A COMPUTER GRAPHICS PACKAGE

by

C RICHARDSON

AD-220 907

UNCLASSIFIED UNLIMITED

ARE TM (UDT)-89111
ACCN. No. 79353
AUGUST 1989
COPY No. 39

ADARALTY RESEARCH ESTABLISHMENT
Procurement Executive, Ministry of Defence,
Southwell, Portland, Dorset.

UNCLASSIFIED UNLIMITED
90 04 25 014
CONDITIONS OF RELEASE

1. This information is released by the UK Government to the recipient Government for defence purposes only.
2. This information must be accorded the same degree of security protection as that accorded thereto by the UK Government.
3. This information may be disclosed only within the Defence Departments of the recipient Government and to its Defence Contractors within its own territory, except as otherwise authorised by the Ministry of Defence. Such recipients shall be required to accept the information on the same conditions as the recipient Government.
4. This information may be subject to privately owned rights.

AMENDMENTS

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reports quoted are not necessarily available to members of the public or to commercial organisations.
A Computer Graphics Package

by

C. Richardson

Controller HMSO London, 1989
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>1. BACKGROUND</td>
<td>2</td>
</tr>
<tr>
<td>2. INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>3. COORDINATE SYSTEMS</td>
<td>3</td>
</tr>
<tr>
<td>4. DEFAULT CONDITIONS</td>
<td>3</td>
</tr>
<tr>
<td>5. SUBROUTINE DESCRIPTIONS</td>
<td>4</td>
</tr>
<tr>
<td>6. METHOD OF USE</td>
<td>10</td>
</tr>
</tbody>
</table>

**Annexes**

<table>
<thead>
<tr>
<th>Annex</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>EXAMPLES</td>
<td>11</td>
</tr>
<tr>
<td>B.</td>
<td>SUMMARY OF SUBROUTINES</td>
<td>45</td>
</tr>
<tr>
<td>C.</td>
<td>FILE CONTENTS SUMMARY</td>
<td>47</td>
</tr>
</tbody>
</table>

## FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A Graphics Example Showing ( \sin(x) ) And ( \cos(x) )</td>
<td>13</td>
</tr>
<tr>
<td>2.</td>
<td>A Graphics Example Showing ( e^{-</td>
<td>x</td>
</tr>
<tr>
<td>3.</td>
<td>Example Of Drawing A Filled Polygon</td>
<td>18</td>
</tr>
<tr>
<td>4.</td>
<td>Example Using Variable Area Fill</td>
<td>21</td>
</tr>
<tr>
<td>5.</td>
<td>Contour Plot Of The Function ( x^2 + y^2 + z^2 = 10^2 )</td>
<td>24</td>
</tr>
<tr>
<td>6.</td>
<td>Surface Plot Of The Function ( x^2 + y^2 + z^2 = 10^2 )</td>
<td>27</td>
</tr>
<tr>
<td>7.</td>
<td>Example Using Greek Characters</td>
<td>29</td>
</tr>
<tr>
<td>8a.</td>
<td>Example Using Grey Shading, Pattern 0</td>
<td>32</td>
</tr>
<tr>
<td>8b.</td>
<td>&quot; &quot; &quot; &quot; &quot; &quot; &quot; 1</td>
<td>33</td>
</tr>
<tr>
<td>8c.</td>
<td>&quot; &quot; &quot; &quot; &quot; &quot; 2</td>
<td>34</td>
</tr>
<tr>
<td>8d.</td>
<td>&quot; &quot; &quot; &quot; &quot; &quot; 3</td>
<td>35</td>
</tr>
<tr>
<td>8e.</td>
<td>&quot; &quot; &quot; &quot; &quot; &quot; 4</td>
<td>36</td>
</tr>
<tr>
<td>9.</td>
<td>Example Showing A Shaded Sphere</td>
<td>40</td>
</tr>
<tr>
<td>10.</td>
<td>Example Of The Use Of Virtual Displays And Windows</td>
<td>44</td>
</tr>
</tbody>
</table>
A Computer Graphics Package

ABSTRACT

1. A general purpose graphics package is described. The package is an updated version of a graphics library first developed in 1964 on an ICL-1900 series computer (described in AUWE Acc 35647) and in use continuously since that time with various modifications and informal documentation improvements. It is currently implemented on a DEC-MicroVAX computer and various hardware devices including the HP series of plotters (using HPGL), Regis devices (VT125, VT340), DEC Workstation, LNO3 laser printer, and certain Ramtek, Sigma and Tektronix devices.

2. Although commercial graphics packages are available this one is simple to understand and use, and together with its familiarity to existing users continues to provide useful graphics support.

3. Examples are given in order to clarify usage of the software and lists of subroutines are also provided.
1 BACKGROUND

1.1 Recent advances in computer technology and software have dramatically changed the computing scene. Before WIMP's computers were used mainly by skilled operators or in rather dedicated applications. Not so long ago suitable software for a particular application was unlikely to be available and computer applications relied on special purpose software being developed.

1.2 In scientific research printed output soon acquired a low reputation. Computer solutions to complex problems frequently produced vast quantities of printed numbers and data was transcribed to graphical form by hand. The graph package described here was first developed in about 1969 (ref: AUWE Acc 35647) for use with a Computer Instrumentation Ltd incremental plotter. Under computer control this machine could do nothing more than move the pen one increment in one of eight directions or raise or lower the pen. Because it was such a basic machine, software was developed so that scientific applications could produce good quality graphics. Features included font designs with rounded characters (unlike the poor quality angular characters which were available), text scaling and rotation, curve fitting to allow cusps to be drawn automatically, and indeed most of the basic features currently available. The package has served the needs of scientific graph plotting for many years and is still in use, including descendant versions adapted by various users at ARE.

1.3 Computer graphics has become a very important and sophisticated subject but has many different applications, including computer art and simulation. Even scientific applications may involve complicated graphics which are not suitable for general purpose use. Sophisticated graph packages are now commercially available and have the advantage of providing a commercially maintained standard, such as GKS. However, this package continues to be attractive because it is simple to use, it is familiar and can be easily modified or extended to cater for different applications or new hardware.

1.4 This report marks a new generation of the software for compatibility with the window environment of a VAX Workstation. Some changes have been incorporated but these have minor implications for existing programs and mainly deal with the Workstation aspects and additional routines.

2 INTRODUCTION

2.1 In scientific research graphical output from computer programs is essential and can be easily obtained using the subroutines described. This graphics package provides comprehensive facilities for two and three dimensional work and is largely device independent. The subroutines can be considered in three groups as follows:

(a) The main routines which actually create a picture;
(b) The supporting routines which modify the actual behaviour of the main routines;
(c) Device dependent routines which are not normally used directly by the application but are available if required;
(d) Special routines which are not intended for general use.

2.2 For simplicity, parameter lists have been kept to a minimum, and consequently the action of the main drawing subroutines depends on 'hidden' parameter values. These hidden parameters are given default values, but may be adjusted by means of the supporting subroutines. For example the subroutine PLOT_TEXT('text',N) plots text using the default character size, but this may be changed by the supporting subroutine CHAR_SIZE(Width,Height).
3 COORDINATE SYSTEMS

3.1 Assume a world cartesian coordinate system \((X,Y)\) is defined on some imaginary drawing surface such as an A4 sheet of paper with its origin at the lower left hand corner and the X-axis horizontal. The units of both \(X\) and \(Y\) are mm. A local cartesian coordinate system \((U,V)\) may be defined with its origin at \(X=X_{\text{shift}}, Y=Y_{\text{shift}}\) (see ORIGIN) and scaled such that one unit of \(U\) is \(X_{\text{scale}}\) mm., and similarly for \(Y\) (see SCALE). Thus:

\[
X = U \times X_{\text{scale}} + X_{\text{shift}} \text{ mm.}
\]

\[
Y = V \times Y_{\text{scale}} + Y_{\text{shift}} \text{ mm.}
\]

3.2 Initially these two systems (world and local) are identical and mapped to the actual display device assuming the drawing surface is A4 landscape in size and orientation. With an HP7750 plotter for example the mapping is to true scale, but on a VT240 some additional scaling is necessary in order to represent the A4 size on the screen. For simple applications it is useful to imagine the graphical output is to be produced on an A4 sheet of paper in landscape or portrait orientation. However, the routine PLOT WINDOW enables the world coordinate system to be modified and PLOT VIEW modifies how the world coordinates are mapped to the physical display device, thus enabling large graphics surfaces to be created. Workstation graphics may create up to 4 virtual displays with 4 windows in each. Although mapping a graph to a display device is defined in terms of a window to a virtual display and a viewport on the device, the full windows environment is not supported on some devices.

3.3 Valid physical graphics devices are recognised by the following mnemonics:

- GF: A pseudo device which saves a file copy of the graphics output.
- RG: A Ramtek series 9000
- VG: A VT340 (or VT125 with some restrictions)
- WG: A Workstation with UIS support
- HP: A Hewlett Packard plotter recognising HPGL

Any hardware combinations can be specified by means of these mnemonics and additionally the mnemonic TT can be used to represent either VG or WG determined by the login device. By default the software automatically selects the appropriate display device you are logged in to (if it is recognised as a valid graphics device) and a hard copy file is produced (See PLOT_INIT). A keyboard and cursor is associated with the first interactive device selected.

