origin, $(x = 0, y = 0)$, and that angles increase in the direction of increasing $y$ from zero at a fixed horizontal line $y = 0$. `POLZER` specifies an alternative convention for polar coordinates.

`RZERO` specifies the radial scale value to correspond with the Cartesian origin. `THZERO` specifies the anticlockwise angle (in degrees) between the horizontal line from the origin to the right, and the chosen position for the fixed line. `IRDIR` and `ITHDIR` indicate the direction of increase of the radial scale and of angular change respectively.

After a call to `POLZER`, all polar pictures are drawn according to the specified convention, and all polar coordinates are interpreted similarly.

### Default

`CALL POLZER(0.0,0,0.0,0)` restores the default.

### See also

`COORDS`, `EQSCAL` and `POLRNG`.

## SUBROUTINE PUTCP7 (IROW, ICOLM, CAP)<br>SUBROUTINE PUTCAP (IROW, ICOLM, CAP, NCAP)

### Name

PUTCP7 – to draw a caption in a specified position in a caption area.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| IROW | INTEGER expression | Line number |
| ICOLM | INTEGER expression | Character position |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

PUTCAP/PUTCP7 draws [2] the single caption, CAP, in the specified position within a caption area defined by DEFCAP (and selected by RESNXT). The position is determined as line IROW (counting down from IROW=1 at top), character position ICOLM (counting across from ICOLM=1 at left).

### Diagnostics

(Caption truncated)$^2$
(No active key/caption area)$^2$

### See also

ADDCP7, DEFCAP, KNUMB and KREAL.

### SUBROUTINE QCURVE (MAXPTS, XARR, YARR, NPTS)

**Name**

QCURVE – to return the coordinates of a stored curve.

**Availability** Section plus, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| MAXPTS | INTEGER expression | Maximum points to receive per call |
| XARR, YARR | REAL arrays | Dimensioned at least MAXPTS to receive coordinates of points |
| NPTS | INTEGER variable | To receive number of points |

**Description**

Some Simpleplot operations can remember coordinates of curves for extraction by QCURVE (see CTHOLD). Each call of QCURVE can return between 0 and MAXPTS coordinates in arrays XARR and YARR; NPTS is set to the number of points returned. The stored coordinates can constitute any number of separate curves which can be extracted by repeated calls of QCURVE. Any curves containing more than MAXPTS points will also require multiple calls of QCURVE.

| NPTS | *Interpretation of replies* |
|---|---|
| NPTS=MAXPTS | The curve may be incomplete either because there are more points to this curve segment or because there is another curve segment; continuation of the current segment is signified by the first point received by next call of QCURVE duplicating the value of last point of the previous segment: (XARR(MAXPTS),YARR(MAXPTS)) |
| 0 < NPTS < MAXPTS | (XARR(NPTS), YARR(NPTS)) is the final point of a curve |
| NPTS=0 | No coordinates retrieved – end of process |

If there are no stored coordinates to retrieve, a diagnostic is issued.

**Diagnostics**

(Curve storage empty)[1]

**See also**

CTHOLD, ZEDGE, ZELEMS, ZZEDGE, ZZELMS, *CONT, and *CUT.

### SUBROUTINE QDEV (NDEV, NSITE, NABS, SNAME, SDESC)

**Name**

QDEV – to inquire details of a menu entry for a specific device option.

**Availability** Section plus, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| NDEV | INTEGER expression | Position of option in S.E.P. menu |
| NSITE | INTEGER variable | To receive *site number* |
| NABS | INTEGER variable | To receive *absolute number* |
| SNAME | STRING variable | To receive *site name* |
| SDESC | STRING variable | To receive *description* |

| NDEV | *Device option* |
|---|---|
| 0 | the currently selected device option |
| >0 | the NDEVth entry in the menu |

**Description**

The Simpleplot Single Entry Point device driver system provides device options which are tabulated when interactive device selection is requested and displayed as a menu. Each device option has associated with it a *site number*, *absolute number*, *site name* and *description.* QDEV inquires the values of NSITE, NABS, SNAME and SDESC for any device option; the value of NDEV indicates which device option is required.

If no device is open and NDEV=0, or if there is no NDEVth entry in the menu, QDEV returns NSITE=0, NABS=0, and SNAME and SDESC are both filled with spaces.

In order to ensure that the full text string is received, SNAME and SDESC should be declared as CHARACTER*20 and CHARACTER*40 respectively.

**See also**

DEVNAM and DEVNO.

## SUBROUTINE QFFSIZ (WCMS, HCMS)

### Name

`QFFSIZ` – to inquire the physical size of current fast area-fill characters.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `WCMS` | `REAL` variable | To receive width (in cms) |
| `HCMS` | `REAL` variable | To receive height (in cms) |

### Description

`QFFSIZ` inquires the width and height (in centimetres) of the current fast area-fill characters. It is only possible to find out this information when a SIMPLEPLOT page has been started; if `QFFSIZ` is called when there is no page, zero width and height are returned. If `QFFSIZ` is to be used in order to establish a suitable picture size, first use `NEWPAG` to start a page.

If `QFFSIZ` is called without a prior call of `FFSIZE`, the current default size for shaded contour maps is returned. The default size for raster pictures (*eg.* shaded surfaces) can be determined by preceding a call to `QFFSIZ` with `CALL FFSIZE(0.5,0.5)`.

### See also

`FFSET`, `FFSIZE` and `NEWPAG`.

### SUBROUTINE QIBRFL (D3ARR, NX, NY, NSETS, ZMIN, ZMAX)

**Name**

`QIBRFL` – to inquire the $z$ scale required by an *SIMPLEPLOT Isometric* 3-D barchart.

**Availability** Section plus, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| `D3ARR` | `REAL` 3-D array | `NX` × `NY` × `NSETS` array of user ($z$) values |
| `NX`, `NY`, `NSETS` | `INTEGER` expressions | Dimensions of 3-D data array |
| `ZMIN` | `REAL` variable | To receive minimum $z$ |
| `ZMAX` | `REAL` variable | To receive maximum $z$ |

**Description**

`QIBRFL` returns the minimum and maximum values required on the $z$ scale for `ISBRFL` to plot the data in array `D3ARR` with the current base $z$ value. This may be used to establish the $z$ scale to set 3-D scales before drawing a stacked barchart.

**Diagnostics**

`(QIBRFL invalid dimensions)`[1]

**See also**

`ISBRBZ` and `ISBRFL`.

### SUBROUTINE QKYCAP (LABARR, NARR, ITYPE, NROWS, NCOLMS)

**Name**

QKYCAP – to inquire the size of a key or caption area to hold a set of captions.

**Availability** Section plus, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| LABARR | STRING array | Set of labels |
| NARR | INTEGER expression | Number of labels in LABARR |
| ITYPE | INTEGER expression | Intended use of text strings |
| NROWS | INTEGER variable | To receive number of rows |
| NCOLMS | INTEGER variable | To receive nominal number of columns |

| ITYPE | *Text usage* |
|---|---|
| 1 | Keys or captions relating to pictures |
| 2 | Keys or captions relating to groups/pages |

**Description**

QKYCAP inquires the number of equivalent rows and columns which a set of a text strings would fill in a key or caption area (see DEFCAP and DEFKEY). Because SIMPLEPLOT scales text according to whether the key (or caption area) relates to the picture or to the page, an additional argument, ITYPE is required to describe the text usage.

QKYCAP cannot be called until a new picture/page has been started.

**See also**

QSTR7, DEFCAP and DEFKEY.

## SUBROUTINE QNODAT (VARW)

### Name

`QNODAT` – to inquire the current value representing no-data.

**Availability** Section 1, released version 2-10.

### Argument

| | | |
|---|---|---|
| `VARW` | `REAL` variable | To receive no-data value |

### Description

`QNODAT` sets the variable `VARW` to the `REAL` value which is currently in use within Simpleplot to represent the absence of a data value. The default no-data value is $-1.0E20$.

### See also

`NODATA`.

## SUBROUTINE QPAGE (WCMS, HCMS)

### Name

`QPAGE` – to inquire the maximum page size of the current device.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `WCMS` | `REAL` variable | To receive page width (in cms) |
| `HCMS` | `REAL` variable | To receive page height (in cms) |

### Description

`QPAGE` inquires the page size of the current device; the values of `WCMS` and `HCMS` returned are centimetre values which can be used as a page size (`CALL PAGE(WCMS,HCMS)`) on the current device, with no shrinkage.

### See also

`PAGE`.

## SUBROUTINE QPIC (X0, Y0, XL, YL)

### Name

QPIC – to inquire the position and size of the current picture.

**Availability** Section 1, released version 2-14.

### Arguments

| | | |
|---|---|---|
| X0, Y0 | REAL variables | To receive centimetre coordinates of the bottom left corner |
| XL | REAL variable | To receive picture width in centimetres |
| YL | REAL variable | To receive picture height in centimetres |

### Description

QPIC inquires the position and size of the current picture. These may be useful for positioning subsequent pictures relative to this one.

A new picture must be started before QPIC is called.

## SUBROUTINE QPIXLS (NX, NY)

### Name

`QPIXLS` – to inquire number of pixels on current device.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `NX` | `INTEGER` variable | To receive number of pixels horizontally |
| `NY` | `INTEGER` variable | To receive number of pixels vertically |

### Description

`QPIXLS` inquires the numbers of pixels horizontally and vertically on the current device.

If `QPIXLS` is called before a device is open, the device opening procedure is performed before the reply is returned. If pixel details are not available (*eg.* on a non-pixel device) `NX`=0 and `NY`=0 are returned.

### See also

`QPAGE`.

## SUBROUTINE QSFLAB (VARMIN, VARMAX, VARSTP)

**Name**

`QSFLAB` – to inquire of the range of data and contour interval on latest 3-D plot.

**Availability** Section 2, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `VARMIN` | `REAL` variable | To receive first $z$ value |
| `VARMAX` | `REAL` variable | To receive last $z$ value |
| `VARSTP` | `REAL` variable | To receive contour interval |

**Description**

`QSFLAB` inquires the limits of the $z$ plotting range, and the interval between contours, on the latest surface plot, and place their `REAL` values in the variables `VARMIN`, `VARMAX` and `VARSTP`.

**See also**

`SFLAB`.

## SUBROUTINE QSPLOT (STR)

**Name**

`QSPLOT` – to inquire the current version of Simpleplot.

**Availability** Section 1, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `STR` | `STRING` variable | to receive software version |

**Description**

The first diagnostic message output by Simpleplot includes the current version; for example:

```
(SIMPLEPLOT Mark 2-15(000)F)
```

`QSPLOT` can be called to get a character representation of this version number, *eg.* `2-15(000)F`.

In order to ensure that the full text string is received, `STR` should be declared as `CHARACTER*10`.

## SUBROUTINE QSTAT (ISTAT)

### Name

`QSTAT` – to inquire the status of the last SIMPLEPLOT call.

**Availability** Section plus, released version 2-12.

### Argument

| | | |
|---|---|---|
| `ISTAT` | `INTEGER` variable | To receive status value |

### Description

Each SIMPLEPLOT subroutine sets a status flag to 0 before it performs any functions. During the execution of the subroutine, the status flag is changed to 1 whenever output is omitted due to lack of space or an invalid request. `QSTAT` returns the value of the status flag (in `ISTAT`) which resulted from the most recently called SIMPLEPLOT subroutine.

If `QSTAT` is the first SIMPLEPLOT subroutine called, `ISTAT` is returned with a value of $-1$.

`DIAGLV(2)` and `DIAGLV(4)` produce explanatory diagnostic messages for most of the conditions which set `ISTAT` to 1.

### See also

`DIAGLV`.

## SUBROUTINE QSTR7 (CAP, ITYPE, WCMS, HCMS)<br>SUBROUTINE QSTR (CAP, NCAP, ITYPE, WCMS, HCMS)

### Name

`QSTR7` – to inquire the physical size of a text string.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CAP` | `STRING` expression | Text string for analysis |
| `ITYPE` | `INTEGER` expression | Intended use of text string |
| `WCMS` | `REAL` variable | To receive width of text string |
| `HCMS` | `REAL` variable | To receive height of string |
| `NCAP` | `INTEGER` expression | Number of characters in `CAP` |

| `ITYPE` | *Text usage* |
|---|---|
| 0 | Normal usage – *eg.* picture or axis annotation |
| 1 | Keys or captions relating to pictures |
| 2 | Keys or captions relating to pages |
| 3 | Titles relating to pictures |
| 4 | Titles relating to pages |

### Description

`QSTR`/`QSTR7` inquire the physical height and width of a text string. Because SIMPLEPLOT scales text according to its use, an additional argument `ITYPE` is required to describe the text usage.

As the size of text depends on the size of the page, `QSTR7` cannot be called until a new picture/page has been started. If `QSTR7` is called with no active page, the values `WCMS`=0.0 and `HCMS`=0.0 are returned.

### See also

`NEWPAG` and `QKYCAP`.

### SUBROUTINE QV3DLM (VARX1, VARX2, VARY1, VARY2, VARZ1, VARZ2)

**Name**

`QV3DLM` – to inquire the three-dimensional limits for plottable data.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `VARX1` | `REAL` variable | To receive value at start of $x$ scale |
| `VARX2` | `REAL` variable | To receive value at end of $x$ scale |
| `VARY1` | `REAL` variable | To receive value at start of $y$ scale |
| `VARY2` | `REAL` variable | To receive value at end of $y$ scale |
| `VARZ1` | `REAL` variable | To receive value at start of $z$ scale |
| `VARZ2` | `REAL` variable | To receive value at end of $z$ scale |

**Description**

*SIMPLEPLOT ViSualization* scales are set to accommodate all data within a three-dimensional box. By default, the $x$, $y$ and $z$ limits of this box are all $-1.0$ to $+1.0$, although `VS3DLM` can change these limits.

`QV3DLM` inquires the data limits used for the current picture.

All points which lie inside the limits after transformations have been applied can be plotted. Other points may lie outside the 2-D SIMPLEPLOT scales.

`QV3DLM` may also be used to calculate the Origin, which is at the centre of the Limiting Box. This is at

$$(\frac{\texttt{VARX1+VARX2}}{2}, \frac{\texttt{VARY1+VARY2}}{2}, \frac{\texttt{VARZ1+VARZ2}}{2})$$

`QV3DLM` gives the limitis for the current picture. It must be called after the scales have been set by `VSNEW`. If `VSNEW` has not been called, all the variables are set to the current 'no data' value.

**Diagnostics**

`[QV3DLM: No current ViSualization picture]`[1]

**See also**

`VS3DLM`.

### SUBROUTINE QVBRFL (Z, NX, NY, NSETS, ZMIN, ZMAX)

**Name**

QVBRFL – to inquire the $z$ scale required by a *SIMPLEPLOT ViSualization* 3-D barchart.

**Availability** Section 7, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| Z | REAL 3-D array | NX × NY × NSETS array of user ($z$) values |
| NX, NY, NSETS | INTEGER expressions | Dimensions of 3-D data array |
| ZMIN | REAL variable | To receive minimum $z$ |
| ZMAX | REAL variable | To receive maximum $z$ |

**Description**

QVBRFL returns the minimum and maximum values required on the $z$ scale for VSBRFL to plot the data in array Z with the current base $z$ value. This may be used to establish the $z$ scale to set 3-D scales before drawing a stacked barchart.

**Diagnostics**

`[QVBRFL: Invalid array dimension]`[1]

**See also**

VS3DLM, VSBRBZ and VSBRFL.

### SUBROUTINE QVDPLM (VARMIN, VARMAX)

**Name**

QVDPLM – to inquire the range of depths for the current *SIMPLEPLOT ViSualization* picture.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| VARMIN | REAL variable | To receive minimum depth |
| VARMAX | REAL variable | To receive maximum depth |

**Description**

QVDPLM gives the range of depths of points inside the *SIMPLEPLOT ViSualization* Limiting Box. The limits for the box are set by VS3DLM, given the current scaling mode set by VSFULL.

Depth increases away from the Viewing Position – only points which are the other side of the Viewing Plane have positive depths.

The range of depths is affected by VSFULL as follows:

- VSFULL(0) (default), VARMIN $\geq -\sqrt{3}$ and VARMAX $\leq +\sqrt{3}$.
- VSFULL(1), VARMIN $= -\sqrt{3}$ and VARMAX $= +\sqrt{3}$.
- VSFULL(2), the range of depths depends on the transformations specified by VSMAG, VSROT, and VSTRAN.

QVDPLM gives the limits for the current picture. It must be called after the scales have been set by VSNEW. If VSNEW has not been called, all the variables are set to the current 'no data' value.

**Diagnostics**

[QVDPLM: No current ViSualization picture] [1]

**See also**

VS3DLM, VSFULL, VSVRTP and VSVXYZ.

### SUBROUTINE QVSCAL (VARXMN, VARXMX, VARYMN, VARYMX)

**Name**

`QVSCAL` – to inquire the scale limits in SIMPLEPLOT user coordinates corresponding to the current projected plotting area.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `VARXMN` | `REAL` variable | To receive minimum value of $x$ scale |
| `VARXMX` | `REAL` variable | To receive maximum value of $x$ scale |
| `VARYMN` | `REAL` variable | To receive minimum value of $y$ scale |
| `VARYMX` | `REAL` variable | To receive maximum value of $y$ scale |

**Description**

`QVSCAL` returns the maximum possible plotting area for the current *SIMPLEPLOT ViSualization* picture. The plotting limits may be altered by calls to `VS3DLM`, `VSFIT`, `VSFITP`, `VSFULL`, `VSVRTP` or `VSVXYZ`.

If a point is within the limits specified by `VS3DLM`, the range of possible $x$-$y$ values is determined by the Scaling Mode selected by `VSFULL`:

- `VSFULL(0)` (default),
  `VARXMN/VARYMN` $\geq -\sqrt{3}$ and `VARXMX/VARYMX` $\leq +\sqrt{3}$.
- `VSFULL(1)`, `VARXMN/VARYMN` $= -\sqrt{3}$, and `VARXMX/VARYMX` $= +\sqrt{3}$.
- `VSFULL(2)`, the 2-D limits depends on the transformations specified by `VSMAG`, `VSROT` and `VSTRAN`.

`QVSCAL` gives the limits for the current picture. It must be called after the scales have been set by `VSNEW`. If `VSNEW` has not been called, all the variables are set to the current 'no data' value.

**Diagnostics**

`[QVSCAL: No current ViSualization picture]`[1]

**See also**

`VS3DLM`, `VSFIT`, `VSFITP`, `VSFULL`, `VSVRTP` and `VSVXYZ`.

## SUBROUTINE QWZSCL (ZSTART, ZSTOP)

### Name

`QWZSCL` – to inquire the current $z$ scales for curves on waterfall charts.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `ZSTART`,<br>`ZSTOP` | `REAL` variables | To receive range of $z$ covered |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. Each curve is drawn on a scale covering the same $z$ range.

`QWZSCL` inquires the current scales for any single curve on the latest waterfall chart.

`QWZSCL` can not return a sensible value if no waterfall scales exist. If `WFCHT`, `WFDRAW` or `WFZSCL` has not been called already, a diagnostic is issued and `ZSTART` and `ZSTOP` are set to the current no-data value.

### Diagnostics

`(No current waterfall Z scale)`[1]

### See also

`KWZVAL` and `WFZSCL`.

### SUBROUTINE RANGE (X1, Y1, X2, Y2)

**Name**

`RANGE` – to draw a line indicating a range of values.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `X1`, `Y1` | `REAL` expressions | Coordinates of the first point |
| `X2`, `Y2` | `REAL` expressions | Coordinates of the second point |

**Description**

`RANGE` draws [1] the mathematically straight line from $(x_1, y_1)$ to $(x_2, y_2)$, with a short perpendicular line across each end. This line may be represented by a curve when plotted on non-linear scales or in polar coordinates.

The points $(x_1, y_1)$ and $(x_2, y_2)$ are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine `KXYXY`.

By default, all drawing is restricted by the limits of the picture scales; range bars can spill over the edge of a picture but the coordinates at either end must not exceed these limits. If coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

**Diagnostics**

`(**`$x_1, y_1, x_2, y_2$`**)`$^2$

**See also**

`CLIPLV`, `DRAWLN` and `BRKNPT`.

## SUBROUTINE RASTER (ICODE)

### Name

`RASTER` – to specify the shading technique to be used for shaded surfaces.

**Availability** Section plus, released version 2-13.

### Argument

`ICODE` `INTEGER` expression Type of shading

| `ICODE` | *Shading technique* |
|---|---|
| 0 | Disable raster emulation (use the *painter's algorithm*) |
| 1 | Enable raster emulation |

### Description

By default, shaded surface pictures are plotted by building up a raster image internally which is output using the current fast area-fill characters. `RASTER` specifies an alternative shading technique for shaded surfaces.

Axes, text and keys are not drawn as part of this raster picture but surface lines (*eg.* outlines, crosshatching) are included in the raster image; for such lines, smaller fast area-fill characters may be desirable (see `FFSIZE`).

There is a limit to the number of different levels (or shading patterns) which can be represented on a raster picture; in the current implementation this limit is 126 levels.

The *painter's algorithm* is an alternative to raster emulation on devices which can overstrike shaded areas (or rub out software shading lines). This technique exploits the representation of a three-dimensional surface on a two-dimensional picture whereby some of the data values are hidden 'behind' the shape which is actually drawn. When using the painter's algorithm, data at the 'back' of the picture is plotted first and then overdrawn where appropriate.

### Default

`CALL RASTER(1)` restores the default.

### See also

`FFSET`, `FFSIZE`, `ISSBAS`, `ISSBCK`, `ISSHAD`, `ISSKRT`, `ISSPAT` and `QFFSIZ`.

## SUBROUTINE REDRAW (ICHAN, IDEV)

### Name

`REDRAW` – to redraw the metafile from the specified I/O channel to the specified device.

**Availability** Section plus, released version 2-8.

### Arguments

| | | |
|---|---|---|
| `ICHAN` | `INTEGER` expression | Channel number |
| `IDEV` | `INTEGER` expression | Device number |

### Description

A SIMPLEPLOT metafile can be created using either `MTFILE` or the META device driver. Once created, the metafile opened on FORTRAN I/O channel `ICHAN` can be redrawn in its entirety by `REDRAW` to the device specified by `IDEV`. `REDRAW` continues reading from channel number `ICHAN` until the end of file is encountered. If the metafile contains details of more than one SIMPLEPLOT page, a succession of separate plots is produced.

If `REDRAW` is called with `ICHAN`=0 while a SIMPLEPLOT page is still being assembled, the current scratch metafile is redrawn.

SIMPLEPLOT can only address one device at a time; if a different device is specified for `REDRAW` from the one in use, SIMPLEPLOT closes the current device, and then opens the device referenced by `IDEV`. Any subsequent plotting from the program automatically reverts to the original device, and diagnostic messages are issued.

`IDEV` can have any value which is available to `DEVNO` including `IDEV`=0 to invoke interactive selection (S.E.P. device driver system only).

### Diagnostics

`(DEVICE CLOSED)`[1]
`(DEVICE OPENED: `*`device`*`)`[1]

### See also

`MTCOMM`, `MTCLOS`, `MTDRAW`, `MTFILE`, `MTMISS`, `MTONLY` and `MTOPEN`.

## SUBROUTINE RESCLR

### Name

`RESCLR` – to cancel all areas reserved for captions and keys.

**Availability** Section 4, released before version 2-5.

### Arguments

None.

### Description

Areas reserved for sequences of captions and keys continue to be allocated with each new picture until the reservation is cancelled by another call of `DEFCAP` or `DEFKEY`. `RESCLR` cancels all reserved areas (for keys and captions) and set the number of reserved areas back to zero.

The limit on the maximum number of reserved areas for keys and captions is 9.

### See also

`DEFCAP` and `DEFKEY`.

## SUBROUTINE RESNXT (VCHAR, HCHAR)

### Name

RESNXT – to specify the reserved area which is to be used next.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of reserved area |
| HCHAR | CHARACTER*1 | Horizontal position of reserved area |

### Description

When more than one area has been reserved for keys or captions, it becomes necessary to indicate which area a key entry or caption is to occupy. If RESNXT is called with arguments VCHAR and HCHAR matching those of one of the reserved areas, subsequent calls of subroutines which draw a key entry or caption are aimed at the next available space in that area. RESNXT has effect until a different area is selected by another call of RESNXT or until a new area is defined by DEFCAP or DEFKEY.

The limit on the maximum number of reserved areas for keys and captions is 9.

RESNXT only interacts with key/caption areas which have been defined using DEFKEY/DEFCAP with ITYPE=3; if RESNXT is called specifying a position for which there is no corresponding reserved area, a diagnostic is issued.

The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

(RESNXT ignored, area not reserved)[1]

### See also

RESCLR, DEFCAP and DEFKEY.

## REAL FUNCTION RGCALC (X, Y, Z2ARR, NX, NY)

### Name

RGCALC – to interpolate $z = f(x, y)$ from 3-D data on a regular grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |

### Description

The value of $z$ at $(x, y)$ is evaluated by two-way linear interpolation within the rectangular cell of the grid in which $(x, y)$ is located.

If X or Y lie outside the limits of the data, RGCALC returns the current no-data value.

### Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]

### See also

NODATA, QNODAT, SFEQX and SFEQY.

## SUBROUTINE RGCNTS (Z2ARR, NX, NY)

### Name

RGCNTS – to draw a contour map from 3-D data on a regular grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |

### Description

RGCNTS draws a contour map on the current picture which represents 3-D data on a regular grid. Contours are drawn for equally-spaced $z$ values within the range of data held in Z2ARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by RGCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set
[2] highest level when more than one level drawn
[3] lowest level of set
[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

### Diagnostics

(Contour map not all in range)$^2$
(INVALID ARGUMENT: ARRAY SIZE)$^1$

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, RIDDLE, SFEQX, SFEQY, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SQBRKN, SQPEN, SQZLAB and SQZVAL.

