D I S L I N 7.1

A Data Plotting

Library

by

Helmut Michels

Contents

Preface to Version 7.1

This manual describes the data plotting library DISLIN written in the programming languages Fortran and C. The name DISLIN is an abbreviation for Device-Independent Software LINdau since applications were designed to run on different computer systems without any changes. The library contains subroutines and functions for displaying data graphically as curves, bar graphs, pie charts, 3-D colour plots, surfaces, contours and maps.

DISLIN is intended to be a powerful and easy to use software package for programmers and scientists that does not require knowledge of hardware features of output devices. The routines in the graphics library are the result of my own work on many projects with different computers and many plotting packages. There are only a few graphics routines with a short parameter list needed to display the desired graphical output. A large variety of parameter setting routines can then be called to create individually customized graphics.

Since the first version of DISLIN was released in Dec. 1986, many changes and corrections have been made and new features and standards have been added to the software. Some of the new features are elementary image routines, a graphical user interface, filled contour lines, flat and smooth shaded surfaces and a C interface for reading binary data from Fortran programs. DISLIN supports now several hardware platforms, operating systems and compilers. A real Fortran 90 library is available for most Fortran 90 compilers.

Although nearly all the routines and utilities of the software package are written by myself, DISLIN would not have been possible without the help of many people. I would like to thank several people at the Max-Planck-Institut in Lindau. First, Dr. W. Degenhardt, Dr. H. J. Müller and Dr. I. Pardowitz who gave their friendly assistance. To all the users of DISLIN, I am grateful for your helpful suggestions and comments. I would especially like to thank the members of the computer center, Friederich Both, Terry Ho, Godehard Monecke and Michael Bruns for their co-operation. Finally, I am grateful to Linda See and Erika Eschebach who corrected the English and German manuals with great carefulness. To all of them, my sincere thanks.

H. Michels Lindau, 15.01.1999

Chapter 1

Introduction

DISLIN is a library of subroutines and functions that display data graphically. The routines can be used with any display device capable of drawing straight lines with the exception of routines that generate 3-D colour graphics which require special devices. Fortran 77, Fortran 90 and C versions of the library are available.

DISLIN stores the graphic information in a device-independent metafile that is converted by a device driver program for plotting on various devices. Two formats can be selected: a GKSLIN metafile coded in ASCII and a CGM binary metafile corresponding to the ANSI norm. Device-dependent plotfiles can also be generated by DISLIN such as PostScript files, HPGL files, Prescribe files for Kyocera printers and X11, VGA and Tektronix emulations for graphic terminals.

Chapter 2 describes the file and page formats and the overall structure of DISLIN programs.

Chapter 3 describes routines for the initialization, termination and plotting of text, numbers and symbols.

Chapter 4 presents the format of two-dimensional axis systems. Axes can be linearly or logarithmically scaled and labeled with linear, logarithmic, time, map and user-defined formats.

Chapter 5 describes the routines for plotting curves. Several curves can appear in one axis system and can be differentiated by colour, line style and pattern.

Chapter 6 summarizes parameter setting routines that overwrite default plotting parameters such as fonts, character size and angle, colours, line styles and patterns.

Chapter 7 presents routines to request the values of plot parameters.

Chapter 8 describes the routines for plotting lines, circles, ellipses, vectors and shaded regions.

Chapter 9 describes the utilities available to transform coordinates, sort data and calculate the lengths of numbers and character strings.

Chapter 10 introduces business graphic routines to create bar graphs and pie charts.

Chapter 11 presents 3-D colour graphics where points can be plotted with coloured or shaded rectangles. A colour graphics device or a PostScript printer is needed for these subroutines and functions.

Chapter 12 describes routines for 3-D coordinate systems. Axis systems, curves and surfaces can be drawn from various angular perspectives. All 2-D plotting routines can be used in a 3-D axis system.

Chapter 13 presents 14 different methods to project geographical coordinates onto a plane surface. Several base maps are stored in the library for map plotting.

Chapter 14 describes routines for contouring three-dimensional functions of the form Z = F(X,Y). Contours can be filled with solid lines.

Chapter 15 offers routines for creating graphical user interfaces in Fortran and C programs.

Chapter 16 describes routines for plotting and modifying the MPAe emblem.

Chapter 2

Basic Concepts and Conventions

2.1 Page Format

In DISLIN, the graphics are limited to a rectangular area called the page. All lines outside of or crossing page borders will be suppressed.

The size of the page is determined by the routines SETPAG and PAGE. SETPAG corresponds to a predefined page while PAGE defines a global page setting. In default mode, there are 100 points per centimeter and the point (0, 0) is located in the upper left corner (Figure 2.1):

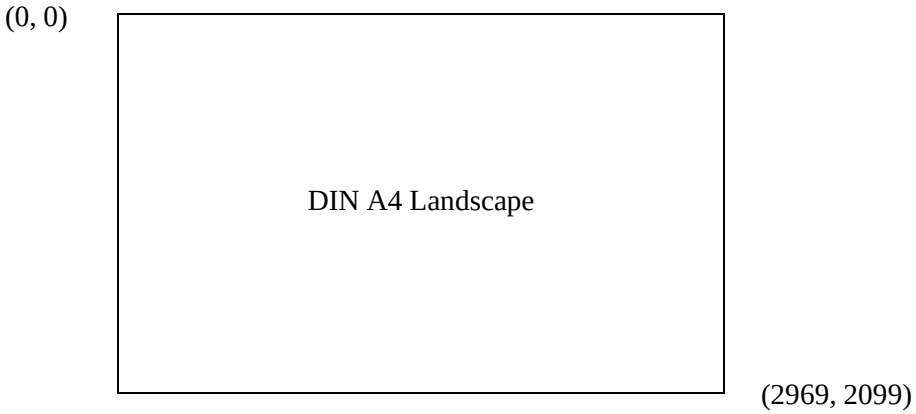

Figure 2.1: Default Page (DA4L)

2.2 File Format

DISLIN can create several types of plotfiles. Device-independent plotfiles or metafiles can be coded in ASCII or binary format. Device-dependent plotfiles are available for several printers and plotters.

The file formats are:

a) a CGM metafile according to the ANSI standard
Plot vectors are coded in binary format with 200 points per cm and with integers in the range -32767 to 32767. Because of binary coding, CGM metafiles are smaller than other plotfiles.

b) a GKSLIN metafile
Plot vectors are stored as floating-point numbers between 0 and 1 in ASCII format. These files are easily transferable from one computer to another.

c) a PostScript file
PostScript is an international standard language that has been developed for laserprinters in the last few years. Some of the PostScript features such as hardware fonts and shading can be used within DISLIN.

d) a Prescribe file
The plotfile is created in the graphics language of Kyocera laserprinters and may also contain hardware features such as shading for rectangles and pies.

e) a HPGL file
Plot vectors and colours are coded in a language recognized by Hewlett-Packard plotters.

f) a Java applet file
Plot vectors and colours are stored in form of a Java applet.

g) a TIFF file
The raster format TIFF can be used for storing graphical output.

h) an IMAGE file
This easy raster format is used by DISLIN to store images. The files contain an ASCII header of 80 bytes and the following image data.

i) a Tektronix, X Window and VGA emulation
Data can be displayed on graphic terminals such as X Window, VGA and Tektronix 4010/4014.

File formats can be set with the routine METAFL. The filename consists of the keyword 'DISLIN' and an extension that depends on the file format. An alternate filename can be chosen by calling the routine SETFIL. Both subroutines must be called before the initialization routine.

2.3 Level Structure of DISLIN

Most routines in DISLIN can be called anywhere during program execution. Certain routines, however, use parameters from other routines and must be called in a fixed order. DISLIN uses a level structure to control the order in which routines are called. The levels are:

0 before initialization or after termination

1 after initialization or a call to ENDGRF

2 after a call to GRAF

3 after a call to GRAF3 or GRAF3D.

Generally, programs should have the following structure:

(1) setting of page format, file format and filename

(2) initialization

(3) setting of plot parameters

(4) plotting of the axis system

(5) plotting the title

(6) plotting data points

(7) termination.

2.4 Conventions

The following conventions appear throughout this manual for the description of routine calls:

- INTEGER variables begin with the character N or I
- CHARACTER variables begin with the character C
- other variables are REAL
- arrays end with the keyword 'RAY'.

Additional notes:

- CHARACTER variables may be specified in upper or lower case and may be shortened to four characters.
- DISLIN stores parameters in common blocks whose names begin with the character 'C'. Common block names in user programs should not begin with the character 'C' to avoid possible name equalities.
- The Fortran logical units 15, 16 and 17 are reserved by DISLIN for plot and parameter files.
- Two types of coordinates are continually referred to throughout the manual: plot coordinates which correspond to the page and always have 100 points per cm, and user coordinates which correspond to the scaling of the axis system.

2.5 Error Messages

When a DISLIN subroutine or function is called with an illegal parameter or not according to the level structure, DISLIN writes a warning to the screen. The call of the routine will be ignored and program execution resumed. Points lying outside of the axis system will also be listed on the screen. Error messages can be suppressed or written to a file with the routines UNIT and NOCHEK.

2.6 Programming in C

There are different DISLIN libraries for the programming languages Fortran 77, Fortran 90 and C. The DISLIN C library is written in the programming language C and useful for C programmers.

Though it is possible to call C routines in Fortran programs and Fortran subroutines in C programs, it is easier to use the corresponding library. Especially, the passing of strings can be complicate in mixed language programming.

The number and meaning of parameters passed to DISLIN routines are identical for all libraries. The Fortran versions use INTEGER, REAL and CHARACTER variables while the C library uses int, float and char variables. A detailed description of the syntax of C routines is given by the utility program DISHLP or can be found in the header file 'dislin.h' which must be included in all C programs.
For example:

```
#include <stdio.h>
#include "dislin.h"
main()
{
    disini ();
    messag ("This is a test", 100, 100);
    disfin ();
}
```

2.7 Programming in Fortran 90

Several DISLIN distributions contain native libraries for the programming language Fortran 90 where the source code of DISLIN is written in Fortran 90. Since the passing of parameters to subroutines and functions can be different in Fortran 90 and Fortran 77, you should not link Fortran 77 programs with Fortran 90 libraries and vice versa.

Important: All program units in Fortran 90 programs that contain calls to DISLIN routines must include the statement 'USE DISLIN'. The module 'DISLIN' contains interfaces for all DISLIN routines and enables the compiler the correct passing and checking of parameters passed to DISLIN routines.

For example:

```
PROGRAM TEST
   USE DISLIN
   CALL DISINI ()
   CALL MESSAG ('This is a test', 100, 100)
   CALL DISFIN ()
END PROGRAM TEST
```

2.8 Linking Programs

The linking of programs with the graphics library depends upon the operating system of the computer. Therefore, DISLIN offers a system-independent link procedure that can be used on all computers in the same way.

Command: DLINK [option] main

option is an optional parameter containing a minus sign and a character. The following options can be used on all computers:

- -c for compiling programs before linking.
- -r for running programs after linking.
- -a for compiling, linking and running programs.

main is the name of the main program.

Additional notes:

- If DLINK is called without parameters, the description of the program will be printed on the screen. There may be other local features available depending upon the operating system used.
- Linking of C programs should be done with the procedure CLINK.
- Linking of Fortran 90 programs should be done with the procedure F90LINK.

2.9 Utility Programs

The following programs are useful for working with DISLIN. They send plotfiles to devices, check the use of DISLIN routines in Fortran programs and print the description of routines on the screen.

D I S H L P

DISHLP prints the description of a DISLIN routine on the screen.

Command: DISHLP routine [options]

routine — is the name of a DISLIN routine or a question mark. For a question mark, all routine names will be listed. An empty input terminates the program.

options — is an optional field of keywords (see DISHLP).

D I S M A N

DISMAN prints an ASCII version of the DISLIN manual on the screen.

Command: DISMAN [options]

options — is an optional field of keywords (see DISMAN).

D I S P R V

DISPRV checks the use of DISLIN routines in a Fortran program. The type and dimension of parameters and the overlapping of common block and routine names with internal DISLIN declarations will be checked.

Command: DISPRV filename[.FOR] [options]

filename — describes the file containing the Fortran code.

options — is an optional field of keywords (see DISPRV).

D I S D R V

DISDRV sends a plotfile to a device. CGM and GKSLIN files can be used for all devices while device-dependent plotfiles can only be sent to corresponding devices.

Command: DISDRV filename[.MET] [device] [options]

filename — is the name of a plotfile.

device — is the name of a device. CONS refers to the graphics screen, XWIN to an X Window terminal, PSCi to a PostScript printer, KYOi to a Kyocera laserprinter with Prescribe and HPLi to a HP-plotter, where i = 1, 2, 3, ..., n is the printer number.

options — is an optional field of keywords (see DISDRV).

D I S H P J

DISHPJ sends a GKSLIN or CGM metafile to a printer using a raster graphics emulation (i.e. HP PCL).

Command: DISHPJ filename[.MET] [device] [options]

filename — is the name of the metafile.

device — is the name of the device.

options — is an optional field of keywords (see DISHPJ).

D I S I M G

DISIMG displays an image file on the screen, or converts it to PostScript and TIFF.

Command: DISIMG filename[.IMG] [device] [options]

filename	is the name of the image file. The file must be created with the routine RIMAGE.
device	is the device name.
options	is an optional field of keywords (see DISIMG).

D I S M O V

DISMOV displays a sequence of image files.