3.4 Special facilities exist for 3D plotting and are described in greater detail later.

4 DEFAULT CONDITIONS

4.1 Initialisation (see PLOT_INIT) resets certain parameters to default values as follows:

- ORIGIN = 0, 0
- SCALE = 1, 1
- CHAR SIZE = 2.4, 3.0
- CHAR ANGLE = 0.0
- CHAR SLOPE = 0.0
- LINE TYPE = 0, 0.0
- SELECT PEN = 1
- MOVE = 0, 0

Various other parameters are reset by device specific initialisation routines (see DEVICE SPECIFIC SUBROUTINES).
5 SUBROUTINE DESCRIPTIONS

5.1 Since the various graphics devices have different capabilities, it is not always possible to ensure that general purpose routines behave identically on the different devices. In particular, character drawing will be subject to resolution differences. Consequently, it may be necessary to make some compromise over the style of graph for a given device. Alternatively different devices may be turned off and on to allow different text styles to be used. However, the following general purpose routines provide a high level of device independence, where the parameter type specifications are INTEGER*4 (if starting with the letters I,J,K,L,M,N) or REAL*4 or as specified in the description:

A4_BOX(I)
- Draw corner marks to define an A4 box. If I>0 the long side is horizontal, if I<0 the long side is vertical.

AXIS_LIN(X0,Y0,X1,Y1,SD,D,P)
- Draw a linear type axis from (X0,Y0) to (X1,Y1), with distance D between tick marks which are of height P(mm.). The starting position through the sequence is given by SD expressed as a fraction of D.

AXIS_LOG(X0,Y0,X1,Y1,SD,D,P)
- Draw a log type axis from (X0,Y0) to (X1,Y1), with distance D between cycles using tick marks which are of height P(mm.). The starting position through the sequence is given by SD expressed as a fraction of D.

CHAR_ABS(CSW,CSH)
- Reposition at the character coordinates CSW*Character width, CSH*Character height, relative to the current origin.

CHAR_ANGLE(PHI)
- Define the direction in which characters are plotted where the angle PHI is in degrees relative to the x-axis and positive anticlockwise. (Default: 0.).

CHAR_POSN(CSW,CSH)
- See CHAR_ABS

CHAR_REL(CSW,CSH)
- Reposition at the character coordinates CSW*Character width, CSH*Character height, relative to the current pen position.

CHAR_SIZE(Width,Height)
- Define character size (mm.). (Default: 2.4,3.0).

CHAR_SLOPE(PHI)
- Causes characters to be slanted at an angle PHI degrees relative to the vertical, with positive to the right. (Default: 0.).

CURSOR(U,V,Ichar)
- Move the cursor under operator control until terminated by a keyboard character or mouse button. The cursor coordinates are returned in U,V and the terminating character or button number in Ichar.

CURVE(F,U Min,U Max,Tol)
- Draw a smooth curve of the function F between U_Min and U_Max. Tol defines the smoothness (segment length in mm.).

DRAW(U,V)
- Reposition at coordinates (U,V) with pen down (see plot). Default coordinates are in mm.
FILL POLY(Xa,Ya,Na,Xb,Yb,Nb)
Using the currently selected colour, fill the area between the two polygons defined by the Na points in arrays Xa,Ya and the Nb points in the arrays Xb,Yb, with polygon a circumscribing polygon b.

FIT(U,V,N,M,S1,S2)
Fit a smooth curve to the N points given in arrays U,V. If:

(a) M=0 the slopes are not defined at the end points, and will be computed
(b) 1 the initial slope is defined in S1
(c) 2 the final slope is defined in S2
(d) 3 both slopes are defined in S1 and S2.

If (U(N),V(N))=(U(1),V(1)) a closed curve is drawn.

GREY(U,V,W)
Draw to coordinates U,V using intensity or colour W.

GREY SCALE(W low,W high)
Set lower and upper bounds of intensity levels for W.

GREY SET
Initialise grey levels.

GRID LAT LONG(XO,YO,X1,Y1,UO,VO,U1,V1,IC)
Draw a latitude v longitude grid where (XO,YO) (X1,Y1) are the world coordinates of the lower left and upper right corners of the grid respectively (mm.) and (UO,VO) (U1,V1) define limits in local coordinates (degrees). If:

(a) IC<0 only the scale and origin are computed.
(b) =0 as above, but the axes are also drawn
(c) >0 as above with the axes numbered.

GRID LIN LIN(XO,YO,X1,Y1,UO,VO,U1,V1,IC)
Draw a linear v linear grid where (XO,YO) (X1,Y1) define the limits in mm., (UO,VO) (U1,V1) define the limits in local units and IC is used as in GRID LAT LONG.

GRID LIN LOG(XO,YO,X1,Y1,UO,VO,U1,V1,IC)
Draw a log v linear grid where (XO,YO) (X1,Y1) define the limits in mm., (UO,VO) (U1,V1) define the limits in local units and IC is used as in GRID LIN LIN.

GRID LOG LIN(XO,YO,X1,Y1,UO,VO,U1,V1,IC)
Draw a linear v log grid where (XO,YO) (X1,Y1) define the limits in mm., (UO,VO) (U1,V1) define limits in local units and IC is used as in GRID LIN LIN.

GRID LOG LOG(XO,YO,X1,Y1,UO,VO,U1,V1,IC)
Draw a log v log grid where (XO,YO) (X1,Y1) define the limits in mm., (UO,VO) (U1,V1) define limits in local units and IC is used as in GRID LIN LIN.

LINE TYPE(I,P)
Define line type I and sequence length P in mm. (Default: 0,0.).

MOVE(U,V)
Reposition at coordinates (U,V) with pen up (see PLOT ABS).
ORIGIN(X Shift,Y Shift)
Redeﬁne the local coordinate origin. (Default: 0.,0.).

PEN_DOWN
- Select pen down.

PEN_UP
- Select pen up.

PLOT_3D(X,Y,Z,IC)
Plots X,Y,Z in three dimensional coordinates, as if viewed from an azimuth relative to the X-axis and an elevation from the X-Y plane, where IC is used as follows:

(a) IC=0 Initialise using azimuth angle -30.0 and elevation 45.0
(b) +1 X,Y are the azimuth and elevation in degrees
(c) +2 X,Y,Z are translation factors
(d) +3 X,Y,Z are scale factors
(e) +4 Move to X,Y,Z
(f) +5 Draw to X,Y,Z without hidden line
(g) +6 Draw to X,Y,Z with hidden line
(h) +7 Similar to 6 but does not draw (ﬁnd envelope)
(i) +8 Reset envelope

PLOT_3D_SURFACE_XY(X,Y,Z,N,M)
Draw the surface deﬁned by X,Y,Z(X,Y) where:

\[ X = X(i) \quad (i=1,2,\ldots,N) \]
\[ Y = Y(j) \quad (j=1,2,\ldots,M) \]
\[ Z = Z(i,j) \]

Before using this routine a 3D coordinate system must be initialised by use of the routine PLOT_3D.

PLOT_ABS(X,Y,IC)
Reposition at world coordinates (X,Y) expressed in mm., where IC is used as follows:

(a) IC<0 Use pen up
(b) +0 No change in pen condition
(c) +0 Use pen down

PLOT_CHAR_1(M,N)
Plot the character number N, using the soft character set number M.

PLOT_CHAR_2(NDAT)
Plot the character which is deﬁned by data in the INTEGER*4 array NDAT. (See source code).

PLOT_CHAR_symbol
Plot the character deﬁned by symbol as follows:

(a) A square-root sign when symbol=SQRROOT
(b) A Greek upper case character when symbol=nameU and name is one of ALPHA, BETA, GAMMA, DELTA, EPSILON, ZETA, ETA, THETA, IOTA, KAPPA, LAMBDA, MU, NU, XI, OMEGA, PI, RHO, SIGMA, TAU, UPSILON, PHI, XI, PSI, OMEGA
(c) A Greek lower case character when symbol=nameL and name is as before
These symbols have been provided in this form for convenience since they commonly occur in notation.

PLOTCONTURE(ZO,Z,M,N,IC,SO)
- Plot the contours of \( Z(X,Y)=Z_0 \) where:

\[
\begin{align*}
X & = i \\
Y & = j \\
Z & = Z(i,j)
\end{align*}
\]

for \( 1 \leq i \leq M \), \( 1 \leq j \leq N \). Special effect are obtained by setting:

\[ IC = 100*K + L \]

where integer \( L (<100) \) is the pen number to be used for annotation, and if \( L \) is zero no annotation is produced. The integer \( K \) defines whether interpolated points on the grid \((i,j)\) are joined by straight lines \((K=0)\) or joined by a smooth curve according \((K \text{ non-zero})\). If selected annotation is placed to begin at a distance \( SO \) along the contour from its start.