### SUBROUTINE RGCONT (ZLEV, Z2ARR, NX, NY)

**Name**

`RGCONT` – to draw a contour curve from 3-D data on a regular grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `ZLEV` | `REAL` expression | Contour level |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |

**Description**

`RGCONT` draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=`ZLEV` from 3-D data (on a regular grid) held in `Z2ARR`.

**Diagnostics**

`(Contour curve not all in range)`$^2$
`(INVALID ARGUMENT: ARRAY SIZE)`$^1$

**See also**

`CTBRKN`, `CTCURV`, `CTHOLD`, `CTLABS`, `CTNUMB`, `QCURVE`, `RIDDLE`, `SFEQX` and `SFEQY`.

## SUBROUTINE RGCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY)

### Name

`RGCUT` – to draw a 2-D curve of a surface section from 3-D data on a regular grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `X1`, `Y1`, `X2`, `Y2` | `REAL` expressions | Coordinates of two points |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |

### Description

If the surface represented by the data in `Z2ARR` were cut along the straight line from $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the data along that line.

`RGCUT` draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable scales must be set up before attempting to draw a section.

### Diagnostics

`(INVALID ARGUMENT: ARRAY SIZE)`[1]
`(Surface section not all in range)`[2]

### See also

`CTBRKN`, `CTCURV`, `CTHOLD`, `QCURVE`, `SFEQX` and `SFEQY`.

### SUBROUTINE RGSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY)

**Name**

RGSHAD – to shade the area between two contour levels from 3-D data on a regular grid.

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV1,<br>ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |

**Description**

RGSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from 3-D data (on a regular grid) held in Z2ARR; the area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \le z < z_{max}$, where $z_{min} = MIN(\texttt{ZLEV1}, \texttt{ZLEV2})$ and $z_{max} = MAX(\texttt{ZLEV1}, \texttt{ZLEV2})$.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)$^1$
(Shaded contour not all in range)$^2$

**See also**

CTLABS, CTNUMB, MPTYPE, RGSHDS, SFEQX, SFEQY, SFMESH and SHEDGE.

## SUBROUTINE RGSHDS (Z2ARR, NX, NY)

### Name

`RGSHDS` – to draw a shaded contour map from 3-D data on a regular grid.

**Availability** Section plus, released version 2-5.

### Arguments

| | | |
|---|---|---|
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |

### Description

`RGSHDS` draws a shaded contour map on the current picture from 3-D data on a regular grid. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in `Z2ARR`; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see `SHEDGE`).

### Diagnostics

`(INVALID ARGUMENT: ARRAY SIZE)`$^{1}$
`(Shaded contours not all in range)`$^{2}$

### See also

`CTLABS`, `CTNUMB`, `MPTYPE`, `RGSHAD`, `RGCNTS`, `SFEQX`, `SFEQY`, `SHEDGE`, `SHPATT` and `SQ*`.

## SUBROUTINE RGSURF (Z2ARR, NX, NY)

### Name

`RGSURF` – to start a new picture and draw a surface from 3-D data on a regular grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |

### Description

`RGSURF` starts a new picture, and draws a surface of the data in `Z2ARR`.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with `COORDS` before calling `RGSURF`.

`RGSURF` uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.

[2] $z < 0$, on underside of surface.

[3] $z \geq 0$, top side of surface.

[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of `ISSHAD`.

### Diagnostics

`(INVALID ARGUMENT: ARRAY SIZE)`$^1$
`(Surface not all within picture)`$^2$
`(SURFACE OMITTED: CONSTANT DATA)`$^1$

### See also

`IS*`, `ISSHAD`, `RIDDLE`, `SFEQX`, `SFEQY` and `SFZSCL`.

### SUBROUTINE RIDDLE (RVAL)

**Name**

`RIDDLE` – to specify the level of elimination of redundant points for polylines.

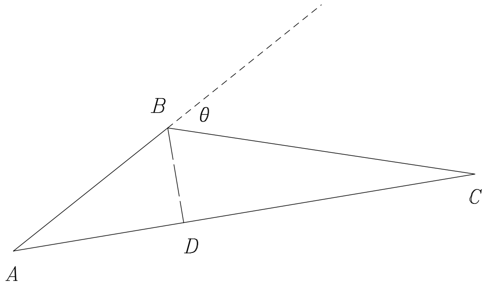
**Availability** Section 1, released version 2-12.

**Argument**

| | | |
|---|---|---|
| `RVAL` | `REAL` expression | Tolerance |

**Description**

By default, no attempt is made to remove co-linear points which make up a polyline. `RIDDLE` specifies that a filtering process is to be applied to all polyline output which throws away intermediate points which are nearly co-linear. `RVAL` specifies the maximum permissible percentage error in thinning-out of polylines, that is, the tolerance to be used to determine co-linearity. `RVAL` must be in range 0.0 to 5.0 inclusive; any other value resets the default.



The filtering criterion relates to angles between vectors. Given a sequence of 3 points $A$, $B$ and $C$ to be joined by 2 vectors $AB$ and $BC$, $B$ is omitted if angle $ABC$ is in the range $180 - \theta$ to $180 + \theta$ where $\tan\theta =$ `RVAL`/25.0.

It can be shown that $BD = AC * (\tan\theta/2)/2$ where $BD$ is the maximum perpendicular distance between the line drawn and any point omitted, and $AC$ is the length of the line. Over the allowed range of `RVAL`, $(\tan\theta)/2$ approximates $\tan(\theta/2)$ so that $BD$ is approximately `RVAL`/100.

**Default**

`CALL RIDDLE(0.0)` restores the default.

### SUBROUTINE SCALES (XSTART, XSTOP, IXTYPE, YSTART, YSTOP, IYTYPE)

**Name**

SCALES – to specify Cartesian scales for all 2-D plotting.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XSTART | REAL expression | $x$ scale value at left edge |
| XSTOP | REAL expression | $x$ scale value at right edge |
| IXTYPE | INTEGER expression | Type of horizontal scale |
| YSTART | REAL expression | $y$ scale value at bottom edge |
| YSTOP | REAL expression | $y$ scale value at top edge |
| IYTYPE | INTEGER expression | Type of vertical scale |

| IXTYPE or IYTYPE | *Type of scale* |
|---|---|
| 0 | Centimetre |
| 1 | Linear |
| 2 | Logarithmic |
| 3 | Normal probability (%) |

**Description**

By default, scales are linear in centimetres starting at zero. SCALES specifies the limits and types of both horizontal and vertical plotting scales. The relative values of the first and last scale values determine whether the scale increases or decreases.

SCALES resets the underlying coordinate system to Cartesian.

Scales are normally set before a picture is started and the same scales used for all relevant 2-D pictures until reset.

**Diagnostics**

```
(Range >=100; linear scale used)¹
(Range through 0; linear scale used)¹
```

**Default**

Calling SCALES with XSTART=XSTOP and/or YSTART=YSTOP, or with IXTYPE=0 and/or IYTYPE=0, restores the default centimetre scaling (the values of all arguments are otherwise ignored).

**See also**

KSCALE, XSCALE, YSCALE and EQSCAL.

## SUBROUTINE SCLEAR

### Name

SCLEAR – to clear the graphics screen independently of starting a new picture.

**Availability** Section 4, released version 2-7.

### Arguments

None.

### Description

On a screen device, Simpleplot always clears the graphics screen immediately after starting a new page. SCLEAR performs the same clearing operation.

On a non-screen device (*eg.* a pen plotter or laser printer), SCLEAR has no effect.

### See also

ENDPAG, OUTBUF, OWNIO and OWNNEW.

### SUBROUTINE SETPNS (IPEN1, IPEN2, IPEN3, IPEN4)

**Name**

SETPNS – to specify the pens associated with the four pen pointers.

**Availability** Section 1, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| IPEN1 | INTEGER expression | Pen associated with pen pointer [1] |
| IPEN2 | INTEGER expression | Pen associated with [2] |
| IPEN3 | INTEGER expression | Pen associated with [3] |
| IPEN4 | INTEGER expression | Pen associated with [4] |

| IPEN*i* | *Pen usage* |
|---|---|
| −1 | plotting is omitted on all devices |
| 0 | plotting is done in background colour; this has no effect on pen plotters |
| 1, 2, 3 . . . | pens/bundles are selected as determined by the current device, palette and bundled attributes |

**Description**

Four pen pointers are identified with different types of plotting operations, and different parts of composite operations, to enable the user to specify a combination of pen usage. The pointers identified with specific plotting operations are indicated by numbers in square brackets within the subroutine specifications. By default, all the pointers are associated with pen number 1. SETPNS specifies alternative pens to be associated with the four pen pointers. IPEN*i* can have values −1, 0, 1, 2, 3, . . . *etc.*

CALL SETPNS(I,I,I,I) is equivalent to CALL PEN(I).

By default, colour is the only line-drawing attribute associated with pens and the distinction between pens depends on the capabilities of the current device. BUNLPR specifies the attributes to be associated with pens when used for drawing lines to differentiate between pens even on monochrome devices.

**Default**

CALL SETPNS(1,1,1,1) restores the default.

**See also**

BUNLPR, PEN, SHPEN, ONEPEN, PENRGB, PENHLS, PENRST and WFPNS.

### SUBROUTINE SFEQX (XSTART, XSTEP)

**Name**

SFEQX – to specify the equally-spaced $x$ values to be associated with gridded 3-D data.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XSTART | REAL expression | $x$ value of first data column |
| XSTEP | REAL expression | $x$ interval between data columns |

**Description**

By default, data on an unspecified regular $x$ grid relates to the current plotting scales:

***Surfaces:*** Current plotting scales do not apply, so XSTART = 1.0 and XSTEP = 1.0 are used.

***Contours:*** XSTART and XSTEP are allocated such that the data cover the current plotting scales; if no scales have been specified, centimetre scaling is used.

SFEQX specifies the $x$ values corresponding to the data grid. SFEQX must be called before one of the RG* (data on a regular grid) or Y* (data on a grid of equally-spaced $x$ values with specified $y$ values) routines.

By default, after a call to SFEQX, an isometric $x$-axis displays the range of values from XSTART to XSTART + XSTEP × (NX − 1), where NX is specified as an argument to RGSURF/YSURF.

**Default**

Calling SFEQX with XSTEP=0.0 restores the default.

**See also**

SFEQY.

### SUBROUTINE SFEQY (YSTART, YSTEP)

**Name**

`SFEQY` – to specify the equally-spaced $y$ values to be associated with gridded 3-D data.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `YSTART` | `REAL` expression | $y$ value of first data row |
| `YSTEP` | `REAL` expression | $y$ interval between data rows |

**Description**

By default, data on an unspecified regular $y$ grid relates to the current plotting scales:

***Surfaces:*** Current plotting scales do not apply, so `YSTART` = 1.0 and `YSTEP` = 1.0 are used.

***Contours:*** `YSTART` and `YSTEP` are allocated such that the data cover the current plotting scales; if no scales have been specified, centimetre scaling is used.

`SFEQY` specifies the $y$ values corresponding to the data grid. `SFEQY` must be called before any of the `RG*` (data on a regular grid) or `X*` (data on a grid of specified $x$ values with equally-spaced $y$ values) routines.

By default, after a call to `SFEQY`, an isometric $y$-axis displays the range of values from `YSTART` to `YSTART` + `YSTEP` × (`NY` − 1), where `NY` is specified as an argument to `RGSURF`/`XSURF`.

**Default**

Calling `SFEQY` with `YSTEP`=0.0 restores the default.

**See also**

`SFEQX`.

## SUBROUTINE SFEQZ (ZSTART, ZSTEP)

### Name

SFEQZ – to specify the equal spacing of contours.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| ZSTART | REAL expression | Offset value for contour levels |
| ZSTEP | REAL expression | Interval between contour levels |

### Description

By default, subroutines which draw a set of equally-spaced contour curves or a set of shaded contour regions, allocate an offset value (ZSTART=0.0) and an interval to produce between 5 and 10 contour levels at suitable values. SFEQZ specifies an alternative offset value and interval for contour levels on both surface pictures and contour maps.

After SFEQZ has been called, sets of equally-spaced contour curves are

drawn at all levels ZSTART + N ∗ ZSTEP (*n*, a +ve or −ve integer) lying within the *z* plotting range.

SFEQZD overrides SFEQZ and *vice versa*.

If ZSTEP is less than $1.0E-4\times$ contour range, the default interval is used instead and, at diagnostic level 2, a diagnostic message is issued (see below).

SFEQZ also affects the interval between the values represented on a map key (see MPK7H and MPK7V).

### Diagnostics

`(Too many contours: default used)`$^2$

### Default

CALL SFEQZ(0.0,0.0) restores the default.

### See also

ISTYPE, MPK7H, MPK7V, SFLIMS, SFEQZD, SFZSCL and SQZVAL.

## SUBROUTINE SFEQZD (NSTEPS, DELTA)

### Name

`SFEQZD` – to specify the equal spacing of contours for discrete data.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `NSTEPS` | `INTEGER` expression | Number of contour intervals |
| `DELTA` | `REAL` expression | Minimum interval between discrete data values |

### Description

By default, subroutines which draw a set of equally-spaced contour curves or a set of shaded contour regions, allocate an offset value and interval to produce between 5 and 10 contour levels at suitable values. `SFEQZD` specifies the number of contours and the interval between discrete data values, for equally-spaced contour levels on both surface pictures and contour maps.

After `SFEQZD` has been called, the $z$ plotting range is divided into `NSTEPS` equal intervals, allowing a margin of `DELTA`×0.5 at each end of the range to accommodate discrete data values correctly. No attempt is made to ensure that the contour levels occur at 'simple' numbers.

`SFEQZ` overrides `SFEQZD` and *vice versa*.

If `DELTA` is less than $1.0E-4\times$ contour range, the default interval is used instead and, at diagnostic level 2, a diagnostic message is issued (see below).

`SFEQZD` also affects the interval between the values represented on a map key (see `MPK7H` and `MPK7V`) and key labels are centred on samples rather than positioned between them.

### Diagnostics

`(Too many contours: default used)`[1]

### Default

`CALL SFEQZD(0,DELTA)` restores the default.

### See also

`ISTYPE`, `MPK7H`, `MPK7V`, `SFLIMS`, `SFEQZ`, `SFZSCL` and `SQZVAL`.

## SUBROUTINE SFLAB

### Name

SFLAB – to annotate the current 3-D picture with the range of values displayed.

**Availability** Section 2, released before version 2-5.

### Arguments

None.

### Description

SFLAB draws [2] a message indicating the $z$ plotting range on a surface picture or contour map. For example,

```
Z:-2 TO 2
```

If equally-spaced contours are included, the interval between them is also given. For example,

```
Z:-2 TO 2 CONTOUR INTERVAL=0.5
```

The message is drawn at the 'B'ottom, 'L'eft of the picture.

### Diagnostics

(Surface label inappropriate)[2]

### See also

QSFLAB.

### SUBROUTINE SFLIMS (Z1, Z2)

**Name**

`SFLIMS` – to specify the $z$ plotting range.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `Z1` | `REAL` expression | $z$ value at start of $z$ range |
| `Z2` | `REAL` expression | $z$ value at end of $z$ range |

**Description**

By default, the full range of the data supplied is plotted on surface pictures and contour maps. `SFLIMS` specifies an alternative range. Any parts of the supplied data which lie outside the range `Z1` to `Z2` are omitted:

***Surface pictures:*** Holes are left in the surface where data values lie above or below the specified limits.

The scale of the picture is not altered to make the cut-off picture fill the space available (see `SFZSCL`).

***Contour maps:*** Blank areas are left in the contour map where data values lie above or below the specified limits. The default contour interval is chosen to suit the reduced data range.

`SFLIMS` also affects the range of values represented on a map key (see `MPK7H` and `MPK7V`): the data range of the key is the intersection of the data range (specified in the call to `MPK7H` or `MPK7V`) with that specified using `SFLIMS` and/or `SFZSCL`.

**Default**

Calling `SFLIMS` with `Z1`=`Z2` restores the default.

**See also**

`MPK7H`, `MPK7V` and `SFZSCL`.

## SUBROUTINE SFMESH (MX, MY)

**Name**

`SFMESH` – to specify a mesh to be used for constructing contours and surfaces.

**Availability** Section plus, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `MX` | `INTEGER` expression | Number of $x$ mesh lines used |
| `MY` | `INTEGER` expression | Number of $y$ mesh lines used |

**Description**

In order to get smoother edges to shaded contour regions, SIMPLEPLOT needs a finer grid; `SFMESH` specifies this grid. By default, the amount of detail in contour curves depends on the data structure:

***Surfaces:*** A grid is used which depends on the $x$-$y$ ranges of the data, possibly modified by a call of `ISMESH`. `SFMESH` specifies a grid to be used, overriding the default for contour curves and surfaces.

***Contour pictures:*** `SFMESH` can be used to change the smoothness of curves by altering the concentration of interpolated points from which curves are constructed; if the mesh used differs from the data configuration, contouring may be slower.

For surfaces, `SFMESH` nominates a grid which is independent of the $x$-$y$ ranges of the data. Please note that the $x$-range is divided into `MX` equal steps and the $y$-range is divided into `MY` equal steps; on surfaces these $x$ steps and $y$ steps are plotted with equal size in the picture.

`ISMESH` has no effect while `SFMESH` is active.

**Default**

Calling `SFMESH` with `MX` and/or `MY` less than 2 restores the default.

**See also**

`ISMESH` and `CTCURV`.

## SUBROUTINE SFZSCL (ZSTART, ZSTOP)

### Name

SFZSCL – to specify the *z* scale of surface pictures and contour maps independently of the data set.

**Availability** Section 2, released version 2-12.

### Arguments

| | | |
|---|---|---|
| ZSTART | REAL expression | Start of *z* scale |
| ZSTOP | REAL expression | End of *z* scale |

### Description

By default, surface pictures are drawn with an increasing *z* scale covering the same range as the data. SFZSCL specifies an alternative *z* scale such that ZSTART lies at the bottom of the picture and ZSTOP at the top. This may be useful when comparable scales are needed for different pictures.

If the user-specified scale can not accommodate the entire surface, part of the picture cannot be drawn; this contributes to the number of incomplete picture tasks and a diagnostic is issued at the end of the picture.

SFZSCL also affects the range of values represented on a map key (see MPK7H and MPK7V): the data range of the key is the intersection of the data range (specified in the call to MPK7H or MPK7V) with that specified using SFZSCL and/or SFLIMS; however, if the data range of the key is null, the scale specified by SFZSCL is used instead.

### Default

Calling SFZSCL with ZSTART=ZSTOP restores the default.

### See also

FNAREA, MPK7H, MPK7V and SFLIMS.

## SUBROUTINE SGEXPL (IVAL, ISEG)

### Name

`SGEXPL` – to specify the amount by which one pie chart segment is to be exploded.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `IVAL` | `INTEGER` expression | Percentage explosion (0–50) |
| `ISEG` | `INTEGER` expression | Segment number (1–*maximum*) |

### Description

By default, the displacement of all segments is zero. `SGEXPL` specifies the amount (as percentage of radius) by which segment number `ISEG` is to be displaced from the centre of the pie, along the radius bisecting its angle. `SGEXPL` must be called before `PIECHT`.

The number of segments of a pie chart is unlimited but the limit on the maximum number of segments which can be omitted or exploded is 32.

The effect of `SGEXPL` can be overridden by `PIEXPL` and *vice versa*.

### Default

`CALL SGEXPL(0,ISEG)` restores the default.

### See also

`PIECHT` and `PIEXPL`.

## SUBROUTINE SGOMIT (TORF, ISEG)

### Name

SGOMIT – to specify whether a pie chart segment is to be omitted.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| TORF | LOGICAL expression | Whether segment is to be omitted |
| ISEG | INTEGER expression | Segment number (1–*maximum*) |

### Description

By default, all data values are represented by a segment in the pie chart and every segment is displayed. SGOMIT specifies whether or not the ISEGth segment of the pie is to be omitted from the picture. All segments are included in the calculation of pie chart segments. SGOMIT must be called before PIECHT.

The number of segments of a pie chart is unlimited but the limit on the maximum number of segments which can be omitted or exploded is 32.

The effect of SGOMIT can be overridden by PIOMIT and *vice versa*.

### Default

CALL SGOMIT(.FALSE.,ISEG) restores the default.

### See also

PIECHT and PIOMIT.

## SUBROUTINE SHANGS (IDEG1, IDEG2, IDEG3, IDEG4)

### Name

`SHANGS` – to specify the angles used in software shading patterns.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `IDEG1,`<br>`IDEG2,`<br>`IDEG3,`<br>`IDEG4` | `INTEGER` expressions | Angles in degrees |

### Description

Software shading patterns consist of sets of equally-spaced parallel lines of one colour. The sequence for a monochrome device is as follows:

- Pattern −1 is empty but is outlined using the pen selected by `SHPEN`, or currently selected for drawing lines (pen pointer 1).
- Pattern 0 fills the area with pen 0 (background). If the device cannot draw in background, pattern 0 is equivalent to pattern −1.
- Pattern 1 fills the area with pen 1 and is not affected by settings for the angle or separation of hatching lines (`SHDESC` or `SHGAP`).
- Patterns 2–5 use the 4 shading angles with a small line separation; patterns 6–9 use the 4 shading angles with a larger line separation, *etc.*

If very large pattern numbers are chosen, very large line separation is used.

By default, four angles are used in shading patterns, 0°, 90°, 45° and 135°. `SHANGS` specifies alternative angles. The number of shading angles is the number of different angles set by `SHANGS` – for example, the number of shading angles is reduced to three if two of the arguments, `IDEG`*n*, are equal. Changing the number of shading angles also changes the relationship between shading patterns and pattern numbers – the use of fewer angles means that the cycle of patterns is repeated sooner.

### Default

`CALL SHANGS(0,90,45,135)` restores the default.

### See also

`SHCOLS`, `SHGAP` and `SHDESC`.

### SUBROUTINE SHCOLS (NCOLS)

**Name**

SHCOLS – to specify a restricted number of colours used for shading patterns.

**Availability** Section 4, released before version 2-5.

**Argument**

| | | |
|---|---|---|
| NCOLS | INTEGER expression | Number of colours |

**Description**

By default, the number of colours used for shading patterns is set to the maximum number of colours available on the device. SHCOLS specifies an alternative number of colours to be used for shading. SHCOLS affects hardware and software shading patterns. SHCOLS must be called before a shading operation.

Changing the number of shading colours also changes the relationship between shading patterns and pattern numbers – the use of fewer colours means that the cycle of patterns is repeated sooner.

**Default**

CALL SHCOLS(0) restores the default. Note that this also resets SHPEN.

**See also**

SHANGS, SHDESC, SHGAP and SHPEN.

### SUBROUTINE SHDEA2 (X1, DX, DARR1, DARR2, NARR, ISHADE)

**Name**

`SHDEA2` – to shade between 2 curves defined by values held in 2 arrays.

**Availability** Section 4, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| `X1` | `REAL` expression | Value of independent variable for plotting `DARR1(1)` and `DARR2(1)` |
| `DX` | `REAL` expression | Independent variable interval between function values |
| `DARR1` | `REAL` array | Function values for first curve |
| `DARR2` | `REAL` array | Function values for second curve |
| `NARR` | `INTEGER` expression | Number of values in each array |
| `ISHADE` | `INTEGER` expression | Shading pattern number |

**Description**

Using shading pattern number `ISHADE`, `SHDEA2` shades the area between two curves:

```
(X1,DARR1(1)), (X1+DX,DARR1(2)),...
                            (X1+(NARR-1)*DX,DARR1(NARR))
(X1,DARR2(1)), (X1+DX,DARR2(2)),...
                            (X1+(NARR-1)*DX,DARR2(NARR))
```

The type of curve depends on `CVTYPE` but the area is enclosed by joining the first points of each curve together by a straight line, and the last points of each curve. Whether the boundary of the area is drawn depends on `SHEDGE`.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

**Diagnostics**

`(Data curve exceeds scales)`$^2$

**See also**

`CLIPLV`, `CVTYPE`, `RIDDLE`, `SHDEC2`, `SHDEAR` and `SHEDGE`.

### SUBROUTINE SHDEAR (X1, DX, DARR, NARR, ISHADE)

**Name**

`SHDEAR` – to draw a curve from an array of values, and shade under it.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| `X1` | `REAL` expression | Value of independent variable for plotting `DARR(1)` |
| `DX` | `REAL` expression | Independent variable interval between function values |
| `DARR` | `REAL` array | Function values for plotting |
| `NARR` | `INTEGER` expression | Number of points |
| `ISHADE` | `INTEGER` expression | Shading pattern number |

**Description**

Using shading pattern number `ISHADE`, `SHDEAR` shades the area enclosed by the curve which connects the following points:

(`X1`,`DARR(1)`), (`X1+DX`,`DARR(2)`),...
(`X1+(NARR-1)*DX`,`DARR(NARR)`)

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ and $x_i = x_1 + (i-1)dx$.

If the curve is not closed, the area shaded depends on `SHTYPE`; the type of curve depends on `CVTYPE` and whether the curve itself is drawn depends on `SHEDGE`.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

**Diagnostics**

`(Data curve exceeds scales)`$^2$

**See also**

`BRKNAR`, `CLIPLV`, `CVTYPE`, `RIDDLE`, `SHDEA2`, `SHDECV`, `SHEDGE` and `SHTYPE`.

## SUBROUTINE SHDEBX (X1, Y1, X2, Y2, ISHADE)

### Name

`SHDEBX` – to draw a shaded box.