Command:	DISMOV filename[.MOV] [device] [options]
filename	is the name of a data file where the filenames of the images are stored (1 line for each filename). The images must be created with the routine RIMAGE.
device	is the device name.
options	is an optional field of keywords (see DISMOV).

D I S T I F

DISTIF displays a TIFF file created by DISLIN on the screen, or converts it to PostScript and an image format.

Command:	DISTIF filename[.TIF] [device] [options]
filename	is the name of the TIFF file. The file must be created with the routine RTIFF.
device	is the device name.
options	is an optional field of keywords (see DISTIF).

D I S G I F

DISGIF displays a GIF file on the screen, or converts it to PostScript and TIFF.

Command:	DISGIF filename[.GIF] [device] [options]
filename	is the name of the GIF file.
device	is the device name.
options	is an optional field of keywords (see DISGIF).

D I S A P S

DISAPS converts an ASCII file to a PostScript file. Several page layouts can be defined.

Command:	DISAPS filename [output] [options]
filename	is the name of the ASCII file.
output	is the name of the output file. By default, the name of the input file and the extension ps will be used.
options	is an optional field of keywords (see DISAPS).
Additional note:	If a utility program is called without parameters, a description of possible parameters will be printed on the screen. DISDRV, for example, lists the local output devices available.

2.10 FTP Sites, WWW Homepage

DISLIN is available via ftp anonymous from the following sites:

ftp://ftp.gwdg.de/pub/grafik/dislin	(fast connection)
ftp://linhmi.mpae.gwdg.de/pub/dislin	(slower connection)

The DISLIN Homepage is:

http://www.linhmi.mpg.de/dislin

2.11 Reporting Bugs

DISLIN is well tested by many users and should be very bug free. However, no software is perfect and every change can cause new bugs. If you have any problems with DISLIN, contact the author:

Helmut Michels
Max-Planck-Institut fuer Aeronomie
D-37191 Katlenburg-Lindau, Max-Planck-Str. 2, Germany
E-Mail: michels@linmpi.mpg.de
Tel.: +49 5556 979 334
Fax: +49 5556 979 240

2.12 License Information

DISLIN is free for the operating systems Linux and FreeBSD and for the GNU compilers GCC+G77/MS-DOS and GCC+G77/Windows95. Other DISLIN versions are available at low charge and can be tested free of charge. Programs linked with DISLIN can be distributed without any royalties together with neccessary shareable DISLIN libraries.

Normally, DISLIN programs check for a valid DISLIN license in the file 'license.dat' in the DISLIN directory. If DISLIN is not installed on a system, a DISLIN program can be executed if the file 'license.dat' is created with the entry 'License: Runtime'. The environment variable 'DISLIN' should be set to the directory where the file 'license.dat' is placed.

A valid DISLIN license can be received by running the program 'license' in the DISLIN directory and sending the output file 'license.lis' to the author.

Chapter 3

Introductory Routines

3.1 Initialization and Termination

DISINI initializes DISLIN by setting default parameters and creating a plotfile. The level is set to 1. DISINI must be called before any other DISLIN routine except for those noted throughout the manual.

The call is:	CALL DISINI	level 0
or:	void disini ();	

DISFIN terminates DISLIN and prints a message on the screen. The level is set back to 0.

The call is:	CALL DISFIN	level 1, 2, 3
or:	void disfin ();	

3.2 Plotting of Text and Numbers

M E S S A G

MESSAG plots text.

The call is:	CALL MESSAG (CSTR, NX, NY)	level 1, 2, 3
or:	void messag (char *cstr, int nx, int ny);	

CSTR	is a character string ($\leq$ 256 characters).
NX, NY	are the plot coordinates of the upper left corner.

N U M B E R

NUMBER plots a floating-point number or integer.

The call is:	CALL NUMBER (X, NDIG, NX, NY)	level 1, 2, 3
or:	void number (float x, int ndig, int nx, int ny);	

X	is a floating-point number.
NDIG	is the number of digits plotted after the decimal point. If NDIG = -1, X will be plotted as an integer. The last digit of X will be rounded up.
NX, NY	are the coordinates of the upper left corner.

RLMESS and RLNUMB are corresponding routines for user coordinates. They can be used for plotting text and numbers in an axis system after a call to GRAF.

The calls are: CALL RLMESS (CSTR, XP, YP) level 2, 3

CALL RLNUMB (X, NDIG, XP, YP) level 2, 3

or: void rlmess (char *cstr, float xp, float yp);

void rlnumb (float x, int ndig, float xp, float yp);

Additional notes:

- To continue character strings and numbers on the same line, the coordinates (999, 999) should be sent to MESSAG and NUMBER. The text or numbers will be plotted after the last plotted text character or number.
- The angle and height of the characters can be changed with the routines ANGLE and HEIGHT.
- The format of numbers can be modified with the routines NUMFMT and NUMODE.
- Text and numbers can be plotted in a box if the routine FRMESS is used.
- The starting point of text and numbers can be interpreted as upper left, upper center and upper right point if the routine TXTJUS is used.

3.3 Plotting Symbols

S Y M B O L

The routine SYMBOL plots symbols.

The call is: CALL SYMBOL (NSYM, NX, NY) level 1, 2, 3

or: void symbol (int nsym, int nx, int ny);

NSYM is a symbol number between 0 and 21. Available symbols are given in the Appendix B.

NX, NY is the centre of the symbol in plot coordinates.

Additional notes:

- The size of symbols can be set with HSYMBL.
- SYMROT (ANGLE) defines a rotation angle for symbols (in degrees). The symbol is rotated in a counter-clockwise direction.

R L S Y M B

RLSYMB plots a symbol where the centre is specified by user coordinates.

The call is: CALL RLSYMB (NSYM, XP, YP) level 2, 3

or: void rlsymb (int nsym, float xp, float yp);

3.4 Plotting a Page Border and Header

P A G E R A

PAGERA plots a border around the page.

The call is:	CALL PAGERA	level 1, 2, 3
or:	void pagera ();	

P A G H D R

PAGHDR plots a page header at a corner of the page. The header line contains date, time and user-defined information.

The call is:	CALL PAGHDR (CSTR1, CSTR2, IOPT, IDIR)	level 1, 2, 3
or:	void paghdr (char *cstr1, char *cstr2, int iopt, int idir);	

CSTR1 is a character string preceding the header line.

CSTR2 is a character string following the header line.

IOPT is the page corner where the header is plotted:

- = 1 is the lower left corner.
- = 2 is the lower right corner.
- = 3 is the upper right corner.
- = 4 is the upper left corner.

IDIR is the direction of the header line:

- = 0 is horizontal.
- = 1 is vertical.

Additional note: The character size of the header line is 0.6 * NH where NH is the parameter used in HEIGHT.

3.5 Sending a Metafile to a Device

A metafile can be converted with a driver program and sent from the operating system to several devices. From within a user program, the SYMFIL routine is used for this purpose.

S Y M F I L

SYMFIL sends a metafile to a device. It must be called after DISFIN.

The call is:	CALL SYMFIL (CDEV, CSTAT)	level 0
or:	void symfil (char *cdev, char *cstat);	

CDEV is the name of the device. 'CONS' refers to the graphics screen, 'XWIN' to a X Window terminal, 'PSCi' to a PostScript printer, 'KYOi' to a Kyocera laserprinter with Prescribe and 'HPLi' to a HP-plotter. The keyword 'NONE' can be used to delete a metafile with no device plotting.

CSTAT is a status parameter and can have the values 'DELETE' and 'KEEP'.

Additional note: SYMFIL calls the DISLIN driver utility DISDRV. The parameter 'REVERS' can be passed to DISDRV from SYMFIL if the routine SCRMOD is called before with the parameter 'REVERS'.

3.6 Including Metafiles into a Graphics

A metafile can be included into a graphics with the routine INCFIL.

I N C F I L

The routine INCFIL includes a GKSLIN or CGM metafile into a graphics.

The call is:	CALL INCFIL (CFIL)	level 1, 2, 3
or:	void incfil (char *cfil);	

CFIL is a character string that contains the filename.

Additional notes:

- The routine FILBOX (NX, NY, NW, NH) defines a rectangular area on the page where the metafile will be included. (NX, NY) are the plot coordinates of the upper left corner, (NW, NH) are the width and length of the box in plot coordinates. By default, the entire page will be used.
- With the statement CALL FILCLR ('NONE'), colour values in a metafile will be ignored. The default is FILCLR ('ALL').

Chapter 4

Plotting Axis Systems and Titles

4.1 Plotting Axis Systems

An axis system defines an area on the page for plotting data. Various axis systems can be plotted to accommodate different applications. For two-dimensional graphics, a maximum of two parallel X- and Y-axes can be drawn. The axis system is scaled to fit the range of data points and can be labeled with values, names and ticks. Two-dimensional axis systems are plotted with a call to the routine GRAF.

G R A F

GRAF plots a two-dimensional axis system.

The call is:	CALL GRAF (XA, XE, XOR, XSTEP, YA, YE, YOR, YSTEP)	level 1
or:	void graf (float xa, float xe, float xor, float xstep, float ya, float ye, float yor, float ystep);	
XA, XE	are the lower and upper limits of the X-axis.	
XOR, XSTEP	are the first X-axis label and the step between labels.	
YA, YE	are the lower and upper limits of the Y-axis.	
YOR, YSTEP	are the first Y-axis label and the step between labels.	

Additional notes:

- GRAF must be called in level 1 and automatically sets the level to 2. When plotting more than 1 axis system on a page, ENDGRF must be called in between each new set of axes in order to set the level back to 1.
- The position of the lower left corner and the size of an axis system can be changed with the routines AXSPOS and AXSLEN.
- The axis scaling is linear by default and can be changed with SCALE. For logarithmic scaling, the corresponding parameters in GRAF must be exponents of base 10.
- One of several label types can be chosen with the routine LABELS or user-defined with MYLAB. Single labels can be suppressed by calling AXENDS.
- The routine NAME defines axis titles.
- The number of ticks between axis labels can be changed with the routine TICKS.
- SETGRF can be used to remove a piece of or complete axis from an axis system.
- If the numerical value of the lower limit of an axis is larger than the upper limit and the label step is negative, axis scaling will be in descending order.

- The routine FRAME defines the thickness of a frame plotted around an axis system. A frame can also be plotted outside of GRAF with the statement CALL BOX2D.
- A crossed axis system can be defined with CALL AXSTYP ('CROSS').

4.2 Termination of Axis Systems

E N D G R F

The routine ENDGRF terminates an axis system and sets the level back to 1.

The call is:	CALL ENDGRF	level 2, 3
or:	void endgrf ();	

4.3 Plotting Titles

T I T L E

This routine plots a title over an axis system. The title may contain up to four lines of text designated with TITLIN.

The call is:	CALL TITLE	level 2, 3
or:	void title ();	
Additional note:	All lines are centred by default but can be left- or right-justified using TITJUS.	

4.4 Plotting Grid Lines

G R I D

The routine GRID overlays a grid on an axis system.

The call is:	CALL GRID (IXGRID, IYGRID)	level 2, 3
or:	void grid (int ixgrid, int iygrid);	
IXGRID, IYGRID	are the numbers of grid lines between labels.	

G R D P O L

The routine GRDPOL plots a polar grid.

The call is:	CALL GRDPOL (IXGRID, IYGRID)	level 2, 3
or:	void grdpol (int ixgrid, int iygrid);	
IXGRID	is the numbers of circles between labels.	
IYGRID	is the numbers of sector lines between 360 degrees.	

Example:

The statements

```
CALL  AXSLEN     (1400,1400)
CALL  GRAF       (-3., 3., -3., 1., -3., 3., -3., 1.)
CALL  GRDPOL     (1, 16)
```

produce the following figure:

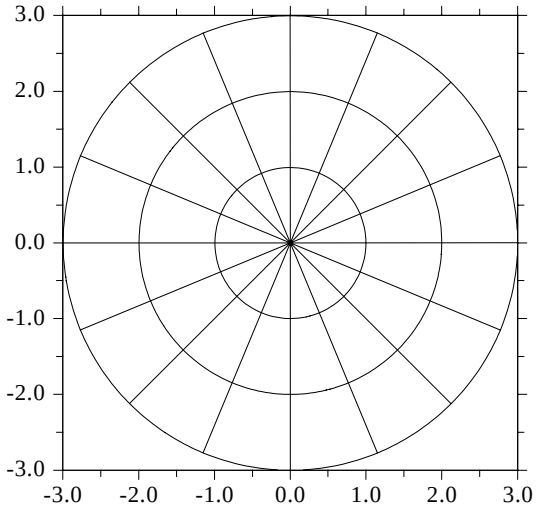

Figure 4.1: GRDPOL

A X G I T

The routine AXGIT plots vertical and horizontal lines through X = 0 and Y = 0.

The call is:	CALL AXGIT	level 2, 3
or:	void axgit ();	
Additional note:	The statement CALL XAXGIT plots only the line Y = 0 while CALL YAXGIT plots only X = 0.	

C R O S S

The routine CROSS plots vertical and horizontal lines with additional ticks through X = 0 and Y = 0.

The call is:	CALL CROSS	level 2, 3
or:	void cross ();	
Additional note:	The statement CALL XCROSS plots only the line Y = 0 while CALL YCROSS plots only X = 0.	

4.5 Secondary Axes

The following routines plot single X- and Y-axes; they are called secondary axes because they do not define or change any of the axis scaling parameters. Secondary axes can be used to add additional labels to the axis systems.