PLOTDEVICE (Text)
- Parameter Text is of type CHARACTER*(*) and is used to give instructions in the form keyword=value[,...] where [,...] means additional expressions may be given. The form of each expression is as follows:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>dd Where dd is a valid device name in use</td>
</tr>
<tr>
<td>STATUS</td>
<td>ON Enable device output</td>
</tr>
<tr>
<td></td>
<td>OFF Disable device output</td>
</tr>
<tr>
<td>ID-VD</td>
<td>n Select a virtual display n (1 to 4)</td>
</tr>
<tr>
<td>ID-WD</td>
<td>n Select a window display n (1 to 4)</td>
</tr>
<tr>
<td>TITLE</td>
<td>&quot;title&quot; The window title</td>
</tr>
</tbody>
</table>

If no device is specified then further action applies to all selected devices. An ID-VD number must refer to a virtual display previously created by PLOT_INIT. An ID-WD number may refer to an existing window, in which case it is selected, or to a new window, in which case PLOT_WINDOW must be used to create the required window (possible preceded by PLOT_VIEW). If no virtual display or window is specified then further action applies to the currently active window (see Examples). An empty character string (Text=' ') causes all selected devices o switch status.

PLOTDEVICE
- Equivalent to PLOTDEVICE_(' '), switch status of currently selected devices.

PLOT_ERASE (X0,Y0,X1,Y1)
- Erase the rectangle defined by the lower left corner at \( X_0,Y_0 \) and the upper right corner at \( X_1,Y_1 \).

PLOT_ERASE
- Erase the complete display or viewport

PLOTFIN (Text)
- Parameter Text is of type CHARACTER*(*) and is used to give instructions in the form keyword=value[,...] where [,...] means additional expressions may be given. The form of each expression is as follows:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>dd Where dd is a valid device name in use</td>
</tr>
<tr>
<td>STATUS</td>
<td>DELETE Delete the file copy</td>
</tr>
</tbody>
</table>
If no device is specified then further action applies to all selected devices. Normally the file copy is saved and other devices are deleted, but the file copy may also be deleted by specifying STATUS=DELETE.

PLOT_FIN
Equivalent to PLOT_FIN(' '), save the file copy (if selected) and delete other devices.

PLOT_INIT(Text)
Parameter Text is of type CHARACTER(*) and is used to give instructions in the form keyword=value[, ...] where [, ...] means additional expressions may be given. The form of each expression is as follows:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVICE</td>
<td>dd</td>
</tr>
<tr>
<td>WINDOW</td>
<td>FULL Select and map to the full screen size in mm.</td>
</tr>
<tr>
<td></td>
<td>A4-H Select and map to A4 landscape</td>
</tr>
<tr>
<td></td>
<td>A4-V Select and map to A4 portrait</td>
</tr>
<tr>
<td></td>
<td>NONE No window created</td>
</tr>
<tr>
<td>ID-VD</td>
<td>n Create virtual display number n (n=1,2,3,4)</td>
</tr>
<tr>
<td>TITLE</td>
<td>&quot;text&quot; The window title</td>
</tr>
</tbody>
</table>

Any hardware combinations can be specified by repeating the DEVICE=dd expression. The mnemonic TT can be used to represent either VG or WG determined by the login device. If no device is specified then 'DEVICE=TT,DEVICE=GF' is assumed. Use of TT allows an interactive device to be selected without GF. If no window is specified a new window is created and mapped to A4-H. The window option NONE is used to over-ride the default window creation. In this case no window is created at this stage (see PLOT WINDOW). An empty character string (Text=' ') is equivalent to 'WINDOW=A4-H'. Initialise the hardware and software as follows:

(a) Set coordinate scale factors (Xscale,Yscale)=(1,1)
(b) Set coordinate origin (Xshift,Yshift)=(0,0)
(c) Set character size (Xchsiz,Ychsiz)=(2.4,3.0)
(d) Set character direction (PHI)=(0,0)
(e) Set character slope (PHI)=(0,0)

PLOT_INIT
Equivalent to PLOT_INIT(' ')

PLOT_LIMITS(Umin,Vmin,Umax,Vmax)
Define plotting limits in current plotter units, where Umin,Vmin defines the lower left corner of a rectangle and Umax,Vmax the upper right corner.

PLOT_MARK(I)
Plot an identification marker at the current position, where I determines the marker shape, in the range 1 to 9. The size of the marker is determined by the current horizontal character size or as specified by PLOT_MARK_SIZE(Size).

PLOT_MARK_SIZE(Size)
Determines the size in mm of marker produced by PLOT_MARK. Size=0. resets to the current horizontal character size.

PLOT_MODE(I)
Define the writing mode as follows:

(a) I=0 Replace writing
(b) 1 Overlay writing
(c) 2 Complement writing
(d) 3 Erase writing
(e) 4 Negative writing off
(f) 5 Negative writing on

PLOT_NUM(Val,N,M)
Plot Val (NB Val is real) as a character string, using the hard character set, with N figures before the decimal point and M after. If M=0 an integer format is used. Positive numbers are preceded by a space, and negative numbers by a - sign.

PLOT_NUM_SOFT(Val,N,M)
Plot Val as a character string, using the soft character set (see PLOT_NUM).

PLOT_REL(X,Y,IC)
Reposition at relative coordinates (X,Y) expressed in mm. See PLOT_ABS

PLOT_TEXT(Text,N)
Plot the N characters in Text, using the hard character set. Special effects can be obtained as follows, where n is an integer value:

(a) "nS" Causes n spaces to be output (the quotes are part of the syntax)
(b) "nC" Causes n newlines, relative to the start point
(c) "P" Causes the character " to be plotted.

After completion the pen remains at the end of the current line.

PLOT_TEXT_SOFT(Text,N)
Plot the N characters in Text, using the soft character set. Special effects can be obtained as follows, where n is an integer value:

(a) "nS" Causes n spaces to be output (the quotes are part of the syntax)
(b) "nc" Causes n newlines, relative to the start point
(c) "nB" Causes characters to be selected from set n
(d) "P" Causes the character " to be plotted.

After completion the pen remains at the end of the current line.

PLOT_VIEW(Xmin,Ymin,Xmax,Ymax)
Define or modify a viewport for the current window using Xmin,Ymin as the lower left corner of a rectangle and Xmax,Ymax as the upper right.

PLOT_WINDOW(Xmin,Ymin,Xmax,Ymax)
Create or modify a window to the current virtual display using Xmin,Ymin as the lower left corner of a rectangle and Xmax,Ymax as the upper right. (See PLOT_DEVICE).

READ KB(Buffer,N1,N2)
Read data from the currently selected keyboard into the CHARACTER string Buffer, where N1 is the maximum number of characters to be read (unless input is terminated by <CR>, and N2 is the actual byte count on completion.

READ KB_NOECHO(Buffer,N1,N2)
Similar to READ KB but without echo.

READ KB_CHECK(N)
Check the keyboard input buffer and set N equal to the number of characters contained in it.
SCALE(X Scale,Y Scale)
Redefine the coordinate scale factors. (Default: 1.,1.).

SCFAC LIN(H0,H1,SD,D)
Calculates suitable scale factors for axis drawing, where H0,H1 are the
given values of the start and finish points and D is the calculated tick
length interval and SD the position of the first tick position (as a
fraction of D). D and SD are chosen to have 'nice' values.

SCFAC LOG(H0,H1,SD,D)
Calculates suitable factors for log-axis drawing, where D is the cycle
interval and SD the start point.

SELECT PEN(I)
Select pen number I. (Default: 1).

SELECT PEN COLOUR(Ipen,Red,Green,Blue)
Define the shade/colour of pen number Ipen as specified by the Red, Green,
Blue values (each in the range 0. to 1.) with special effects as follows:

<table>
<thead>
<tr>
<th>Ipen</th>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&gt;=0</td>
<td>r</td>
<td>g</td>
<td>b</td>
</tr>
</tbody>
</table>

6 METHOD OF USE

6.1 The FORTRAN examples provided demonstrate the use of many of the routines
described. Each program, for example EXAMPLE_1, should be compiled:

FOR Example_1
linked with the shareable library (named GRAPH here):

LINK Example_1,GRAPH/LIBR

and run:

RUN Example_1

Example_1 uses automatic device selection and therefore a file copy is produced
with the file name PLOT.PLT. In addition a display copy is produced if the
program is run from a suitable interactive device. The file copy may be
displayed on a suitable interactive device by use of the command DISP, or
plotted by the command PLUT. Occasionally it is also necessary to link with
other libraries (such as MATHS). Several examples are listed in Annex A.
Annex A. **Examples**

A.1 The following examples are included in order to demonstrate the use of various routines. Calculations associated with developing graphics images are not proposed as realistic or optimum, but provide a simple image which may be displayed or plotted using the routines described.
PROGRAM EXAMPLE_1

C Description: Graphics example showing how to draw a simple graph. The default
C initialization is used to automatically define an A4 drawing
C surface mapped to the device in landscape orientation.