**Availability** Section 4, released version 2-8.

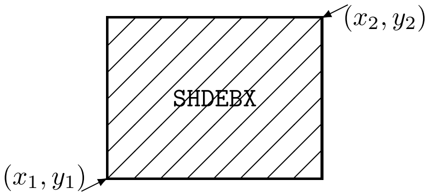
### Arguments

| | | |
|---|---|---|
| `X1`, `Y1`, `X2`, `Y2` | `REAL` expressions | Coordinates of opposite corners of the box, specified in units of the plotting scales |
| `ISHADE` | `INTEGER` expression | Shading pattern number |

### Description

Using shading pattern number `ISHADE`, `SHDEBX` shades the box extending from $x_1$ to $x_2$ by $y_1$ to $y_2$. If $x_1$=$x_2$ or $y_1$=$y_2$, then nothing is drawn. Whether a boundary is drawn depends on `SHEDGE`.

The points $(x_1, y_1)$ and $(x_2, y_2)$ are specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, `KXYXY`.



By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(**`$x_1, y_1, x_2, y_2$`**)`$^2$
`(Null box, nothing drawn)`$^2$

### See also

`BRKNBX`, `CLIPLV` and `SHEDGE`.

### SUBROUTINE SHDEC2 (XARR1, YARR1, NARR1, XARR2, YARR2, NARR2, ISHADE)

**Name**

SHDEC2 – to shade the area between 2 curves specified by sets of coordinates.

**Availability** Section 4, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR1,<br>YARR1 | REAL arrays | Coordinates of first curve |
| NARR1 | INTEGER expression | Number of points in first curve |
| XARR2,<br>YARR2 | REAL arrays | Coordinates of second curve |
| NARR2 | INTEGER expression | Number of points in second curve |
| ISHADE | INTEGER expression | Shading pattern number |

**Description**

Using shading pattern number ISHADE, SHDEC2 shades the area between two curves:

(XARR1(1), YARR1(1)), . . . , (XARR1(NARR1), YARR1(NARR1))

(XARR2(1), YARR2(1)), . . . , (XARR2(NARR2), YARR2(NARR2))

The type of curve depends on CVTYPE but the area is enclosed by joining the first points of each curve together by a straight line, and the last points of each curve. Whether the boundary of the area is drawn depends on SHEDGE.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

**Diagnostics**

(Data curve exceeds scales)$^2$

**See also**

CLIPLV, CVTYPE, RIDDLE, SHDEA2, SHDECV and SHEDGE.

## SUBROUTINE SHDECL (XCENT, YCENT, RADIUS, ISHADE)

### Name

`SHDECL` – to draw a circular function with specified centre and radius, and to shade inside it.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `XCENT`,<br>`YCENT` | `REAL` expressions | Coordinates of the centre of the circle in units of the plotting scale |
| `RADIUS` | `REAL` expression | Circle radius in plotting scale units |
| `ISHADE` | `INTEGER` expression | Shading pattern number |

### Description

Using shading pattern number `ISHADE`, `SHDECL` shades the circle (or the part of the circle which lies within the range of the plotting scales) centred at point $(x_{cent}, y_{cent})$, with radius `RADIUS`. The resulting shape is only geometrically circular when both scales are equal and linear. Whether a boundary is drawn depends on `SHEDGE`.

The centre point and the radius are both specified in units of the current plotting scales (*ie.* user coordinates); device or page coordinates can be converted to user coordinates using the conversion subroutine, `KXYXY`.

By default, all drawing is restricted by the limits of the picture scales; if it exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Circle exceeds scales)`$^2$

### See also

`BRKNCL`, `CLIPLV`, `EQSCAL`, `KXYXY` and `SHEDGE`.

## SUBROUTINE SHDECV (XARR, YARR, NARR, ISHADE)

### Name

SHDECV – to draw a curve from a set of coordinates, and shade under it.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | Coordinates of points in units of the plotting scales |
| NARR | INTEGER expression | Number of points |
| ISHADE | INTEGER expression | Shading pattern number |

### Description

Using shading pattern number ISHADE, SHDECV shades the area enclosed by the curve corresponding to the following points:

(XARR(1), YARR(1)), (XARR(2), YARR(2)), . . .
(XARR(NARR), YARR(NARR))

A closed curve can be obtained by using data in which XARR(NARR)=XARR(1) and YARR(NARR)=YARR(1). If the curve is not closed, the area shaded depends on SHTYPE. The type of curve drawn depends on CVTYPE and whether the curve itself is drawn depends on SHEDGE.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Data curve exceeds scales)$^2$

### See also

BRKNCV, CLIPLV, CVTYPE, MARKCV, RIDDLE, SHEDGE and SHTYPE.

## SUBROUTINE SHDEFN (FUNX, ISHADE)

### Name

`SHDEFN` – to draw a curve of a user-defined function, $y = f(x)$, and shade under it.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `FUNX` | function name | `REAL` function with one `REAL` argument, also declared in an `EXTERNAL` statement |
| `ISHADE` | `INTEGER` expression | Shading pattern number |

### Description

Using shading pattern number `ISHADE`, `SHDEFN` shades the area enclosed by the curve of a used-defined function, `FUNX`, and the axes or edge of the picture. Whether a boundary is drawn depends on `SHEDGE`.

With Cartesian coordinates, the specified function, `FUNX`, must calculate $y$ given $x$, and with polar coordinates `FUNX` must calculate $r$ given $\theta$. The intervals of the independent variable over which `SHDEFN` evaluates the function, vary with the curvature of the curve.

Care must be taken in the programming of the `FUNX` to ensure that it is continuous and cannot fail for any arguments within the plotting range. Discontinuities may cause SIMPLEPLOT to loop indefinitely.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using `CLIPLV`.

### Diagnostics

`(Function curve exceeds scales)`$^2$

### See also

`BRKNFN`, `CLIPLV`, `FNRNGE`, `POLY`, `RIDDLE` and `SHEDGE`.

## SUBROUTINE SHDEK7 (ISHADE, CAP)
## SUBROUTINE SHDEKY (ISHADE, CAP, NCAP)

### Name

SHDEK7 – to draw an annotated sample of a shading pattern in a key.

**Availability** Section 4, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| ISHADE | INTEGER expression | Shading pattern number |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

SHDEK7/SHDEKY inserts a key to a shading pattern in the next empty space in the current key or caption area (defined by DEFKEY/DEFCAP or selected by RESNXT).

A rectangular sample of shading pattern ISHADE is drawn on the left of the entry; the caption, CAP, is drawn [2] alongside the sample. Whether a boundary is drawn around the sample shading pattern depends on SHEDGE.

REAL and INTEGER values can be converted to text strings using the conversion subroutines, KREAL and KNUMB respectively.

### Diagnostics

(Caption truncated)[2]<br>
(Key/caption area full)[2]<br>
(No active key/caption area)[2]

### See also

ADDJST, BLNKKY, DEFKEY, FILLK7, MPK7H, MPK7V, SHEDGE, SHKEYS, SHPATT, SHSET and SQSHAD.

## SUBROUTINE SHDEPR (SUBR, T1, T2, ISHADE)

### Name

SHDEPR – to draw the user-defined parametric function, and shade inside it.

**Availability** Section 4, released version 2-9.

### Arguments

| | | |
|---|---|---|
| SUBR | subroutine name | Subroutine with three REAL arguments, also declared in an EXTERNAL statement |
| T1, T2 | REAL expressions | Values of the function parameter between which curve is to be drawn |
| ISHADE | INTEGER expression | Shading pattern number |

### Description

Using shading pattern number ISHADE, SHDEPR shades the area enclosed by the curve of a used-defined parametric function. If the curve is not closed, a closed area is formed by joining the first and last points with a straight line; whether a boundary is drawn depends on SHEDGE.

The parametric function is specified in subroutine SUBR; SUBR must have three REAL arguments: T, the value of the function argument, and VARX and VARY, to receive coordinates corresponding o T. SUBR is called by SHDEPR with values of T ranging from T1 to T2.

By default, all drawing is restricted by the limits of the picture scales; if coordinates exceed these limits, plotting is clipped to the picture limits and every such occurrence contributes to the number of incomplete picture tasks. An alternative clipping strategy can be specified using CLIPLV.

### Diagnostics

(Parametric function exceeds scales)$^2$

### See also

BRKNPR, CLIPLV, RIDDLE and SHEDGE.

### SUBROUTINE SHDESC (IDEG1, IDEG2, IDEG3, IDEG4, NCOLS, GAP)

**Name**

SHDESC – to specify all characteristics of shading patterns.

**Availability** Section 4, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| IDEG1,<br>IDEG2,<br>IDEG3,<br>IDEG4 | INTEGER expressions | Angles in degrees |
| NCOLS | INTEGER expression | Number of colours |
| GAP | REAL expression | Minimum separation between shading lines (in cms) |

**Description**

Each software shading pattern consists of a set of equally-spaced parallel lines of one colour. On a monochrome device the sequence is as follows: pattern −1 is empty but is outlined using the pen selected by SHPEN, or pen pointer 1; patterns 0 and 1 are as near to solid as the device permits – 0 uses background if the device permits, otherwise it is equivalent to pattern −1; patterns 2–5 use the 4 shading angles with a small line separation; patterns 6–9 use a larger line separation, *etc.* If very large pattern numbers are chosen, very large line separation is used.

By default, IDEG1 is equivalent to 0, IDEG2 to 90, IDEG3 to 45, IDEG4 to 135, NCOLS to the maximum number of colours available on the device, and GAP to the thickness of the lines drawn on the device.

SHDESC specifies alternatives for all these shading characteristics – the number of shading angles used is the number of different angles specified; the number of shading colours can be any positive value and the minimum separation between shading lines can be any value greater than or equal to the standard thickness of lines on the device. SHDESC must be called before a shading operation.

NCOLS also affects hardware shading patterns but the angles and line separation only affect software patterns. If fewer colours or angles are selected, the cycle of patterns is repeated sooner.

**Default**

CALL SHDESC(0,90,45,135,0,0.0) restores the default

**See also**

SHANGS, SHCOLS and SHGAP.

## SUBROUTINE SHEDGE (ITYPE)

### Name

`SHEDGE` – to specify whether to draw a boundary around a shaded area.

**Availability** Section plus, released version 2-11.

### Argument

| | | |
|---|---|---|
| `ITYPE` | `INTEGER` expression | Boundary option |

| `ITYPE` | *Shading edge* |
|---|---|
| −1 | Default mode, determined by SIMPLEPLOT |
| 0 | No edge drawn |
| 1 | Edge drawn |

### Description

By default, SIMPLEPLOT draws some shaded areas with a boundary and some without. For example, pie charts, bar charts and two-dimensional shaded curves are drawn with a boundary, but shaded areas on a contour map are drawn without a boundary. `SHEDGE` overrides this default behaviour.

The boundary of a shaded area is drawn with the same pen as the shading pattern itself except for pattern −1; pattern −1 is outlined using the pen selected by `SHPEN` or (if none selected) the pen currently selected for drawing lines, [1]. The drawing of this boundary line is not affected by bundled line-drawing attributes.

### Default

`CALL SHEDGE(-1)` restores the default.

### See also

`MPK7H`, `MPK7V`, `SHSTRT`, `SHSTOP`, `SHDE*`, `*SHAD` and `*SHDS`.

## SUBROUTINE SHGAP (GAP)

### Name

`SHGAP` – to specify the spacing of software shading lines.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `GAP` | `REAL` expression | Minimum separation between lines (in cms) |

### Description

Software shading patterns consist of sets of equally-spaced parallel lines of one colour. The sequence for a monochrome device is as follows:

- Pattern −1 is empty but is outlined using the pen selected by `SHPEN`, or the pen currently selected for drawing lines (pen pointer 1).
- Pattern 0 is as near to solid as the device permits, using pen 0. If the device cannot draw in background, pattern 0 is equivalent to pattern −1.
- Pattern 1 fills the area with pen 1 and is not affected by settings for the angle or separation of hatching lines.
- Patterns 2–5 use the 4 shading angles with a small line separation.
- Patterns 6–9 use the 4 shading angles with a larger line separation.
- *etc.*

If very large pattern numbers are chosen, very large line separation is used.

By default, the minimum separation between shading lines is set to the thickness of lines drawn on the device. `SHGAP` specifies the minimum separation between lines as any value greater than or equal to the standard thickness of lines on the device. Changing the separation between shading lines also changes the relationship between shading patterns and pattern numbers.

### Default

`CALL SHGAP(0.0)` restores the default.

### See also

`SHANGS`, `SHCOLS` and `SHDESC`.

### SUBROUTINE SHKEYS (VCHAR, HCHAR, LABARR, NARR, CAP)

**Name**

SHKEYS – to draw a complete key to a sequence of shading patterns.

**Availability** Section 4, released version 2-11.

**Arguments**

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of area |
| HCHAR | CHARACTER*1 | Horizontal position of area |
| LABARR | STRING array | Set of captions |
| NARR | INTEGER expression | Number of labels in LABARR |
| CAP | STRING expression | Header caption for key |

**Description**

SHKEYS constructs a complete key to the first NARR shading patterns of the currently defined sequence (see SQSHAD and SHPATT).

The key is positioned on the SIMPLEPLOT page in the same way as other keys; VCHAR and HCHAR are single characters representing the initial letters of vertical and horizontal positions.

CAP provides a heading which, by default, is centred at the top of the key (see ADDJST). A box is drawn [1] around the key unless cancelled by BOXKY.

The key is associated only with the current picture but area is masked to prevent overdrawing. To avoid the key being overdrawn by other plotting (*eg.* if the key is positioned 'inside' the picture), it is best to call SHKEYS after NEWPIC but before the main picture is drawn (this may be slower because SIMPLEPLOT must check for overlapping).

The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

**Diagnostics**

(Key/caption height reduced)$^2$
(Key/caption width reduced)$^2$
(Maximum no. of keys/captions reached)$^2$
(No room for key/caption area)$^2$

**See also**

ADDJST, BOXKY, DEFKEY, DEFKYW, DEFPOS, MPK7H, MPK7V, SHDEK7, SHPATT, SHSET and SQSHAD.

## SUBROUTINE SHPATT (ISHADE, IPOS)

### Name

`SHPATT` – to specify one of a sequence of shading patterns.

**Availability** Section 4, released version 2-5.

### Arguments

| | | |
|---|---|---|
| `ISHADE` | `INTEGER` expression | Pattern number |
| `IPOS` | `INTEGER` expression | Indicating which of the sequence is to be set (1–32) |

| `ISHADE` | *Shading pattern* |
|---|---|
| −1 | an empty area |
| 0 | solid fill with background colour |
| 1, 2, 3. . . | hardware/software patterns |

### Description

By default, subroutines which shade a sequence of areas to distinguish them from each other use the shading patterns in numerical order 1, 2, 3 *etc.* `SHPATT` specifies an alternative pattern, `ISHADE`, to be associated with the `IPOS`th shaded area. `SHPATT` must be called before the subroutine which performs the shading.

`ISHADE` can have values −1, 0, 1, 2, 3, . . . *etc.*

If the boundary of a shaded area is to be drawn (see `SHEDGE`), it is drawn with the same pen as the shading pattern itself except for pattern −1; pattern −1 is outlined using the pen selected by `SHPEN` or (if none selected) the pen currently selected for drawing lines, [1].

The number of shading patterns is unlimited but the limit on the *maximum* number of patterns which can be specified as a sequence is 32.

### Default

`CALL SHPATT(`*n*`,`*n*`)` restores the default for each position, *n*.

### See also

`SQSHAD`, `SHPEN` and `SHEDGE`.

## SUBROUTINE SHPEN (IPEN)

### Name

SHPEN – to specify single colour shading patterns with the specified pen.

**Availability** Section 4, released version 2-7.

### Argument

IPEN  INTEGER expression  Pen number

| IPEN | *Pen usage* |
|---|---|
| −1 | plotting is omitted on all devices |
| 0 | plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted |
| 1, 2, 3 . . . | pens/colours are selected as determined by the device driver |

### Description

By default, the number of colours used for shading patterns is set to the maximum number of colours available on the device. SHPEN requests monochromatic shading with pen number, IPEN. SHPEN must be called before a shading operation.

IPEN can have values −1, 0, 1, 2, 3, . . . *etc.*

Please note that changing the number of shading colours changes the relationship between shading patterns and pattern numbers.

Software shading is not affected by bundled line-drawing attributes.

### Default

CALL SHCOLS(0) restores the default.

### See also

SHCOLS, SHSET, PEN and SETPNS.

## SUBROUTINE SHSET (ISET)

### Name

SHSET – to specify an alternative source of area-fill patterns for polygon shading.

**Availability** Section 4, released version 2-8.

### Argument

ISET INTEGER expression Code number for area fill source

| ISET | *Polygon area-fill source* |
|---|---|
| $\leq -1$ | Alternative hardware/firmware sets (if available) |
| 0 | Hardware/firmware area fill (if available) |
| 1 | Device independent software shading |

### Description

For efficient operation and clarity, SIMPLEPLOT is set up to shade areas using area fill from device hardware or firmware wherever suitable. SHSET selects an alternative source of area fill patterns.

On devices which can use hardware area fill, the hardware patterns may be used to take the place of software shading. When no more hardware patterns are available, SIMPLEPLOT returns to the software pattern that would have occurred in this position.

### Default

CALL SHSET(0) restores the default.

### See also

CHSET, FFSET, LINSET and MKSET.

## SUBROUTINE SHSTOP

### Name

`SHSTOP` – to terminate the description of a user-defined shaded shape.

**Availability** Section plus, released version 2-11.

### Arguments

None.

### Description

`SHSTOP` terminates a user-defined shape for shading started by `SHSTRT`. The graphics output of the shape is not started until the drawing has been terminated.

If the shape does not consist entirely of closed areas, nothing is drawn and a diagnostic is issued.

### Diagnostics

`(Unclosed shaded area)`[1]

### See also

`SHSTRT`.

## SUBROUTINE SHSTRT (ISHADE)

### Name

`SHSTRT` – to start a user-defined shaded shape.

**Availability** Section plus, released version 2-11.

### Argument

| | | |
|---|---|---|
| `ISHADE` | `INTEGER` expression | Shading pattern number |

### Description

A user-defined shape which can be drawn using any of the `DRAW*` or `BRKN*` subroutines, `JOINPT` and `BREAK` is shaded if the drawing of the shape is preceded by a call of `SHSTRT` and followed by a call of `SHSTOP`. `ISHADE` selects the shading pattern.

The drawing of the shape for shading should normally be terminated by `SHSTOP`, but a call of any SIMPLEPLOT subroutine other than those allowed for drawing the shape, causes the shape to be abandoned and a diagnostic is issued.

The shape can be made up from one or more closed areas, and the boundaries traced in any order; if there are any gaps in the boundaries of the shape, the unclosed curves are omitted and a diagnostic is issued.

Whether the boundary is drawn depends on `SHEDGE`.

### Diagnostics

`(Unclosed shaded area)`$^1$
`(USER SHADING ABANDONED)`$^1$

### See also

`SHSTOP` and `SHEDGE`.

## SUBROUTINE SHTYPE (ITYPE)

### Name

`SHTYPE` – to specify the shading boundary used for unclosed curves.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `ITYPE` | `INTEGER` expression | Mode of shading |

| `ITYPE` | *Shading area for unclosed curves* |
|---|---|
| 0 | Area contained by joining last point of curve to first with a straight line |
| 1 | Area between curve and horizontal axis or base of graph |
| 2 | Area between curve and vertical axis or left hand edge of graph |

### Description

By default, when `SHDECV` or `SHDEAR` defines a curve which is not closed, shading applies to the area between the curve and the $x$-axis or bottom horizontal boundary when no axis exists. Alternative areas can be chosen by calling `SHTYPE`.

`SHTYPE` does not apply to polar charts nor to functions drawn by `SHDEFN` or `SHDEPR`. The curves drawn by `SHDEA2` and `SHDEC2` are always closed and therefore unaffected by `SHTYPE`.

### Default

`CALL SHTYPE(1)` restores the default.

### See also

`SHDECV` and `SHDEAR`.

## SUBROUTINE SHUNIF (TORF)

### Name

`SHUNIF` – to specify whether software shading lines are to be drawn in only one direction.

**Availability** Section 4, released before version 2-5.

### Argument

| | | |
|---|---|---|
| `TORF` | `LOGICAL` expression | Whether uniform shading is required |

### Description

By default, software shading lines are drawn in alternating directions. On mechanical devices, it is faster to draw shading lines in alternate directions than to draw them all one way; but on pen plotters, the shaded area may look more uniform if the lines are all drawn the same way. After a call of `SHUNIF(.TRUE.)`, all software generated shading lines are drawn in one direction.

### Default

`CALL SHUNIF(.FALSE.)` restores the default.

### See also

`SHDESC`.

## SUBROUTINE SQBRKN (IARR, NARR)

### Name

SQBRKN – to specify a sequence of broken line patterns.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| IARR | INTEGER array | Pattern numbers for each contour line |
| NARR | INTEGER expression | Number of elements in IARR (1–32) |

### Description

By default, subroutines which draw a sequence of contour lines use a single broken line pattern (which can be specified by CTBRKN) for every line. SQBRKN specifies a sequence of broken line patterns.

The number of software line patterns is unlimited but patterns beyond the usual range, $-6 \ldots 6$, have longer patterns and may not be easily distinguishable from one another. The number of hardware broken line patterns is also unlimited in theory but, in practice, there are fewer than are available in software. Whether hardware or software patterns are used, there is a limit on the *maximum* number of patterns which can be specified as a sequence – only 32 patterns can be defined as a sequence.

If line style has been requested as a bundled attribute, the broken line patterns specified using SQBRKN override the line style of the bundles (specified by PEN, SETPNS or SQPEN) unless

IARR($i$) $= -999$ which indicates that all the current bundled line-drawing attributes are to be used.

When the sequence is exhausted, the line pattern specified by CTBRKN is used.

### Default

CALL SQBRKN(IARR,0) restores the default.

### See also

BUNLPR, CTBRKN, LINSET and SQ*.

## SUBROUTINE SQPEN (IARR, NARR)

### Name

SQPEN – to specify a sequence of pens/bundles.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| IARR | INTEGER array | Pen numbers for each contour |
| NARR | INTEGER expression | Number of elements in IARR (1–32) |

| IARR(*i*) | *Pen usage* |
|---|---|
| −1 | plotting is omitted on all devices |
| 0 | plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted |
| 1, 2, 3 ... | pens/bundles are selected as determined by the current device, palette (see PENHLS and PENRGB) and the current bundled attributes (see BUNLPR) |

### Description

By default, subroutines which draw a sequence of contour lines, use a single pen (specified by PEN) or use the pens assigned by pen pointers:

[1] positive levels except lowest and highest of set

[2] highest level when more than one level drawn

[3] lowest level of set

[4] negative and zero levels except lowest and highest of set

SQPEN specifies an alternative sequence of pens. SQPEN also affects surface pictures which are drawn with crosshatching or contour lines (see ISTYPE, *CNTS). A different pen is used for each contour range which the crosshatched lines cross and at the edge of a surface on which contours are drawn. IARR(*i*) can have values −1, 0, 1, 2, 3, ... *etc.*

The number of pens is unlimited but the maximum number of pens which can be specified as a sequence is 32. When the sequence is exhausted, the pen pointed to by pen pointer [1] is used (see SETPNS and PEN).

### Default

CALL SQPEN(IARR,0) restores the default.

### See also

BUNLPR, PEN, PENHLS, PENRGB, SETPNS and SQ*.

## SUBROUTINE SQSHAD (IARR, NARR)

### Name

`SQSHAD` – to specify a sequence of shading patterns.

**Availability** Section 4, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `IARR` | `INTEGER` array | Pattern numbers for each shaded area |
| `NARR` | `INTEGER` expression | Number of elements in `IARR` (1–32) |

| `IARR(`*i*`)` | *Shading pattern* |
|---|---|
| −1 | an empty area |
| 0 | solid fill with background colour |
| 1, 2, 3 . . . | hardware/software patterns |

### Description

By default, subroutines which shade a sequence of areas to distinguish them from each other, use the shading patterns in numerical order 1, 2, 3 *etc.* `SQSHAD` specifies an alternative sequence of pattern numbers.

`SQSHAD` must be called before the subroutine which performs the shading.

`IARR` can have values −1, 0, 1, 2, 3, . . . *etc.*

If the boundary of a shaded area is to be drawn (see `SHEDGE`), it is drawn with the same pen as the shading pattern itself except for pattern −1; pattern −1 is outlined using the pen selected by `SHPEN` or (if none selected) the pen currently selected for drawing lines, [1].

The number of shading patterns is unlimited but the limit on the maximum number of patterns which can be specified as a sequence is 32.

`SQSHAD` also affects the shading pattern samples represented on map keys (see `MPK7H` and `MPK7V`) and shaded keys (see `SHKEYS`).

### Default

`CALL SQSHAD(IARR,0)` restores the default.

### See also

`MPK7H`, `MPK7V`, `SHEDGE`, `SHKEYS`, `SHPEN` and `SQ*`.

## SUBROUTINE SQZLAB (LABARR, NARR)

### Name

SQZLAB – to specify a sequence of contour labels.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| LABARR | STRING array | Contour labels |
| NARR | INTEGER expression | Number of elements in LABARR (1–32) |

### Description

By default, contour maps are not labelled but CTLABS or CTNUMB can be called to specify that automatically generated numeric labels are to be drawn. SQZLAB specifies an alternative set of textual labels to be used. SQZLAB also determines the labels used on a map key (see MPK7H and MPK7V).

The number of contour labels is unlimited but the limit on the maximum number of labels which can be specified as a sequence is 32. Also, the length of individual labels is restricted to a maximum of 20 characters; longer labels are truncated.

When the sequence is exhausted, labelling ceases.

### Default

CALL SQZLAB(LABARR,0) restores the default.

### See also

CTLABS, CTNUMB, MPK7H, MPK7V, SQZVAL, *CNTS and *SHDS.

### SUBROUTINE SQZVAL (DARR, NARR)

**Name**

SQZVAL – to specify a sequence of contour levels.