The plotting routines for secondary axes are:

XAXIS	plots a linear X-axis.	level 1, 2, 3
YAXIS	plots a linear Y-axis.	level 1, 2, 3
XAXLG	plots a logarithmic X-axis.	level 1, 2, 3
YAXLG	plots a logarithmic Y-axis.	level 1, 2, 3

The call is: CALL XAXIS (A, B, OR, STEP, NL, CSTR, IT, NX, NY)

or: void xaxis (float a, float b, float or, float step, int nl, char *cstr, int it, int nx, int ny);

A, B	are the lower and upper limits of the axis.
OR, STEP	are the first label and the step between labels.
NL	is the length of the axis in plot coordinates.
CSTR	is a character string containing the axis name.
IT	indicates how ticks, labels and the axis name are plotted. If IT = 0, they are plotted in a clockwise direction. If IT = 1, they are plotted in an counter-clockwise direction.
NX, NY	are the plot coordinates of the axis start point. The X-axis will be plotted from left to right and the Y-axis from bottom to top.

Analog: YAXIS, XAXLG, YAXLG

Additional notes:

- Secondary axes can be called from level 1, 2 or 3. Note again that secondary axes do not change the scaling of an axis system defined by GRAF. Similarly, curves cannot be plotted with only secondary axes, they require a call to GRAF.
- As in GRAF, the parameters of logarithmic axes must be exponents of base 10.
- User-defined labels may also be plotted on secondary axes with MYLAB and the argument 'USER' in the routine LABELS. The number of ticks can be changed by calling TICKS.

Chapter 5

Plotting Curves

This chapter describes how to plot curves with lines and symbols. Several curves can be plotted in one axis system and can be differentiated by colour, line style and pattern. Curve attributes can be plotted in a legend.

5.1 Plotting Curves

C U R V E

CURVE connects data points with lines or plots them with symbols.

The call is:	CALL CURVE (XRAY, YRAY, N)	level 2, 3
or:	void curve (float *xray, float *yray, int n);	
XRAY, YRAY	are arrays that contain X- and Y-coordinates.	
N	is the number of data points.	

Additional notes:

- CURVE must be called after GRAF from level 2 or 3.
- By default, data points that lie outside of an axis system are listed on the screen. The listing can be suppressed with the routine NOCHEK.
- For a logarithmic scaling of an axis, CURVE suppresses the plotting of curves and prints a warning if some corresponding data coordinates have non positive values. After the statement CALL NEGLOG (EPS), where EPS is a small positiv floating-point number, CURVE will use the value EPS for non positive values.
- CURVE suppresses lines outside the borders of an axis system. Suppressing can be disabled with NOCLIP or the margins of suppression can be changed with GRACE.
- INCMRK determines if CURVE plots lines or symbols.
- When plotting several curves, attributes such as colour and line style can be changed automatically by DISLIN or directly by the user. The routine CHNCRV defines which attributes are changed automatically. The routines COLOR or SETCLR are used to define colours, SOLID, DOT, DASH, CHNDOT, CHNDSH, DOTL, DASHM and DASHL to define line styles and MARKER to define symbols plotted with the routine CURVE.
- Different data interpolation methods can be chosen with POLCRV.

5.2 Plotting Legends

To differentiate multiple curves in an axis system, legends with text can be plotted. DISLIN can store up to 30 curve attributes such as symbols, thicknesses, line styles and colours and these can be incorporated in a legend.

Legends are created with the following steps:

(1) define a character variable used to store the lines of text in the legend
(2) initialize the legend
(3) define the lines of text
(4) plot the legend.

The corresponding routines are:

L E G I N I

LEGINI initializes a legend.

The call is: CALL LEGINI (CBUF, NLIN, NMAXLN) level 1, 2, 3
or: void legini (char *cbuf, int nlin, int nmaxln);

CBUF is a character variable used to store the lines of text in the legend. The variable must be defined by the user to have at least NLIN * NMAXLN characters.

NLIN is the number of text lines in the legend.

NMAXLN is the number of characters in the longest line of text.

L E G L I N

LEGLIN stores lines of text for the legend.

The call is: CALL LEGLIN (CBUF, CSTR, ILIN) level 1, 2, 3
or: void leglin (char *cbuf, char *cstr, int ilin);

CBUF see LEGINI.

CSTR is a character string that contains a line of text for the legend.

ILIN is the number of the legend line between 1 and NLIN.

L E G E N D

LEGEND plots legends.

The call is: CALL LEGEND (CBUF, NCOR) level 2, 3
or: void legend (char *cbuf, int ncor);

CBUF see LEGINI.

NCOR indicates the position of the legend:
= 1 is the lower left corner of the page.
= 2 is the lower right corner of the page.
= 3 is the upper right corner of the page.
= 4 is the upper left corner of the page.
= 5 is the lower left corner of the axis system.
= 6 is the lower right corner of the axis system.
= 7 is the upper right corner of the axis system.
= 8 is the upper left corner of the axis system.

Additional notes: The following routines change the position and appearance of a legend. They must be called after LEGINI except for the routines FRAME and LINESP.

- LEGTIT (CTIT) sets the title of the legend.
  Default: CTIT = 'Legende'.
- LEGPOS (NX, NY) defines a global position for the legend where NX and NY are the plot coordinates of the upper left corner. After a call to LEGPOS, the second parameter in LEGEND will be ignored.
- NLX = NXLEGN (CBUF) and NYL = NYLEGN (CBUF) return the length and the height of a legend in plot coordinates.
- FRAME (NFRA) defines the thickness of a frame plotted around a legend.
- LINESP (XF) changes the spacing of lines in a legend.
- LEGCLR retains the same colour for curves and lines of text in the legend.
- The statement CALL MIXLEG enables multiple text lines in legends. By default, the character '/' is used as a newline character but can be changed with the routine SETMIX.

L E G P A T

The routine LEGPAT stores curve attributes plotted in legends. Normally, this is done automatically by routines such as CURVE and BARS.

The call is: CALL LEGPAT (ITYP, ITHK, ISYM, ICLR, IPAT, ILIN) level 1, 2, 3

or: void legpat (int ityp, int ithk, int isym, int iclr, long ipat, int ilin);

ITYP is the line style between -1 and 7 (see LINTYP). IF ITYP = -1, no line will be plotted in the legend line.

ITHK defines the thickness of lines (> 0).

ISYM is the symbol number between -1 and 21. If ISYM = -1, no symbol will be plotted in the legend line.

ICLR is the colour value between -1 and 255. If ICLR = -1, the current colour will be used.

IPAT is the shading pattern (see SHDPAT). If IPAT = -1, no pattern will be plotted in the legend line.

ILIN is the legend line between 1 and NLIN.

Additional notes:
- The routine LEGPAT is useful to create legends without calls to CURVE.
- LEGPAT must be called after LEGINI.

L E G O P T

The routine LEGOPT modifies the appearance of legends.

The call is: CALL LEGOPT (XF1, XF2, XF3) level 1, 2, 3

or: void legopt (float xf1, float xf2, float xf3);

XF1 is a multiplier for the length of the pattern field. The length is XF1 * NH, where NH is the current character height. If XF1 = 0., the pattern field will be suppressed.

XF2 — is a multiplier for the distance between legend frames and text. The distance is XF2 * NH * XSPC, where XSPC is the spacing between legend lines (see LINESP).

XF3 — is a multiplier for the spacing between multiple text lines. The space is XF3 * NH * XLINSP.

Default: (4.0, 0.5, 1.0).

5.3 Plotting Shaded Areas between Curves

S H D C R V

SHDCRV plots a shaded area between two curves.

The call is: CALL SHDCRV (X1RAY, Y1RAY, N1, X2RAY, Y2RAY, N2) level 2, 3

or: void shdcrv (float *x1ray, float *y1ray, int n1, float *x2ray, float *y2ray, int n2);

X1RAY, Y1RAY — are arrays with the X- and Y-coordinates of the first curve. Values are not changed by SHDCRV.

N1 — is the number of points in the first curve.

X2RAY, Y2RAY — are arrays with the X- and Y-coordinates of the second curve. Values are not changed by SHDCRV.

N2 — is the number of points in the second curve.

Additional notes:

- The maximum number of data points cannot be greater than 2000.
- Different shading patterns can be selected with SHDPAT. The pattern number will automatically be incremented by 1 after a call to SHDCRV.
- Legends may be plotted for shaded curves.
- The routine NOARLN will suppress border lines around shaded areas.

5.4 Plotting Error Bars

E R R B A R

The routine ERRBAR plots error bars.

The call is: CALL ERRBAR (XRAY, YRAY, E1RAY, E2RAY, N) level 2, 3

or: void errbar (float *xray, float *yray, float *e1ray, float *e2ray, int n);

XRAY, YRAY — are arrays that contain the X- and Y-coordinates.

E1RAY, E2RAY — are arrays that contain the errors. Lines will be drawn from YRAY - E1RAY to YRAY + E2RAY.

N — is the number of data points.

Additional notes:

- Horizontal bars will be drawn after CALL BARTYP ('HORI').
- A symbol can be selected with MARKER and the symbol size with HSYMBL.

5.5 Plotting Vector Fields

F I E L D

The routine FIELD plots a vector field.

The call is:	CALL FIELD (X1RAY, Y1RAY, X2RAY, Y2RAY, N, IVEC)	level 2, 3
or:	void field (float *x1ray, float *y1ray, float *x2ray, float *y2ray, int n, int ivec);	
X1RAY, Y1RAY	are arrays that contain the X- and Y-coordinates of the start points.	
X2RAY, Y2RAY	are arrays that contain the X- and Y-coordinates of the end points.	
N	is the number of vectors.	
IVEC	is a four digit number that specifies the vector (see VECTOR).	

Chapter 6

Parameter Setting Routines

All parameters in DISLIN have default values set by the initialization routine DISINI. This chapter summarizes subroutines that allow the user to alter default values. The following routines can be called from level 1, 2 or 3 except for those noted throughout the chapter. Subroutines that can only be called from level 0 must appear before DISINI. In general, parameter setting routines should be called between DISINI and the plotting routines they affect.

6.1 Basic Routines

R E S E T

RESET sets parameters back to their default values.

The call is:	CALL RESET (CNAME)	level 1, 2, 3
or:	void reset (char *cname);	
CNAME	is a character string containing the name of the routine whose parameters will be set back to default values. If CNAME = 'ALL', all parameters in DISLIN will be reset.	

U N I T

UNIT defines the logical unit used for printing error messages and listing data points that lie outside of the axis scaling.

The call is:	CALL UNIT (NU)	level 1, 2, 3
or:	void unit (FILE *nu);	
NU	is the logical unit. If NU = 0, all messages will be suppressed. Default: NU = 6	

O R I G I N

In DISLIN, all lines are plotted relative to the origin which is a point located in the upper left corner of the page. Modifying this point by ORIGIN produces a shifting of plot vectors on the page.

The call is:	CALL ORIGIN (NX0, NY0)	level 1
or:	void origin (int nx0, int ny0);	
NX0, NY0	are the coordinates of the origin.	Default: (0, 0).

C O L O R

COLOR defines the colours used for plotting text and lines.

The call is:	CALL COLOR (CNAME)	level 1, 2, 3
or:	void color (char *cname);	

CNAME — is a character string that can have the values 'BLACK', 'RED', 'GREEN', 'BLUE', 'CYAN', 'YELLOW', 'ORANGE', 'MAGENTA', 'WHITE' and 'FORE'. The keyword 'FORE' resets the color to the default value.

Additional note: Colours can also be defined with SETCLR which selects a colour index from an actual colour table (see chapter 11).

P A G E

PAGE determines the size of the page.

The call is:	CALL PAGE (NXP, NYP)	level 0
or:	void page (int nxp, int nyp);	

NXP, NYP — are the length and height of the page in plot coordinates. The lower right corner of the page is the point (NXP-1, NYP-1).

Default: (2970, 2100).

S E T P A G

SETPAG selects a predefined page format.

The call is:	CALL SETPAG (CPAGE)	level 0
or:	void setpag (char *cpage);	

CPAGE — is a character string that defines the page format.

= 'DA4L'	DIN A4,	landscape,	2970 * 2100 points.
= 'DA4P'	DIN A4,	portrait,	2100 * 2970 points.
= 'DA3L'	DIN A3,	landscape,	4200 * 2970 points.
= 'DA3P'	DIN A3,	portrait,	2970 * 4200 points.
= 'DA2L'	DIN A2,	landscape,	5940 * 4200 points.
= 'DA2P'	DIN A2,	portrait,	4200 * 5940 points.
= 'DA1L'	DIN A1,	landscape,	8400 * 5940 points.
= 'DA1P'	DIN A1,	portrait,	5940 * 8400 points.
= 'PS4L'	PostScript A4,	landscape,	2800 * 1950 points.
= 'PS4P'	PostScript A4,	portrait,	1950 * 2800 points.
= 'KY4L'	Kyocera A4,	landscape,	2870 * 2000 points.
= 'KY4P'	Kyocera A4,	portrait,	2000 * 2870 points.
= 'HP4L'	HP-plotter A4,	landscape,	2718 * 1900 points.
= 'HP4P'	HP-plotter A4,	portrait,	1900 * 2718 points.
= 'HP3L'	HP-plotter A3,	landscape,	3992 * 2718 points.
= 'HP3P'	HP-plotter A3,	portrait,	2718 * 3992 points.
= 'HP2L'	HP-plotter A2,	landscape,	5340 * 3360 points.
= 'HP2P'	HP-plotter A2,	portrait,	3360 * 5340 points.
= 'HP1L'	HP-plotter A1,	landscape,	7570 * 5340 points.
= 'HP1P'	HP-plotter A1,	portrait,	5340 * 7570 points.