C Author: C Richardson, ARE (Portland)

C History: Issue 1.0 4 November 1983

EXTERNAL F1,F2 ;functions to plot

Main Entry Point:

100 FORMAT(A)

CALL PLOT_INIT ;Initialise plotter
CALL CHAR_SIZE(1.5,2.0) ;Select character size
CALL CHAR_SLOPE(20.)
CALL GRID_LIN_LIN(60.,60.,180.,0.,-1.,360.,1.,1.)
CALL MOVE(0.,0.) ;Draw grid
CALL DRAW(360.,0.) ;Draw a line at Y=0.
CALL CURVE(F1,0.,360.,1.) ;Draw sin(x)
CALL SELECT_PEN(2) ;Select pen 2
CALL LINE_TYPE(2,10.0) ;Select another line type
CALL CURVE(F2,0.,360.,1.) ;Draw cos(x)
CALL SELECT_PEN(1) ;Select pen 1
CALL CHAR_SIZE(1.6,2.4)
CALL PLOT_ABS(160.,40.,-1) ;Position for X label
CALL CHAR_POSN(0.,-2.) ;X axis label
CALL PLOT_ABS(60.,90.,-1) ;Position for Y label
CALL CHAR_POSN(-4.,0.) ;Y axis label
CALL CHAR_ANGLE(90.)
CALL PLOT_TEXT(‘Sin(x)’,8) ;Y axis label
CALL SELECT_PEN(2)
CALL PLOT_TEXT(‘Cos(x)’,6)
CALL CHAR_ANGLE(0.)
CALL SELECT_PEN(1)
CALL PLOT_ABS(40.,170.,-1) ;Position for title
CALL CHAR_ANGLE(-90.)
CALL PLOT_TEXT(‘Fig 1.’,7)
CALL PLOT_TEXT(‘A Graphics Example’,19)
CALL PLOT_TEXT(‘Showing Sin(x) And Cos(x)’,26)
ACCEPT 100,N
CALL PLOT_FIN ;End of plotting
STOP ‘Finished’
END

FUNCTION F1(X)
F1=SIN(X/57.29578) ;F1 is sin(x)
RETURN
END

FUNCTION F2(X)
F2=COS(X/57.29578) ;F2 is cos(x)
RETURN
END
Fig 1. A Graphics Example Showing $\sin(x)$ And $\cos(x)$
CLEAR

C Description:
C An example showing a function which contains a cusp plotted by
C the routine CURVE and mapped to the device in portrait
C orientation.

C Author: C Richardson, ARE (Portland)

C History: Issue 1.0 4 November 1983

EXTERNAL Fn

C Main Entry Point:

100 FORMAT(A)
U0=-10.,
U1=10.,
V0=-0.1
V1=1.1
CALL PLOT_INIT('WINDOW=A4-V')
CALL CHAR_SIZE(1.5,2.0)
CALL CHAR_SLOPE(20.)
CALL GRID_LIN_LIN(40.,50.,140.,170.,U0,V0,U1,V1,1)
CALL CURVE(Fn,U0,U1,1.
CALL CHAR_SIZE(1.8,2.4)
CALL PLOT_TEXT('x',1.)
CALL MOVE((U0+U1)/2.,V0)
CALL PLOT_WIDTH(0.2.,2.)
CALL PLOT_TEXT('y',1.)
CALL MOVE(U0,(V0+V1)/2.)
CALL CHAR_POSN(0.,-1.)
CALL PLOT_TEXT('Fig 1.1
CALL PLOT_&8S(20.,20.,-1)
CALL PLOT_LABEL('A Graphics Example',19)
CALL PLOT_LABEL('Showing e^x',10)
CALL CHAR_POSN(0.,0.25)
CALL CHAR_SIZE(1.5,2.0)
CALL PLOT_TEXT('-|x|',4.)
ACCEPT 100,N
CALL PLOT_FIN
STOP 'Finished'
END
FUNCTION Fn(X)
Fn=exp(-ABS(X))
RETURN
END
Fig 2. A Graphics Example Showing $e^{-|x|}$
PROGRAM FILL_POLY_TEST

C Description:
C An example to draw closed polygons which are shaded with a given
C colour. The letters E and P are drawn, where the first requires
C a single polygon to be specified and the second requires two.
C
C Author:
C C Richardson, ARE (Portland)
C
C History:
C Issue 1.0 6 November 1986
C
C Local Variables:

REAL X(60), Y(60), Xi(13), Yi(13)
* 0., 0., 15., 15., 5., 5., 15., 15., 5., 5., 15., 15., 0., 0.,
* 0., 25., 25., 20., 15., 15., 10., 10., 5., 5., 0., 0./

C Main Entry Point:

100 FORMAT(A)
X(1)=0.  !Define the outer polygon
Y(1)=0.  ! for letter P
X(2)=0.  !Define the inner polygon
Y(2)=25.
DO n=0,24
  T=7.5*n
  X(n+3)=7.5+7.5*SIND(T)
  Y(n+3)=17.5+7.5*COSD(T)
ENDDO
X(28)=5.
Y(28)=10.
X(29)=5.
Y(29)=0.
X(30)=5.
Y(30)=20.
DO n=0,24
  T=7.5*n
  X(n+3)=7.5+2.5*SIND(T)
  Y(n+3)=17.5+2.5*COSD(T)
ENDDO
X(56)=5.
Y(56)=15.

CALL PLOT_INIT ('WINDOW-A4-V')  !Initialise with portrait format
CALL CHAR_SIZE(1.5,2.0)  !Enlarge the letters E, P
CALL SCALE(3.,3.)  !Position of letter E
CALL ORIGIN(50.,50.)  !Draw letter E shaded
CALL SELECT_PEN(4)  ! with colour 4
CALL FILL_POLY(X(1),Y(1),13,0.,0.,0.)  !Draw the outline using
CALL SELECT_PEN(2)  ! colour 2
CALL MOVE(X(1),Y(1))
DO n=2,13
  CALL DRAW(X(n),Y(n))
ENDDO

CALL ORIGIN(100.,50.)  !Position of letter P
CALL SELECT_PEN(4)  !Draw letter P shaded
CALL FILL_POLY(X(1),Y(1),29,X(30),Y(30),27)  ! with colour 4
CALL SELECT_PEN(2)  !Draw the outline using
CALL MOVE(X(1),Y(1))
DO n=2,29
  CALL DRAW(X(n),Y(n))
ENDDO

CALL DRAW(X(1),Y(1))  !Draw the inner outline using
CALL DRAW(X(30),Y(30))  ! colour 3

CALL ORIGIN(0.,0.)  !Reset scale and origin
CALL SCALE(1.,1.)  !Select pen 1
CALL MOVE(20.,20.)  !Plot a title
CALL PLOT_TEXT('Fig 3.,?)
CALL PLOT_TEXT(' Example Of Drawing A Filled Polygon.,)

ACCEPT 100,n
CALL PLOT_FIN
END
Fig 3. Example Of Drawing A Filled Polygon
PROGRAM PLOTVAFTEST

C Description: Example using variable area fill.

C Author: C Richardson, ARE (Portland)

C History: Issue 1.0 6 November 1986

C Local Variables:
REAL XX(201),YY(201) !Simulated data
COMMON T1,T2,T3,T4,A,B,C,D

C Main Entry Point:
100 FORMAT(A)

N=201 !Number of points
A=100. !Data generation parameters
B=20.
C=5.
D=2.

CALL PLOT_INIT !Initialise and draw a rectangular grid
CALL CHAR_SLOPE(20.)
CALL GRID_LIN_LIN(50.,50.,250.,186.,0.,0.,200.,17.,1)
CALL MOVE(100.,0.) !Annotate x axis
CALL CHAR_POSN(-5.5,-2.)
CALL PLOT_TEXT('Time (msec)',11)
CALL MOVETO.,8.5) !Annotate y axis
CALL CHAR_POSN(-5.,0.)
CALL CHAR_ANGLE(90.)
CALL CHAR_POSN(-8.5,0.)
CALL PLOT_TEXT('Hydrophone number',17)
CALL CHAR_ANGLE(0.)
CALL ORIGIN(0.,0.)
CALL MOVE(20.,20.)
CALL PLOT_TEXT('Fig 4.1')
CALL PLOT_TEXT('Example Using Variable Area Fill',.)

DO I=1,N !Calculate x values
   XX(I)=I+49.
ENDDO

DO I=1,16 !Draw trace for 16 hydrophones
   IX=20.+5.*I
   T1=50.+8.5*I
   T2=80.+10.**I
   T3=8.5+50.
   DO IX=1,N !Calculate trace for this hydrophone
      Y(I)=F(IX)+Y
   ENDDO
   CALL PLOT_VAF(XX,YY,N,Y+1.) !Draw trace with filled peaks
ENDDO