**Availability** Section plus, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| DARR | REAL array | Contour levels |
| NARR | INTEGER expression | Number of elements in DARR (1–32) |

**Description**

By default, a set of contour curves is drawn with a regular contour interval (see SFEQZ and SFEQZD); SQZVAL specifies an alternative set of values at which to draw contours. The number of contour levels is unlimited but the limit on the maximum number of values which can be specified as a sequence is 32.

SQZVAL also affects the values represented on a map key (see MPK7H and MPK7V).

SQZVAL overrides the effects of both SFEQZ and SFEQZD.

**Diagnostics**

(DATA GRID NOT MONOTONIC)[1]

**Default**

CALL SQZVAL(DARR,0) restores the default.

**See also**

MPK7H, MPK7V, SFEQZ, SFEQZD, SQZLAB, *CNTS and *SHDS.

## SUBROUTINE STRCHS (CHESC, CHOPEN, CHCLOS)

### Name

`STRCHS` – to specify the three special characters to be used in escape sequences.

**Availability** Section plus, released version 2-11.

### Arguments

| | | |
|---|---|---|
| `CHESC` | `CHARACTER*1` | Escape character |
| `CHOPEN` | `CHARACTER*1` | Opening bracket |
| `CHCLOS` | `CHARACTER*1` | Closing bracket |

### Description

By default, SIMPLEPLOT recognizes `'!'`, `'['` and `']'` as the special characters used to identify embedded commands within text strings; `STRCHS` specifies alternative characters.

Alphanumeric characters may not be used as escape or bracketing characters; the same character can be used for both bracket characters, but the escape character must be different from both brackets.

`STRCHS` can be called at any time, regardless of whether `STRLV` has been called to enable recognition of these characters. If `STRCHS` is called with invalid arguments, the use of escape sequences is disabled until a valid call of `STRCHS`.

### Default

`CALL STRCHS('!','[',']')` restores the default.

### See also

`QSTR7` and `STRLV`.

## SUBROUTINE STRLV (ILEVEL)

### Name

`STRLV` – to specify the level of interpretation of embedded commands in text strings.

**Availability** Section plus, released version 2-11.

### Argument

`ILEVEL` `INTEGER` expression Level of interpretation

| `ILEVEL` | *Action* |
|---|---|
| 0 | Disable interpretation (default) |
| 1 | Enable interpretation of subcommands |

### Description

`STRING` arguments to SIMPLEPLOT subroutines which supply text to be drawn on pictures, can include embedded subcommands `!N`, `!B`, `!H`, `!L`, `!U`, `!P`, `!F`.

| *Embedded subcommand* | *Action* |
|---|---|
| `!N` | new line |
| `!B` | backspace |
| `!H[`*t*`ext]` | superscript |
| `!L[`*t*`ext]` | subscript |
| `!U[`*t*`ext]` | underline |
| `!P`*n*`[`*t*`ext]` | colour/pen change |
| `!F`*n*`[`*t*`ext]` | font change |

These subcommands use ordinary characters as escape sequences to distinguish them from the text to be drawn. `STRLV` sets the level of interpretation of these escapes sequences within text strings.

### Default

`CALL STRLV(0)` restores the default.

### See also

`QSTR7` and `STRCHS`.

## SUBROUTINE TEXTAR (RATIO)

### Name

TEXTAR – to specify the aspect ratio of text characters.

**Availability** Section 4, released version 2-10.

### Argument

| | | |
|---|---|---|
| RATIO | REAL expression | Width/height |

### Description

TEXTAR can be used to specify the height to width ratio of text characters. If hardware text is currently selected, TEXTAR only has effect if the following criteria are met:

- the current device is capable of adjusting the aspect ratio of text, and
- the device driver has been suitably modified.

TEXTAR affects all software text except the adjustable character sets (ISET=51 and 52).

By default, 'Simple software' characters (as selected by CHSET(1)) are constrained such that they are drawn using a whole number of device steps (or pixels) per step in order to achieve satisfactory shapes. If a non-default aspect ratio is specified, this constraint is removed and text may appear misshapen especially on low-resolution devices.

### Default

Calling TEXTAR with RATIO $\leq 0$ restores the default.

### See also

CHSET and TEXTSZ.

## SUBROUTINE TEXTLF (FACTOR)

### Name

`TEXTLF` – to specify the vertical separation of lines of text.

**Availability** Section 1, released version 2-11.

### Argument

| | | |
|---|---|---|
| `FACTOR` | `REAL` expression | Factor to multiply text height |

### Description

`TEXTLF` modifies the vertical spacing between successive lines of title, between entries in key and caption areas, and between multiple lines of text containing encoded linefeeds. `FACTOR` can be any `REAL` value greater than or equal to 1.0.

### Default

`CALL TEXTLF(1.2)` restores the default or any call of `TEXTLF` with `FACTOR` $\leq 0.0$.

### See also

`TEXTSZ`.

## SUBROUTINE TEXTMG (FACTOR)

### Name

TEXTMG – to specify the magnification of text.

**Availability** Section 1, released before version 2-5.

### Argument

| | | |
|---|---|---|
| FACTOR | REAL expression | Magnification factor for text size |

### Description

By default, SIMPLEPLOT aims to produce text which is both clear and well proportioned. A target size of text used is chosen to relate to the physical size of the picture, and to be appropriate to its purpose – title, key, axis annotation *etc.* This target size can not be achieved on all devices – some devices can reproduce hardware text at any chosen size, some have a few discrete sizes, and some can only draw one size of text; software text may vary according to the pixel resolution of the device.

If TEXTMG has been called, the target size is multiplied by FACTOR; this new target size is then adjusted as described above.

### Default

CALL TEXTMG(1.0) restores the default.

### See also

TEXTMN and TEXTSZ.

## SUBROUTINE TEXTMN (CMS)

**Name**

`TEXTMN` – to specify the minimum text size.

**Availability** Section 1, released version 2-6.

**Argument**

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Minimum character width (in cms) |

**Description**

By default, SIMPLEPLOT aims to produce text which is both clear and well proportioned. A target size of text used is chosen to relate to the physical size of the picture, and to be appropriate to its purpose – title, key, axis annotation *etc.* This target size can not be achieved on all devices – some devices can reproduce hardware text at any chosen size, some have a few discrete sizes, and some can only draw one size of text; software text may vary according to the pixel resolution of the device.

The target width is always forced to be no less than some minimum value. After `TEXTMN` has been called, the specified value is used in place of all smaller targets. If `TEXTMN` has not been called, a minimum set in the device driver is used; but when no minimum is set in the device driver either, 0.14 centimetres is used.

For proportionally-spaced fonts (*eg.* the Hershey fonts), the target text size is set to the width of digits which is uniform for all digits within a font.

**Default**

`CALL TEXTMN(0.0)` restores the default.

**See also**

`TEXTMG` and `TEXTSZ`.

## SUBROUTINE TEXTSL (ANGLE)

### Name

`TEXTSL` – to specify the slant of hardware text.

**Availability** Section 4, released version 2-10.

### Argument

| | | |
|---|---|---|
| `ANGLE` | `REAL` expression | Clockwise angle of slant from the upright (in degrees) |

### Description

`TEXTSL` can be used to specify the angle of slant of hardware text but only has effect if the following criteria are met:

- hardware text is currently selected,
- the current device is capable of adjusting the slope of text, and
- the device driver has been modified to implement this facility.

The slant is specified, in degrees, as the clockwise angle from the upright. Positive angles produce forward sloping characters and negative angles produce backward sloping characters. The maximum slope is device dependent.

### Default

`CALL TEXTSL(0.0)` restores the default.

### See also

`CHSET` and `TEXTAR`.

## SUBROUTINE TEXTSZ (CMS)

**Name**

`TEXTSZ` – to specify the fixed character width of all text.

**Availability** Section 1, released version 2-5.

**Argument**

| | | |
|---|---|---|
| `CMS` | `REAL` expression | Character width (in cms) |

**Description**

By default, Simpleplot aims to produce text which is both clear and well proportioned. A target size of text used is chosen to relate to the physical size of the picture, and to be appropriate to its purpose – title, key, axis annotation *etc.* This target size can not be achieved on all devices – some devices can reproduce hardware text at any chosen size, some have a few discrete sizes, and some can only draw one size of text; software text may vary according to the pixel resolution of the device.

`TEXTSZ` fixes the target size of text to a value specified in centimetres. For the reasons described above, this target may not always be achieved.

For the proportionally-spaced fonts (*eg.* the Hershey fonts) the text size corresponds to the width of digits which is uniform for all digits within a font.

For subscripts and superscripts, the ordinary target size is multiplied by 0.75.

**Default**

`CALL TEXTSZ(0.0)` restores the default.

**See also**

`TEXTMG` and `TEXTMN`.

## SUBROUTINE THCKMG (CHTYPE, FACTOR)

### Name

THCKMG – to specify the pen thickness to be used for all plotting.

**Availability** Section 1, released version 2-13.

### Arguments

| | | |
|---|---|---|
| CHTYPE | CHARACTER*1 | Type of plotting to be affected by FACTOR |
| FACTOR | REAL expression | Magnification factor for thickening |

| CHTYPE | *Type of plotting* |
|---|---|
| 'L' | Line drawing |
| 'T' | Selected software text |
| 'M' | Software marker symbols |
| '*' | All of the above |

### Description

By default, SIMPLEPLOT performs all plotting with a single pen thickness (FACTOR=1.0). After THCKMG has been called, the type of plotting specified by CHTYPE is performed using the pen thickness specified by FACTOR.

Software shading lines are not affected by THCKMG.

If line thickness has been specified as a bundled attribute (see BUNLPR), each member of the sequence of line thicknesses (1, 2, 3, 4) is multiplied by FACTOR.

### Default

CALL THCKMG('*',1.0) restores the default.

### See also

BUNLPR.

### SUBROUTINE TIMFMT (COMPNT, LBTYPE, NCHARS)

**Name**

TIMFMT – to specify the format for one component of *time-date* data.

**Availability** Section plus, released version 2-12.

**Arguments**

| | | |
|---|---|---|
| COMPNT | CHARACTER*2 | *Time-date* component |
| LBTYPE | CHARACTER*1 | Label type |
| NCHARS | INTEGER expression | Number of characters |

| LBTYPE | *Label type* |
|---|---|
| 'N' | Number alone |
| 'P' | Prefixed number |
| 'S' | Suffixed number |
| 'T' | Text – only valid for 'DW', 'DM', 'MO', and 'HO' |
| 'D' | Default – the value of NCHARS is ignored |

**Description**

*Time-date* annotations are built from combinations of components, each of which may be expressed in different formats. The possible interpretations of COMPNT are listed in Appendix T of the *SIMPLEPLOT Supplement*.

For text labels, if NCHARS $> 0$, the component will contain NCHARS characters, truncating from the right if the string is too long. If NCHARS $\leq 0$ the whole string is used, excluding trailing spaces.

For numeric labels, if NCHARS=2 and the component value is a whole number, two digits are used, including zeros. Otherwise, SIMPLEPLOT uses as many digits as are needed to represent the value.

If LBTYPE='P' or 'S', a single character prefix or suffix, specified by TIMLDC, is appended to the numerical value of the component. The individual components of axis labels are separated by one-character separators which can be controlled independently by AXLBSP.

**Diagnostics**

```
(Invalid component type: XX)1
(Text label unavailable: XX)2
```

**Default**

CALL TIMFMT(COMPNT,'D',0) restores the default.

**See also**

AXLBSP, AXLBTP, TIMLDA and TIMLDC.

## SUBROUTINE TIMLDA (COMPNT, LABARR, NARR)

### Name

TIMLDA – to load a new set of text labels for one component of *time-date* axis annotations.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| COMPNT | CHARACTER*2 | *Time-date* component |
| LABARR | STRING array | Containing set of labels |
| NARR | INTEGER expression | Number of items in LABARR |

| COMPNT | *Meaning* | *Max NARR* |
|---|---|---|
| 'MO' | Month | 12 |
| 'DM' | Day of the month | 31 |
| 'DW' | Day of the week | 7 |
| 'HO' | Hour | 24 |

### Description

*Time-date* annotations are built from up to three components. All components can be labelled numerically but some components can also be labelled with text from lists stored by SIMPLEPLOT. These standard labels are listed in Appendix T of the *SIMPLEPLOT Supplement*. TIMLDA changes the contents of the list for one component.

If NARR=0, LABARR is not examined and the standard labels for that component are loaded; otherwise, up to NARR labels from array LABARR are stored for use as text labels. If a non-zero value of NARR is smaller than the maximum for that component, the first NARR labels are stored, and used in conjunction with the rest of the existing set of text labels.

Please note that TIMLDA is only used to change the text labels – TIMFMT is used to switch between the use of text labels and numeric labels.

### Diagnostics

```
(Invalid component type: XX)1
(Inappropriate component type: XX)2
```

### Default

CALL TIMLDA(COMPNT,LABARR,0) restores the default.

### See also

AXLBTP, TIMFMT and TIMLDC.

## SUBROUTINE TIMLDC (COMPNT, CHTYPE)

### Name

`TIMLDC` – to load a new prefix/suffix character for numeric labels of one component of *time-date* axis annotations.

**Availability** Section plus, released version 2-12.

### Arguments

| | | |
|---|---|---|
| `COMPNT` | `CHARACTER*2` | *Time-date* component |
| `CHTYPE` | `CHARACTER*1` | Character to add to numeric labels |

### Description

*Time-date* annotations are built from up to three components. Each component can be labelled numerically, with a single-character prefix or suffix. `TIMLDC` sets the stored character for one component.

`COMPNT` identifies the component; the possible interpretations of `COMPNT` are listed in Appendix T of the *SIMPLEPLOT Supplement*. `CHTYPE` specifies the character to use. If `CHTYPE=' '`, the default character is loaded.

`TIMLDC` only changes the stored character; `TIMFMT` must be called to switch between the use of prefixed/suffixed numeric labels and alternative forms of label. The individual components of labels are separated by one-character separators controlled independently by `AXLBSP`.

### Diagnostics

`(Invalid component type: `*`X`*`X)`[1]

### Default

`CALL TIMLDC(COMPNT,' ')` restores the default.

### See also

`AXLBSP`, `AXLBTP`, `TIMFMT` and `TIMLDA`.

## SUBROUTINE TITLE7 (VCHAR, HCHAR, CAP)<br>SUBROUTINE TITLE (VCHAR, HCHAR, CAP, NCAP)

### Name

TITLE7 – to draw a text string as a title to the picture, group or page.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of title |
| HCHAR | CHARACTER*1 | Horizontal position of title |
| CAP | STRING expression | Caption |
| NCAP | INTEGER expression | Number of characters in CAP |

### Description

TITLE/TITLE7 draws [2] the caption, CAP, as a title in the specified position. Several lines of text can be drawn by successive calls of TITLE7/TITLE in the same position. The only restriction on how many titles can be drawn in any position, is when the bottom of the page is reached.

VCHAR and HCHAR are single characters representing the initial letters of vertical and horizontal positions. The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

'N'orth and 'H'igher both position titles as high as possible on a page but differ in their horizontal positioning – 'N'orth titles use the full width of the page and can be centred within the limits of the page whereas 'H'igher titles use the limits of the group.

The position of the title also determines its horizontal limit – the full width of the page, the width of the group or the width of the picture.

### Diagnostics

(None of title will fit in)[2]<br>
(Text clipped)[2]<br>
(Title attempted with no page)[2]<br>
(Title attempted with no picture)[2]<br>
(Title omitted below bottom)[2]<br>
(Title too tall)[2]<br>
(Title truncated)[2]

### See also

ADDCP7, CLIPLV, CP7LB, CP7PT, CP7XC, CP7YC, DEFCAP and KPOSXY.

### SUBROUTINE VS3DLM (XSTART, XSTOP, YSTART, YSTOP, ZSTART, ZSTOP)

**Name**

VS3DLM – to set the three-dimensional Limiting Box for plottable data.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| XSTART | REAL expression | Value at start of $x$ scale |
| XSTOP | REAL expression | Value at end of $x$ scale |
| YSTART | REAL expression | Value at start of $y$ scale |
| YSTOP | REAL expression | Value at end of $y$ scale |
| ZSTART | REAL expression | Value at start of $z$ scale |
| ZSTOP | REAL expression | Value at end of $z$ scale |

**Description**

*SIMPLEPLOT ViSualization* scales are set to accommodate all data within a three-dimensional box. By default, the $x$, $y$ and $z$ limits of this box are all $-1.0$ to $+1.0$. VS3DLM changes these limits.

All points which lie inside the Limiting Box after transformation can be plotted. Note that, although a point may have $(x, y, z)$ coordinates within the limits set by VS3DLM, it may still lie outside the Limiting Box if any transformations have been applied.

For more information about the effect of transformations on scales, see documentation on the transformation subroutines VSMAG, VSROT and VSTRAN, and the scaling subroutines VSFIT, VSFITP and VSFULL.

VS3DLM is also used to set the Origin, which is at the centre of the Limiting Box. By default, this is at $(0.0, 0.0, 0.0)$, although after

```
CALL VS3DLM(XSTART, XSTOP, YSTART, YSTOP, ZSTART, ZSTOP)
```

it is at

$$\left(\frac{\mathtt{XSTART+XSTOP}}{2}, \frac{\mathtt{YSTART+YSTOP}}{2}, \frac{\mathtt{ZSTART+ZSTOP}}{2}\right)$$

VS3DLM does not take effect until the next call of VSNEW.

**Default**

Calling VS3DLM with XSTART=XSTOP, YSTART=YSTOP or ZSTART=ZSTOP restores the default for all scales.

**See also**

QV3DLM, VSFIT, VSFITP, VSFULL and VSINIT.

## SUBROUTINE VSAXDR (CAPX, CAPY, CAPZ)

### Name

VSAXDR – to draw 3-D axes on a *SIMPLEPLOT ViSualization* picture.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| CAPX | STRING expression | $x$-axis caption |
| CAPY | STRING expression | $y$-axis caption |
| CAPZ | STRING expression | $z$-axis caption |

### Description

VSAXDR can be called to draw $x$-$y$-$z$ axes for the current *SIMPLEPLOT ViSualization* picture. VSAXDR must be called after 3-D scales have been applied by VSNEW. The axes are drawn directly on the picture, not into the raster image. It is preferable to draw axes after the image has been transmitted by VSOUT, to avoid the axis being overdrawn.

### Diagnostics

[VSAXDR: No current ViSualization picture][1]

### See also

AX*, VSAXFC, VSNEW and VSOUT.

### SUBROUTINE VSAXFC (IXFIX, IYFIX, IZFIX)

**Name**

`VSAXFC` – to specify the colours for filled rear planes of *SIMPLEPLOT ViSualization* pictures.

**Availability** Section 7, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| `IXFIX` | `INTEGER` expression | Fill index for rear $x$ plane |
| `IYFIX` | `INTEGER` expression | Fill index for rear $y$ plane |
| `IZFIX` | `INTEGER` expression | Fill index for rear $z$ plane |

**Description**

`VSAXFC` specifies colour indices for filling the rear planes of the *SIMPLEPLOT ViSualization* Limiting Box. If a non-negative value is specified for any plane, it is filled whenever a new image is started. `VSAXFC` must therefore be called before `VSNEW`.

`IXFIX` specifies the colour index for the rear plane(s) with $x$ constant, `IYFIX` specifies the colour index for the rear plane(s) with $y$ constant, and `IZFIX` specifies the colour index for the rear plane(s) with $z$ constant.

If colour index $-1$ is specified (default), the associated plane is not filled. Zero, one or two planes may be filled for each variable, depending on the Viewing Position set by `VSVRTP`/`VSVXYZ`.

In this release of the software, data may be mapped onto colour indices 0 to 95 inclusive.

**Default**

`CALL VSAXFC(-1,-1,-1)` restores the default.

**Diagnostics**

`[VSAXFC: Colour index` *I* `out of range]`$^1$

**See also**

`VSAXGC`, `VSOUT`, `VSVRTP` and `VSVXYZ`.

## SUBROUTINE VSAXGC (IXCIX, IYCIX, IZCIX)

### Name

`VSAXGC` – to specify the colour indices for grid lines on the rear planes of *SIMPLEPLOT ViSualization* pictures.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `IXCIX` | `INTEGER` expression | Colour index for $x$ |
| `IYCIX` | `INTEGER` expression | Colour index for $y$ |
| `IZCIX` | `INTEGER` expression | Colour index for $z$ |

### Description

`VSAXGC` specifies colour indices for grid lines on the rear planes of the *SIMPLEPLOT ViSualization* Limiting Box. When grids have been requested, they are drawn in the raster image whenever a new image is started. `VSAXGC` must therefore be called before `VSNEW`.

`IXCIX` specifies the colour index for grid lines on the rear plane(s) with $x$ constant, `IYCIX` specifies the colour index for grid lines on the rear plane(s) with $y$ constant, and `IZCIX` specifies the colour index for grid lines on the rear plane(s) with $z$ constant.

If index $-1$ is specified, grids on the associated planes are not drawn, and if rear planes are filled, grid lines are formed by gaps in the filling. If index $-2$ is specified, the associated grids are omitted. This is the default.

### Default

`CALL VSAXGC(-2,-2,-2)` restores the default.

### Diagnostics

`[VSAXGC: Colour index` *`I`* `out of range]`[1]

### See also

`VSAXFC`.

## SUBROUTINE VSBLFL (X1, X2, Y1, Y2, Z1, Z2)

### Name

`VSBLFL` – to fill a 3-D block on a *SIMPLEPLOT ViSualization* picture.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `X1`, `X2` | `REAL` expressions | Range of $x$ values |
| `Y1`, `Y2` | `REAL` expressions | Range of $y$ values |
| `Z1`, `Z2` | `REAL` expressions | Range of $z$ values |

### Description

`VSBLFL` draws a single block on the current *SIMPLEPLOT ViSualization* picture. This block is filled with the current fill index set by `VSFILC`, and subject to edge visibility and colour.

If the range of any dimension is zero (`X1`=`X2`, `Y1`=`Y2` or `Z1`=`Z2`) nothing is drawn.

### Diagnostics

`[VSBLFL: No current ViSualization picture]`[1]

### See also

`VSBLTN`, `VSBRFL`, `VSEDGC`, `VSEDGV` and `VSFILC`.

### SUBROUTINE VSBLTN (X1, X2, Y1, Y2, Z1, Z2, U1, U2)

**Name**

`VSBLTN` – to tint a 3-D block on a *SIMPLEPLOT ViSualization* picture.

**Availability** Section 7, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| `X1`, `X2` | `REAL` expressions | Range of $x$ values |
| `Y1`, `Y2` | `REAL` expressions | Range of $y$ values |
| `Z1`, `Z2` | `REAL` expressions | Range of $z$ values |
| `U1`, `U2` | `REAL` expressions | Range of $u$ values |

**Description**

`VSBLTN` draws a single block on the current *SIMPLEPLOT ViSualization* picture. The block is tinted with colours ranging from the colour derived from `U1` at $z =$ `Z1` to the colour derived from `U2` at $z =$ `Z2`. The block is subject to edge visibility and colour.

**Diagnostics**

`[VSBLTN: No current ViSualization picture]`[1]

**See also**

`VSBLFL`, `VSBRTN`, `VSEDGC` and `VSEDGV`.

### SUBROUTINE VSBRBZ (ZVAL)

**Name**

`VSBRBZ` – to specify the base $z$ value for bars in *SIMPLEPLOT ViSualization* 3-D barcharts.

**Availability** Section 7, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `ZVAL` | `REAL` expression | Required base $z$ value |

**Description**

By default, *SIMPLEPLOT ViSualization* 3-D barcharts show each bar extending from $z = 0.0$, to a $z$ value from an array. `VSBRBZ` can be called before `VSBRFL` or `VSBRTN` to set an alternative base z value.

**Default**

`CALL VSBRBZ(0.0)` restores the default.

**See also**

`VSBRFL` and `VSBRTN`.

## SUBROUTINE VSBRFC (ICODE)

### Name

`VSBRFC` – to specify how colour sequences should be used on subsequent *SIMPLEPLOT ViSualization* 3-D barcharts.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `ICODE` | `INTEGER` expression | Specify use of colour (see below) |

| `ICODE` | *Different colours* |
|---|---|
| 0 | Faces of bars distinguished by pen pointers (default) |
| 1 | Data Rows distinguished by colour sequence |
| 2 | Data Columns distinguished by colour sequence |
| 3 | Data Sets distinguished by colour sequence |
| 4 | Data magnitudes distinguished by colour sequence |

### Description

`VSBRFL` fills a *SIMPLEPLOT ViSualization* 3-D barchart. `VSBRFC` can be called before `VSBRFL`, to specify how colours are used for the bars.

When `ICODE` = 0, three colour indices distinguish the three planes of each bar: the $x$ plane uses the first colour index, the $y$ plane uses the second, and the $z$ plane uses the third.

When `ICODE` > 0, all planes of a bar are shaded with the same colour index. `SQSHAD` may be called to specify the sequence of colour indices. When `SQSHAD` has not been called, the colour indices are used in the natural sequence 1, 2, 3, ...

### Diagnostics

`[VSBRFC: Integer` *I* `out of range]`[1]

### Default

`CALL VSBRFC(0)` restores the default.

### See also

`SQSHAD` and `VSBRFL`.

## SUBROUTINE VSBRFL (Z, NX, NY, NSETS)

### Name

VSBRFL – to fill a *SIMPLEPLOT ViSualization* 3-D barchart.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| Z | REAL 3-D array | NX × NY × NSETS array of user $(z)$ values |
| NX, NY, NSETS | INTEGER expressions | Dimensions of 3-D data array |

### Description

A *SIMPLEPLOT ViSualization* 3-D barchart is drawn representing NSETS sets of values in array Z, stacked on top of each other. The sets may be stacked downward with the first set at the top, or upward with the first set at the bottom, as specified by VSBRUP.