Default: CPAGE = 'DA4L'.

M E T A F L

METAFL defines the metafile format.

The call is: CALL METAFL (CFMT) level 0

or: void metafl (char *cfmt);

CFMT is a character string that defines the file format.

= 'GKSL' defines a GKSLIN metafile.

= 'CGM' defines a CGM metafile.

= 'POST' defines a greyscaled PostScript file. The colour table 'RGREY' is loaded by DISINI.

= 'PSCL' defines a coloured PostScript file. The background is filled black and the colour table 'RAINBOW' is loaded by DISINI.

= 'KYOC' defines a Kyocera file.

= 'HPGL' defines a HPGL file.

= 'JAVA' defines a Java applet file.

= 'TIFF' defines a TIFF file.

= 'IMAG' defines an image file.

= 'VIRT' defines a virtual file. The metafile is hold in a raster format in computer memory and can be saved on a file with the routines RIMAGE and RTIFF.

= 'CONS' defines a graphics output on the screen.

= 'XWIN' defines an X Window display.

= 'XWIi' defines an X Window display, where i is the window number between 1 and 5. By default, window 1 is situated in the lower right corner, window 2 in the upper right corner, window 3 in the upper left corner, window 4 in the lower left corner and window 5 in the centre of the screen.

Default: CFMT = 'GKSL'.

Notes:

- The default size of JAVA, TIFF, IMAGE and virtual files is set to 853 x 603 points but can be modified with the routine WINSIZ.
- JAVA applet files created by DISLIN can be compiled with Java and then displayed in a browser. The class names of the applets are identical with the filenames of the output files. They can be changed with the routine SETFIL.
- The default colour table loaded by DISINI is 'RGREY' for greyscaled PostScript files and 'RAINBOW' for the other file formats. Coloured PostScript files with a white background can be created with the keyword 'PSCL' and the statement CALL SCRMOD ('REVERS') before DISINI, or with the keyword 'POST' and the statement CALL SETVLT ('RAINBOW') after DISINI.

S E T F I L

By default, the plotfile name consists of the keyword 'dislin' and an extension that depends on the file format. An alternate filename can be set with SETFIL.

The call is: CALL SETFIL (CFIL) level 0

or: void setfil (char *cfil);

CFIL is a character string that contains the filename.

E R R F I L

By default, the name of the error file is 'dislin.err'. An alternate filename can be set with ERRFIL.

The call is: CALL ERRFIL (CFIL) level 0

or: void errfil (char *cfil);

CFIL is a character string that contains the filename.

E R R D E V

The routine ERRDEV defines the output device for DISLIN warnings. By default, warnings are written to the screen.

The call is: CALL ERRDEV (COPT) level 0

or: void errdev (char *copt);

COPT is a character string that can have the values 'CONS' and 'FILE'.
Default: COPT = 'CONS'.

S C L F A C

SCLFAC sets the scaling factor for an entire plot.

The call is: CALL SCLFAC (XFAC) level 0

or: void sclfac (float xfac);

XFAC is the scaling factor by which the entire plot is scaled up or down. Default: XFAC = 1.

S C L M O D

The method by which graphics are scaled to the hardware pages of devices such as a graphics terminal can be selected with the routine SCLMOD.

The call is: CALL SCLMOD (CMOD) level 0

or: void sclmod (char *cmod);

CMOD = 'DOWN' means that graphics will be scaled down if the hardware page of a device is smaller than the plotting page.

= 'FULL' means that the graphics will be scaled up or down depending upon the size of the hardware page.

Default: CMOD = 'DOWN'.

Additional notes:
- The size of a graphics screen will be interpreted as DIN A4 landscape. This means that by default graphics which are smaller than DIN A4 will not fill the entire screen.
- SCLFAC and SCLMOD can affect each other.

F I L M O D

The routine FILMOD determines if a new plotfile name is created for existing files.

The call is: CALL FILMOD (CMOD) level 0, 1, 2, 3

or: void filmod (char *cmod);

CMOD is a character string containing the mode.

= 'COUNT' means that a new file version will be created.

= 'DELETE' means that the existing file will be overwritten.

= 'BREAK' means that the program will be terminated by DISINI.

Default: CMOD = 'COUNT'.

P A G M O D

GKSLIN and CGM files can be rotated by 90 degrees to use the full hardware page of a device. In general, this is done automatically by the driver program.

The call is: CALL PAGMOD (CMOD) level 0

or: void pagmod (char *cmod);

CMOD = 'LAND' means that the metafile is not rotated.

= 'PORT' means that the metafile is rotated by 90 degrees.

= 'NONE' can be used to disable automatic plotfile rotation in the driver program (i.e. for PostScript files).

Default: CMOD = 'LAND'.

Figure 6.1 shows the effect of PAGMOD:

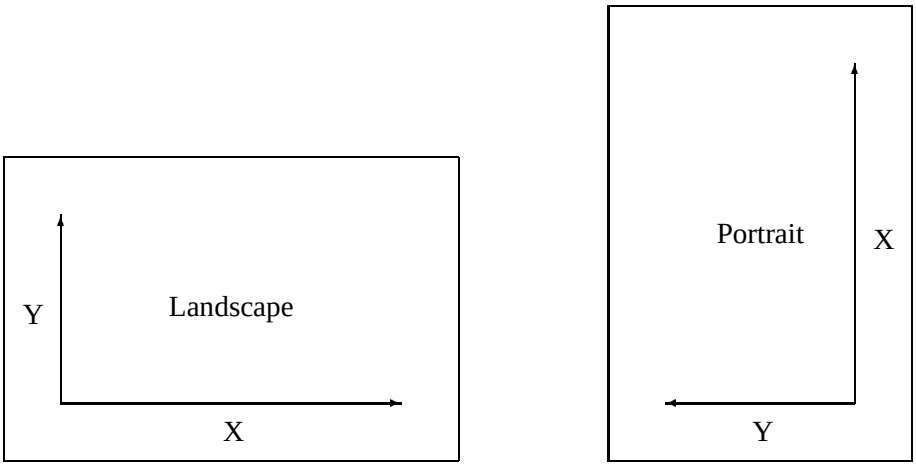

Figure 6.1: PAGMOD

S C R M O D

Normally, the background of screens and coloured PostScript files is set to 'BLACK' and the foreground colour is set to 'WHITE'. With the routine SCRMOD, the back and foreground colours can be swapped without changing the colour table.

The call is: CALL SCRMOD (CMOD) level 0

or: void scrmod (char *cmod);

CMOD = 'REVERS' means that the background colour is set to 'WHITE' and the foreground colour to 'BLACK'.

CMOD = 'NOREV' means that the background colour is set to 'BLACK' and the foreground colour to 'WHITE'.

Default: CMOD = 'NOREV'.

N E W P A G

NEWPAG creates a new page.

The call is:	CALL NEWPAG	level 1
or:	void newpag ();	

Additional notes:
- NEWPAG is not useful for GKSLIN files.
- The number of pages for CGM files is limited to a length of 327 cm in the X-direction.
- On X Window terminals, NEWPAG is waiting for a mouse button 2 event before displaying the next page. On other terminals, NEWPAG has the same effect as ERASE.

W I N D O W

This routine defines, for X Window terminals, a region on the screen where the graphics will be displayed. By default, the window size is set to 2/3 of the screen size and located in the lower right corner of the screen.

The call is:	CALL WINDOW (NX, NY, NW, NH)	level 0, 1, 2, 3
or:	void window (int nx, int ny, int nw, int nh);	

NX, NY — are the screen coordinates of the upper left corner.

NW, NH — are the width and height of the window in screen coordinates.

Additional note: In general, the screen size is 1280 * 1024 pixels.

W I N S I Z

This routine defines the size of windows. By default, the window size is set to 2/3 of the screen size.

The call is:	CALL WINSIZ (NW, NH)	level 0, 1, 2, 3
or:	void winsiz (int nw, int nh);	

NW, NH — are the width and height of the window in screen coordinates.

W I N A P P

The routine WINAPP defines if a DISLIN program should look like a Windows console, or more like a Windows program. If Windows mode is selected, all warnings are written to an error file and the protocol in disfin is displayed in a widget.

The call is:	CALL WINAPP (COPT)	level 0
or:	void winapp (char *copt);	

COPT — is a character string that can have the values 'CONSOLE' and 'WINDOWS'.
Default: COPT = 'CONSOLE'.

H W P A G E

The routine HWPAGE defines the size of the PostScript hardware page.

The call is:	CALL HWPAGE (NW, NH)	level 0

or: void hwpage (int nw, int nh);

NW, NH are the width and height of the PostScript hardware page in plot coordinates.
Default: (1950, 2800).

H W O R I G

The routine HWORIG defines the hardware origin of the PostScript hardware page.

The call is: CALL HWORIG (NX, NY) level 0

or: void hworig (int nx, int ny);

NX, NY are the plot coordinates of the hardware origin.
Default: (75, 100).

S E T X I D

The routine SETXID defines an external X graphics window or pixmap for DISLIN. All graphical output is sent to the external window or pixmap.

The call is: CALL SETXID (ID, CTYPE) level 0

or: void setxid (int id, char *ctype);

ID is the window or pixmap ID.

CTYPE is a character string that can have the values 'WINDOW' and 'PIXMAP'.

Additional notes:

- If an external pixmap is used, backing store must also be enabled with the routine X11MOD.
- An external window is not erased by DISINI. This can be done with the routine ERASE.
- External windows are not blocked in DISFIN (see WINMOD).

6.2 Axis Systems

This section describes subroutines that allow the user to modify axis systems. The position of an axis system, the size, the scaling, ticks, labels and axis titles can be altered in any way. Some of the routines defining axis attributes can also be used with secondary axes. Routines that set axis attributes can be used for one or for any combination of axes. The axes are identified by a character string that can contain the characters 'X', 'Y' and 'Z' in any combination.

6.2.1 Modifying the Type

A X S T Y P

The routine AXSTYP defines the type of an axis system. Axis systems can be plotted as rectangles or in a crossed form. For crossed axis systems, the scaling must be linear and the axis limits must contain the origin.

The call is:	CALL AXSTYP (COPT)	level 1
or:	void axstyp (char *copt);	
COPT	is a character string defining the type.	
= 'RECT'	defines a rectangular axis system.	
= 'CROSS'	defines a crossed axis system.	

Default: COPT = 'RECT'.

The following figure shows a rectangular and a crossed axis system:

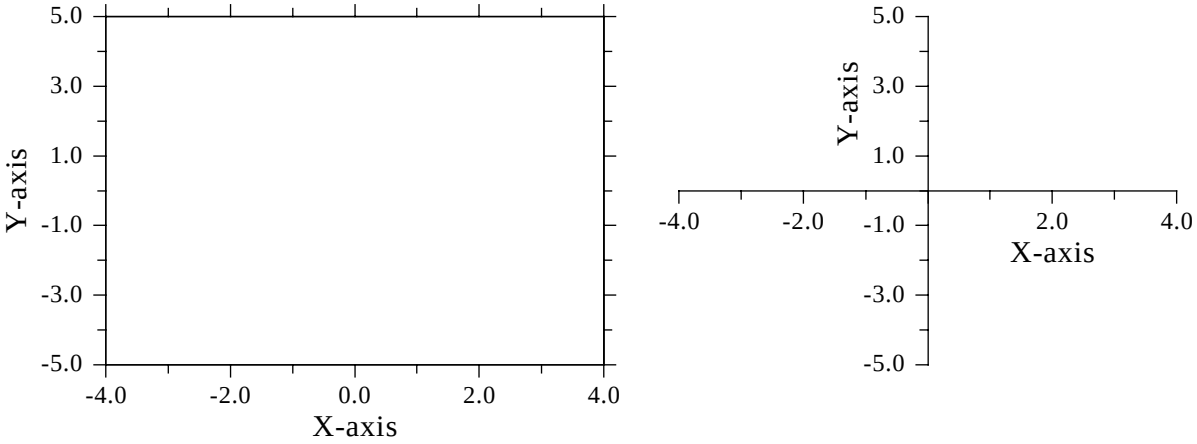

Figure 6.2: Rectangular and Crossed Axis Systems

6.2.2 Modifying the Position and Size

A X S P O S

AXSPOS determines the position of an axis system.

The call is:	CALL AXSPOS (NXA, NYA)	level 1
or:	void axspos (int nxa, int nya);	
NXA, NYA	are plot coordinates that define the lower left corner of an axis system. By default, axis systems are centred in the X-direction while NYA is set to the value (page height - 300).	

A X S O R G

AXSORG is an alternate routine for defining the position of a crossed axis system.

The call is: CALL AXSORG (NX, NY) level 1

or: void axsorg (int nx, int ny);

NX, NY are plot coordinates that define the position of the origin of a crossed axis system.

A X S L E N

AXSLEN defines the size of an axis system.

The call is: CALL AXSLEN (NXL, NYL) level 1

or: void axslen (int nxl, int nyl);

NXL, NYL are the length and height of an axis system in plot coordinates. The default values are set to 2/3 of the page length and height.

C E N T E R

A call to the routine CENTER will centre the axis system on the page. All elements of an axis system, including titles, axis labels and names, will be taken into consideration. The centralisation is done by GRAF through changing the position of the origin. Therefore, all plotting routines called after GRAF will work with the new origin.