ACCEPT 100,1
CALL PLOT_FIN
END
FUNCTION F(X)
COMMON T1,T2,T3,S,A,B,C,D
XX=XX-T1
IF (ABS(XX) .LT. 0.000001) THEN
Y=A
ELSE
Y=A*SIN(XX)/(XX)
ENDIF
XX=XX-T2
IF (ABS(XX) .LT. 0.000001) THEN
Y=Y+A
ELSE
Y=Y+A*SIN(XX)/(XX)
ENDIF
XX=XX-T3
IF (ABS(XX) .LT. 0.000001) THEN
Y=Y+A
ELSE
Y=Y+A*SIN(XX)/(XX)
ENDIF
Y=Y+B*SIN(X/5.0)*C*SIN(X/2.0)*D*SIN(X)
F=2*Y
RETURN
END
PROGRAM PLOT_CONTOUR_TEST
C
Description: Example showing a contour plot.
C
Author: C Richardson, ARE (Portland)
C
History: Issue 1.0 6 November 1986
C
Local Variables:
REAL Z(15,11) : Grid map
C
Main Entry Point:
100 FORMAT(A)
101 FORMAT(16P8.3)
N=15
M=11
DO i=1,M
X=i-6.
DO j=1,M
Y=j-6
20=x**2+y**2-v**2
IF (20 .LE. 0.) THEN
Z(i,j)=0.
ELSE
Z(i,j)=SQRT(20)
ENDIF
ENDDO
ENDDO
CALL PLOT_INIT(‘WINDOW=Ax-V’)
CALL CHARSIZE(15,20.)
CALL ORIGYN(10.,50.)
CALL SCALE(10.,10.)
X=M
Y=N
CALL MOVE(1.,1.)
CALL DRAW(X,1.)
CALL DRAW(X,Y)
CALL DRAW(1.,1.)
DO J=2,M-1
Y=J
CALL MOVE(1.,Y)
CALL DRAW(X,Y)
ENDDO
Y=N
DO I=2,M-1
X=I
CALL MOVE(X,1.)
CALL DRAW(X,Y)
ENDDO
DO I=1,M
X=I
CALL MOVE(X,1.)
CALL CHAR_POSN(-3.5,0.)
CALL PLOTNUM(X-6.,2.0)
ENDDO
DO J=1,M
Y=J
CALL MOVE(1.,Y)
CALL CHAR_POSN(-3.5,0.)
CALL PLOTNUM(Y-6.,2.0)
ENDDO
CALL MOVE((M+1.)/2.,1.)
CALL CHAR_POSN(-4.5,-2.)
CALL PLOTTEXT(‘X’,1.1)
CALL MOVE((N+1.)/2.,1.)
CALL CHAR_POSN(-5.,0.)
CALL PLOTTEXT(‘Y’,1.1)
CALL SELECT_PEN(2)
IC=103
S0=0.
DO IS=12,19
S0=S0+1.
ENDDO
CALL PLOT_CONTOUR(10,2,3,1,IC,50)
END

CALL SCALE(1.,1.)
CALL ORIGIN(0.,0.)
CALL MOVE(20.,20.)
CALL SELECT_PEN(1)
CALL PLOT_TEXT("Fig 5.")
CALL PLOT_TEXT(" Contour Plot Of The Function X ",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + X",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + Y",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + Z",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + T",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + R",11)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT(" + S",11)
CALL CHAR_POSN(0.,0.25)
ACCEPT 100.52
CALL PLOT_FIN
END

CALL PLOT_FIG
Fig 5. Contour Plot Of The Function $x^2 + y^2 + z^2 = 10^2$
PROGRAM PLOT_SURFACE_TEST

C Description:
C Example showing a surface plot.

C Author:
C Richardson, ARE (Portland)

C History:
C Issue 1.0 5 December 1988

C Local Variables:

REAL XX(29),YY(21), ZZ(29,21) ! Node coordinates
              ! Surface function

C Main Entry Point:

100 FORMAT(A)

NM=29              ! Surface X dimension
NY=21              ! Surface Y dimension
NR=10.             ! Radius of sphere
DO n=1,NM
   XX(n)=(NM+1.)/2.
DO m=1,NM
   YY(m)=(NM+1.)/2.
   ZZ(m,n)=-(XX(m)**2+YY(n)**2)*2
   IF (Z0 .LE. 0.) THEN
      ZZ(m,n)=0.
   ELSE
      ZZ(m,n)=SQRT(Z0)
   ENDIF
ENDDO

CALL PLOT_INIT  ! Initialise graphics
CALL CHAR_SIZE(1.5,2.0)
CALL CHAR_SLOPE(20.)
CALL ORIGIR(50.,100.)
CALL PLOT_3D(0.,0.,0.,0.2)  ! 3D Origin
CALL PLOT_3D(5.5,5.5,3)    ! 3D Scale
CALL PLOT_3D(XX(1),YY(1),1.0,4)  ! Draw axes
CALL PLOT_3D(XX(MM),YY(1),1.0,5)
CALL PLOT_3D(XX(1),YY(NN),1.0,5)
CALL PLOT_3D(XX(MM),YY(NN),1.0,5)

CALL CHAR_ANGLE(45.)  ! Annotate X axis
CALL CHAR_SLOPE(45.)
Y=1.2*YY(1)-0.2*YY(2)
DO m=1,NM
   CALL PLOT_3D(XX(m),YY(1),0.4)
   CALL PLOT_3D(XX(m),YY(NN),0.4)
   CALL PLOT_3D(XX(m),Y,0.5)
   CALL PEN_UP
   CALL CHAR_POSN(-4.0,0.)
   IF (MOD(m,2).EQ.1) CALL PLOT_NUM(XX(m),2.0)
ENDDO

X=(XX(1)+XX(MM))/2.0
CALL PLOT_3D(X,Y,0.0,4)  ! Plot label (X)
CALL CHAR_POSN(-3.5,-1.0)
CALL CHAR_SLOPE(-30.0)
CALL CHAR_ANGLE(30.)
CALL CHAR_SLOPE(-30.)
CALL CHAR_POSN(-0.5,0.0)
CALL PLOT_3D('X',1)

X=0.8*XX(MM)+0.2*XX(MM-1)
DO m=1,NM
   CALL PLOT_3D(XX(m),YY(m),0.4)
   CALL PLOT_3D(XX(m),YY(n),0.5)
   CALL PEN_UP
   CALL CHAR_POSN(-0.25,0.0)
   IF (MOD(m,2).EQ.1) CALL PLOT_NUM(YY(m),2.0)
ENDDO

Y=(YY(1)+YY(NN))/2.0
CALL PLOT_3D(X,Y,0.0,4)  ! Plot label (Y)
CALL CHAR_POSN(-3.5,-1.0)
CALL CHAR_SLOPE(45.)
CALL CHAR_SLOPE(45.)
CALL CHAR_POSN(-0.5,0.0)
CALL PLOT_TEXT('Y', 1)
CALL CHAR_ANGLE(-30.)
CALL CHAR_SLOPE(-30.)
X=1.2*X(1)-0.2*X(2)
DO k=0,10
   CALL PLOT_1D(X(1),X(1),Z,4)
   CALL PLOT_3D(X(1),Z,5)
   CALL PEN_UP
   CALL CHARSLOPE(-30.)
   IF (MOD(K,2).EQ.0) CALL PLOT_NUM(Z,2,0)
ENDO
CALL PLOT_3D(X(1),Y(1),Z,4)
CALL PLOT_TEXT('Z',1)
CALL PLOT_3D_SURFACE_XY(X(1),Y(1),Z,2,6,MM,NN)
CALL CHAR_ANGLE(0.)
CALL CHAR_SLOPE(20.)
CALL ORIGIN(0.,0.)
CALL SCALE(1.,1.)
CALL MOVE(20.,20.)
CALL PLOT_TEXT('Fig 6.', 7)
CALL PLOT_TEXT('Surface Plot Of The Function X^2, Y')
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Y^2', 4)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('X+Y', 4)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('X', 4)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Y', 4)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Z', 4)
CALL CHARSLOPE(20.)
CALL PLOT_TEXT('X', 7)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Y', 7)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Z', 7)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('X', 7)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Y', 7)
CALL CHAR_POSN(0.,0.25)
CALL PLOT_TEXT('Z', 7)
CALL CHAR_POSN(0.,0.25)
CALL SELECT_PEN(2)
ACCEPT 100,N
CALL PLOT_FIM
END
PROGRAM SYMGREEKTEST
C
Description: Test greek characters
C
Author: C Richardson, ARE (Portland)
C
History: Issue 1.0 2 October 1988
C
Local Variables:
C
Main Entry Point:

100 FORMAT(A)

CALL PLOT_INIT('WINDOW=A4-V')
CALL CHAR_SIZE(3.0,4.0)
CALL CHAR_SLOPE(10.0)
CALL MOVE(20,150.)
CALL PLOT_TEXT('Upper case ','11')
CALL PLOT_CHAR_ALPHA_U
CALL PLOT_CHAR_BETA_U
CALL PLOT_CHAR_GAMMA_U
CALL PLOT_CHAR_DELTA_U
CALL PLOT_CHAR_EPSILON_U
CALL PLOT_CHAR_ZETA_U
CALL PLOT_CHAR_ETA_U
CALL PLOT_CHAR_THETA_U
CALL PLOT_CHAR_IOTA_U
CALL PLOT_CHAR_KAPPA_U
CALL PLOT_CHAR_LAMDA_U
CALL PLOT_CHAR_MU_U
CALL PLOT_CHAR_NU_U
CALL PLOT_CHAR_SIGMA_U
CALL PLOT_CHAR_OMEGAL
CALL CHAR_SIZE(1.5,2.0)
CALL MOVE(20,20.)
CALL PLOT_TEXO('Fig 7.',7)
CALL PLOT_TEXT('Example Using Greek Characters',)
ACCEPT 101,1
CALL PLOT FIN
END
Upper case ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

Lower case αβγδεζηθικλμνξοπρστυφχψω

Fig 7. Example Using Greek Characters
DESCRIPTION:
Example plots using the routine GREY. The surfaces which are
generated do not represent solutions to realistic problems,
but provide simple examples.