Different colours may be used to distinguish between data columns, data rows, or data sets, or data magnitudes, as specified by VSBRFC.

### Diagnostics

[VSBRFL: No current ViSualization picture][1]
[VSBRFL: Invalid array dimension][1]

### See also

QVBRFL, VSBRBZ, VSBRFC, VSBRSZ, VSBRTN and VSBRUP.

## SUBROUTINE VSBRSZ (X1, X2, Y1, Y2)

### Name

`VSBRSZ` – to specify the widths of bars in *SIMPLEPLOT ViSualization* 3-D barcharts.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `X1`, `X2` | `REAL` expressions | Limits of bars in $x$ direction |
| `Y1`, `Y2` | `REAL` expressions | Limits of bars in $y$ direction |

### Description

By default, each bar in *SIMPLEPLOT ViSualization* 3-D barcharts extends from 0.1 to 0.9 of the space available in both $x$ and $y$. `VSBRSZ` can be called to specify alternative limits.

When `X2` is not greater than `X1`, or either lies outside the range 0.0 to 1.0, the default in $x$ is restored; when `Y2` is not greater than `Y1`, or either lies outside the range 0.0 to 1.0, the default in $y$ is restored.

### Default

`CALL VSBRSZ(0.0, 0.0, 0.0, 0.0)` restores the default.

### See also

`VSBRFL` and `VSBRTN`.

## SUBROUTINE VSBRTN (Z, NX, NY)

### Name

VSBRTN – to tint a *SIMPLEPLOT ViSualization* 3-D barchart.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| Z | REAL 2-D array | NX×NY user ($z$) values |
| NX, NY | INTEGER expressions | Dimensions of 2-D data array |

### Description

VSBRTN draws a *SIMPLEPLOT ViSualization* 3-D barchart representing the values in array Z. The colours representing different $z$ values are controlled by VSUTOC, and by default map onto colour indices 1 to 15.

### Diagnostics

[VSBRTN: No current ViSualization picture][1]
[VSBRTN: Invalid array dimension][1]

### See also

VSBRBZ, VSBRFL, VSBRSZ and VSUTOC.

## SUBROUTINE VSBRUP (IUP)

### Name

`VSBRUP` – to specify the stacking direction for *SIMPLEPLOT ViSualization* 3-D barcharts.

**Availability** Section 7, released version 2-14.

### Arguments

`IUP` `INTEGER` expression Whether bars stack upwards

| `IUP` | *Direction* |
|---|---|
| 0 | Downward (default) |
| 1 | Upward |

### Description

`VSBRUP` is called before `VSBRFL` to specify whether multiple data sets should be stacked downward with the first set at the top, or upward with the first set at the bottom:

### Diagnostics

`[VSBRUP: Integer` *I* `out of range]`[1]

### Default

`CALL VSBRUP(0)` restores the default.

### See also

`VSBRFL` and `VSBRTN`.

### SUBROUTINE VSCHLS (ICIX, HUE, BRIGHT, SATN)

**Name**

VSCHLS – to set hue, lightness and saturation levels for the *SIMPLEPLOT ViSualization* palette.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| ICIX | INTEGER expression | Colour index to be changed |
| HUE | REAL expression | Hue (degrees) |
| BRIGHT | REAL expression | Lightness (0.0–1.0) |
| SATN | REAL expression | Saturation (0.0–1.0) |

**Description**

*SIMPLEPLOT ViSualization* maintains its own target palette for output. Palette requests on devices with limited colour capabilities are rendered using Error Diffusion techniques.

BRIGHT and SATN must be between 0.0 and 1.0. If they are not, a diagnostic is issues and the palette is unchanged.

VSCHLS sets the device palette if palette control is available.

**Diagnostics**

[VSCHLS: Colour index *I* out of range][1]
[VSCHLS: Colour specification *R* out of range][1]

**See also**

VSCRGB and VSCRNG.

## SUBROUTINE VSCRGB (ICIX, RED, GREEN, BLUE)

### Name

`VSCRGB` – to specify red, green and blue levels for the *SIMPLEPLOT ViSualization* palette.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `ICIX` | `INTEGER` expression | Colour index to be specified |
| `RED` | `REAL` expression | Proportion of Red (0.0–1.0) |
| `GREEN` | `REAL` expression | Proportion of Green (0.0–1.0) |
| `BLUE` | `REAL` expression | Proportion of Blue (0.0–1.0) |

### Description

*SIMPLEPLOT ViSualization* maintains its own target palette for output. Palette requests on devices with limited colour capabilities are rendered using Error Diffusion techniques.

`RED`, `GREEN` and `BLUE` must be between 0.0 and 1.0. If they are not, a diagnostic is issued and the palette is unchanged.

`VSCRGB` sets the device palette if palette control is available.

### Diagnostics

`[VSCRGB: Colour index` *I* `out of range]`[1]
`[VSCRGB: Colour specification` *R* `out of range]`[1]

### See also

`VSCHLS` and `VSCRNG`.

## SUBROUTINE VSCRNG (ICIX1, ICIX2)

### Name

`VSCRNG` – to set a graded range of colours in the *SIMPLEPLOT ViSualization* palette.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `ICIX1` | `INTEGER` expression | first pen index of graded range |
| `ICIX2` | `INTEGER` expression | last pen index of graded range |

### Description

`VSCRNG` sets the colours for all indices between `ICIX1` and `ICIX2`. The existing values of colours `ICIX1` and `ICIX2` are left unchanged and a graded range of colours is interpolated for all indices between them.

### See also

`VSCHLS` and `VSCRGB`.

## SUBROUTINE VSEDGC (ICIX)

### Name

VSEDGC – to specify the colour index for the edge of filled areas.

**Availability** Section 7, released version 2-13.

### Argument

| | | |
|---|---|---|
| ICIX | INTEGER expression | Colour index for edge of filled areas |

### Description

VSEDGC sets the colour index for the edge of *SIMPLEPLOT ViSualization* filled areas. This value is then applied if the edge visibility is set to ON by VSEDGV(1). In this release of software, data may be mapped on to colour indices 0 to 95 inclusive.

If colour index −1 is specified, subsequent edges are omitted. This differs from selecting colour index 0, which draws the edge in background, and from VSEDGV(0) which fills the interior up to and including the edge.

### Diagnostics

[VSEDGC: Colour index *I* out of range][1]

### Default

CALL VSEDGC(1) restores the default.

### See also

VSBLFL, VSBLTN, VSBRFL, VSBRTN, VSEDGV, VSPGFL, VSRG, VSRGU, VSPGTN, VSXYZ, VSXYZU, VSZ and VSZU.

### SUBROUTINE VSEDGV (LEVEL)

**Name**

VSEDGV – to specify whether the edge of polygons is to be drawn.

**Availability** Section 7, released version 2-13.

**Argument**

| | | |
|---|---|---|
| LEVEL | INTEGER expression | Whether to draw the edge in edge colour index |

| LEVEL | *Drawn* |
|---|---|
| 0 | OFF – no edge drawn |
| 1 | ON – edge drawn |

**Description**

By default, the edge visibility is OFF, and the edge of filled areas is drawn in the same colour as the interior. By setting the edge visibility ON, the edge of each polygon is drawn in the edge colour index (see VSEDGC). By drawing the edge of polygons separately from the interior, the underlying data structure can be displayed.

**Diagnostics**

[VSEDGV: Integer *I* out of range][1]

**Default**

CALL VSEDGV(0) restores the default.

**See also**

VSBLFL, VSBLTN, VSBRFL, VSBRTN, VSEDGC, VSEDGI, VSIS, VSISU, VSPGFL, VSPGTN, VSRG, VSRGU, VSXYZ, VSXYZU, VSZ and VSZU.

## SUBROUTINE VSEQX (XSTART, XSTEP)

### Name

VSEQX – to specify the equally-spaced $x$ values to be associated with *SIMPLEPLOT ViSualization* gridded 3-D data.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| XSTART | REAL expression | $x$ value of first data column |
| XSTEP | REAL expression | Interval between data columns |

### Description

By default, data on a regular grid drawn by VSRG, VSRGU, VSIS or VSISU are scaled to fit within the current Limiting Box specified by VS3DLM.

VSEQX specifies the $x$ values corresponding to the data grid, and may be used in conjunction with VS3DLM to position the $y-z$ plane in 3-D space. XSTART specifies the $x$ coordinate to be associated with the first column of the data array, and XSTEP specifies the interval between columns.

VSEQX has no effect if VS3DLM has not been called to set the *SIMPLEPLOT ViSualization* 3-D scales explicitly.

### Default

Calling VSEQX with XSTEP=0.0 restores the default.

### See also

VSEQY, VSEQZ, VSRG, VSRGU, VSIS, and VSISU.

## SUBROUTINE VSEQY (YSTART, YSTEP)

### Name

VSEQY − to specify the equally-spaced $y$ values to be associated with *SIMPLEPLOT ViSualization* gridded 3-D data.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| YSTART | REAL expression | $y$ value of first data row |
| YSTEP | REAL expression | Interval between data rows |

### Description

By default, data on a regular grid drawn by VSRG, VSRGU, VSIS or VSISU are scaled to fit within the current Limiting Box specified by VS3DLM.

VSEQY specifies the $y$ values corresponding to the data grid, and may be used in conjunction with VS3DLM to position the $x - z$ plane in 3-D space. YSTART specifies the $y$ coordinate to be associated with the first row of the data array, and YSTEP specifies the interval between rows.

VSEQY has no effect if VS3DLM has not been called to set the *SIMPLEPLOT ViSualization* 3-D scales explicitly.

### Default

Calling VSEQY with YSTEP=0.0 restores the default.

### See also

VSEQX, VSEQZ, VSRG, VSRGU, VSIS and VSISU.

## SUBROUTINE VSFILC (ICIX)

### Name

`VSFILC` – to specify the colour index to be used by *SIMPLEPLOT ViSualization* area fill subroutines.

**Availability** Section 7, released version 2-13.

### Argument

| | | |
|---|---|---|
| `ICIX` | `INTEGER` expression | Colour index for filling |

### Description

`VSFILC` sets the colour index for subsequent filling by `VSPGFL` or `VSBLFL`. In this release of software, data may be mapped on to colour indices 0 to 95 inclusive. If colour index −1 is specified, subsequent areas are not filled.

### Diagnostics

`[VSFILC: Colour index` *I* `out of range]`[1]

### Default

`CALL VSFILC(1)` restores the default.

### See also

`VSBLFL`, `VSCHLS`, `VSCRGB` and `VSPGFL`.

## SUBROUTINE VSFIT (CHAXES)

### Name

`VSFIT` – to specify which axis scales are measured in similar units.

**Availability** Section 7, released version 2-13.

### Argument

`CHAXES` `STRING` expression axes which are related

| `CHAXES` | *similar axes* |
|---|---|
| `'XYZ'` | all scales in same units (default) |
| `'XY'` | $x$ and $y$ scales are similar |
| `'XZ'` | $x$ and $z$ scales are similar |
| `'YZ'` | $y$ and $z$ scales are similar |
| `' '` | all scales are unrelated |

### Description

2-D Simpleplot scales are set to accommodate all data within the three-dimensional box specified by `VS3DLM`. By default, each of these scales is measured in the same units. If the ranges of these scales are of different magnitudes, the resulting picture is distorted.

`VSFIT` specifies which scales are comparable. If scales are unrelated, each axis defines the edge of a cube which contains the data. If only 2 scales are similar, the length of the third axis is equal to the longer of the other two.

`CHAXES` may be in upper or lower case, but if it does not correspond to one of the strings above, an error message is written and nothing is altered.

If non-default scales are used, the relationship between axes is affected by the transformation routines, `VSMAG`, `VSROT` and `VSTRAN`.

### Diagnostics

`[VSFIT: Invalid string` *`XX`*`]`[1]

### Default

`CALL VSFIT('XYZ')` restores the default.

### See also

`VS3DLM` and `VSFITP`.

## SUBROUTINE VSFITP (XLEN, YLEN, ZLEN)

### Name

VSFITP – to specify the relative lengths of the $x$, $y$ and $z$ axes.

**Availability** Section 7, released version 2-14.

### Argument

| | | |
|---|---|---|
| XLEN | REAL expression | Length of $x$-axis |
| YLEN | REAL expression | Length of $y$-axis |
| ZLEN | REAL expression | Length of $z$-axis |

### Description

By default, the lengths of the *SIMPLEPLOT ViSualization* scales are proportional to the data range they cover. This means that if the ranges are not comparable, the picture is unduly distorted.

VSFITP specifies that the axis lengths are in proportion to each other. After CALL VSFITP(1.0,1.0,1.0), for example, all axes are of the same length. After CALL VSFITP(1.0,2.0,4.0), the $x$-axis is half the length of the $y$-axis, which is half the length of the $z$-axis.

If any argument is non-zero, the default is restored – scales are proportional to data range.

### Default

CALL VSFITP(0.0,0.0,0.0) restores the default.

### See also

VS3DLM and VSFIT.

## SUBROUTINE VSFULL (MODE)

### Name

`VSFULL` – to specify how the Limiting Box is to fill the SIMPLEPLOT picture.

**Availability** Section 7, released version 2-13.

### Argument

| | | |
|---|---|---|
| `MODE` | `INTEGER` expression | Scaling Mode used |

### Description

By default, the 2-D SIMPLEPLOT scales are set to accommodate the extent of the Limiting Box. The underlying SIMPLEPLOT scales are Linear Cartesian with 1 unit in $x$ the same as 1 unit in $y$.

If a point or object is *Transformed* (Rotated, Translated or Magnified), its $(x, y, z)$ coordinates change. This means that it may no longer fit on the picture, even though before transformation it lies within the limits specified by `VS3DLM`. After `CALL VSFULL(1)`, `VSNEW` sets scales to allow all possible rotations. The relative size of the image is comparable for successive pictures.

After `CALL VSFULL(2)`, `VSNEW` sets all points within the 3-D limits to be plottable after the current requested Transformations. The resulting images are larger than those produced by Scaling Mode 1, but the scales of successive pictures are not directly comparable.

`VSFULL` does not take effect until the next call of `VSNEW`.

### Diagnostics

`[VSFULL: Integer` *I* `out of range]`[1]

### Default

`CALL VSFULL(0)` restores the default.

### See also

`VS3DLM`, `VSINIT`, `VSMAG`, `VSROT` and `VSTRAN`.

## SUBROUTINE VSINIT (LEVEL)

### Name

VSINIT – to reset *SIMPLEPLOT ViSualization* defaults.

**Availability** Section 7, released version 2-13.

### Argument

| | | |
|---|---|---|
| LEVEL | INTEGER expression | Level of initialization |

| LEVEL | *Initializes* |
|---|---|
| 1 | Rotation, Translation and Magnification |
| 2 | Scales |
| 4 | Drawing attributes |

### Description

VSINIT may be called to restore some or all defaults of attributes set in *SIMPLEPLOT ViSualization*:

The following subroutines are affected by VSINIT:

| LEVEL | *Subroutines affected* |
|---|---|
| 1 | VSMAG, VSROT, VSTRAN |
| 2 | VS3DLM, VSEQX, VSEQY, VSEQZ, VSFIT,<br>VSFITP, VSFULL, VSVRTP, VSVXYZ |
| 4 | VSAXFC, VSAXGC, VSBRBZ, VSBRFC, VSBRSZ, VSBRUP,<br>VSCRGB, VSEDGC, VSEDGC, VSEDGV, VSFILC, VSCHLS,<br>VSLBOX,<br>VSLINC, VSPMC, VSPMMG, VSUTOC |

VSINIT(2) takes effect at the next call to VSNEW. VSINIT(1) and VSINIT(4) take effect immediately.

Multiple categories of attributes may be set by adding the appropriate values of LEVEL, *eg.* CALL VSINIT(3) is equivalent to calling VSINIT(1) and VSINIT(2). CALL VSINIT(7) restores all defaults.

### Diagnostics

[VSINIT: Integer *I* out of range][1]

### See also

VSNEW.

## SUBROUTINE VSK7H (VCHAR, HCHAR, ULEFT, URIGHT, CAP)

### Name

`VSK7H` – to draw a horizontal key.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| `VCHAR` | `CHARACTER*1` | Vertical position of key |
| `HCHAR` | `CHARACTER*1` | Horizontal position of key |
| `ULEFT` | `REAL` expression | $u$ value at left of key |
| `URIGHT` | `REAL` expression | $u$ value at right of key |
| `CAP` | `STRING` expression | Single caption for key |

### Description

`VSK7H` draws a key to the current patterns representing user data values on tinted *SIMPLEPLOT ViSualization* plots. A horizontal rectangle gives a sample of colours, and a $u$-axis, labelled with the caption `CAP`, gives the data values.

The rectangle is drawn as a *SIMPLEPLOT ViSualization* image, and `VSK7H` should not be called between `VSNEW` and `VSOUT`. A box is drawn round the key unless cancelled by `CALL BOXKY(.FALSE.)`.

`VCHAR` and `HCHAR` are the single character initial letters of vertical and horizontal positions of the key. The possible interpretations of `VCHAR` and `HCHAR` are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

`[VSK7H: Key omitted - active ViSualization picture]` [1]

### See also

`BOXKY`, `VSBLTN`, `VSBRTN`, `VSK7V`, `VSPGTN`, `VSPLTN`, `VSRG`, `VSRGU`, `VSUTOC`, `VSXYZ`, `VSXYZU`, `VSZ` and `VSZU`.

## SUBROUTINE VSK7V (VCHAR, HCHAR, UTOP, UBOTTM, CAP)

### Name

VSK7V – to draw a vertical key.

**Availability** Section 7, released version 2-14.

### Arguments

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of key |
| HCHAR | CHARACTER*1 | Horizontal position of key |
| UTOP | REAL expression | $u$ value at top of key |
| UBOTTM | REAL expression | $u$ value at bottom of key |
| CAP | STRING expression | Single caption for key |

### Description

VSK7V draws a key to the current patterns representing user data values on tinted *SIMPLEPLOT ViSualization* plots. A vertical rectangle gives a sample of colours, and a $u$-axis, labelled with the caption CAP, gives the data values.

The rectangle is drawn as a *SIMPLEPLOT ViSualization* image, and VSK7V should not be called between VSNEW and VSOUT. A box is drawn round the key unless cancelled by CALL BOXKY(.FALSE.).

VCHAR and HCHAR are the single character initial letters of vertical and horizontal positions of the key. The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

### Diagnostics

[VSK7V: Key omitted - active ViSualization picture][1]

### See also

BOXKY, VSBLTN, VSBRTN, VSK7H, VSPGTN, VSPLTN, VSRG, VSRGU, VSUTOC, VSXYZ, VSXYZU, VSZ and VSZU.

### SUBROUTINE VSKEYS (VCHAR, HCHAR, LABARR, NARR, CAP)

**Name**

VSKEYS – to draw a complete key to a sequence of distinct shading patterns.

**Availability** Section 7, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| VCHAR | CHARACTER*1 | Vertical position of area |
| HCHAR | CHARACTER*1 | Horizontal position of area |
| LABARR | STRING array | Set of captions |
| NARR | INTEGER expression | Number of labels in LABARR |
| CAP | STRING expression | Header caption for key |

**Description**

VSKEYS constructs a complete key to the first NARR shading patterns of the currently defined sequence (see SQSHAD and SHPATT).

The key is positioned on the SIMPLEPLOT page in the same way as other keys; VCHAR and HCHAR are single characters representing the initial letters of vertical and horizontal positions. The possible interpretations of VCHAR and HCHAR are listed in Appendix T of the *SIMPLEPLOT Supplement*.

CAP provides a heading which, by default, is centred at the top of the key. The key is associated only with the current picture but area is masked to prevent overdrawing.

VSKEYS should not be called between VSNEW and VSOUT. A box is drawn round the key unless cancelled by CALL BOXKY(.FALSE.).

**See also**

ADDJST, BOXKY, DEFKEY, DEFKYW, DEFPOS, SHPATT, SHSET SQSHAD, VSK7H, VSK7V.

### SUBROUTINE VSLBOX (LEVEL)

**Name**

`VSLBOX` – to specify whether Limiting Boxes are to be drawn around *SIMPLEPLOT ViSualization* images.

**Availability** Section 7, released version 2-13.

**Argument**

| | | |
|---|---|---|
| `LEVEL` | `INTEGER` expression | Whether images are to be boxed |

| `LEVEL` | *Action taken* |
|---|---|
| 0 | No box drawn |
| 1 | Box is drawn |

**Description**

By default, images are not boxed. `VSLBOX` specifies whether or not a 3-D box is to be drawn around every subsequent image.

After `CALL VSLBOX(1)`, a 3-D box is drawn around the area defined by the Limiting Box whenever `VSNEW` is called to start a new image. The box is drawn in the current Line Colour Index set by `VSLINC`. If an image has already been started, no box is drawn until the next image.

**Default**

`CALL VSLBOX(0)` restores the default.

**Diagnostics**

`[VSLBOX: Integer` *I* `out of range]`[1]

**See also**

`VS3DLM`, `VSFULL`, `VSLINC` and `VSNEW`.

## SUBROUTINE VSLINC (ICIX)

### Name

VSLINC – to specify the colour index to be used by line drawing subroutines.

**Availability** Section 7, released version 2-13.

### Argument

| | | |
|---|---|---|
| ICIX | INTEGER expression | Colour index for line drawing |

### Description

VSLINC sets the colour index for subsequent line drawing by VSLBOX or VSPLDR. In this release of software, data may be drawn in colour indices 0 to 95 inclusive. If colour index $-1$ is specified, subsequent lines are omitted.

### Diagnostics

[VSLINC: Colour index *I* out of range][1]

### Default

CALL VSLINC(1) restores the default.

### See also

VSCHLS, VSLBOX, VSPLDR and VSCRGB.

## SUBROUTINE VSMAG (XFACT, YFACT, ZFACT)

### Name

`VSMAG` – to magnify *SIMPLEPLOT ViSualization* coordinates along any or all of the $x$-$y$-$z$ axes.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `XFACT` | `REAL` expression | Magnification factor along the $x$-axis |
| `YFACT` | `REAL` expression | Magnification factor along the $y$-axis |
| `ZFACT` | `REAL` expression | Magnification factor along the $z$-axis |

### Description

By default, *SIMPLEPLOT ViSualization* interprets $(x, y, z)$ coordinates literally. However, following a call to `VSMAG`, a magnification factor is applied to subsequent coordinates.

For example, following a call to `VSMAG(XFACT,1.0,1.0)`, the point $(x, y, z)$ is interpreted as $(x \times \texttt{XFACT}, y, z)$, and all subsequent objects are magnified by `XFACT` along the $x$-axis.

Similar magnification can be applied to the $y$ and $z$-axes by setting `YFACT` and `ZFACT` to values other than 1.0.

After a point has been magnified, its $(x, y, z)$ position changes. This means that it may not fit on the current picture, even if it lies within the limits specified by `VS3DLM`. `VSFULL(2)` may be called before `VSNEW` to ensure that $(x, y, z)$ points within the Limiting Box before transformation stay on the picture.

Transformation is cumulative, so two successive calls to `VSMAG(2.0,2.0,2.0)` quadruple the size along each axis.

The transformed coordinate is a combination of the affects of `VSMAG`, `VSROT` and `VSTRAN`, which means that default magnification cannot be restored on its own. All transformation variables are restored by `CALL VSINIT(1)`.

### See also

`VSFULL`, `VSINIT`, `VSROT` and `VSTRAN`.

## SUBROUTINE VSNEW

### Name

`VSNEW` – to start a new *SIMPLEPLOT ViSualization* picture.

**Availability** Section 7, released version 2-13.

### Arguments

None.

### Description

`VSNEW` starts a new *SIMPLEPLOT ViSualization* picture and updates the variables which affect scaling. These are attributes set by `VS3DLM`, `VSEQX`, `VSEQY`, `VSEQZ`, `VSFIT`, `VSFITP`, `VSFULL`, `VSVRTP` and `VSVXYZ`.

`VSNEW` sets the underlying SIMPLEPLOT scales which are used by the internal *SIMPLEPLOT ViSualization* subroutines when constructing and transmitting an image. If another type of graph is drawn after a call to `VSNEW`, the SIMPLEPLOT scales should be set explicitly by `SCALES` or `EQSCAL`.

Backdrops and grids specified by `VSAXFC` and `VSAXGC` are drawn into the raster image when `VSNEW` is called.

### Diagnostics

`[VSNEW: Radius too short - no perspective applied]`[1]

### See also

`VS3DLM`, `VSAXFC`, `VSAXGC`, `VSFIT`, `VSFITP`, `VSFULL`, `VSVRTP` and `VSVXYZ`.

## SUBROUTINE VSOUT

### Name

`VSOUT` – to flush the raster image.

**Availability** Section 7, released version 2-13.

### Arguments

None.

### Description

`VSOUT` sends the raster image to the display device. With this release of *SIMPLEPLOT ViSualization* the internal raster image is not retained for subsequent re-processing, and `VSNEW` must be called before starting a new image.

### Diagnostics

`[VSOUT: *n* incomplete tasks]`[1]

### See also

`VSNEW`.

## SUBROUTINE VSPGFL (XARRAY, YARRAY, ZARRAY, NPTS)

### Name

`VSPGFL` – to fill a planar polygon with the current fill index in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `XARRAY` | `REAL` array | `NPTS` $x$ values |
| `YARRAY` | `REAL` array | `NPTS` $y$ values |
| `ZARRAY` | `REAL` array | `NPTS` $z$ values |
| `NPTS` | `INTEGER` expression | Number of points in polygon |

### Description

The current transformations are applied to the $x$-$y$-$z$ data. Successive points are joined by a straight line, and the interior area is filled with the current fill colour index. If the Edge Visibility is ON, following a call to `VSEDGV(1)`, the edge is drawn in the current Edge Colour Index.