The call is: CALL CENTER level 1, 2, 3

or: void center ();

Additional notes:
- If there are several axis systems on the page, the origin will be changed only by the first call to GRAF.
- The character height of titles should be defined with HTITLE if it is different from the current character height in GRAF.

6.2.3 Axis Scaling

AXSSCL

This routine sets the axis scaling to logarithmic or linear.

The call is: CALL AXSSCL (CSCL, CAX) level 1, 2, 3

or: void axsscl (char *cscl, char *cax);

CSCL = 'LIN' denotes linear scaling.

= 'LOG' denotes logarithmic scaling.

CAX is a character string that defines the axes.

Default: ('LIN', 'XYZ').

Additional notes: - For logarithmic scaling, the corresponding parameters in GRAF must be exponents of base 10.

- The routine AXSSCL replaces the DISLIN routine SCALE because SCALE is also a Fortran 90 intrinsic function.

S E T S C L

The parameters in GRAF will be calculated automatically by DISLIN if the routine SETSCL is used. In this case, GRAF must have dummy parameters in which DISLIN returns the calculated values.

The call is: CALL SETSCL (XRAY, N, CAX) level 1, 2, 3

or: void setscl (float *xray, int n, char *cax);

XRAY is a vector that contains user coordinates. SETSCL calculates the minimum and maximum values of the data and stores them in a common block.

N is the number of points in XRAY.

CAX is a character string that defines the axes.

Additional notes:
- SETSCL can be used with linear and logarithmic scaling and with all label types.
- The calculation of scaling and label values is done by GRAF. The minimum and maximum of the data are always used for the lower and upper limits of an axis while even values are calculated for the labels.
- The number of digits after the decimal point will be set automatically.
- If the scaling of an axis is logarithmic, labels will be plotted with the format 'LOG'.

6.2.4 Modifying Ticks

T I C K S

This routine is used to define the number of ticks between axis labels.

The call is: CALL TICKS (NTIC, CAX) level 1, 2, 3

or: void ticks (int ntic, char *cax);

NTIC is the number of ticks (≥ 0).

CAX is a character string that defines the axes.

Default: (2, 'XYZ').

T I C P O S

This routine defines the position of ticks.

The call is: CALL TICPOS (CPOS, CAX) level 1, 2, 3

or: void ticpos (char *cpos, char *cax);

CPOS is a character string defining the position.

= 'LABELS' means that ticks will be plotted on the same side as labels.

= 'REVERS' means that ticks will be plotted inside of an axis system.

= 'CENTER' means that ticks will be centred on the axis line.

CAX is a character string that defines the axes.

Default: ('LABELS', 'XYZ').

T I C L E N

TICLEN sets the lengths of major and minor ticks.

The call is:	CALL TICLEN (NMAJ, NMIN)	level 1, 2, 3
or:	void ticlen (int nmaj, int nmin);	
NMAJ	is the length of major ticks in plot coordinates (> 0).	
NMIN	is the length of minor ticks in plot coordinates (> 0).	
	Default: (24, 16).	

L O G T I C

The appearance of minor ticks on logarithmic axes differs slightly from linear axes. By default, logarithmic minor ticks are generated automatically if the label step is 1 or -1 and if the number of ticks in TICKS is greater than 1. If the step has another value, minor ticks are plotted as specified in TICKS. This algorithm can be modified with LOGTIC.

The call is:	CALL LOGTIC (CMOD)	level 1, 2, 3
or:	void logtic (char *cmod);	
CMOD	is a character string defining the appearance of logarithmic ticks.	
= 'AUTO'	defines default ticks.	
= 'FULL'	means that logarithmic minor ticks will be generated for every cycle even if the label step is not 1 but some other integer.	
	Default: CMOD = 'AUTO'.	

6.2.5 Modifying Labels

L A B E L S

LABELS determines which label types will be plotted on an axis.

The call is:	CALL LABELS (CLAB, CAX)	level 1, 2, 3
or:	void labels (char *clab, char *cax);	

CLAB — is a character string that defines the labels.

- = 'NONE' — will suppress all axis labels.
- = 'FLOAT' — will plot labels in floating-point format.
- = 'EXP' — will plot floating-point labels in exponential format where fractions range between 1 and 10.
- = 'FEXP' — will plot labels in the format fEn where f ranges between 1 and 10.
- = 'LOG' — will plot logarithmic labels with base 10 and the corresponding exponents.
- = 'CLOG' — is similar to 'LOG' except that the entire label is centred below the tick mark; with 'LOG', only the base '10' is centred.
- = 'ELOG' — will plot only the logarithmic values of labels.
- = 'TIME' — will plot time labels in the format 'hhmm'.
- = 'HOURS' — will plot time labels in the format 'hh'.
- = 'SECONDS' — will plot time labels in the format 'hhmmss'.
- = 'MAP' — defines geographical labels which are plotted as non negative floating-point numbers with the following characters 'W', 'E', 'N' and 'S'.
- = 'LMAP' — is simular to 'MAP' except that lowercase characters are used.
- = 'DMAP' — selects labels that are plotted as floating-point numbers with degree symbols.
- = 'MYLAB' — selects labels that are defined with the routine MYLAB.

CAX — is a character string that defines the axes.

Default: ('FLOAT', 'XYZ').

Additional notes:

- The values 'LOG', 'CLOG' and 'ELOG' can be only used with logarithmic scaling. If these label types are used with linear scaling, DISLIN will change them to 'FLOAT'.
- For the values 'TIME', 'HOURS' and 'SECONDS', the corresponding parameters in GRAF must be in seconds since midnight.

M Y L A B

MYLAB defines user labels.

The call is:	CALL MYLAB (CSTR, ITICK, CAX)	level 1, 2, 3
or:	void mylab (char *cstr, int itick, char *cax);	

CSTR — is a character string containing a label ($\leq$ 16 characters).

ITICK — is the tick number where the label will be plotted ($\leq$ 20). Tick numbering starts with 1.

CAX — is a character string that defines the axes.

L A B T Y P

LABTYP defines horizontal or vertical labels.

The call is:	CALL LABTYP (CTYPE, CAX)	level 1, 2, 3
or:	void labtyp (char *ctype, char *cax);	

CTYPE	is a character string defining the direction.
= 'HORI'	defines horizontal labels.
= 'VERT'	defines vertical labels.
CAX	is a character string that defines the axes.

Default: ('HORI', 'XYZ').

L A B P O S

LABPOS defines the position of labels.

The call is:	CALL LABPOS (CPOS, CAX)	level 1, 2, 3
or:	void labpos (char *cpos, char *cax);	

CPOS	is a character string defining the position.
= 'TICKS'	means that labels will be plotted at major ticks.
= 'CENTER'	means that labels will be centred between major ticks.
= 'SHIFT'	means that the starting and end labels will be shifted.
CAX	is a character string that defines the axes.

Default: ('TICKS', 'XYZ').

L A B J U S

LABJUS defines the alignment of axis labels.

The call is:	CALL LABJUS (CJUS, CAX)	level 1, 2, 3
or:	void labjus (char *cjus, char *cax);	

CJUS	is a character string defining the alignment of labels.
= 'AUTO'	means that labels are automatically justified.
= 'LEFT'	means that labels are left-justified.
= 'RIGHT'	means that labels are right-justified.
= 'OUTW'	means that labels are left-justified on the left and lower axes of an axis system. On the right and upper axes, labels are right-justified.
= 'INWA'	means that labels are right-justified on the left and lower axes of an axis system. On the right and upper axes, labels are left-justified.
CAX	is a character string that defines the axes.

Default: ('AUTO', 'XYZ').

L A B D I G

This routine sets the number of digits after the decimal point displayed in labels.

The call is:	CALL LABDIG (NDIG, CAX)	level 1, 2, 3
or:	void labdig (int ndig, char *cax);	

NDIG	= -1	defines integer labels.
	= 0	defines integer labels followed by a decimal point.
	= n	defines the number of digits after the decimal point. The last digit will be rounded up.
CAX		is a character string that defines the axes. Default: (1, 'XYZ').
Additional note:		The routine LABDIG replaces the DISLIN routine DIGITS because DIGITS is also a Fortran 90 intrinsic function.

I N T A X

With the routine INTAX, all axes will be labeled with integers.

The call is:	CALL INTAX	level 1, 2, 3
or:	void intax ();	

L A B D I S

This routine sets the distance between labels and ticks.

The call is:	CALL LABDIS (NDIS, CAX)	level 1, 2, 3
or:	void labdis (int ndis, char *cax);	

NDIS	is the distance in plot coordinates.
CAX	is a character string that defines the axes. Default: (24, 'XYZ').

T I M O P T

With TIMOPT time labels can be plotted in the format 'hh:mm'. The default is 'hhmm'.

The call is:	CALL TIMOPT	level 1, 2, 3
or:	void timopt ();	

R G T L A B

The routine RGTLAB right-justifies user labels. By default, user labels are left-justified.

The call is:	CALL RGTLAB	level 1, 2, 3
or:	void rgtlab ();	

6.2.6 Modifying Axis Titles

N A M E

NAME defines axis titles.

The call is:	CALL NAME (CSTR, CAX)	level 1, 2, 3
or:	void name (char *cstr, char *cax);	
CSTR	is a character string containing the axis title ($\leq$ 60 characters).	
CAX	is a character string that defines the axes. Default: (' ', 'XYZ').	

H N A M E

HNAME defines the character height for axis names.

The call is:	CALL HNAME (NHNAME)	level 1, 2, 3
or:	void hname (int nhname);	
NHNAME	is the character height in plot coordinates. Default: NHNAME = 36	

N A M D I S

NAMDIS sets the distance between axis names and labels.

The call is:	CALL NAMDIS (NDIS, CAX)	level 1, 2, 3
or:	void namdis (int ndis, char *cax);	
NDIS	is the distance in plot coordinates.	
CAX	is a character string that defines the axes. Default: (30, 'XYZ').	

N A M J U S

The routine NAMJUS defines the alignment of axis titles.

The call is:	CALL NAMJUS (CJUS, CAX)	level 1, 2, 3
or:	void namjus (char *cjus, char *cax);	
CJUS	is a character string that can have the values 'CENT', 'LEFT' and 'RIGHT'.	
CAX	is a character string that defines the axes. Default: ('CENT', 'XYZ').	

R V Y N A M

The routine RVYNAM is used to plot names on right Y-axes and colour bars at an angle of 90 degrees. By default, they are plotted at an angle of 270 degrees.

The call is:	CALL RVYNAM	level 1, 2, 3
or:	void rvynam ();	

6.2.7 Suppressing Axis Parts

N O L I N E

After a call to NOLINE the plotting of axis lines will be suppressed.

The call is:	CALL NOLINE (CAX)	level 1, 2, 3
or:	void noline (char *cax);	

CAX is a character string that defines the axes.

A X E N D S

With a call to AXENDS certain labels can be suppressed.

The call is:	CALL AXENDS (COPT, CAX)	level 1, 2, 3
or:	void axends (char *copt, char *cax);	

COPT is a character string that defines which labels will be suppressed.

= 'NONE'	means that all labels will be displayed.
= 'FIRST'	means that only the starting label will be plotted.
= 'NOFIRST'	means that the starting label will not be plotted.
= 'LAST'	means that only the ending label will be plotted.
= 'NOLAST'	means that the ending label will not be plotted.
= 'ENDS'	means that only the start and end labels will be plotted.
= 'NOENDS'	means that start and end labels will be suppressed.

CAX is a character string that defines the axes.

Default: ('NONE', 'XYZ').

N O G R A F

The routine NOGRAF suppresses the plotting of an axis system.

The call is:	CALL NOGRAF	level 1
or:	void nograf ();	

A X 2 G R F

The routine AX2GRF suppresses the plotting of the upper X- and left Y-axis.

The call is:	CALL AX2GRF	level 1, 2, 3
or:	void ax2grf ();	

S E T G R F

SETGRF removes a part of an axis or a complete axis from an axis system.

The call is:	CALL SETGRF (C1, C2, C3, C4)	level 1, 2, 3
or:	void setgrf (char *c1, char *c2, char *c3, char *c4);	

Ci — are character strings corresponding to the four axes of an axis system. C1 corresponds to the lower X-axis, C2 to the left Y-axis, C3 to the upper X-axis and C4 to the right Y-axis. The parameters can have the values 'NONE', 'LINE', 'TICKS', 'LABELS' and 'NAME'. With 'NONE', complete axes will be suppressed, with 'LINE', only axis lines will be plotted, with 'TICKS', axis lines and ticks will be plotted, with 'LABELS' axis lines, ticks and labels will be plotted and with 'NAME', all axis elements will be displayed.
Default: ('NAME', 'NAME', 'TICKS', 'TICKS').

Additional notes:
- By default, GRAF plots a frame of thickness 1 around axis systems. Therefore, in addition to the parameter 'NONE', FRAME should be called with the parameter 0 for suppressing complete axes.
- SETGRF does not reset the effect of NOGRAF and NOLINE. This must be done using RESET.

6.2.8 Modifying Clipping

C L P W I N

The routine CLPWIN defines a rectangular clipping area on the page.

The call is: CALL CLPWIN (NX, NY, NW, NH) — level 1, 2, 3
or: void clpwin (int nx, int ny, int nw, int nh);

NX, NY — are the plot coordinates of the upper left corner.

NW, NH — are the width and height of the rectangle in plot coordinates.