AUTHOR:
Richardson, ARE (Portland)

HISTORY:
Issue 1.0 6 November 1986

LOCAL VARIABLES:
INTEGER ASKN

MAIN ENTRY POINT:

100 FORMAT(A)
IN=ASKN('Pattern number')
IF (IN.EQ.0) GOTO 10
CALL PLOT_INIT
CALL CHAR_SIZE(1.8,2.4)
CALL ORIGIN(-20.,0.)
CALL GREY_SET
CALL GREY_SCALE(0.,10.)
CALL GREY_SELECT(10.)
CALL MOVE210.,160.)
CALL CHAR_ANGLE(-90.)
CALL PLOT_TEXT('Example Using Grey Shading, Pattern Number',42)
CALL PLOTNUM(FLOAT(IN),2,0)
IF (IN.NE.0) GOTO 10
DO J=0,100
X=12.*J/100.
Y=J*2.
CALL MOVE(0.,Y)
DO I=0,100
X=I/100.
S=5.*X/G2
CALL GREY(X,Y,S)
CALL READ_KS
ENDO
ENDO
GOTO 90

10 IF (IN.EQ.1) GOTO 20
A=1./1000.
DO J=-50,50
Y=J/2.
CALL MOVE(0.,Y+100.)
DO I=-50,50
X=I/2.
S=A*X**2+Y**2
CALL GREY(X+100.,Y+105.,Z)
CALL READ_KS
ENDO
ENDO
GOTO 90

20 IF (IN.EQ.2) GOTO 30
A=360./142.
C=1000.
DO J=-50,50
Y=J/2.
CALL MOVE(0.,Y+100.)
DO I=-50,50
X=I/2.
S=A*X*COSD(A*X)
Z=10.*COSD(A*X)
CALL GREY(X=100.,Y=105.,Z)
CALL READ_KS
ENDO
ENDO
GOTO 90

30
IF (I.NE.3) GOTO 40
A=CDSD(45.)*360./142.
B=A
C=0.001
DO J=0,100
Y=J*.2
CALL MOVE(0.,Y)
DO I=0,100
X=I*.2
R=SQRT(X*X+Y*Y)
E=2.5*(CDSD(A*X+B*Y)+CDSD(A*X-B*Y))
Z=-2.*EXP(-C*R)
CALL GREY(X,Y,Z)
CALL READ KB CHECK(N)
IF (N.NE.3) GOTO 91
ENDDO
GOTO 90
DO J=1,100
T=J*.2
CALL MOVE(0.,Y)
DO I=1,100
X=I*.2
T=ATAN2(Y,X)
T=18.*[T-.785396]
Z=1.
IF (T.NE.0.) Z=SQRT(T)/T
Z=10.*(.4+.Z)
CALL GREY(X,Y,Z)
CALL READ KB CHECK(N)
IF (N.NE.0.) GOTO 91
ENDDO
ENDDO
ACCEPT 100.I
90 ACCEPT 100.I
91 CALL PLOT FIN
END

Pattern 3. A vibrating plate

Pattern 4. A k,f diagram
Fig 8a. Example Using Grey Shading, Pattern 0
Fig 8b. Example Using Grey Shading, Pattern 1
Fig 8c. Example Using Grey Shading, Pattern 2
Fig 8d. Example Using Grey Shading, Pattern 3
Fig 8c. Example Using Grey Shading, Pattern 4
PROGRAM PLOT_SPHERE

Description:
Draw a sphere on a chequered board. A simple calculation is used in order to demonstrate the use of colour. The program does not model perspective drawing or light reflection in a sophisticated way.

Author: C Richardson, ARE (Portland)

History: Issue 1.0 26 January 1988

Local Variables:
PARAMETER Max=8 !Size of board
REAL X(0:Max,0:Max),Y(0:Max,0:Max),
* X(0:100,0:100),Y(0:100,0:100)
CHARACTER*16 Buf

Main Entry Point:
Calculate various coordinates for the board and sphere. The board is drawn in perspective, but the sphere is not because it looks distorted if drawn in perspective.

Max_Pen=150
Max=Max
X=0.
Y=200.
Xb=200.
Yb=200.
X0=100.
Y0=100.
X1=100.
Y1=100.
R=20.

CALL INTERSECT(Xa, Ya, X0, Y0, Xb, Yb, X1, Y1, Xe, Ye)
CALL INTERSECT(Xa, Ya, X1, Y1, Xb, Yb, X0, Y0, Xe, Ye)
S=SQRT((Xo-Xa)**2+(Yo-Ya)**2)
Sb=SQRT((Xo-Xb)**2+(Yo-Yb)**2)

CALL INTERSECT(Xs, Ye, Xb, Yb, X1, Y1, Xe, Ye)
CALL INTERSECT(Xs, Ye, X0, Y0, X1, Y1, Xe, Ye)

CALL INTERSECT(Xs, Ye, Xb, Yb, X0, Y0, X1, Y1, Xe, Ye)

ENDO

ENDDO

DO 1=0, 77
Thm=5.4
Ph=5.4-90.
U=R*COSD(Th)*COSD(Ph)
V=RSIND(Th)*COSD(Ph)
W=R&SIND(90)
Xs(1,J)=X(4,4)+U
Ys(I,J)=Y(4,4)+V

37
Using the data calculated above draw the board and sphere

CALL PLOT_INIT('DEVICE=TT') ; Initialise
CALL SELECT_PEN_COLOUR(-1.0,1.0) ; Shades of green
CALL SELECT_PEN_COLOUR(1.1,0.0) ; Pen 1 set to red
CALL SELECT_PEN_COLOUR(0.0,0.1) ; Last pen set to white
DO I=1,N

DO J=1,N

CALL SELECT_PEN(IP+1) ; Alternate the colour
CALL FILL_POLY(XP,YP,4,..0) ; Fill the square
ENDDO
ENDDO

CALL PLOT_MODE(0) ; Select replace writing
TH0=300. ; Set the lighting direction
PH0=45.
PI=COSD(TH0)*COSD(PH0) ; Find lighting vector
P2=SIND(TH0)*COSD(PH0)
P3=SIND(PH0)
DO I=3,72

DO J=1,36

CALL SELECT_PEN(IP) ; Draw sphere
CALL PLOT_POLY(XP,PP,4,..0) ; Convert to a pen number
IF (IP .GT. Max Pen) IP=Max Pen
IF (IP .LT. 3) IP=3
END DO
END DO

CALL MOVE(50.,35.)
CALL CHAR_SIZE(3.0,4.0)
CALL CHAR_SLOPE(20.)
CALL SELECT_PEN(Max Pen)
CALL PLOT_TEXT('Example Showing A Shaded Sphere...')

CALL READ_RK_NOECHO(Buf,1,1) ; Overlay mode
CALL PLOT_MODE(1)
CALL SELECT_PEN_COLOUR(0.0,0.0) ; Reset colours
CALL PLOT_PEN
END
SUBROUTINE INTERSECT(Xa1,Ya1,Xa2,Ya2,Xb1,Yb1,Xb2,Yb2,X,Y)

S1=(Xa1-Xa2)/(Ya1-Ya2)
S2=(Xb1-Xb2)/(Yb1-Yb2)
Y=(S1*Ya2-S2*Yb2+Xb2-Xa2)/(S1-S2)
X=S1*(Y-Ya2)+Xa2
RETURN
END
Fig 9. Example Showing A shaded Sphere
PROGRAM WINDOWS_TEST

Description:
An example to demonstrate the use of multiple virtual displays and/or windows on a workstation. Two virtual displays are created - green and blue images respectively. In each virtual display there are three windows each mapped to the screen and selected by using the keys A,B,1,2,3 to select virtual display 1 or 2, or the window viewports 1,2,3.

Author: C Richardson, ARE (Portland)

History:
Issue 1.0 26 January 1988

Local Variables:
REAL Xs(0:72,0:36),Ys(0:72,0:36)
CHARACTER*16 Buf
CHARACTER*12 Title(2,2)/'Green Sphera','Green Cube'/'Blue Sphere','Blue Cube'/

Main Entry Point:
IFP1=100 !Pen number range for virtual display number 1
IFP2=101 !Pen number range for virtual display number 2
N=25.
DO 2=0,72
Th=5.4*
DO J=0,36
Ph=9.*J-90.
U=R*COSD(Th)*COSD(Ph)
V=R*SIND(Th)*COSD(Ph)
W=R+SIND(Ph)
Xs(I,J)=U
Ys(I,J)=W
ENDDO
ENDDO

C Initialise virtual display number 1 and create window number 1
mapped to the coordinates 0.0, 200.200
CALL PLOT_INIT ('DEVICE-WG,WINDOW-WNONE')
CALL PLOTDEVICE ('ID-WD=1,TITLE="Window Number 1,1"')
CALL PLOTVIEW(0.0,200.,200.)
CALL PLOTWINDOW(0.,0.,200.,200.)
CALL CHAR_SIZE(3.0,4.0)
CALL CHARSLOPE(20.)
C Draw a green sphere and a green cube
DO I=IP10,IP1I
U=FLOAT(I-IP10)/(IP11-IP10) !Shades of green
ENDDO
CALL SELECT_PEN_COLOUR(1.0..U,0.)
ENDDO
CALL SELECT_PEN_COLOUR(IP11-IP10,1.0..U,0.)
CALL ORIGIN(50..40.)
CALL SPHERE(IP10,IP11.Xs.Ys)
CALL MOVE(-25.,-10.)
CALL SELECT_PEN(IP11)
CALL PLOTTEXT(Title(1,1),)
CALL ORIGIN(150..40.)
CALL CUBE(IP10,IP11)
CALL MOVE(-25.,-10.)
CALL SELECT_PEN(IP11)
CALL PLOTTEXT(Title(1,1),)
CALL ORIGIN(150..40.)
CALL CUBE(IP10,IP11)
CALL MOVE(-25.,-10.)
CALL SELECT_PEN(IP11)
CALL PLOTTEXT(Title(1,1),)