Polygons must be planar, convex and between 3 and 200 points. If `NPTS` $< 3$, a diagnostic is issued and nothing is drawn.

Hidden line and surface removal are applied to all drawing.

### Diagnostics

`[VSPGFL: No current ViSualization picture]`[1]
`[VSPGFL: Invalid array dimension]`[1]

### See also

`VSEDGC`, `VSEDGV`, `VSFILC` and `VSPGTN`.

### SUBROUTINE VSPGTN (XARRAY, YARRAY, ZARRAY, UARRAY, NPTS)

**Name**

VSPGTN – to tint a planar polygon from user data values in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| XARRAY | REAL array | NPTS $x$ values |
| YARRAY | REAL array | NPTS $y$ values |
| ZARRAY | REAL array | NPTS $z$ values |
| UARRAY | REAL array | NPTS user data values |
| NPTS | INTEGER expression | Number of points in polygon |

**Description**

The current transformations are applied to the $x$-$y$-$z$ data. Successive points are joined by a straight line, and the interior area is filled by changing colours interpolated from the corresponding user data values. If the Edge Visibility is ON, following a call to VSEDGV(1), the edge is drawn in the current Edge Colour Index.

Polygons must be planar, convex and between 3 and 200 points. If NPTS $< 3$, a diagnostic is issued and nothing is drawn.

Hidden line removal is applied to all drawing.

**Diagnostics**

```
[VSPGTN: No current ViSualization picture]
[VSPGTN: Invalid array dimension]
```

**See also**

VSEDGC, VSEDGV, VSPGFL and VSUTOC.

### SUBROUTINE VSPLDR (XARRAY, YARRAY, ZARRAY, NPTS)

**Name**

`VSPLDR` – to draw a polyline through a succession of $(x, y, z)$ data points in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `XARRAY` | `REAL` array | `NPTS` $x$ values |
| `YARRAY` | `REAL` array | `NPTS` $y$ values |
| `ZARRAY` | `REAL` array | `NPTS` $z$ values |
| `NPTS` | `INTEGER` expression | Number of points in polyline |

**Description**

The current transformations are applied to the input data and successive points are joined by a straight line, using the current line colour index. Hidden line removal is applied to all drawing.

If `NPTS` $< 2$, a diagnostic is issued and nothing is drawn.

**Diagnostics**

`[VSPLDR: No current ViSualization picture]`$^1$
`[VSPLDR: Invalid array dimension]`$^1$

**See also**

`VSLINC` and `VSPLTN`.

### SUBROUTINE VSPLTN (XARRAY, YARRAY, ZARRAY, UARRAY, NPTS)

**Name**

`VSPLTN` – to tint a polyline based on user data values in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `XARRAY` | `REAL` array | `NPTS` $x$ values |
| `YARRAY` | `REAL` array | `NPTS` $y$ values |
| `ZARRAY` | `REAL` array | `NPTS` $z$ values |
| `UARRAY` | `REAL` array | `NPTS` user data values |
| `NPTS` | `INTEGER` expression | Number of points in polyline |

**Description**

The current *SIMPLEPLOT ViSualization* transformations are applied to the $x$-$y$-$z$ data. Successive points are joined by a straight line, using changing colours interpolated from the corresponding user data values.

If `VSPLTN` $< 2$, a diagnostic is issued and nothing is drawn.

Hidden line removal is applied to all drawing.

**Diagnostics**

`[VSPLTN: No current ViSualization picture]`[1]
`[VSPLTN: Invalid array dimension]`[1]

**See also**

`VSPLDR` and `VSUTOC`.

### SUBROUTINE VSPMC (ICIX)

**Name**

`VSPMC` – to specify the colour index for subsequent markers.

**Availability** Section 7, released version 2-14.

**Argument**

| | | |
|---|---|---|
| `ICIX` | `INTEGER` expression | Colour index for markers |

**Description**

`VSPMC` sets the colour index for subsequent marker symbols. In this release of software, data may be drawn in colour indices 0 to 95 inclusive. If colour index $-1$ is specified, subsequent markers are omitted.

**Diagnostics**

`[VSPMC: Colour index` *I* `out of range]`$^1$

**Default**

`CALL VSPMC(1)` restores the default.

**See also**

`VSCHLS`, `VSPMDR` and `VSCRGB`.

### SUBROUTINE VSPMDR (XARRAY, YARRAY, ZARRAY, NPTS, MKTYPE)

**Name**

VSPMDR – to draw a set of markers of type MKTYPE in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-14.

**Arguments**

| | | |
|---|---|---|
| XARRAY | REAL array | NPTS $x$ coordinates |
| YARRAY | REAL array | NPTS $y$ coordinates |
| ZARRAY | REAL array | NPTS $z$ coordinates |
| NPTS | INTEGER expression | Number of points in polymarker |
| MKTYPE | INTEGER expression | Marker type |

**Description**

VSPMDR draws a polymarker – a set of marker symbols – of type MKTYPE, centred on each of the points held in the $x$-$y$-$z$ arrays. The current transformations are applied to the $x$-$y$-$z$ data.

Markers are drawn in the current marker colour with the current marker magnification factor applied by VSPMMG. Markers are drawn face on to the viewer, orthogonal to the line of sight.

By default, markers are scaled to occupy 2% of the smaller of the width or height of the picture size, with a minimum size of 0.14cm. When perspective is applied to the *SIMPLEPLOT ViSualization* image, the target marker size applies at a point half way between the front and back of the image. The size of markers decreases away from the viewer.

**Diagnostics**

[VSPMDR: No Current ViSualization picture] $^1$
[VSPMDR: Invalid array dimension] $^1$
[VSPMDR: Integer *I* out of range] $^1$

**See also**

VSPMC and VSPMMG.

## SUBROUTINE VSPMMG (FACTOR)

### Name

`VSPMMG` – to specify the scaling factor for subsequent markers.

**Availability** Section 7, released version 2-14.

### Argument

| | | |
|---|---|---|
| `FACTOR` | `REAL` expression | Factor by which marker size to be scaled |

### Description

By default, markers are scaled to occupy 2% of the smaller of the width or height of the picture size, with a minimum marker size of 0.14cm. When perspective is applied to the *SIMPLEPLOT ViSualization* image, the target marker size applies at a point half way between the front and back of the image. The size of markers decreases away from the viewer.

After `CALL VSPMMG(FACTOR)`, the size of markers is increased by a factor of `FACTOR`. `FACTOR` must be greater than 0.0. If it is not, the default is used.

### Diagnostics

`[VSPMMG: Value` $R$ `out of range]`[1]

### Default

`CALL VSPMMG(0.0)` restores the default.

### See also

`VSPMDR`.

## SUBROUTINE VSRG (Z2ARR, NX, NY)

### Name

VSRG – to draw a perspective surface picture from a 2-D array in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | NX×NY array of user ($z$) values |
| NX, NY | INTEGER expression | Dimensions of 2-D data array |

### Description

The data values held in the array Z2ARR are drawn as a filled surface. The colours representing different heights are controlled by VSUTOC.

By default, the $x$, $y$ and $z$ ranges of the data are mapped onto the Limiting Box. If VS3DLM has been called to set the scales explicitly, the $z$ values map onto the specified limits, and the $x$ and $y$ scales can be controlled by VSEQX and VSEQY.

### Diagnostics

[VSRG: No current ViSualization picture][1]
[VSRG: Invalid array dimension][1]

### See also

VS3DLM, VSEDGC, VSEDGV, VSEQX, VSEQY, VSRGU and VSUTOC.

## SUBROUTINE VSRGU (Z2ARR, U2ARR, NX, NY)

### Name

VSRGU – to draw contours of data from one 2-D array on a surface picture from another 2-D array in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | NX×NY array of ($z$) values |
| U2ARR | REAL 2-D array | NX×NY array of data values |
| NX, NY | INTEGER expressions | Dimensions of 2-D arrays |

### Description

Contours representing U2ARR are drawn on the surface defined by Z2ARR. The colours representing different contour levels are controlled by VSUTOC.

By default, the $x$, $y$ and $z$ ranges of the data are mapped onto the Limiting Box. If VS3DLM has been called to set the scales explicitly, the $z$ values map onto the specified limits, and the $x$ and $y$ scales can be controlled by VSEQX and VSEQY.

### Diagnostics

[VSRGU: No current ViSualization picture][1]
[VSRGU: Invalid array dimension][1]

### See also

VS3DLM, VSEDGC, VSEDGV, VSEQX, VSEQY, VSRG and VSUTOC.

## SUBROUTINE VSROT (CHAXIS, ROTATE)

### Name

VSROT – to rotate *SIMPLEPLOT ViSualization* coordinates around one of the *x*-*y*-*z* axes.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| CHAXIS | CHARACTER*2 | Axis type |
| ROTATE | REAL expression | Rotation (in degrees) |

| CHAXIS | *Type of axis* |
|---|---|
| 'X3' | 3-D perspective *x*-axis |
| 'Y3' | 3-D perspective *y*-axis |
| 'Z3' | 3-D perspective *z*-axis |

### Description

By default, *SIMPLEPLOT ViSualization* interprets $(x, y, z)$ coordinates literally. However, following a call to VSROT, subsequent coordinates are rotated around one of the axes.

CHAXIS signifies axis type. If you are looking down the axis in a positive direction, rotation is clockwise.

Following a call to VSROT('X3',45.0), the point $(x, y, z)$ is rotated by 45° around the *x*-axis. Transformation is cumulative, so two successive calls to VSROT('Y3',30.0) rotate points 60° around the *y*-axis.

After a point has been rotated, its $(x, y, z)$ position changes. This means that it may not fit on the current picture, even if it lies within the limits specified by VS3DLM. VSFULL(1) or VSFULL(2) may be called before VSNEW to ensure that $(x, y, z)$ points within the Limiting Box before transformation stay on the picture.

The transformed coordinate is a combination of the effects of VSMAG, VSROT and VSTRAN, which means that default rotation cannot be restored on its own. All transformation variables are restored by CALL VSINIT(1).

### Diagnostics

[VSROT: Invalid string *XX*][1]

### See also

VSFULL, VSINIT, VSMAG and VSTRAN.

## SUBROUTINE VSTRAN (XTRANS, YTRANS, ZTRANS)

### Name

`VSTRAN` – to translate *SIMPLEPLOT ViSualization* coordinates along any or all of the $x$-$y$-$z$ axes.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `XTRANS` | `REAL` expression | Translation along the $x$-axis |
| `YTRANS` | `REAL` expression | Translation along the $y$-axis |
| `ZTRANS` | `REAL` expression | Translation along the $z$-axis |

### Description

By default, *SIMPLEPLOT ViSualization* interprets $(x, y, z)$ coordinates literally. However, following a call to `VSTRAN`, a translation value is applied to subsequent coordinates.

For example, following a call to `VSTRAN(XTRANS,0.0,0.0)`, the point $(x, y, z)$ is interpreted as $(x + \texttt{XTRANS}, y, z)$, and all subsequent objects are translated by `XTRANS` along the $x$-axis.

Similar translation can be applied to the $y$ and $z$-axes by setting `YTRANS` and `ZTRANS` to non-zero values.

After a point has been translated, its $(x, y, z)$ position changes. This means that it may not fit on the current picture, even if it lies within the limits specified by `VS3DLM`. `VSFULL(2)` may be called before `VSNEW` to ensure that $(x, y, z)$ points within the Limiting Box before transformation stay on the picture.

Transformation is cumulative, so two successive calls to `VSTRAN(2.0,2.0,2.0)` move points 4.0 units along each axis.

The transformed coordinate is a combination of the effects of `VSMAG`, `VSROT` and `VSTRAN`, which means that default translation cannot be restored on its own. All transformation variables are restored by `CALL VSINIT(1)`.

### See also

`VSFULL`, `VSINIT`, `VSMAG` and `VSROT`.

### SUBROUTINE VSUTOC (UMIN, UMAX, ICIX1, ICIX2)

**Name**

VSUTOC – to define mapping from user data values to colour indices.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| UMIN, UMAX | REAL expressions | Range of user data values to be mapped |
| ICIX1, ICIX2 | INTEGER expressions | Corresponding range of colour indices |

**Description**

By default, *SIMPLEPLOT ViSualization* maps the full range of user data given in output subroutines on to colour indices 1 to 15. VSUTOC selects different ranges of colour associated with a *SIMPLEPLOT ViSualization* picture.

VSUTOC maps data within a given range onto a specified set of pen indices. After CALL VSUTOC(UMIN,UMAX,ICIX1,ICIX2), each index from ICIX1 to (ICIX2 − 1) is associated with an equal range of data between UMIN and UMAX. Each range measures $\frac{\texttt{UMAX}-\texttt{UMIN}}{\texttt{ICIX2}-\texttt{ICIX1}}$ units.

Index ICIX2 represents all data values greater than or equal to UMAX. Index ICIX1 is also used for data less than UMIN.

The default action may be restored by setting UMIN greater than UMAX.

In this release of software, data may be mapped on to colour indices 0 to 95 inclusive.

**Diagnostics**

[VSUTOC: Colour index *I* out of range][1]

**Default**

CALL VSUTOC(0.0, -1.0, 0, 0) restores the default.

**See also**

VSCRGB and VSCHLS.

## SUBROUTINE VSVRTP (RADIUS, THETA, PHI)

### Name

`VSVRTP` – to set the Viewing Position in terms of a radius and 2 angles for a projected picture.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| `RADIUS` | `REAL` expression | Distance of the Viewing Position from the Origin |
| `THETA` | `REAL` expression | Horizontal angle |
| `PHI` | `REAL` expression | Vertical angle |

### Description

`VSVRTP` sets a Viewing Position for *SIMPLEPLOT ViSualization* in terms of a radius and 2 angles.

`RADIUS` is the distance between the Viewing Position and the Origin, where the maximum possible distance from the Origin to any point in the Limiting Box is $\sqrt{3}$. The Origin and Limiting Box are set by `VS3DLM`.

`RADIUS` is interpreted as being positive. A radius of $-R$ is equivalent to a radius of $+R$.

If the Viewing Position is too close to the Origin, points could be projected to infinity, or be otherwise unplottable. To prevent this happening, if the absolute value of `RADIUS` $\leq \sqrt{3}$, a diagnostic is issued at the next call to `VSNEW`, and no perspective is applied.

`THETA` is the horizontal angle in degrees between the Viewing Position and the Origin on the $x$-$y$ plane; `PHI` is the vertical angle in degrees between this plane and the Line of Sight. By default, `THETA`=30.0, and `PHI`=15.0.

`THETA` $= 0.0$ along the $x$-axis (*ie.* at $y = 0.0$). `PHI` $= 0.0$ on the $x$-$y$ plane (*ie.* at $z = 0.0$).

The change in Viewing Position does not take effect until the next call of `VSNEW`. `VSVRTP` overrides any previous call to `VSVXYZ`.

### Default

`CALL VSVRTP(0.0, 30.0, 15.0)` restores the default.

### See also

`VS3DLM`, `VSINIT`, `VSNEW` and `VSVXYZ`.

## SUBROUTINE VSVXYZ (XVIEW, YVIEW, ZVIEW)

### Name

VSVXYZ – to set the Viewing Position for a projected picture in terms of $(x, y, z)$.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| XVIEW | REAL expression | $x$ component of Viewing Position |
| YVIEW | REAL expression | $y$ component of Viewing Position |
| ZVIEW | REAL expression | $z$ component of Viewing Position |

### Description

VSVXYZ sets a Viewing Position for *SIMPLEPLOT ViSualization* in terms of $(x, y, z)$.

XVIEW, YVIEW and ZVIEW are in the underlying three-dimensional coordinates before transformation.

If the Viewing Position is too close to the Origin, points could be projected to infinity, or be otherwise unplottable. To prevent this from happening, the distance from (XVIEW, YVIEW, ZVIEW) to the Origin should be greater than the radius of the Bounding Sphere specified by VS3DLM.

If

$$\mathtt{XVIEW}^2 + \mathtt{YVIEW}^2 + \mathtt{ZVIEW}^2 <$$
$$(\frac{\mathtt{XSTOP-XSTART}}{2})^2 + (\frac{\mathtt{YSTOP-YSTART}}{2})^2 + (\frac{\mathtt{ZSTOP-ZSTART}}{2})^2$$

a diagnostic is issued at the next call to VSNEW, and no perspective is applied.

The change in Viewing Position does not take effect until the next call of VSNEW. VSVXYZ overrides any previous call to VSVRTP.

### Default

See VSVRTP.

### See also

VS3DLM, VSINIT, VSNEW and VSVRTP.

### SUBROUTINE VSXYZ (X2ARR, Y2ARR, Z2ARR, NX, NY)

**Name**

VSXYZ – to draw a surface picture from three 2-D arrays in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| X2ARR,<br>Y2ARR,<br>Z2ARR | REAL 2-D arrays | NX×NY arrays of user $(x, y, z)$ values |
| NX, NY | INTEGER expression | Dimensions of 2-D arrays |

**Description**

The quadrilateral data structure held in the $x$-$y$-$z$ arrays is represented by a surface. Elements are constructed from each of the 4 adjacent elements in the 2-D arrays. The colours representing different $z$-values are controlled by VSUTOC.

**Diagnostics**

[VSXYZ: No current ViSualization picture][1]
[VSXYZ: Invalid array dimension][1]

**See also**

VS3DLM, VSEDGC, VSEDGV, VSUTOC and VSXYZU.

## SUBROUTINE VSXYZU (X2ARR, Y2ARR, Z2ARR, U2ARR, NX, NY)

### Name

VSXYZU – to draw contours of data in one 2-D array on a surface picture from three 2-D arrays in a *SIMPLEPLOT ViSualization* image.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| X2ARR,<br>Y2ARR,<br>Z2ARR | REAL 2-D arrays | NX×NY arrays of user $(x, y, z)$ values |
| U2ARR | REAL 2-D array | NX×NY User data values |
| NX, NY | INTEGER expression | Dimensions of 2-D arrays |

### Description

The quadrilateral data structure held in the $x$-$y$-$z$ arrays is represented by a surface. Contours from the corresponding U2ARR array are drawn on this surface. Elements are constructed from each of the four adjacent elements in the $x$-$y$-$z$ arrays. The colours representing different $u$ values are controlled by VSUTOC.

### Diagnostics

[VSXYZU: No current ViSualization picture] [1]
[VSXYZU: Invalid array dimension] [1]

### See also

VS3DLM, VSEDGC, VSEDGV, VSUTOC and VSXYZ.

## SUBROUTINE VSZ (XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELS)

### Name

VSZ – to draw a surface from 3-D data structured into elements.

**Availability** Section 7, released version 2-13.

### Arguments

| | | |
|---|---|---|
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values<br>in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELS | INTEGER expression | Number of elements |

### Description

VSZ draws a surface representing data structured into elements and held in parallel arrays, XARR(NPTS), YARR(NPTS) and ZARR(NPTS).

Before using VSZ, the element array I2ARR(NNODES,NELS), must be set up. The SIMPLEPLOT-PLUS subroutines ZZORDR or ZZORDN can only be used for this purpose if the surface is such that there is a single function value z for any $(x, y)$, otherwise the user must construct an index array identifying the indices in the $x$, $y$ and $z$ arrays to each node in each element.

The colours representing different contour levels are controlled by VSUTOC.

### Diagnostics

```
[VSZ: No current ViSualization picture]
[VSZ: Invalid array dimension]
```
[1] [1]

### See also

VSEDGC, VSEDGV, VSUTOC, and VSZU.

### SUBROUTINE VSZU (XARR, YARR, ZARR, UARR, NPTS, I2ARR, NNODES, NELS)

**Name**

`VSZU` – to draw contours of data in an array on a surface from 3-D data structured into elements.

**Availability** Section 7, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `X2ARR,`<br>`Y2ARR,`<br>`Z2ARR` | `REAL` arrays | $(x, y, z)$ coordinates of data values in parallel arrays |
| `U2ARR` | `REAL` array | `NPTS` user data values |
| `NPTS` | `INTEGER` expression | Number of data points |
| `I2ARR` | `INTEGER` 2-D array | Data element structure |
| `NNODES` | `INTEGER` expression | Number of nodes per element |
| `NELS` | `INTEGER` expression | Number of elements |

**Description**

`VSZU` draws a contour map of the data values in `UARR(NPTS)`, on the surface representing data structured into planar elements and held in parallel arrays `XARR(NPTS)`, `YARR(NPTS)` and `ZARR(NPTS)`.

Before using `VSZU`, the element array `I2ARR(NNODES,NELS)` must have been set up. The Simpleplot-Plus subroutines `ZZORDR` or `ZZORDN` can be used for this purpose if the surface is such that there is a single function value $z$ for any $(x, y)$. Otherwise, the user must construct an index array identifying the indices in the $x$, $y$ and $z$ arrays to each node in each element.

The colours representing different contour levels are controlled by `VSUTOC`.

**Diagnostics**

`[VSZU: No current ViSualization picture]`[1]
`[VSZU: Invalid array dimension]`[1]

**See also**

`VSEDGC`, `VSEDGV`, `VSUTOC` and `VSZ`.

## SUBROUTINE WFCHT (Z2ARR, NPTS, NCURVS, CAPN, CAPL)

### Name

WFCHT – to draw a waterfall chart on the current picture.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NPTS | INTEGER expression | Number of columns in Z2ARR |
| NCURVS | INTEGER expression | Number of rows in Z2ARR |
| CAPN | STRING expression | Numeric axis caption |
| CAPL | STRING expression | Label axis caption |

### Description

A waterfall chart has a numeric scale for an independent variable and a label scale over which a series of staggered curves are plotted. WFCHT draws a waterfall chart on the current picture of data held in the two-dimensional array, Z2ARR(NPTS,NMAX), where NMAX is greater or equal to NCURVS, the number of curves to be plotted Each curve corresponding to the points Z2ARR(1,*n*) to Z2ARR(NPTS,*n*) is offset from the last by an equal step and masked by curves already drawn on the waterfall chart.

NPTS and NCURVS must both be positive INTEGERs; if they are not, a diagnostic is generated and nothing is drawn. A waterfall curve must have between 2 and 1024 points.

WFCHT draws numeric and label axes which can be configured using appropriate AX* subroutines with CHAXIS='NW' and 'LW' respectively. Other characteristics of a waterfall chart can be specified using prior calls of WF*.

WFCHT draws a waterfall chart on the current 2-D picture and should therefore be preceded by the *new picture* subroutine, NEWPIC.

### Diagnostics

(Constant data: default z scale used)[1]
(Too many points in waterfall curve)[1]
(Waterfall chart not all in range)[2]
(WATERFALL OMITTED: INVALID DIMENSIONS)[1]

### See also

WF*.

## SUBROUTINE WFDRAW (ZARR, NPTS)

### Name

WFDRAW – to draw a single curve on a waterfall chart.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| ZARR | REAL array | Data values of curve |
| NPTS | INTEGER expression | Number of points in ZARR |

### Description

WFDRAW draws a single waterfall curve on the current picture. Each curve corresponding to the points ZARR(1) to ZARR(NPTS) is offset from the last by an equal step, and masked by waterfall curves already drawn.

Different parts of the curves are drawn using different pen pointers so that independent changes to the visibility or appearance of the parts can be achieved through pointer adjustments:

| *Pointer* | *Pen usage* |
|---|---|
| [1] | Unmasked part of curve with $z \geq$ ZLEVEL |
| [2] | Masked part of curve with $z \geq$ ZLEVEL |
| [3] | Unmasked part of curve with $z <$ ZLEVEL |
| [4] | Masked part of curve with $z <$ ZLEVEL |

The first CALL WFDRAW(NPTS) on a chart sets the number of points for curve on the whole chart. Subsequent curves on the same chart are either short or incomplete if they have different values of NPTS. NPTS must be a positive INTEGER; if it is not, a diagnostic is generated and nothing is drawn. The maximum number of points per waterfall curve is limited to 1024.

WFDRAW draws a waterfall curve on the current 2-D picture therefore the first call to WFDRAW must be preceded by the *new picture* subroutine, NEWPIC.

### Diagnostics

(Constant data: default z scale used)$^1$
(INVALID ARGUMENT: ARRAY SIZE)$^1$
(Too many waterfall curves on picture)$^1$
(Too many points in waterfall curve)$^1$
(Waterfall curve not all in range)$^2$

### See also

WFEQL, WFEQN, WFNCVS, WFPNS, WFSTEP, WFZLEV and WFZSCL.

## SUBROUTINE WFEQL (CSTART, CSTEP)

### Name

WFEQL – to specify the equally-spaced values on the label scale of a waterfall chart.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| CSTART | REAL expression | Minimum scale value |
| CSTEP | REAL expression | Interval between curves |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. Standard SIMPLEPLOT calls may be used to add axes to annotate the chart. By default, the label scale is defined to identify the curves by 1.0, 2.0, 3.0 *etc.* in the order of drawing.

WFEQL may be called to specify CSTART, the scale value to associate with the beginning of the span of the first curve, and CSTEP, the scale interval between curves.

The label scale defined by WFEQL affects axis annotations, but has no effect on the curves.

WFEQL takes effect at the start of the next waterfall chart.

### Default

Calling WFEQL with CSTEP=0.0 restores the default.

### See also

AXIS7, WFCHT and WFDRAW.

### SUBROUTINE WFEQN (ESTART, ESTEP)

**Name**

WFEQN – to specify the equally-spaced values on the numeric scale of a waterfall chart.

**Availability** Section plus, released 2-13.

**Arguments**

| | | |
|---|---|---|
| ESTART | REAL expression | Value of first data column |
| ESTEP | REAL expression | Interval between data columns |

**Description**

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves is plotted. By default, curves cover the full range of the numeric scale.

By default, the numeric scale on a waterfall chart is 1.0 to REAL(NPTS) , where NPTS is the number of points in the first curve drawn. WFEQN relates the numeric scale to the data values. WFEQN must be called before a waterfall chart has been started.