C L P B O R

The routine CLPBOR sets the clipping area to the entire page or to the axis system.

The call is: CALL CLPBOR (COPT) — level 1, 2, 3
or: void clpbor (char *copt);

COPT — is a character string that can have the values 'PAGE' and 'AXIS'.
Default: COPT = 'PAGE'.

N O C L I P

The suppressing of lines outside of the borders of an axis system can be disabled with NOCLIP.

The call is: CALL NOCLIP — level 1, 2, 3
or: void noclip ();

G R A C E

GRACE defines a margin around axis systems where lines will be clipped.

The call is: CALL GRACE (NGRA) — level 1, 2, 3
or: void grace (int ngra);

NGRA — is the width of the margin in plot coordinates. If NGRA is negative, lines will be clipped inside the axis system.
Default: NGRA = -1

6.2.9 Framing Axis Systems

F R A M E

FRAME defines the thickness of frames plotted by routines such as GRAF and LEGEND.

The call is:	CALL FRAME (NFRM)	level 1, 2, 3
or:	void frame (int nfrm);	

NFRM — is the thickness of the frame in plot coordinates. If NFRM is negative, the frame will be thickened from the inside. If positive, the frame will be thickened towards the outside.

Default: NFRM = 1

6.2.10 Setting Colours

A X C L R S

AXCLRS selects colours for single parts of axes.

The call is:	CALL AXCLRS (NCLR, COPT, CAX)	level 1, 2, 3
or:	void axclrs (int nclr, char *copt, char *cax);	

NCLR — is a colour number between -1 and 255. If NCLR = -1, the actual colour is used.

COPT — is a character string that can have the values 'LINE', 'TICKS', 'LABELS', 'NAME' and 'ALL'.

CAX — is a character string that defines the axes.

Default: (-1, 'ALL', 'XYZ').

6.2.11 Axis System Titles

T I T L I N

This subroutine defines up to four lines of text used for axis system titles. The text can be plotted with TITLE after a call to GRAF.

The call is:	CALL TITLIN (CSTR, N)	level 1, 2, 3
or:	void titlin (char *cstr, int n);	

CSTR — is a character string ($\leq$ 60 characters).

N — is an integer that contains a value between 1 and 4 or -1 and -4. If N is negative, the line will be underscored.

Default: All lines are filled with blanks.

T I T J U S

The routine TITJUS defines the alignment of title lines.

The call is:	CALL TITJUS (CJUS)	level 1, 2, 3
or:	void titjus (char *cjus);	

CJUS — is a character string that can have the values 'CENT', 'LEFT' and 'RIGHT'.
Default: CJUS = 'CENT'.

L F T T I T

Title lines are centred above axis systems by default but can be left-justified with a call to LFTTIT. This routine has the same meaning as TITJUS ('LEFT').

The call is: CALL LFTTIT — level 1, 2, 3
or: void lfttit ();

T I T P O S

The routine TITPOS defines the position of title lines which can be plotted above or below axis systems.

The call is: CALL TITPOS (CPOS) — level 1, 2, 3
or: void titpos (char *cpos);

CPOS — is a character string that can have the values 'ABOVE' and 'BELOW'.
Default: CPOS = 'ABOVE'.

L I N E S P

LINESP defines the spacing between title and legend lines.

The call is: CALL LINESP (XFAC) — level 1, 2, 3
or: void linesp (float xfac);

XFAC — The space between lines is set to XFAC * character height.
Default: XFAC = 1.5

H T I T L E

HTITLE defines the character height for titles. The character height defined by HEIGHT will be used if HTITLE is not called.

The call is: CALL HTITLE (NHCHAR) — level 1, 2, 3
or: void htitle (int nhchar);

NHCHAR — is the character height in plot coordinates.

V K Y T I T

The space between titles and axis systems can be enlarged or reduced with VKYTIT. By default, the space is 2 * character height.

The call is: CALL VKYTIT (NV) — level 1, 2, 3
or: void vkytit (int nv);

NV — is an integer that determines the spacing between axis systems and titles. If NV is negative, the space will be reduced by NV plot coordinates. If NV is positive, the space will be enlarged by NV plot coordinates.
Default: NV = 0

6.3 Text and Numbers

H E I G H T

HEIGHT defines the character height.

The call is:	CALL HEIGHT (NHCHAR)	level 1, 2, 3
or:	void height (int nhchar);	

NHCHAR is the character height in plot coordinates.

Default: NHCHAR = 36

A N G L E

This routine modifies the direction of text plotted with the routines MESSAG, NUMBER, RLMESS and RLNUMB.

The call is:	CALL ANGLE (NDEG)	level 1, 2, 3
or:	void angle (int ndeg);	

NDEG is an angle measured in degrees and a counter-clockwise direction.

Default: NDEG = 0

T X T J U S

The routine TXTJUS defines the alignment of text plotted with the routines MESSAG and NUMBER.

The call is:	CALL TXTJUS (CJUS)	level 1, 2, 3
or:	void txtjus (char *cjus);	

CJUS is a character string that can have the values 'LEFT', 'RIGHT' and 'CENT'. The starting point of text and numbers will be interpreted as upper left, upper right and upper centre point.

Default: CJUS = 'LEFT'.

F R M E S S

FRMESS defines the thickness of frames around text plotted by MESSAG.

The call is:	CALL FRMESS (NFRM)	level 1, 2, 3
or:	void frmess (int nfrm);	

NFRM is the thickness of frames in plot coordinates. If NFRM is negative, frames will be thickened from the inside. If positive, frames will be thickened towards the outside.

Default: NFRM = 0

N U M F M T

NUMFMT modifies the format of numbers plotted by NUMBER and RLNUMB.

The call is:	CALL NUMFMT (COPT)	level 1, 2, 3
or:	void numfmt (char *copt);	

COPT is a character string defining the format.

= 'FLOAT'	will plot numbers in floating-point format.
= 'EXP'	will plot numbers in exponential format where fractions range between 1 and 10.
= 'FEXP'	will plot numbers in the format fEn where f ranges between 1 and 10.
= 'LOG'	will plot numbers logarithmically with base 10 and the corresponding exponents. The exponents must be passed to NUMBER and RLNUMB.

Default: COPT = 'FLOAT'.

Additional note: SETEXP and SETBAS alter the position and size of exponents.

N U M O D E

NUMODE alters the appearance of numbers plotted by NUMBER and RLNUMB.

The call is: CALL NUMODE (CDEC, CGRP, CPOS, CFIX) level 1, 2, 3

or: void numode (char *cdec, char *cgrp, char *cpos, char *cfix);

CDEC	is a character string that defines the decimal notation.
= 'POINT'	defines a point.
= 'COMMA'	defines a comma.
CGRP	is a character string that defines the grouping of 3 digits.
= 'NONE'	means no grouping.
= 'SPACE'	defines a space as separator.
= 'POINT'	defines a point as separator.
= 'COMMA'	defines a comma as separator.
CPOS	is a character string that defines the sign preceding positive numbers.
= 'NONE'	means no preceding sign.
= 'SPACE'	defines a space as a preceding sign.
= 'PLUS'	defines a plus as a preceding sign.
CFIX	is a character string specifying character spacing.
= 'NOEQUAL'	is used for proportional spacing.
= 'EQUAL'	is used for non-proportional spacing.

Default: ('POINT','NONE','NONE','NOEQUAL').

C H A S P C

CHASPC affects intercharacter spacing.

The call is: CALL CHASPC (XSPC) level 1, 2, 3

or: void chaspc (float xspc);

XSPC is a real number that contains a multiplier. If XSPC < 0, the intercharacter spacing will be reduced by XSPC * NH plot coordinates where NH is the current character height. If XSPC > 0, the spacing will be enlarged by XSPC * NH plot coordinates.

Default: XSPC = 0.

C H A W T H

CHAWTH affects the width of characters.

The call is:	CALL CHAWTH (XWTH)	level 1, 2, 3
or:	void chawth (float xwth);	

XWTH — is a real number between 0 and 2. If XWTH $<$ 1, the character width will be reduced. If XWTH $>$ 1, the character width will be enlarged.
Default: XWTH = 1.

C H A A N G

CHAANG defines an inclination angle for characters.

The call is:	CALL CHAANG (ANGLE)	level 1, 2, 3
or:	void chaang (float angle);	

ANGLE — is the inclination angle between characters and the vertical direction in degrees (-60. $\leq$ ANGLE $\leq$60).
Default: ANGLE = 0.

F I X S P C

All fonts in DISLIN except for the default font are proportional. After a call to FIXSPC the characters of a proportional font will also be plotted with a constant character width.

The call is:	CALL FIXSPC (XFAC)	level 1, 2, 3
or:	void fixspc (float xfac);	

XFAC — is a real number containing a scaling factor. Characters will be centred in a box of width XFAC * XMAX where XMAX is the largest character width of the current font.

6.4 Fonts

The following routines define character sets of varying style and plot velocity. All fonts except for the default font DISALF are proportional. Each font provides 6 alphabets.

The calls are:	CALL DISALF	- default font, single stroke, low resolution
	CALL SIMPLX	- single stroke font
	CALL COMPLX	- complex font
	CALL DUPLX	- double stroke font
	CALL TRIPLX	- triple stroke font
	CALL GOTHIC	- gothic font
	CALL SERIF	- complex shaded font
	CALL HELVE	- shaded font
	CALL HELVES	- shaded font with small characters

Additional note: If one of the shaded fonts SERIF, HELVE or HELVES is used, only the outlines of characters are plotted to minimize plotting time. With the statement CALL SHDCHA characters will be shaded.

P S F O N T

PSFONT defines a PostScript font.

The call is: CALL PSFONT (CFONT) level 1, 2, 3

or: void psfont (char *cfont);

CFONT is a character string containing the font. Standard font names in PostScript are:

Times-Roman	Courier
Times-Bold	Courier-Bold
Times-Italic	Courier-Oblique
Times-BoldItalic	Courier-BoldOblique
Helvetica	AvantGarde-Book
Helvetica-Bold	AvantGarde-Demi
Helvetica-Oblique	AvantGarde-BookOblique
Helvetica-BoldOblique	AvantGarde-DemiOblique
Helvetica-Narrow	Bookman-Light
Helvetica-Narrow-Bold	Bookman-LightItalic
Helvetica-Narrow-Oblique	Bookman-Demi
Helvetica-Narrow-BoldOblique	Bookman-DemiItalic
NewCenturySchlbk-Roman	Palatino-Roman
NewCenturySchlbk-Italic	Palatino-Italic
NewCenturySchlbk-Bold	Palatino-Bold
NewCenturySchlbk-BoldItalic	Palatino-BoldItalic
ZapfChancery-MediumItalic	Symbol
ZapfDingbats	

Additional notes:

- The file format must be set to 'POST' or to 'PSCL' with the routine METAFL.
- Font names cannot be shortened. Some printers provide additional non-standard fonts. These fonts should be specified exactly in upper and lower characters as they are described in the printer manuals. PostScript suppresses any graphics if there is a syntax error in the font name. Standard font names are not case-sensitive.
- A call to a DISLIN font resets PostScript fonts.

W I N F N T

WINFNT defines a TrueType font for screen output on Windows displays.

The call is: CALL WINFNT (CFONT) level 1, 2, 3

or: void winfnt (char *cfont);

CFONT is a character string containing the font. The following fonts can normally be used on the Windows 95/NT operating system:

Courier New
Courier New Bold
Courier New Italic
Courier New Bold Italic

Times New Roman
Times New Roman Bold
Times New Roman Italic
Times New Roman Bold Italic
Arial
Arial Bold
Arial Italic
Arial Bold Italic

H W F O N T

The routine HWFONT sets a standard hardware font if hardware fonts are supported by the current file format. For example, if the file format is PostScript, the font 'Times Roman' is used, if the file format is 'CONS' or 'XWIN', 'Times New Roman' is used for Windows 95/NT. If no hardware fonts are supported, COMPLX is used.

The call is: CALL HWFONT — level 1, 2, 3

or: void hwfont ();

B A S A L F

BASALF defines the base alphabet.

The call is: CALL BASALF (CALPH) — level 1, 2, 3

or: void basalf (char *calph);

CALPH — is a character string that can have the values 'STANDARD', 'ITALIC', 'GREEK', 'SCRIPT', 'RUSSIAN' and 'MATHEMATIC'. These alphabets can be used with all fonts.

Default: 'STANDARD'.

S M X A L F

SMXALF defines shift characters to shift between the base and an alternate alphabet.

The call is: CALL SMXALF (CALPH, C1, C2, N) — level 1, 2, 3

or: void smxalf (char *calph, char *c1, char *c2, int n);

CALPH — is a character string containing an alphabet. In addition to the names in BASALF, CALPH can have the value 'INSTRUCTION'.

C1 — is a character that shifts to the alternate alphabet.

C2 — is a character that shifts back to the base alphabet. C1 and C2 may be identical. After the last plotted character of a character string, DISLIN automatically shifts back to the base alphabet.

N — is an integer between 1 and 6. Up to 6 alternate alphabets can be defined.

E U S H F T

EUSHFT defines a shift character to plot special European characters.

The call is: CALL EUSHFT (CNAT, CHAR) — level 1, 2, 3

or: void eushft (char *cnat, char *char);

CNAT — is a character string that can have the values 'GERMAN', 'FRENCH', 'SPANISH', 'DANISH', 'ACUTE', 'GRAVE' and 'CIRCUM'.