C Initialise virtual display number 2 and create window number 1
mapped to the coordinates 100.50, 300.350
CALL PLOT_INIT ('DEVICE-WG,ID-WD=2,WINDOW-WNONE')
CALL PLOTDEVICE ('ID-WD=1,TITLE="Window Number 2,1"')
CALL PLOTVIEW(100.50,300.350)
CALL PLOTWINDOW(0.0,200.,200.)
CALL CHAR_SIZE(3.0,4.0)
CALL CHARSLOPE(20.)
C Draw a blue sphere and a blue cube
DO I=IP20,IP21
U=FLOAT(1-IP20)/(IP21-IP20,)
41
CALL SELECT_PEN_COLOUR(1,0.,0.,0.) !Shades of blue
END
CALL SELECT_PEN_COLOUR(1P21,1.,1.,1.) !Last Pen set to white
CALL ORIGIN(150.,40.)
CALL SPHERE(1P20,1P21,Xs,Ys)
CALL MOVE(-25.,-10.)
CALL SELECT_PEN(1P21)
CALL PLOT_TSTX(Ttitle(l,2,)) !Plot title
CALL ORIGIN(50.,40.)
CALL CUBE(1P20,1P21)
CALL MOVE(-25.,-10.)
CALL SELECT_PEN(1P21)
CALL PLOT_TSTX(Ttitle(2,2,)) !Plot title
C Using virtual display 1, create window 2
C mapped to the coodinates 0.0, 100,100
CALL PLOT_DEVICE_1('ID-VD=1, ID-WD=2,TITLE="Window Number 1,2"')
CALL PLOT_WINDOW(0.,0.,100.,100.)
C Continuing to use virtual display 1, create window 3
C mapped to the coordinates 100,100, 200,200
CALL PLOT_DEVICE_1('ID-WD=3,TITLE="Window Number 1,3"')
CALL PLOT_VIEW(100.,100.,200.,200.)
CALL PLOT_WINDOW(100.,0.,200.,100.)
C Using virtual display 2, create window 2
C mapped to the coordinates 100,0, 200,100
CALL PLOT_DEVICE_2('ID-VD=2, ID-WD=2,TITLE="Window Number 2,2"')
CALL PLOT_WINDOW(150.,50.,200.,150.)
CALL PLOT_WINDOW(0.,0.,100.,100.)
C Continuing to use virtual display 2, create window 3
C mapped to the coordinates 200,150, 300,250
CALL PLOT_DEVICE_2('ID-WD=3,TITLE="Window Number 2,3"')
CALL PLOT_VIEW(200.,150.,300.,250.)
CALL PLOT_WINDOW(100.,0.,200.,100.)
C Switch between virtual displays 1 and 2 by pressing one of the keys
C A or B and switch between windows 1,2,3 by pressing one of the keys
C 1,2,3. Any other key terminates the program.

BUFF(1:1) = '1'
DO WHILE (((BUFF(1:1) .GE. '1') .AND. (BUFF(1:1) .LT. '4')).OR.
         ((BUFF(1:1) .GE. 'A') .AND. (BUFF(1:1) .LT. 'C'))) !Check key.
   CALL READ_KEYBOARD(BUFF(l:1))
   IF (BUFF(I:1) = 'A') THEN
      CALL PLOT_DEVICE_1('ID-VD=1') !Select Virtual display 1
      ELSE IF (BUFF(I:1) = 'B') THEN
         CALL PLOT_DEVICE_2('ID-VD=2') !Select Virtual display 2
      END IF
   ELSE IF (BUFF(l:1) .EQ. '2') THEN
      CALL PLOT_DEVICE_2('ID-WD=2') !Select window 2
   ELSE IF (BUFF(1:1) .EQ. '3') THEN
      CALL PLOT_DEVICE_2('ID-WD=3') !Select window 3
   ELSE
      CALL PLOT_DEVICE_2('ID-WD=1') !Select window 1
   END IF
END IF
END
CALL SELECT_PEN_COLOUR(0,0.,0.,0.) !Reset colours
CALL PLOT_FTH
END
SUBROUTINE SPH_DER(X0, Y0, Z0, X, Y, Z)
REAL X0, Y0, Z0, X, Y, Z

TH0 = 0.0, PH0 = 0.0, P1 = COSD(TH0) * COSD(PH0)
F2 = SIND(TH0) * COSD(PH0)
F3 = SIND(PH0)
DO 1 = 1, 72
TH = 5.5
DO 3 = 1, 7
PH = 5.5
XP(1) = X0(1, 1, 1)
YP(1) = Y0(1, 1, 1)
QP3 = COSD(TH) * COSD(PH)
QP4 = SIND(TH) * SIND(PH)
QP5 = SIND(PH)
QP6 = XP(1) * QP2 + YP(1) * QP3 + ZP(1) * QP4
QP7 = QP2 * QP2 + QP3 * QP3 + QP4 * QP4
IF (QP7 .GT. QP1) QP1 = QP7
CALL SELECT PEN(IP)
CALL FILL POLY(XP, YP, ..., 0)
ENDDO
RETURN
END

SUBROUTINE CUBE(IPO, IP1)
REAL XP(4), YP(4)
R = 25.0
XP(1) = 0.0
YP(1) = 0.0
XP(2) = XP(1)
YP(2) = YP(1)
XP(3) = 0.666 * R
YP(3) = 1.5 * R
XP(4) = XP(3)
YP(4) = 0.5 * R
IF = NINT(0.3333 * (IP1 - IPO + IPO)
CALL SELECT PEN(IP)
CALL FILL POLY(XP, YP, ..., 0)
XP(1) = 0.866 * R
YP(1) = 1.75 * R
XP(4) = 0.5 * R
YP(4) = 0.5 * R
IF = NINT(0.6667 * (IP1 - IPO) + IPO)
CALL SELECT PEN(IP)
CALL FILL POLY(XP, YP, ..., 0)
XP(1) = -R
YP(1) = -R
XP(2) = -R
YP(2) = -R
XP(3) = 0.666 * R
YP(3) = 1.5 * R
XP(4) = 0.866 * R
YP(4) = 1.75 * R
IF = IP1
CALL SELECT PEN(IP)
CALL FILL POLY(XP, YP, ..., 0)
RETURN
END

SUBROUTINE INTERSECT(XA1, YA1, XA2, YA2, XB1, YB1, XB2, YB2)
1 = (X1 - X2) / (Y1 - Y2)
2 = (X1 - XB2) / (Y1 - YB2)
3 = (XB1 - X2) / (XB1 - XB2)
4 = (X1 - Y2) * Y1
RETURN
END
Fig 10. Example Of The Use Of Virtual Displays And Windows
Annex B. Summary Of Subroutines