After a call to WFEQN, the numeric waterfall axis displays the range of values from ESTART to ESTART + ESTEP × (NPTS − 1), where NPTS is specified as an argument to WFCHT/WFDRAW.

**Default**

Calling WFEQN with ESTEP=0.0 restores the default.

**See also**

WFCHT, WFDRAW, WFEQL and WFNSCL.

## SUBROUTINE WFINIT

### Name

`WFINIT` – to reset all defaults for waterfall charts.

**Availability** Section plus, released version 2-13.

### Arguments

None.

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. The characteristics of waterfall charts can be controlled by various specification subroutines to change values used when plotting waterfall charts.

`WFINIT` cancels the effects of calls of all the waterfall specification subroutines, resetting all values to their defaults.

`WFINIT` has no effect on the axis characteristics of a waterfall chart.

### See also

`INITSP`, `WFEQL`, `WFEQN`, `WFNCVS`, `WFPNS`, `WFSTEP`, `WFZLEV` and `WFZSCL`.

### SUBROUTINE WFNCVS (NCURVS)

**Name**

`WFNCVS` – to specify the number of curves to be accommodated on waterfall charts.

**Availability** Section plus, released version 2-13.

**Argument**

| | | |
|---|---|---|
| `NCURVS` | `INTEGER` expression | Total number of waterfall curves |

**Description**

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. The curves drawn on the chart can all be plotted by a single call of `WFCHT`, or by a succession of calls of `WFDRAW`.

By default, a waterfall chart drawn by `WFCHT` is scaled to accommodate the number of curves in the supplied data; if only one waterfall curve is drawn by `WFDRAW`, the scale accommodates this single curve. `WFNCVS` specifies an alternative number of curves, `NCURVS`.

Following a call to `WFNCVS`, the label scale is calculated to accommodate `NCURVS` curves for each waterfall chart. In this case, a waterfall chart drawn by `WFCHT` is scaled to accommodate `NCURVS` curves and only this number are drawn. Similarly, `NCURVS` curves can be drawn on the same picture by `NCURVS` successive calls of `WFDRAW`.

`NCURVS` must be a positive `INTEGER`; if not, the default is restored. The maximum number of curves that can be accommodated on a waterfall chart is 1024.

**Default**

`CALL WFNCVS(0)` restores the default.

**See also**

`WFCHT`, `WFDRAW` and `WFSTEP`.

### SUBROUTINE WFNSCL (START, STOP)

**Name**

`WFNSCL` – to specify the numeric scale for waterfall charts.

**Availability** Section plus, released version 2-13.

**Arguments**

| | | |
|---|---|---|
| `START` | `REAL` expression | Value at start of numeric scale |
| `STOP` | `REAL` expression | Value at end of numeric scale |

**Description**

By default, the numeric scale on a waterfall chart is 1.0 to `REAL(NPTS)`, where `NPTS` is the number of points in the first curve drawn. If `WFEQN` has been called to relate the plotting scales to the data values, the default numeric scale is `ESTART` to `ESTART` + (`ESTEP` × (`NPTS` − 1)). `WFNSCL` sets an alternative scale from `START` to `STOP`. These new scale limits can extend beyond the default limits or restrict the default limits; also, the direction of the scale can be reversed by making `START` greater than `STOP`.

**Default**

Calling `WFNSCL` with `START`=`STOP` restores the default.

**See also**

`WFEQL` and `WFEQN`.

## SUBROUTINE WFPNS (IPEN1, IPEN2, IPEN3, IPEN4)

### Name

WFPNS – to specify the pens used in waterfall charts.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| IPEN1 | INTEGER expression | Pen associated with [1] |
| IPEN2 | INTEGER expression | Pen associated with [2] |
| IPEN3 | INTEGER expression | Pen associated with [3] |
| IPEN4 | INTEGER expression | Pen associated with [4] |

| *Pointer* | *Pen usage* |
|---|---|
| [1] | Unmasked part of curve with $z \geq$ ZLEVEL |
| [2] | Masked part of curve with $z \geq$ ZLEVEL |
| [3] | Unmasked part of curve with $z <$ ZLEVEL |
| [4] | Masked part of curve with $z <$ ZLEVEL |

| IPEN*i* | *Pen usage* |
|---|---|
| −1 | plotting is omitted on all devices |
| 0 | drawn in background colour |
| 1, 2, 3 ... | pens/bundles as specified by PENHLS/PENRGB and BUNLPR |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. Different parts of the curves are distinguished by the use of different pen pointers.

IPEN*i* can have values −1, 0, 1, 2, 3, ... *etc.*

Pointers set by WFPNS are used for waterfall curves independently of pointers set by SETPNS. This allows different defaults, and simultaneous setting of one set of pointers for the waterfall charts for axes.

### Default

CALL WFPNS(1,-1,1,-1) restores the default.

### See also

AXIS7, BUNLPR, SETPNS, WFCHT and WFDRAW.

### SUBROUTINE WFSTEP (ZDISP)

**Name**

WFSTEP – to specify the displacement between waterfall curves.

**Availability** Section plus, released version 2-13.

**Argument**

| | | |
|---|---|---|
| ZDISP | REAL expression | Displacement between curves relative to the span of a curve |

**Description**

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. Each curve is scaled to occupy an equal portion of the label scale, and the curves are displaced from each other by an equal step. By default, each curve is displaced from the next by one third of the span of a curve.

WFSTEP specifies an alternative value for ZDISP. Subsequently, label scales are calculated to cover a range dependent on a function of ZDISP, the number of curves being accommodated, and the physical length of the scale.

The value of ZDISP must be greater than $-1.0$ (the value of ZDISP for which the next curve would be completely obscured); if ZDISP $\leq -1.0$ the default is restored.

**Default**

CALL WFSTEP(-1.0) restores the default.

**See also**

WFCHT, WFDRAW and WFNCVS.

## SUBROUTINE WFZLEV (ZLEVEL)

### Name

WFZLEV – to specify the data value at which waterfall curves can change pens.

**Availability** Section plus, released version 2-13.

### Argument

| | | |
|---|---|---|
| ZLEVEL | REAL expression | Data value for pen change |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. If pen pointers are used to discriminate between parts of the curves, a switch of pens is made at a particular data value. By default, the switch occurs at 0.0; WFZLEV may be called to specify an alternative value, ZLEVEL, for switching pens.

### Default

CALL WFZLEV(0.0) restores the default.

### See also

WFCHT, WFDRAW and WFPNS.

## SUBROUTINE WFZSCL (ZSTART, ZSTOP)

### Name

WFZSCL – to specify the $z$ scales for curves on subsequent waterfall charts.

**Availability** Section plus, released version 2-13.

### Arguments

| | | |
|---|---|---|
| ZSTART,<br>ZSTOP | REAL expressions | Range of $z$ covered |

### Description

A waterfall chart has a numeric scale for an independent variable, and a label scale over which a series of staggered curves are plotted. Each curve is drawn on a scale covering the same $z$ range.

By default, the $z$ scale extends from the minimum $z$ to the maximum $z$ of the data as it is known at the start of a waterfall chart. This involves all the data when WFCHT is used for the chart, but only the first curve of a chart drawn by WFDRAW. WFZSCL specifies an alternative $z$ scale to be used for all the curves on a waterfall chart. WFZSCL only takes effect when the next waterfall chart is started.

### Default

Calling WFZSCL with ZSTART=ZSTOP restores the default scaling.

### See also

KSCALE, KWZVAL, WFCHT and WFDRAW.

### REAL FUNCTION XCALC (X, Y, Z2ARR, NX, NY, XARR)

**Name**

`XCALC` – to interpolate $z = f(x, y)$ from 3-D data on an $x$-specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `X`, `Y` | `REAL` expressions | Coordinates of a point |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |
| `XARR` | `REAL` array | `NX` $x$ values of the grid |

**Description**

The value of $z$ at $(x, y)$ is evaluated by two-way linear interpolation within the rectangular cell of the grid in which $(x, y)$ is located. If `X` or `Y` lie outside the limits of the data, `XCALC` returns the current no-data value.

Simpleplot can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and the current no-data value is returned.

**Diagnostics**

`(DATA GRID NOT MONOTONIC)`[1]
`(INVALID ARGUMENT: ARRAY SIZE)`[1]

**See also**

`NODATA`, `QNODAT` and `SFEQY`.

## SUBROUTINE XCNTS (Z2ARR, NX, NY, XARR)

### Name

XCNTS – to draw a contour map from 3-D data on an $x$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |

### Description

XCNTS draws a contour map on the current picture representing 3-D data on an $x$-specified tartan grid. Contours are drawn for equally-spaced $z$ values within the range of data held in Z2ARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by XCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set

[2] highest level when more than one level drawn

[3] lowest level of set

[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

SIMPLEPLOT can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

### Diagnostics

(Contour map not all in range)$^2$
(DATA GRID NOT MONOTONIC)$^1$
(INVALID ARGUMENT: ARRAY SIZE)$^1$

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, RIDDLE, SFEQY, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SQBRKN, SQPEN, SQZLAB and SQZVAL.

## SUBROUTINE XCONT (ZLEV, Z2ARR, NX, NY, XARR)

### Name

`XCONT` – to draw a contour curve from 3-D data on an $x$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `ZLEV` | `REAL` expression | Contour level |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |
| `XARR` | `REAL` array | `NX` $x$ values of the grid |

### Description

`XCONT` draws [1] a contour curve on the current picture. The contour curve corresponds to $z =$ `ZLEV` from 3-D data (on an $x$-specified tartan grid) held in `Z2ARR`.

Simpleplot can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no contour is drawn.

### Diagnostics

```
(Contour curve not all in range)2
(DATA GRID NOT MONOTONIC)1
(INVALID ARGUMENT: ARRAY SIZE)1
```

### See also

`CTBRKN`, `CTCURV`, `CTHOLD`, `CTLABS`, `CTNUMB`, `QCURVE`, `RIDDLE` and `SFEQY`.

### SUBROUTINE XCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, XARR)

**Name**

`XCUT` – to draw a 2-D curve of a surface section from 3-D data on an $x$-specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `X1`, `Y1`, `X2`, `Y2` | `REAL` expressions | Coordinates of two points |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |
| `XARR` | `REAL` array | `NX` $x$ values of the grid |

**Description**

If the surface represented by the data in `Z2ARR` were cut along the straight line from $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the data along that line.

`XCUT` draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable scales must be set up before attempting to draw a section.

SIMPLEPLOT can only interpret data on an $x$-specified grid if the grid values in `XARR` are supplied in increasing or decreasing order (*ie.* they are *monotonic*). If the grid values are not monotonic, a diagnostic is issued and no curve is drawn.

**Diagnostics**

```
(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]
(Surface section not all in range)[2]
```

**See also**

`CTBRKN`, `CTCURV`, `CTHOLD`, `QCURVE` and `SFEQY`.

## SUBROUTINE XSCALE (XSTART, XSTOP, IXTYPE)

### Name

XSCALE – to specify the Cartesian horizontal scale for all 2-D plotting.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| XSTART | REAL expression | Horizontal scale value at left edge |
| XSTOP | REAL expression | Horizontal scale value at right edge |
| IXTYPE | INTEGER expression | Type of horizontal scale |

| IXTYPE | *Type of scale* |
|---|---|
| 0 | Centimetre |
| 1 | Linear |
| 2 | Logarithmic |
| 3 | Normal probability (%) |

### Description

By default, the horizontal scale is linear in centimetres starting at zero. XSCALE can be used to specify the horizontal scale of a 2-D picture, leaving the vertical scale unchanged. The scale increases in the conventional direction when $x_{start} < x_{stop}$.

XSCALE resets the underlying coordinate system to Cartesian.

The same scale is used for all relevant 2-D pictures until reset.

### Diagnostics

```
(Range >=100; linear scale used)
(Range through 0; linear scale used)
```

[1] [1]

### Default

Calling XSCALE with XSTART=XSTOP, or IXTYPE=0, restores the default (the values of all arguments are otherwise ignored).

### See also

EQSCAL, SCALES and YSCALE.

## SUBROUTINE XSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, XARR)

### Name

XSHAD – to shade the area between two contour levels from 3-D data on an $x$-specified grid.

**Availability** Section plus, released version 2-5.

### Arguments

| | | |
|---|---|---|
| ZLEV1, ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |

### Description

XSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from 3-D data (on an $x$-specified grid) held in Z2ARR; the area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min} = \min(\texttt{ZLEV1}, \texttt{ZLEV2})$ and $z_{max} = \max(\texttt{ZLEV1}, \texttt{ZLEV2})$.

SIMPLEPLOT can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

### Diagnostics

(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contour not all in range)[2]

### See also

CTLABS, CTNUMB, XSHDS, MPTYPE, SFEQY, SFMESH and SHEDGE.

## SUBROUTINE XSHDS (Z2ARR, NX, NY, XARR)

### Name

`XSHDS` – to draw a shaded contour map from 3-D data on an $x$-specified grid.

**Availability** Section plus, released version 2-5.

### Arguments

| | | |
|---|---|---|
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |
| `XARR` | `REAL` array | `NX` $x$ values of the grid |

### Description

`XSHDS` draws a shaded contour map on the current picture from 3-D data on a $x$-specified grid. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in `Z2ARR`; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see `SHEDGE`).

Simpleplot can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

### Diagnostics

```
(DATA GRID NOT MONOTONIC)1
(INVALID ARGUMENT: ARRAY SIZE)1
(Shaded contours not all in range)2
```

### See also

`CTLABS`, `CTNUMB`, `MPTYPE`, `SFEQY`, `SHEDGE`, `SHPATT`, `SQ*`, `XSHAD` and `XCNTS`.

## SUBROUTINE XSURF (Z2ARR, NX, NY, XARR)

### Name

XSURF – to start a new picture and draw a surface from 3-D data on an $x$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |

### Description

XSURF starts a new picture, and draws a surface of the data in Z2ARR.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with COORDS before calling XSURF.

XSURF uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.<br>
[2] $z < 0$, on underside of surface.<br>
[3] $z \geq 0$, top side of surface.<br>
[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of ISSHAD.

SIMPLEPLOT can only interpret data on an $x$-specified grid if the grid values, $x_i$, are supplied in increasing or decreasing order (*ie.* $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no surface is drawn.

### Diagnostics

(DATA GRID NOT MONOTONIC)$^1$<br>
(INVALID ARGUMENT: ARRAY SIZE)$^1$<br>
(Surface not all within picture)$^2$<br>
(SURFACE OMITTED: CONSTANT DATA)$^1$

### See also

IS*, ISSHAD, RIDDLE, SFEQY, SFEQZ, SFEQZD, SFLIMS, SFMESH and SFZSCL.

### REAL FUNCTION XYCALC (X, Y, Z2ARR, NX, NY, XARR, YARR)

**Name**

XYCALC – to interpolate $z = f(x, y)$ from 3-D data on an $x$-$y$ specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

The value of $z$ at $(x, y)$ is evaluated by two-way linear interpolation within the rectangular cell of the grid in which $(x, y)$ is located. If X or Y lie outside the limits of the data, XYCALC returns the current no-data value.

Simpleplot can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and the current no-data value is returned.

**Diagnostics**

(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

NODATA and QNODAT.

## SUBROUTINE XYCNTS (Z2ARR, NX, NY, XARR, YARR)

### Name

XYCNTS – to draw a contour map from 3-D data on an $x$-$y$ specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

### Description

XYCNTS draws a contour map on the current picture representing 3-D data on an $x$-$y$ specified tartan grid. Contours are drawn for equally-spaced $z$ values within the range of data held in Z2ARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by XYCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set

[2] highest level when more than one level drawn

[3] lowest level of set

[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

SIMPLEPLOT can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no curves are drawn.

### Diagnostics

```
(Contour map not all in range)²
(DATA GRID NOT MONOTONIC)¹
(INVALID ARGUMENT: ARRAY SIZE)¹
```

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, RIDDLE, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SQBRKN, SQPEN, SQZLAB and SQZVAL.

### SUBROUTINE XYCONT (ZLEV, Z2ARR, NX, NY, XARR, YARR)

**Name**

XYCONT – to draw a contour curve from 3-D data on an $x$-$y$ specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV | REAL expression | Contour level |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

XYCONT draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=ZLEV from 3-D data (on an $x$-$y$ specified tartan grid) held in Z2ARR.

SIMPLEPLOT can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no curve is drawn.

**Diagnostics**

(Contour curve not all in range)[2]
(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

CTBRKN, CTCURV, CTHOLD, CTLABS, CTNUMB, QCURVE and RIDDLE.

## SUBROUTINE XYCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, XARR, YARR)

### Name

XYCUT – to draw a 2-D curve of a surface section from 3-D data on an *x*-*y* specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of two points |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

### Description

If the surface represented by the data in Z2ARR were cut along the straight line $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the data along that line.

XYCUT draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable scales must be set up before attempting to draw a section.

SIMPLEPLOT can only interpret data on an *x*-*y* specified grid if the grid values in XARR and YARR are each supplied in increasing or decreasing order (*ie.* they are *monotonic*). If the grid values are not monotonic, a diagnostic is issued and no curve is drawn.

### Diagnostics

```
(DATA GRID NOT MONOTONIC)¹
(INVALID ARGUMENT: ARRAY SIZE)¹
(Surface section not all in range)²
```

### See also

CTBRKN, CTCURV, CTHOLD and QCURVE.

### SUBROUTINE XYSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, XARR, YARR)

**Name**

XYSHAD – to shade the area between two contour levels from 3-D data on an $x$-$y$ specified grid.

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV1,<br>ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

XYSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from 3-D data (on an $x$-$y$ specified grid) held in Z2ARR; the area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min} = \min(\texttt{ZLEV1}, \texttt{ZLEV2})$ and $z_{max} = \max(\texttt{ZLEV1}, \texttt{ZLEV2})$.

SIMPLEPLOT can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no curves are drawn.

**Diagnostics**

(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contour not all in range)[2]

**See also**

CTLABS, CTNUMB, MPTYPE, SFMESH, SHEDGE and XYSHDS.

### SUBROUTINE XYSHDS (Z2ARR, NX, NY, XARR, YARR)

**Name**

XYSHDS – to draw a shaded contour map from 3-D data on an $x$-$y$ specified grid.

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

XYSHDS draws a shaded contour map on the current picture from 3-D data on a $x$-$y$ specified grid. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in Z2ARR; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see SHEDGE).

SIMPLEPLOT can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no curves are drawn.

**Diagnostics**

(DATA GRID NOT MONOTONIC)$^1$
(INVALID ARGUMENT: ARRAY SIZE)$^1$
(Shaded contours not all in range)$^2$

**See also**

CTLABS, CTNUMB, MPTYPE, SHEDGE, SHPATT, SQ*, XYSHAD and XYCNTS.

### SUBROUTINE XYSURF (Z2ARR, NX, NY, XARR, YARR)

**Name**

XYSURF – to start a new picture and draw a surface from 3-D data on an $x$-$y$ specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| XARR | REAL array | NX $x$ values of the grid |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

XYSURF starts a new picture, and draws a surface of the data in Z2ARR.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with COORDS before calling XYSURF.

XYSURF uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.
[2] $z < 0$, on underside of surface.
[3] $z \geq 0$, top side of surface.
[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of ISSHAD.

SIMPLEPLOT can only interpret data on an $x$-$y$ specified grid if the grid values, $y_i$ and $x_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$, and $x_1 < x_2 < \ldots < x_n$ or $x_1 > x_2 > \ldots > x_n$). If the grid values are not monotonic, a diagnostic is issued and no surface is drawn.

**Diagnostics**

```
(DATA GRID NOT MONOTONIC)1
(INVALID ARGUMENT: ARRAY SIZE)1
(Surface not all within picture)2
(SURFACE OMITTED: CONSTANT DATA)1
```

**See also**

IS*, ISSHAD, RIDDLE, SFEQZ, SFEQZD, SFLIMS, SFMESH and SFZSCL.

## REAL FUNCTION YCALC (X, Y, Z2ARR, NX, NY, YARR)

### Name

`YCALC` – to interpolate $z = f(x, y)$ from 3-D data on a $y$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `X`, `Y` | `REAL` expressions | Coordinates of a point |
| `Z2ARR` | `REAL` 2-D array | Data values over a grid |
| `NX`, `NY` | `INTEGER` expressions | Number of columns and rows in `Z2ARR` |
| `YARR` | `REAL` array | `NY` $y$ values of the grid |

### Description

The value of $z$ at $(x, y)$ is evaluated by two-way linear interpolation within the rectangular cell of the grid in which $(x, y)$ is located. If `X` or `Y` lie outside the limits of the data, `YCALC` returns the current no-data value.

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and the current no-data value is returned.

### Diagnostics

`(DATA GRID NOT MONOTONIC)`[1]
`(INVALID ARGUMENT: ARRAY SIZE)`[1]

### See also

`NODATA`, `QNODAT` and `SFEQX`.

## SUBROUTINE YCNTS (Z2ARR, NX, NY, YARR)

### Name

YCNTS – to draw a contour map from 3-D data on a $y$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

### Description

YCNTS draws a contour map on the current picture representing 3-D data on a $y$-specified grid. Contours are drawn for equally-spaced $z$ values within the range of data held in Z2ARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by YCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set
[2] highest level when more than one level drawn
[3] lowest level of set
[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

### Diagnostics

(Contour map not all in range)$^2$
(DATA GRID NOT MONOTONIC)$^1$
(INVALID ARGUMENT: ARRAY SIZE)$^1$

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, RIDDLE, SFEQX, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SQBRKN, SQPEN, SQZLAB and SQZVAL.

### SUBROUTINE YCONT (ZLEV, Z2ARR, NX, NY, YARR)

**Name**

YCONT – to draw a contour curve from 3-D data on a $y$-specified grid.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV | REAL expression | Contour level |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

YCONT draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=ZLEV from 3-D data (on a $y$-specified tartan grid) held in Z2ARR.

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and no contour is drawn.

**Diagnostics**

(Contour curve not all in range)[2]
(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

CTBRKN, CTCURV, CTHOLD, CTLABS, CTNUMB, QCURVE, RIDDLE and SFEQX.

## SUBROUTINE YCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, YARR)

### Name

YCUT – to draw a 2-D curve of a surface section from 3-D data on a $y$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of two points |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

### Description

If the surface represented by the data in Z2ARR were cut along the straight line joining $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the data along that line.

YCUT draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable scales must be set up before attempting to draw a section.

Simpleplot can only interpret data on a $y$-specified grid if the grid values in YARR are supplied in increasing or decreasing order (*ie.* they are *monotonic*). If the grid values are not monotonic, a diagnostic is issued and no curve is drawn.

### Diagnostics

```
(DATA GRID NOT MONOTONIC)¹
(INVALID ARGUMENT: ARRAY SIZE)¹
(Surface section not all in range)²
```

### See also

CTBRKN, CTCURV, CTHOLD, QCURVE and SFEQX.

## SUBROUTINE YSCALE (YSTART, YSTOP, IYTYPE)

### Name

`YSCALE` – to specify the Cartesian vertical scale for all 2-D plotting.

**Availability** Section 1, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `YSTART` | `REAL` expression | $y$ scale at bottom edge |
| `YSTOP` | `REAL` expression | $y$ scale value at top edge |
| `IYTYPE` | `INTEGER` expression | Type of vertical scale |

| `IYTYPE` | *Type of scale* |
|---|---|
| 0 | Centimetre |
| 1 | Linear |
| 2 | Logarithmic |
| 3 | Normal probability (%) |

### Description

By default, the vertical scale is linear in centimetres starting at zero. `YSCALE` can be used to specify the vertical scale of a 2-D picture, leaving the horizontal scale unchanged. The scale increases in the conventional direction when $y_{start} < y_{stop}$.

`YSCALE` resets the underlying coordinate system to Cartesian.

The same scale is used for all relevant 2-D pictures until reset.

### Diagnostics

```
(Range >=100; linear scale used)
(Range through 0; linear scale used)
```

(both lines carry superscript [1])

### Default

Calling `YSCALE` with `YSTART`=`YSTOP`, or `IYTYPE`=0, restores the default (the values of all arguments are otherwise ignored).

### See also

`EQSCAL`, `SCALES` and `XSCALE`.

## SUBROUTINE YSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, YARR)

### Name

YSHAD – to shade the area between two contour levels from 3-D data on a $y$-specified grid.

**Availability** Section plus, released version 2-5.

### Arguments

| | | |
|---|---|---|
| ZLEV1,<br>ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

### Description

YSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from 3-D data (on a $y$-specified grid) held in Z2ARR; the area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min} = \min(\texttt{ZLEV1}, \texttt{ZLEV2})$ and $z_{max} = \max(\texttt{ZLEV1}, \texttt{ZLEV2})$.

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

### Diagnostics

(DATA GRID NOT MONOTONIC)[1]
(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contour not all in range)[2]

### See also

CTLABS, CTNUMB, MPTYPE, SFEQX, SFMESH, SHEDGE and YSHDS.

### SUBROUTINE YSHDS (Z2ARR, NX, NY, YARR)

**Name**

YSHDS – to draw a shaded contour map from 3-D data on a $y$-specified grid.

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

**Description**

YSHDS draws a shaded contour map on the current picture from 3-D data on a $y$-specified grid. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in Z2ARR; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see SHEDGE).

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and no contours are drawn.

**Diagnostics**

(DATA GRID NOT MONOTONIC)[1]<br>
(INVALID ARGUMENT: ARRAY SIZE)[1]<br>
(Shaded contours not all in range)[2]

**See also**

CTLABS, CTNUMB, MPTYPE, SFEQX, SHEDGE, SHPATT, SQ*, YSHAD and YCNTS.