CHAR		is a shift character. For example, with CNAT = 'GERMAN', the characters A, O, U, a, o, u and s placed directly after CHAR will be plotted as Ä, Ö, Ü, ä, ö, ü and ß.
Additional notes:	-	Shift characters can be defined multiple where the characters must be different.
	-	European characters are supported by PostScript fonts and by COMPLX.

DISALF

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	B	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ϵ
34	"	*"*	"	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	E	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	X	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	H	106	j	*j*	?
39	'	*'*	'	73	I	*I*	I	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	K	109	m	*m*	μ
42	*	**	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	M	111	o	*o*	ο
44	,	*,*	,	78	N	*N*	N	112	p	*p*	π
45	−	*−*	−	79	O	*O*	O	113	q	*q*	φ
46	.	*.*	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	P	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	T	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Y	123	{	*{*	{
56	8	*8*	8	90	Z	*Z*	Z	124	|	*/*	|
57	9	*9*	9	91	[	*[*	[	125	}	*}*	}
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	`	*`*	`	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	ϑ	133	ß	*ß*	?

Figure 6.3: DISALF Character Set

SIMPLX

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	Β	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	"	*"*	"	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	Ε	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	Χ	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	Η	106	j	*j*	?
39	'	*'*	'	73	I	*I*	Ι	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	Κ	109	m	*m*	μ
42	*	**	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	Μ	111	o	*o*	ο
44	,	*,*	,	78	N	*N*	Ν	112	p	*p*	π
45	—	*—*	—	79	O	*O*	Ο	113	q	*q*	θ
46	.	*.*	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	Ρ	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	Τ	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Υ	123	≷	*≷*	≷
56	8	*8*	8	90	Z	*Z*	Ζ	124	—	*/*	—
57	9	*9*	9	91	[	*[*	[	125	≷	*≷*	≷
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	`	*`*	`	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.4: SIMPLX Character Set

COMPLX

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	B	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	''	*''*	''	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	E	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	X	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	H	106	j	*j*	?
39	'	*'*	'	73	I	*I*	I	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	K	109	m	*m*	μ
42	*	**	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	M	111	o	*o*	*o*
44	,	*,*	,	78	N	*N*	N	112	p	*p*	π
45	-	*-*	-	79	O	*O*	O	113	q	*q*	θ
46	.	*.*	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	P	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	T	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Y	123	{	*{*	{
56	8	*8*	8	90	Z	*Z*	Z	124	|	*/*	|
57	9	*9*	9	91	[	*[*	[	125	}	*}*	}
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	`	*`*	`	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.5: COMPLX Character Set

COMPLX

ASCII	SCRI.	RUSS.	MATH.	ASCII	SCRI.	RUSS.	MATH.	ASCII	SCRI.	RUSS.	MATH.
32				66	$\mathcal{B}$	Б	≡	100	d	д	↓
33	$!$	!	!	67	$\mathcal{C}$	Э	⟩	101	e	й	⇒
34	″	″	″	68	$\mathcal{D}$	Д	×	102	f	ф	⇐
35	#	#	#	69	$\mathcal{E}$	Й	÷	103	g	г	⇔
36	$	Ъ	$	70	$\mathcal{F}$	Ф	±	104	h	ж	⊕
37	%	Ы	%	71	$\mathcal{G}$	Г	∓	105	i	и	⊖
38	&	Ь	&	72	$\mathcal{H}$	Ж	≤	106	j	ч	⊙
39	′	'	′	73	$\mathcal{I}$	И	≠	107	k	к	∀
40	(	(	(	74	$\mathcal{J}$	Ч	≥	108	l	л	∞
41	)	)	)	75	$\mathcal{K}$	К	⊥	109	m	м	∂
42	*	*	*	76	$\mathcal{L}$	Л	∩	110	n	н	∇
43	+	+	+	77	$\mathcal{M}$	М	∪	111	o	о	¬
44	,	,	,	78	$\mathcal{N}$	Н	⊂	112	p	п	∨
45	—	—	—	79	$\mathcal{O}$	О	⊃	113	q	ш	∧
46	.	.	.	80	$\mathcal{P}$	П	∧	114	r	р	∑
47	/	/	/	81	$\mathcal{Q}$	Ш	∨	115	s	с	∏
48	0	0	0	82	$\mathcal{R}$	Р	⊆	116	t	т	⋃
49	1	1	1	83	$\mathcal{S}$	С	⊇	117	u	ю	⋂
50	2	2	2	84	$\mathcal{T}$	Т	‖	118	v	в	∫
51	3	3	3	85	$\mathcal{U}$	Ю	∃	119	w	щ	∮
52	4	4	4	86	$\mathcal{V}$	В	∄	120	x	х	√
53	5	5	5	87	$\mathcal{W}$	Щ	∈	121	y	у	∅
54	6	6	6	88	$\mathcal{X}$	Х	∉	122	z	з	≅
55	7	7	7	89	$\mathcal{Y}$	У	∋	123	{	е	{
56	8	8	8	90	$\mathcal{Z}$	З	∌	124	|	ё	—
57	9	9	9	91	[	Е	[	125	}	ц	}
58	:	:	:	92	\	Ё	\	126	~	я	~
59	;	;	;	93	]	Ц	]	127	$\ddot{\mathcal{A}}$	?	?
60	<	ъ	<	94	^	Я	^	128	$\ddot{\mathcal{O}}$	?	?
61	=	ы	=	95	_	_	_	129	$\ddot{\mathcal{U}}$	?	?
62	>	ь	>	96	`	`	`	130	$\ddot{a}$	?	?
63	?	?	?	97	a	а	→	131	$\ddot{o}$	?	?
64	@	@	@	98	b	б	↑	132	$\ddot{u}$	?	?
65	$\mathcal{A}$	A	⟨	99	c	э	←	133	β	?	?

Figure 6.6: COMPLX Character Set

GOTHIC

ASCII	STAN.	ITAL.	SCRI.	ASCII	STAN.	ITAL.	SCRI.	ASCII	STAN.	ITAL.	SCRI.
32				66	𝔅	𝔅	𝔅	100	𝔡	𝔡	𝔡
33	!	!	!	67	ℭ	𝔇	ℭ	101	𝔢	𝔢	𝔢
34	"	"	"	68	𝔇	𝔇	𝔇	102	𝔣	𝔣	𝔣
35	#	#	#	69	𝔈	𝔈	𝔈	103	𝔤	𝔤	𝔤
36	$	$	$	70	𝔉	𝔉	𝔉	104	𝔥	𝔥	𝔥
37	%	%	%	71	𝔊	𝔊	𝔊	105	𝔦	𝔦	𝔦
38	&	&	&	72	ℌ	ℌ	ℌ	106	𝔧	𝔧	𝔧
39	'	'	'	73	ℑ	ℑ	ℑ	107	𝔨	𝔨	𝔨
40	(	(	(	74	𝔍	𝔍	𝔍	108	𝔩	𝔩	𝔩
41	)	)	)	75	𝔎	𝔎	𝔎	109	𝔪	𝔪	𝔪
42	*	*	*	76	𝔏	𝔏	𝔏	110	𝔫	𝔫	𝔫
43	+	+	+	77	𝔐	𝔐	𝔐	111	𝔬	𝔬	𝔬
44	,	,	,	78	𝔑	𝔑	𝔑	112	𝔭	𝔭	𝔭
45	-	-	-	79	𝔒	𝔒	𝔒	113	𝔮	𝔮	𝔮
46	.	.	.	80	𝔓	𝔓	𝔓	114	𝔯	𝔯	𝔯
47	/	/	/	81	𝔔	𝔔	𝔔	115	𝔰	𝔰	ſ
48	0	0	0	82	ℜ	ℜ	ℜ	116	𝔱	𝔱	𝔱
49	1	1	1	83	𝔖	𝔖	𝔖	117	𝔲	𝔲	𝔲
50	2	2	2	84	𝔗	𝔗	𝔗	118	𝔳	𝔳	𝔳
51	3	3	3	85	𝔘	𝔘	𝔘	119	𝔴	𝔴	𝔴
52	4	4	4	86	𝔙	𝔙	𝔙	120	𝔵	𝔵	𝔵
53	5	5	5	87	𝔚	𝔚	𝔚	121	𝔶	𝔶	𝔶
54	6	6	6	88	𝔛	𝔛	𝔛	122	𝔷	𝔷	𝔷
55	7	7	7	89	𝔜	𝔜	𝔜	123	≀	≀	≀
56	8	8	8	90	ℨ	ℨ	ℨ	124	—	—	—
57	9	9	9	91	[	[	[	125	≀	≀	≀
58	:	:	:	92	╲	╲	╲	126	~	~	~
59	;	;	;	93	]	]	]	127	?	?	Ä
60	<	<	<	94	^	^	^	128	?	?	Ö
61	=	=	=	95	_	_	_	129	?	?	Ü
62	>	>	>	96	`	`	`	130	?	?	ä
63	?	?	?	97	𝔞	𝔞	𝔞	131	?	?	ö
64	@	@	@	98	𝔟	𝔟	𝔟	132	?	?	ü
65	𝔄	𝔄	𝔄	99	𝔠	𝔠	𝔠	133	?	?	ß

Figure 6.7: GOTHIC Character Set

HELVE

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	Β	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	”	*”*	”	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	Ε	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	Χ	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	Η	106	j	*j*	?
39	’	*’*	’	73	I	*I*	Ι	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	Κ	109	m	*m*	μ
42	*	**	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	Μ	111	o	*o*	ο
44	,	*,*	,	78	N	*N*	Ν	112	p	*p*	π
45	–	*–*	–	79	O	*O*	Ο	113	q	*q*	θ
46	.	*.*	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	Ρ	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	Τ	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Υ	123	{	*{*	{
56	8	*8*	8	90	Z	*Z*	Ζ	124	|	*/*	|
57	9	*9*	9	91	[	*[*	[	125	}	*}*	}
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	‘	*‘*	‘	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.8: HELVE Character Set

Times-Roman

ASCII	CHAR	ASCII	CHAR	ASCII	CHAR	ASCII	CHAR	ASCII	CHAR
32		62	>	92	\	122	z	152	Ú
33	!	63	?	93	]	123	{	153	á
34	"	64	@	94	^	124	|	154	é
35	#	65	A	95	_	125	}	155	í
36	$	66	B	96	‘	126	~	156	ó
37	%	67	C	97	a	127	Ä	157	ú
38	&	68	D	98	b	128	Ö	158	À
39	’	69	E	99	c	129	Ü	159	È
40	(	70	F	100	d	130	ä	160	Ì
41	)	71	G	101	e	131	ö	161	Ò
42	*	72	H	102	f	132	ü	162	Ù
43	+	73	I	103	g	133	ß	163	à
44	,	74	J	104	h	134	Å	164	è
45	-	75	K	105	i	135	Ø	165	ì
46	.	76	L	106	j	136	Æ	166	ò
47	/	77	M	107	k	137	å	167	ù
48	0	78	N	108	l	138	ø	168	Â
49	1	79	O	109	m	139	æ	169	Ê
50	2	80	P	110	n	140	Ñ	170	Î
51	3	81	Q	111	o	141	ñ	171	Ô
52	4	82	R	112	p	142	Ç	172	Û
53	5	83	S	113	q	143	ç	173	â
54	6	84	T	114	r	144	Ë	174	ê
55	7	85	U	115	s	145	Ï	175	î
56	8	86	V	116	t	146	ë	176	ô
57	9	87	W	117	u	147	ï	177	û
58	:	88	X	118	v	148	Á	178	
59	;	89	Y	119	w	149	É	179	
60	<	90	Z	120	x	150	Í	180	
61	=	91	[	121	y	151	Ó	181	

Figure 6.9: Times-Roman Character Set

PostScript Fonts

This is Times-Roman
This is Times-Bold
This is Times-Italic
This is Times-BoldItalic
This is Helvetica
This is Helvetica-Bold
This is Helvetica-Oblique
This is Helvetica-BoldOblique
This is Helvetica-Narrow
This is Helvetica-Narrow-Bold
This is Helvetica-Narrow-Oblique
This is Helvetica-Narrow-BoldOblique
This is NewCenturySchlbk-Roman
This is NewCenturySchlbk-Italic
This is NewCenturySchlbk-Bold
This is NewCenturySchlbk-BoldItalic
This is ZapfChancery-MediumItalic
✶❋❉▲ ❉▲ ❋❁❒❄⌘❉■✵✲❁▼▲
This is Courier
This is Courier-Bold
This is Courier-Oblique
This is Courier-BoldOblique
This is AvantGarde-Book
This is AvantGarde-Demi
This is AvantGarde-BookOblique
This is AvantGarde-DemiOblique
This is Bookman-Light
This is Bookman-LightItalic
This is Bookman-Demi
This is Bookman-DemiItalic
This is Palatino-Roman
This is Palatino-Italic
This is Palatino-Bold
This is Palatino-BoldItalic
Τηισ ισ Σψμβολ

Figure 6.10: PostScript Fonts

6.5 Indices and Exponents

Indices and exponents can be plotted by using control characters in characters strings. There are 3 predefined control characters in DISLIN which can be altered with the routines NEWMIX and SETMIX. The predefined character

[is used for exponents. The character height is reduced by the scaling factor FEXP and the pen is moved up FBAS * NH plot coordinates where NH is the current character height.

] is used for indices. The pen is moved down FBAS * NH plot coordinates and the character height is reduced by the scaling factor FEXP.

$ is used to move the pen back to the base-line. This will automatically be done at the end of a character string.