AXBOX(I)
AXIS_LINE(XO,YO,X1,Y1,SD,D,P)
AXIS_LOG(XO,YO,X1,Y1,SD,D,P)
CHAR_ABS(CSW,CSH)
CHAR_ANGLE(Phi)
CHAR_POSN(CSW,CSH)
CHAR_REL(CSW,CSH)
CHAR_SIZE(Width,Height)
CHAR_SLOPE(Phi)
CURSOR(U,V,Char)
CURVE(P,Usin,Umaj,TOL)
DRAW(U,V)
FILL_POLY(Xa,Ya,Na,Xb,Yb,Nb)
FIT(U,V,M,S1,S2)
GREY(X,Y,Z)
GREY_SCALE(Z_low,Z_high)
GRID_LAT_LON(XO,YO,X1,Y1,UO,VO,U1,V1,I)
GRID_LINE(XO,YO,X1,Y1,UO,VO,U1,V1,I)
GRID_LOG_LINE(XO,YO,X1,Y1,UO,VO,U1,V1,I)
GRID_LOG_LOG(XO,YO,X1,Y1,UO,VO,U1,V1,I)
LINE_TYPE(I,P)
MOVE(U,V)
ORIGIN(X_Shift,Y_Shift)
PEN_DOWN
PEN_UP
PLOT_3D(UV,W,I)
PLOT_3D_SURFACE(ZZ,M,N,Interp)
PLOT_ABS(X,Y,IC)
PLOT_CHAR ALPHA L
PLOT_CHAR_ALPHA U
PLOT_CHAR_BETA L
PLOT_CHAR_BETA U
PLOT_CHAR_GAMMA L
PLOT_CHAR_GAMMA U
PLOT_CHAR_DELTA L
PLOT_CHAR_DELTA U
PLOT_CHAR_EPSILON_L
PLOT_CHAR_EPSILON_U
PLOT_CHAR_ZETA_L
PLOT_CHAR_ZETA_U
PLOT_CHAR_eta L
PLOT_CHAR_eta U
PLOT_CHAR_THETA_L
PLOT_CHAR_THETA_U
PLOT_CHAR_IOTA_L
PLOT_CHAR_IOTA_U
PLOT_CHAR_KAPPA L
PLOT_CHAR_KAPPA U
PLOT_CHAR_LAMDA_L
PLOT_CHAR_LAMDA_U
PLOT_CHAR_MU_L
PLOT_CHAR_MU_U
PLOT_CHAR_NU_L
PLOT_CHAR_NU_U
PLOT_CHAR_XI_L
PLOT_CHAR_XI_U
PLOT CHAR OMRICON L
PLOT_CHAR_OMRICON_U
PLOT_CHAR_PI L
PLOT_CHAR_PI_U
PLOT_CHAR_RHO L
PLOT_CHAR_RHO_U
PLOT_CHAR_SIGMA L
PLOT_CHAR_SIGMA_U
PLOT_CHAR_TAU L
PLOT_CHAR_TAU_U
PLOT_CHAR_UPSILON L
PLOT_CHAR_UPSILON_U
PLOT_CHAR_PHI L
PLOT_CHAR_PHI_U
PLOT_CHAR_XI L
PLOT_CHAR_XI_U
PLOT_CHAR_PSI L
PLOT_CHAR_PSI_U
PLOT_CHAR_OMEGA L
PLOT_CHAR_OMEGA_U
PLOT_CHAR_SQRT
PLOT_CONTOUR(Z0,Z1,N,IC,S0)
PLOT_COPY(FILE,N,M)
PLOT_DEVICE
PLOT_DEVICE(Text)
PLOT_ERASE
PLOT_ERASE(X0,Y0,X1,Y1)
PLOT_FIN
PLOT_FIN(Text)
PLOT_INIT
PLOT_INIT(Text)
PLOT_LIMITS(Umin,Vmin,Umax,Vmax)
PLOT_MARK(Select)
PLOT_MARK_SIZE(Size)
PLOT_MODE(I)
PLOT_NUM(Val,N,M)
PLOT_NUM_SOFT(Val,N,M)
PLOT_REL(X,Y,IC)
PLOT_TEXT(Text,N)
PLOT_TEXT_SOFT(Text,N)
PLOT_VAF([J,V,N,VO)
PLOT_VIEW(XMIN,YMIN,XMAX,YMAX)
PLOT_WINDOW(XMIN,YMIN,XMAX,YMAX)
READ Kb(Buffer,N1,N2)
READ Kb_NOCHECK(Buffer,N1,N2)
SCALE(X,Scale,Y,Scale)
SCFAC_LIN(N0,R1,DD,D)
SCFAC_LOG(N0,B1,DD,D)
SELECT_PEN(I)
SELECT_PEN_COLOUR(I,Red,Green,Blue)
Annex C. File Contents

A4BOX.FOR
A4BOX(I)

AXIS.FOR
AXIS_LIN(X0,Y0,X1,Y1,SD,D,P)
AXIS_LOG(X0,Y0,X1,Y1,SD,D,P)

CHAR_DATA.FOR
CHAR_DATA(Set,N,Char_buf)

COORDS.FOR
SCALE(X Scale,Y Scale)
ORIGIN(X Shift,Y_Shift)

CURSOR.FOR
CURSOR(X,Y,Char)

CURVE.FOR
CURVE(F,X_MIN,X_MAX,Tol)

FIT.FOR
FIT(X,Y,N,M,S1,S2)

GRIDS.FOR
SCFAC_LIN(H0,H1,DD,D)
SCFAC_LOG(H0,H1,DD,D)
GRID_LAT_LONG(X0,Y0,X1,Y1,U0,VO,UI,VI,I)
GRID_LIN_LIN(X0,Y0,X1,Y1,U0,VO,UI,VI,1)
GRID_LIN_LOG(X0,Y0,X1,Y1,U0,VO,UI,VI,1)
GRID_LOG_LIN(X0,Y0,X1,Y1,U0,VO,UI,VI,1)
GRID_LOG_LOG(X0,Y0,X1,Y1,U0,VO,UI,VI,1)

PLOT_1.FOR
PLOT_INIT(Text)
PLOT_VIEW(Xmin,Ymin,Xmax,Ymax)
PLOT_DEVICE(Text)
PLOT_FIN(Text)
MOVE(U,V)
DRAW(U,V)
PLOT_REL(X,Y,IC)
PLOT_ABS(X,Y,IC)

PLOT_2.FOR
PEN_UP
PEN_DOWN
PLOT_MODE(I)
PLOT_ERASE(X0,Y0,X1,Y1)
SELECT_PEN-COLOUR(I,Red,Green,Blue)
SELECT_PEN(I)
LINE_TYPE(I,P)
CHAR_SIZE(WIDTH,HEIGHT)
CHAR_SLOPE(PHI)
CHAR_ANGLE(PHI)

47
CHAR_SOFT
  CHAR_HARD
CHAR_POSN(CSW,CSH)
CHAR_REL(CSW,CSH)
CHAR_ABS(CSW,CSH)
PLOT_LIMITS(Umin,Vmin,Umax,Vmax)

PLOT_3D.FOR
  PLOT_3D(XX,YY,ZZ,CON)

PLOT_4.FOR
  FILL_POLY(Xa,Ya,Na,Xb,Yb,Nb)
  GREY(U,V,W)
  GREY_SET
  GREY_SCALE(V_lo,v_hi)
  PLOT_VAF(U,V,N,V0)

PLOT_CONTOUR.FOR
  PLOT_CONTOUR(ZO,Z,M,N,IC,SO)

PLOT_CHAR.FOR
  PLOT_CHAR_1(M,N)
  PLOT_CHAR_2(NDAT)
  PLOT_CHAR_2M(NDAT,IC)
  PLOT_CH(X,Y)
  GET_CH(NDAT,I,J)
  SET_CH

PLOT_MARK.FOR
  PLOT_MARK(Select)
  PLOT_MARK_SIZE(Size)

PLOT_SURFACE.FOR
  PLOT_3D_SURFACE(ZZ,M,N,Interp)

PLOT_TEXT_1.FOR
  PLOT_TEXT(Text,N)
  PLOT_NUM(Val,N,M)

PLOT_TEXT_2.FOR
  PLOT_TEXT_SOFT(Text,N)
  PLOT_NUM_SOFT(Val,N,M)

READ_KB.FOR
  READ_KB(Buffer,N1,N2)
  READ_KB_NOECHO(Buffer,N1,N2)
  READ_KB_CHECK(N)

SYM_GREEK.FOR
  PLOT_CHAR_ALPHA
  PLOT_CHAR_ALPHA_L
  PLOT_CHAR_ALPHA_U
  PLOT_CHAR_BETA
  PLOT_CHAR_BETA_L
  PLOT_CHAR_BETA_U
  PLOT_CHAR_GAMMA
  PLOT_CHAR_GAMMA_L
  PLOT_CHAR_GAMMA_U
  PLOT_CHAR_DELTA
  PLOT_CHAR_DELTA_L
  PLOT_CHAR_DELTA_U
PLOT_CHAR_EPSILON
PLOT_CHAR_EPSILON_L
PLOT_CHAR_EPSILON_U
PLOT_CHAR_ZETA
PLOT_CHAR_ZETA_L
PLOT_CHAR_ZETA_U
PLOT_CHAR_ETA
PLOT_CHAR_ETA_L
PLOT_CHAR_ETA_U
PLOT_CHAR_THETA
PLOT_CHAR_THETA_L
PLOT_CHAR_THETA_U
PLOT_CHAR_IOTA
PLOT_CHAR_IOTA_L
PLOT_CHAR_IOTA_U
PLOT_CHAR_KAPPA
PLOT_CHAR_KAPPA_L
PLOT_CHAR_KAPPA_U
PLOT_CHAR_LAMDA
PLOT_CHAR_LAMDA_L
PLOT_CHAR_LAMDA_U
PLOT_CHAR_MU
PLOT_CHAR_MU_L
PLOT_CHAR_MU_U
PLOT_CHAR_NU
PLOT_CHAR_NU_L
PLOT_CHAR_NU_U
PLOT_CHAR_XI
PLOT_CHAR_XI_L
PLOT_CHAR_XI_U
PLOT_CHAR_OMRICON
PLOT_CHAR_OMRICON_L
PLOT_CHAR_OMRICON_U
PLOT_CHAR_Phil
PLOT_CHAR_Phil_L
PLOT_CHAR_Phil_U
PLOT_CHAR_PI
PLOT_CHAR_PI_L
PLOT_CHAR_PI_U
PLOT_CHAR_RHO
PLOT_CHAR_RHO_L
PLOT_CHAR_RHO_U
PLOT_CHAR_SIGMA
PLOT_CHAR_SIGMA_L
PLOT_CHAR_SIGMA_U
PLOT_CHAR_TAU
PLOT_CHAR_TAU_L
PLOT_CHAR_TAU_U
PLOT_CHAR_UPSILON
PLOT_CHAR_UPSILON_L
PLOT_CHAR_UPSILON_U
PLOT_CHAR_PHI
PLOT_CHAR_PHI_L
PLOT_CHAR_PHI_U
PLOT_CHAR_XI
PLOT_CHAR_XI_L
PLOT_CHAR_XI_U
PLOT_CHAR_PSI
PLOT_CHAR_PSI_L
PLOT_CHAR_PSI_U
PLOT_CHAR_OMEGA
PLOT_CHAR_OMEGA_L
PLOT_CHAR_OMEGA_U
SYM_MATHS.FOR
PLOT_CHAR_SQROOT