## SUBROUTINE YSURF (Z2ARR, NX, NY, YARR)

### Name

YSURF – to start a new picture and draw a surface from 3-D data on a $y$-specified grid.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| Z2ARR | REAL 2-D array | Data values over a grid |
| NX, NY | INTEGER expressions | Number of columns and rows in Z2ARR |
| YARR | REAL array | NY $y$ values of the grid |

### Description

YSURF starts a new picture, and draws a surface of the data in ZARR.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with COORDS before calling YSURF.

YSURF uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.<br>
[2] $z < 0$, on underside of surface.<br>
[3] $z \geq 0$, top side of surface.<br>
[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of ISSHAD.

Simpleplot can only interpret data on a $y$-specified grid if the grid values, $y_i$, are supplied in increasing or decreasing order (*ie.* $y_1 < y_2 < \ldots < y_n$ or $y_1 > y_2 > \ldots > y_n$). If the grid values are not monotonic, a diagnostic is issued and no surface is drawn.

### Diagnostics

```
(DATA GRID NOT MONOTONIC)1
(INVALID ARGUMENT: ARRAY SIZE)1
(Surface not all within picture)2
(SURFACE OMITTED: CONSTANT DATA)1
```

### See also

IS*, RIDDLE, SFEQX, SFEQZ, SFEQZD, SFLIMS, SFMESH and SFZSCL.

### REAL FUNCTION ZCALC (X, Y, XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZCALC – to interpolate $z = f(x, y)$ from ungridded 3-D data (without neighbours).

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point |
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values<br>in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

The value of $z$ at $(x, y)$ is interpolated on a plane triangle across 3 data points surrounding $(x, y)$. The data points are selected from the the ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR.

If NNODES is greater than 3, an extra point in each element is generated by taking the mean of each of the coordinates of the data points of the element, and triangular sub-elements are constructed by joining each of the nodes to this extra point.

If X or Y lies outside the limits of the data, ZCALC returns the current no-data value.

Before using ZCALC, the element array I2ARR(NNODES,NELEMS) must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZCALC is more efficient than ZCALC.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

NODATA, QNODAT, ZORDEN, ZORDER and ZZCALC.

**SUBROUTINE ZCNTS (XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)**

### Name

ZCNTS – to draw a contour map from ungridded 3-D data (without neighbours).

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values<br>in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

### Description

ZCNTS draws a contour map on the current picture representing ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR. Contours are drawn for equally-spaced $z$ values within the range of data held in ZARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by ZCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set
[2] highest level when more than one level drawn
[3] lowest level of set
[4] negative and zero levels except lowest and highest of set

CTBRKN overrides the bundled line-drawing attributes of pens associated with the four pen pointers.

Before using ZCNTS the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZCNTS is more efficient than ZCNTS.

### Diagnostics

(Contour map not all in range)$^{2}$
(INVALID ARGUMENT: ARRAY SIZE)$^{1}$

### See also

CTBRKN, CTCURV, CTLABS, CTNUMB, SFEQZ, SFLIMS, SQBRKN, SQPEN, SQZLAB, SQZVAL, ZORDEN, ZORDER and ZZCNTS.

### SUBROUTINE ZCONT (ZLEV, XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZCONT – to draw a contour curve from ungridded 3-D data (without neighbours).

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV | REAL expression | Contour level |
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values<br>in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZCONT draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=ZLEV from the ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR.

Before using ZCONT the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZCONT is more efficient than ZCONT.

**Diagnostics**

```
(Contour curve not all in range)2
(INVALID ARGUMENT: ARRAY SIZE)1
```

**See also**

CTBRKN, CTCURV, CTHOLD, CTLABS, CTNUMB, QCURVE, RIDDLE, ZORDEN, ZORDER and ZZCONT.

### SUBROUTINE ZCUT (X1, Y1, X2, Y2, XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

## Name

ZCUT – to draw a 2-D curve of a surface section from ungridded 3-D data (without neighbours).

**Availability** Section 2, released before version 2-5.

## Arguments

| | | |
|---|---|---|
| X1, Y1, X2, Y2 | REAL expressions | Coordinates of two points |
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

## Description

If the surface represented by the 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR, was cut along the straight line from $(x_1, y_1)$ to $(x_2, y_2)$, the side view of the section would show as a two-dimensional curve of the variation of the data along that line.

ZCUT draws [1] the curve of surface height *vs.* distance from $(x_1, y_1)$ on the current picture – suitable 2-D scales must be set up before attempting to draw a section.

Before using ZCUT the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

## Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Surface section not all in range)[2]

## See also

CTBRKN, CTCURV, CTHOLD, QCURVE, ZORDEN and ZORDER.

### SUBROUTINE ZEDGE (XARR, YARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZEDGE – to draw around the boundary of the data area covered by a set of elements (without neighbours).

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZEDGE searches the $(x, y)$ coordinates held in parallel arrays, XARR and YARR, and draws [1] straight lines between all pairs of nodes which are adjacent in only one element. These straight lines constitute the boundary of the area in which ZCNTS and ZCONT can draw contours with this element structure.

Before using ZEDGE the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZEDGE is more efficient than ZEDGE.

**Diagnostics**

(Element boundary not all in range)[2]
(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

CTBRKN, CTHOLD, QCURVE, ZORDEN, ZORDER and ZZEDGE.

### SUBROUTINE ZELEM (JELE, XARR, YARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZELEM – to draw a single area element.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| JELE | INTEGER expression | Element number to be drawn |
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZELEM draws [1] the boundary of the JELEth area element on the current 2-D picture.

Before using ZELEM, the element array I2ARR(NNODES,NELEMS) must be set up; ZORDER or ZORDEN can be used for this purpose.

If any part of the element is off the current picture scales, this part is omitted and a diagnostic is issued.

**Diagnostics**

(Element not all in range)[2]<br>
(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

CTBRKN, ZELEMS, ZNUMB, ZORDEN, ZORDER and ZZELMS.

### SUBROUTINE ZELEMS (XARR, YARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZELEMS – to draw the configuration of a set of area elements (without neighbours).

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZELEMS draws [1] the boundaries of all the area elements on the current 2-D picture. The elements represent the 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR.

Before using ZELEMS the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

If any part of the element is off the current picture scales, this part is omitted and a diagnostic is issued.

ZZELMS is more efficient than ZELEMS.

**Diagnostics**

```
(Elements not all in range)²
(INVALID ARGUMENT: ARRAY SIZE)¹
```

**See also**

CTBRKN, CTHOLD, QCURVE, ZELEM, ZORDEN, ZORDER and ZZELMS.

## SUBROUTINE ZNEIGH (I2ARR, N2ARR, NNODES, NELEMS)

### Name

ZNEIGH – to set up an array of neighbours from an array of elements.

**Availability** Section 2, released version 2-5.

### Arguments

| | | |
|---|---|---|
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | To receive element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

### Description

Some computations with ungridded data are slow because of the need to search repeatedly through the elements. Considerable savings in searching time are made possible by the introduction of an additional INTEGER array, the array of neighbours, with the same dimensions as the array of elements. ZNEIGH sets up an array of neighbours N2ARR(NNODES,NELEMS,f)rom the array of elements I2ARR(NNODES,NELEMS).

For each subroutine, Z*, operating on ungridded $(x, y, z)$ coordinates, a more efficient ZZ* subroutine is available; each ZZ* subroutine has the array N2ARR in addition to the arguments of the corresponding ZZ* subroutine.

If an array of elements needs to be set up as well as an array of neighbours, ZZORDR (or ZZORDN) is more efficient than using ZORDER (or ZORDEN) and ZNEIGH separately.

Memory for N2ARR is allocated internally, and must be freed using SPCA_free.

### Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]

### See also

ZZORDN and ZZORDR.

## SUBROUTINE ZNUMB (NTORF, ETORF)

### Name

`ZNUMB` – to specify whether nodes and/or elements are to be numbered.

**Availability** Section 2, released before version 2-5.

### Arguments

| | | |
|---|---|---|
| `NTORF` | `LOGICAL` expression | Whether to number nodes |
| `ETORF` | `LOGICAL` expression | Whether to number elements |

### Description

`ZNUMB` modifies the actions of subsequent calls to `ZELEM`, `ZELEMS` or `ZZELMS` – in addition to drawing element boundaries, they will also label each node drawn with the node number (*ie.* its position in `XARR` and `YARR`) if `NTORF` = `.TRUE.`, and/or label the centre of each element drawn with the element number (*ie.* its position in the element array) if `ETORF` = `.TRUE.`.

### Default

`CALL ZNUMB(.FALSE.,.FALSE.)` restores the default.

### See also

`LABANG`, `ZELEM`, `ZELEMS` and `ZZELMS`.

## SUBROUTINE ZORDEN (XARR, YARR, NPTS, I2ARR, NVAR, ISIZE)

### Name

`ZORDEN` – to reconfigure $(x, y)$ coordinates into triangular area elements using normalized values.

**Availability** Section 2, released version 2-9.

### Arguments

| | | |
|---|---|---|
| `XARR`, `YARR` | `REAL` arrays | $(x, y)$ coordinates of data values in parallel arrays |
| `NPTS` | `INTEGER` expression | Number of data points |
| `I2ARR` | `INTEGER` 2-D array | To receive data element structure, `I2ARR(3,ISIZE)` |
| `NVAR` | `INTEGER` variable | To receive number of elements |
| `ISIZE` | `INTEGER` expression | second dimension of `I2ARR` |

### Description

`ZORDEN` organizes a set of $(x, y)$ coordinates, held in parallel arrays `XARR` and `YARR`, into a structure of non-overlapping triangular area elements covering the data, in a similar way to `ZORDER`. Unlike `ZORDER`, `ZORDEN` allows for dissimilar orders of magnitude in $x$ and $y$, by triangulating $x$ and $y$ values normalized over identical ranges; the values in the arrays are not altered.

The number of triangles generated is placed into `NVAR`. If `NVAR` reaches `ISIZE` before the process is complete, triangulation is incomplete – a diagnostic is issued and `I2ARR` is filled with the partial structure and `NVAR`=`ISIZE`.

Memory for `I2ARR` is allocated internally, and must be freed using `SPCA_free`.

### Diagnostics

`(INVALID ARGUMENT: ARRAY SIZE)`[1]
`(Triangulation data duplicated)`[1]
`(Triangulation incomplete)`[1]
`(Triangulation failed)`[1]
`(Triangulation impossible)`[1]

### See also

`ZORDER`, `ZZORDN` and `ZZORDR`.

### SUBROUTINE ZORDER (XARR, YARR, NPTS, I2ARR, NVAR, ISIZE)

**Name**

ZORDER – to reconfigure $(x, y)$ coordinates into triangular area elements.

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | To receive data element structure,<br>I2ARR(3,ISIZE) |
| NVAR | INTEGER variable | To receive number of elements |
| ISIZE | INTEGER expression | second dimension of I2ARR |

**Description**

ZORDER organizes a set of $(x, y)$ coordinates, held in parallel arrays XARR and YARR, into a structure of non-overlapping triangular area elements covering the data; three node numbers (*ie.* their positions in XARR and YARR) for each triangle are placed in I2ARR.

For example, I2ARR$(1, n) = i$, I2ARR$(2, n) = j$, I2ARR$(3, n) = k$, means that the $n$th triangle consists of (XARR$(i)$, YARR$(i)$), (XARR$(j)$, YARR$(j)$) and (XARR$(k)$, YARR$(k)$).

The number of triangles generated is placed into NVAR. If NVAR reaches ISIZE before the process is complete, triangulation is incomplete – a diagnostic is issued and I2ARR is filled with the partial structure and NVAR=ISIZE.

Memory for I2ARR is allocated internally, and must be freed using SPCA_free.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Triangulation data duplicated)[1]
(Triangulation incomplete)[1]
(Triangulation failed)[1]
(Triangulation impossible)[1]

**See also**

ZORDEN, ZZORDN and ZZORDR.

### SUBROUTINE ZSHAD (ZLEV1, ZLEV2, ISHADE, XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

## Name

ZSHAD – to shade the area between two contour levels from ungridded 3-D data (without neighbours).

**Availability** Section plus, released version 2-5.

## Arguments

| | | |
|---|---|---|
| ZLEV1, ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

## Description

ZSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR. The area is drawn on the current picture using shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min} = \min(\text{ZLEV1}, \text{ZLEV2})$ and $z_{max} = \max(\text{ZLEV1}, \text{ZLEV2})$.

Before using ZSHAD the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZSHAD is more efficient than ZSHAD.

## Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contour not all in range)[2]

## See also

CTLABS, CTNUMB, MPTYPE, SFMESH, SHEDGE, ZORDEN, ZORDER, ZSHDS and ZZSHAD.

### SUBROUTINE ZSHDS (XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

ZSHDS – to draw a shaded contour map from ungridded 3-D data (without neighbours).

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values<br>in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZSHDS draws a shaded contour map on the current picture from ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in ZARR; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see SHEDGE).

Before using ZSHDS the element array, I2ARR(NNODES,NELEMS), must be set up; ZORDER or ZORDEN can be used for this purpose.

ZZSHDS is more efficient than ZSHDS.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contours not all in range)[2]

**See also**

CTLABS, CTNUMB, MPTYPE, SHEDGE, SHPATT, SQ*, ZCNTS, ZORDEN, ZORDER, ZSHAD and ZZSHDS.

### SUBROUTINE ZSURF (XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

**Name**

`ZSURF` – to start a new picture and draw a surface from ungridded 3-D data (without neighbours).

**Availability** Section 2, released before version 2-5.

**Arguments**

| | | |
|---|---|---|
| `XARR`, `YARR`, `ZARR` | `REAL` arrays | $(x, y, z)$ coordinates of data values |
| `NPTS` | `INTEGER` expression | Number of data points |
| `I2ARR` | `INTEGER` 2-D array | Data element structure |
| `NNODES` | `INTEGER` expression | Number of nodes per element |
| `NELEMS` | `INTEGER` expression | Number of elements |

**Description**

`ZSURF` starts a new picture, and draws a surface representing the ungridded data (without neighbours) held in parallel arrays, `XARR`, `YARR` and `ZARR`.

In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with `COORDS` before calling `ZSURF`.

`ZSURF` uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.
[2] $z < 0$, on underside of surface.
[3] $z \geq 0$, top side of surface.
[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of `ISSHAD`.

Before using `ZSURF` the element array, `I2ARR(NNODES,NELEMS)`, must be set up; `ZORDER` or `ZORDEN` can be used for this purpose.

`ZZSURF` is more efficient than `ZSURF`.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Surface not all within picture)[2]
(SURFACE OMITTED: CONSTANT DATA)[1]

**See also**

`IS*`, `RIDDLE`, `SFEQZ`, `SFEQZD`, `SFLIMS`, `SFMESH`, `SFZSCL`, `ZORDEN`, `ZORDER` and `ZZSURF`.

### REAL FUNCTION ZZCALC (X, Y, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZCALC – to interpolate $z = f(x, y)$ from ungridded 3-D data (with neighbours).

**Availability** Section 2, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| X, Y | REAL expressions | Coordinates of a point |
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

The value of $z$ at $(x, y)$ is interpolated on a plane triangle across 3 data points surrounding $(x, y)$. The data points are selected from the the ungridded 3-D data (without neighbours) held in parallel arrays, XARR, YARR and ZARR.

If NNODES is greater than 3, an extra point in each element is generated by taking the mean of each of the coordinates of the data points of the element, and triangular sub-elements are constructed by joining each of the nodes to this extra point.

If X or Y lies outside the limits of the data, ZZCALC returns the current no-data value.

Before using ZZCALC the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZCALC is more efficient than ZCALC.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]

**See also**

NODATA, QNODAT, ZZORDN and ZZORDR.

### SUBROUTINE ZZCNTS (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZCNTS – to draw contours from ungridded 3-D data (with neighbours).

**Availability** Section 2, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZZCNTS draws a contour map on the current picture representing ungridded 3-D data (with neighbours) held in parallel arrays, XARR, YARR and ZARR. Contours are drawn for equally-spaced $z$ values within the range of data held in ZARR. By default, an interval is allocated to give between 5 and 10 contour levels over the range.

The contours drawn by ZZCNTS use the four pen pointers as follows:

[1] positive levels except lowest and highest of set
[2] highest level when more than one level drawn
[3] lowest level of set
[4] negative and zero levels except lowest and highest of set

The element data structures specified by I2ARR and N2ARR may be created by ZZORDR or ZZORDN if required.

ZZCNTS is more efficient than ZCNTS.

**Diagnostics**

(Contour map not all in range)$^2$
(INVALID ARGUMENT: ARRAY SIZE)$^1$

**See also**

CTBRKN, CTCURV, CTLABS, CTNUMB, SFEQZ, SFEQZD, SFLIMS, SFZSCL, SQBRKN, SQPEN, SQZLAB, SQZVAL, ZZORDN, ZZORDR and ZZSHDS.

## SUBROUTINE ZZCONT (ZLEV, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

### Name

ZZCONT – to draw a contour curve from ungridded 3-D data (with neighbours).

**Availability** Section 2, released version 2-5.

### Arguments

| | | |
|---|---|---|
| ZLEV | REAL expression | Contour level |
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

### Description

ZZCONT draws [1] a contour curve on the current picture. The contour curve corresponds to $z$=ZLEV from the ungridded 3-D data (with neighbours) held in parallel arrays, XARR, YARR and ZARR

Before using ZZCONT the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZCONT is more efficient than ZCONT.

### Diagnostics

```
(Contour curve not all in range)²
(INVALID ARGUMENT: ARRAY SIZE)¹
```

### See also

CTBRKN, CTCURV, CTHOLD, CTLABS, CTNUMB, QCURVE, RIDDLE, ZZORDN and ZZORDR.

### SUBROUTINE ZZEDGE (XARR, YARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZEDGE – to draw around the boundary of the data area covered by a set of elements (with neighbours).

**Availability** Section 2, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZZEDGE searches the $(x, y)$ coordinates held in parallel arrays, XARR and YARR, and draws [1] straight lines between all pairs of nodes which are adjacent in only one element. These straight lines constitute the boundary of the area in which ZZCNTS and ZZCONT can draw contours with this element structure.

Before using ZZEDGE the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZEDGE is more efficient than ZEDGE.

**Diagnostics**

```
(Element boundary not all in range)2
(INVALID ARGUMENT: ARRAY SIZE)1
```

**See also**

CTBRKN, CTHOLD, QCURVE, ZZORDN and ZZORDR.

### SUBROUTINE ZZELMS (XARR, YARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZELMS – to draw the configuration of a set of area elements (with neighbours).

**Availability** Section 2, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZZELMS draws [1] the boundaries of all the area elements on the current 2-D picture. The elements represent the 3-D data (with neighbours) held in parallel arrays, XARR, YARR and ZARR.

Before using ZZELMS the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZELMS is more efficient than ZELEMS.

**Diagnostics**

(Elements not all in range)$^2$
(INVALID ARGUMENT: ARRAY SIZE)$^1$

**See also**

CTBRKN, CTHOLD, QCURVE, ZELEM, ZNUMB, ZZORDN and ZZORDR.

### SUBROUTINE ZZORDN (XARR, YARR, NPTS, I2ARR, N2ARR, NVAR, ISIZE)

## Name

ZZORDN – to reconfigure $(x, y)$ coordinates into triangular elements and an array of neighbours (using normalized values).

**Availability** Section 2, released version 2-9.

## Arguments

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values in parallel arrays |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | To receive data element structure, I2ARR(3,ISIZE) |
| N2ARR | INTEGER 2-D array | To receive neighbour array, N2ARR(3,ISIZE) |
| NVAR | INTEGER variable | To receive number of elements |
| ISIZE | INTEGER expression | second dimension of I2ARR and N2ARR |

## Description

ZZORDN organizes a set of $(x, y)$ coordinates, held in parallel arrays XARR and YARR, into a structure of non-overlapping triangular area elements covering the data. The element numbers of neighbouring elements are placed into N2ARR for use by the ZZ* subroutines.

Unlike ZZORDR, ZZORDN allows for dissimilar orders of magnitude in $x$ and $y$, by triangulating $x$ and $y$ values normalized over identical ranges.

If NVAR reaches ISIZE before completion, a diagnostic is issued, I2ARR is filled with the partial structure and NVAR=ISIZE.

## Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Triangulation data duplicated)[1]
(Triangulation incomplete)[1]
(Triangulation failed)[1]
(Triangulation impossible)[1]

## See also

ZNEIGH, ZORDEN, ZORDER and ZZORDR.

## SUBROUTINE ZZORDR (XARR, YARR, NPTS, I2ARR, N2ARR, NVAR, ISIZE)

### Name

ZZORDR – to reconfigure $(x, y)$ coordinates into triangular elements and an array of neighbours.

**Availability** Section 2, released version 2-5.

### Arguments

| | | |
|---|---|---|
| XARR, YARR | REAL arrays | $(x, y)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | To receive data element structure, I2ARR(3,ISIZE) |
| N2ARR | INTEGER 2-D array | To receive neighbour array, N2ARR(3,ISIZE) |
| NVAR | INTEGER variable | To receive number of elements |
| ISIZE | INTEGER expression | second dimension of I2ARR and N2ARR |

### Description

ZZORDR organizes a set of $(x, y)$ coordinates, held in parallel arrays XARR and YARR, into a structure of non-overlapping triangular area elements covering the data. Three node numbers (*ie.* their positions in XARR and YARR) for each triangle are placed in I2ARR. At the same time, the element numbers of neighbouring elements are placed in N2ARR for use by the ZZ* subroutines.

If NVAR reaches ISIZE before the process is complete, a diagnostic is issued, I2ARR is filled with the partial structure and NVAR=ISIZE.

### Diagnostics

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Triangulation data duplicated)[1]
(Triangulation incomplete)[1]
(Triangulation failed)[1]
(Triangulation impossible)[1]

### See also

ZNEIGH, ZORDEN, ZORDER and ZZORDN.

### SUBROUTINE ZZSHAD (ZLEV1, ZLEV2, ISHADE, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZSHAD – to shade the area between two contour levels from ungridded 3-D data (with neighbours).

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| ZLEV1, ZLEV2 | REAL expressions | Contour levels |
| ISHADE | INTEGER expression | Shading pattern number |
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZZSHAD shades the area between contour levels $z$=ZLEV1 and ZLEV2 from ungridded 3-D data (with neighbours) held in parallel arrays, XARR, YARR and ZARR. The area is filled with shading pattern ISHADE. By default, the contours themselves are not drawn (see SHEDGE).

If data values, $z$, coincide with contour levels, only those equal to $z_{min}$ are included within the shaded region, *ie.* $z_{min} \leq z < z_{max}$, where $z_{min}$ = min(ZLEV1, ZLEV2) and $z_{max}$ = max(ZLEV1, ZLEV2).

Before using ZZSHAD the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZSHAD is more efficient than ZSHAD.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)$^1$
(Shaded contour not all in range)$^2$

**See also**

CTLABS, CTNUMB, MPTYPE, SFMESH, SHEDGE, ZZORDN, ZZORDR and ZZSHDS.

### SUBROUTINE ZZSHDS (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

**Name**

ZZSHDS – to draw a shaded contour map from ungridded 3-D data (with neighbours).

**Availability** Section plus, released version 2-5.

**Arguments**

| | | |
|---|---|---|
| XARR, YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

**Description**

ZZSHDS draws a shaded contour map on the current picture from ungridded 3-D data (with neighbours) held in parallel arrays, XARR, YARR and ZARR. By default, the contour curves are drawn at equally-spaced $z$ values using an interval chosen to give between 5 and 10 contours levels over the range of data held in ZARR; each of the contour intervals is shaded with a sequence of shading patterns; the contours themselves are not drawn (see SHEDGE).

Before using ZZSHDS the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose.

ZZSHDS is more efficient than ZSHDS.

**Diagnostics**

(INVALID ARGUMENT: ARRAY SIZE)[1]
(Shaded contours not all in range)[2]

**See also**

CTLABS, CTNUMB, MPTYPE, SHEDGE, SHPATT, SQ*, ZZCNTS, ZZORDN, ZZORDR and ZZSHAD.

## SUBROUTINE ZZSURF (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

### Name

ZZSURF – to start a new picture and draw a surface from ungridded 3-D data (with neighbours).

**Availability** Section 2, released version 2-5.

### Arguments

| | | |
|---|---|---|
| XARR,<br>YARR, ZARR | REAL arrays | $(x, y, z)$ coordinates of data values |
| NPTS | INTEGER expression | Number of data points |
| I2ARR | INTEGER 2-D array | Data element structure |
| N2ARR | INTEGER 2-D array | Element numbers of neighbours |
| NNODES | INTEGER expression | Number of nodes per element |
| NELEMS | INTEGER expression | Number of elements |

### Description

ZZSURF starts a new picture, and draws a surface representing the ungridded data (with neighbours) held in parallel arrays, XARR, YARR and ZARR. In order to draw a surface picture of polar data, $z = f(r, \theta)$, specify polar units with COORDS before calling ZZSURF.

ZZSURF uses the four pens pointers as follows:

[1] $z \geq 0$, on underside of surface.
[2] $z < 0$, on underside of surface.
[3] $z \geq 0$, top side of surface.
[4] $z < 0$, on top side of surface.

A shaded surface can be drawn according to a prior call of ISSHAD.

Before using ZZSURF the element array, I2ARR(NNODES,NELEMS), and the neighbour array, N2ARR(NNODES,NELEMS), must be set up; ZZORDR or ZZORDN can be used for this purpose. ZZSURF is more efficient than ZSURF.

### Diagnostics

```
(INVALID ARGUMENT: ARRAY SIZE)¹
(Surface not all within picture)²
(SURFACE OMITTED: CONSTANT DATA)¹
```

### See also

IS*, RIDDLE, SFEQZ, SFEQZD, SFLIMS, SFMESH, SFZSCL, ZZORDN and ZZORDR.

# Index