FBAS and FEXP have the default values 0.6 and 0.8, respectively, these values can be changed with the routines SETBAS and SETEXP.

M I X A L F

This routine instructs DISLIN to search for control characters in character strings.

The call is: CALL MIXALF level 1, 2, 3

or: void mixalf ();

S E T B A S

SETBAS defines the position of indices and exponents. This routine also affects logarithmic axis labels.

The call is: CALL SETBAS (FBAS) level 1, 2, 3

or: void setbas (float fbas);

FBAS is a real number used as a scaling factor. The pen will be moved up or down by FBAS * NH plot coordinates to plot exponents or indices. NH is the current character height.

Default: FBAS = 0.6.

S E T E X P

SETEXP sets the character height of indices and exponents.

The call is: CALL SETEXP (FEXP) level 1, 2, 3

or: void setexp (float fexp);

FEXP is a real number used as a scaling factor. The character height of indices and exponents is set to FEXP * NH where NH is the current character height.

Default: FEXP = 0.8

N E W M I X

NEWMIX defines an alternate set of control characters for plotting indices and exponents. The default characters '[', ']' and '$' are replaced by '^', '_' and '%'.

The call is: CALL NEWMIX level 1, 2, 3

or: void newmix ();

S E T M I X

SETMIX defines global control characters for plotting indices and exponents.

The call is:	CALL SETMIX (C, CMIX)	level 1, 2, 3
or:	void setmix (char *c, char *cmix);	

C — is a new control character.

CMIX — is a character string that defines the function of the control character. CMIX can have the values 'EXP', 'IND', 'RES' and 'LEG' for exponents, indices, resetting the base-line, and for multiple text lines in legends, respectively.

Additional note: The routines NEWMIX and SETMIX only modify the control characters. A call to MIXALF is always necessary to plot indices and exponents.

6.6 Instruction Alphabet

The instruction alphabet contains commands that control pen movements and character sizes during the plotting of character strings. It is provided for the representation of complicated formulas.
The instruction alphabet can be used in the same way as other alphabets in DISLIN. Shift characters must be defined with the routine SMXALF to switch between the base and the instruction alphabet.
The commands of the instruction alphabet consist of a single character and an optional parameter. If the parameter is omitted, DISLIN will use default values. A parameter can be a real number, an integer or the character 'X' which resets the parameter back to the entry value at the beginning of the character string.
Commands of the instruction alphabet can only change plot parameters temporarily within a character string. At the end of a character string, all parameters are reset to their entry values.

The following table summarizes all instruction commands. The character r means a real parameter and i an integer. The base-line of character strings is placed directly below them. Commands can be given in uppercase or lowercase letters. Real parameters can be specified without decimal points while integer parameters cannot have decimal points. Several commands can follow one another. Blanks between commands will be ignored.

Instruction-Alphabet

Command	Parameter	Default	Description
A	real	1.	moves the pen horizontally by r * NH plot coordinates where NH is the current character height. If r $<$ 0, the pen will be moved backwards.
C	integer	1	moves the pen horizontally by i character spaces. If i $<$ 0, the pen will be moved backwards.
D	real	1.	moves the pen down from the base-line by r * NH plot coordinates. If r $>$ 0, NH is the entry character height. If r $<$ 0, NH is the current character height.
E			moves the pen up by 0.75 * character height and reduces the character height by the scaling factor 0.6 (for exponents).
F	integer	1	moves the pen horizontally by i spaces. If i is negative, the pen is moved backwards.
G	integer	1	moves the pen horizontally to the tab position with the index i, where 1 $\leq$ i $\leq$ 20.
H	real	0.6	sets the character height to r * NH. If r $>$ 0, NH is the entry character height. If r $<$ 0, NH is the current character height.
I			moves the pen down by 0.35 * character height and multiplies the character height by 0.6 (for indices).
J	integer	1	underscores twice from the tab position i to the current pen position.
K	real	0.8	is used to plot characters with constant widths. Characters will be centred in a box with the width r * W where W is the largest character length in the current font. The global routine is FIXSPC.
L	integer	1	underscores from the tab position i to the current pen position.
M	integer	1	defines the base alphabet. (1 = STAND., 2 = GREEK, 3 = MATH., 4 = ITAL., 5 = SCRIPT, 6 = RUSSIAN).

Command	Parameter	Default	Description
N	integer	1	sets a colour i, where $0 \leq i \leq 255$). The global routine is SETCLR.
O	real	0.	moves the base-line vertically by r * character height. If $r < 0$ the base-line is moved down.
P	integer	1	defines a horizontal tab position with the index i at the current pen position, where $1 \leq i \leq 20$. All tab positions are initialized to the beginning of the string.
R			resets the character height and the base-line to their entry values.
S	integer	0	plots a symbol with the number i, where $0 \leq i \leq 21$.
T	integer	0	moves the pen horizontally from the beginning of the string by i plot coordinates.
U	real	1.	moves the pen up from the base-line by r * NH plot coordinates. If $r > 0$, NH is the entry character height. If $r < 0$, NH is the current character height.
V	integer	1	plots a horizontal line from the tab position i to the current pen position. The line is moved up from the base-line by 0.5 * character height plot coordinates.
W	real	1.	affects the width of characters. The global routine is CHAWTH.
Y	real	0.	affects the character spacing. The global routine is CHASPC.
Z	real	0.	defines an inclination angle for characters, where $-60 \leq r \leq 60$. The global routine is CHAANG.

For the following examples, the characters '{' and '}' are defined with

```
CALL  SMXALF ('INST', '{', '}', 1)
```

to switch between the instruction and the base alphabet.

Instruction Alphabet

1.) Character{H0.5} height{RZ−30} incli{Z30}nation {ZW0.5} ratio {WK} fixed width

Character height \\c*nation* ratio fixed width

2.) Underscoring{L} {P}twice{J} vectors {PA8V}

Underscoring twice vectors

3.) {M2}C{M4}(x) = {M3P}v{M4}e{E}−t{R}t{E}x−1{R}dt{GDH0.4F−1}0{U1.4M3F3}1

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt$$

4.) lim{GDHC}x{M3CD1.1}a{C}1{RM} (1 + {PUH} 1 {RVGD0.5H} x {R}){U1.2H}x{R} = e

$$\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e$$

Figure 6.11: Instruction Alphabet

6.7 Curve Attributes

C H N C R V

CHNCRV defines attributes that will be automatically changed by CURVE after a certain number of calls to the routine CURVE.

The call is:	CALL CHNCRV (CATT)	level 1, 2, 3
or:	void chncrv (char *catt);	

CATT = 'NONE'	means that CURVE changes no attributes.
= 'COLOR'	means that colours will be changed.
= 'LINE'	means that line styles will be changed.
= 'BOTH'	means that colours and line styles will be changed.

Default: CATT = 'NONE'.

Additional note: The sequence of colours is WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN and MAGENTA.
The sequence of line styles is SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL.
The symbol number is always changed. It will be incremented by 1 starting with the current symbol defined by MARKER.

The following three routines are useful when automatic attribute setting is selected and the routine CURVE is called several times to plot a single curve.

I N C C R V

INCCRV defines the number of calls after which CURVE will automatically change attributes.

The call is:	CALL INCCRV (NCRV)	level 1, 2, 3
or:	void inccrv (int ncrv);	

NCRV is the number of curves that will be plotted with identical attributes.

Default: NCRV = 1

C H N A T T

CHNATT is an alternative routine to INCCRV. It is useful when the number of curves plotted with identical attributes varies. CHNATT defines new attributes that will be used by CURVE during the next call.

The call is:	CALL CHNATT	level 1, 2, 3
or:	void chnatt ();	

Additional notes:
- CHNATT changes only attributes specified with CHNCRV.
- Attributes cannot be skipped by calling CHNATT several times; the order of the attribute cycles must be changed.

R E S A T T

In general, curve attributes will be repeated after 8 changes. With the routine RESATT, the attributes can be reset earlier.

The call is: CALL RESATT level 1, 2, 3

or: void resatt ();

I N C M R K

INCMRK selects line or symbol mode for CURVE.

The call is: CALL INCMRK (NMRK) level 1, 2, 3

or: void incmrk (int nmrk);

NMRK = - n means that CURVE plots only symbols. Every n-th point will be marked by a symbol.

= 0 means that CURVE connects points with lines.

= n means that CURVE plots lines and marks every n-th point with a symbol.

Default: NMRK = 0

M A R K E R

The symbols used to plot points can be selected with the routine MARKER. The symbol number will be incremented by 1 after a certain number of calls to CURVE defined by INCCRV.

The call is: CALL MARKER (NSYM) level 1, 2, 3

or: void marker (int nsym);

NSYM is the symbol number between 0 and 21. The symbols are shown in appendix B.

Default: NSYM = 0

H S Y M B L

HSYMBL defines the size of symbols.

The call is: CALL HSYMBL (NHSYM) level 1, 2, 3

or: void hsymbl (int nhsym);

NHSYM is the size of symbols in plot coordinates.

Default: NHSYM = 35

T H K C R V

THKCRV defines the thickness of curves.

The call is: CALL THKCRV (NTHK) level 1, 2, 3

or: void thkcrv (int nthk);

NTHK is the thickness of curves in plot coordinates.

Default: NTHK = 1

G A P C R V

GAPCRV defines a data gap used in the routine CURVE. If the distance between two neightbouring X coordinates is greater than the gap value, CURVE will not connect these data points.

The call is: CALL GAPCRV (XGAP) level 1, 2, 3

or: void gapcrv (float xgap);

XGAP is the gap value.

P O L C R V

POLCRV defines an interpolation method used by CURVE to connect points.

The call is:	CALL POLCRV (CPOL)	level 1, 2, 3
or:	void polcrv (char *cpol);	

CPOL is a character string containing the interpolation method.

= 'LINEAR'	defines linear interpolation.
= 'STEP'	defines step interpolation.
= 'STAIRS'	defines step interpolation.
= 'BARS'	defines bar interpolation.
= 'STEM'	defines stem interpolation.
= 'SPLINE'	defines spline interpolation.
= 'PSPLINE'	defines parametric spline interpolation.

Default: CPOL = 'LINEAR'.

Additional notes:

- The width of bars can be set with BARWTH.
- For spline interpolation, the X-coordinates must have different values and be in ascending order. There is no restriction for a parametric spline. The order of spline polynomials and the number of interpolated points can be modified with SPLMOD.

S P L M O D

SPLMOD defines the order of polynomials and the number of interpolated points used for the interpolation methods 'SPLINE' and 'PSPLINE'.

The call is:	CALL SPLMOD (NGRAD, NPTS)	level 1, 2, 3
or:	void splmod (int ngrad, int npts);	

NGRAD is the order of the spline polynomials (2 - 10). It affects the number of points accepted by CURVE which is determined by the formula (2 * NGRAD + 1) * N $\leq$ 1000. For example, with a cubic spline, up to 142 points can be passed to CURVE.

NPTS is the number of points that will be interpolated in the range XRAY(1) to XRAY(N).

Default: (3, 200).

B A R W T H

BARWTH sets the width of bars plotted by CURVE.

The call is:	CALL BARWTH (XWTH)	level 1, 2, 3
or:	void barwth (float xwth);	

XWTH defines the bar width. If positive, the absolute value of XWTH * (XRAY(1)-XRAY(2)) is used. If negative, the absolute value of XWTH is used where XWTH is specified in plot coordinates.

Default: XWTH = 0.75

N O C H E K

The routine NOCHEK can be used to suppress the listing of points that lie outside of the axis scaling.

The call is:	CALL NOCHEK	level 1, 2, 3
or:	void nochek ();	

6.8 Line Attributes

L I N E S T Y L E S

The routines SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL define different line styles. They are called without parameters. The routine LINTYP (NTYP) can also be used to set line styles where NTYP is an integer between 0 and 7 and corresponds to the line styles above. The routine MYLINE sets user-defined line styles.

M Y L I N E

MYLINE defines a global line style.

The call is:	CALL MYLINE (NRAY, N)	level 1, 2, 3
or:	void myline (int *nray, int n);	

NRAY — is an array of positive integers characterizing the line style. Beginning with pen-down, a pen-down and pen-up will be done alternately according to the specified lengths in NRAY. The lengths must be given in plot coordinates.

N — is the number of elements in NRAY.

Examples: The values of NRAY for the predefined line styles are given below:

SOLID :	NRAY = {1}
DOT :	NRAY = {1, 10}
DASH :	NRAY = {10, 10}
CHNDSH:	NRAY = {30, 15, 10, 15}
CHNDOT:	NRAY = {1, 15, 15, 15}
DASHM :	NRAY = {20, 15}
DOTL :	NRAY = {1, 20}
DASHL :	NRAY = {30, 20}

L I N W I D

The routine LINWID sets the line width.

The call is:	CALL LINWID (NWIDTH)	level 1, 2, 3
or:	void linwid (int nwidth);	

NWIDTH — is the line width in plot coordinates. Default: NWIDTH = 1

Additional note: To define smaller line widhts than 1 (i.e. for PostScript files), the routine PENWID (XWIDTH) can be used where XWIDTH has the same meaning as NWIDTH.

L N C A P

The routine LNCAP sets the current line cap parameter.

The call is:	CALL LNCAP (CAP)	level 1, 2, 3
or:	void lncap (char *cap);	

CAP — is a character string defining the line cap.

- = 'ROUND' defines rounded caps.
- = 'CUT' defines square caps.
- = 'LONG' defines square caps where stroke ends will be continued equal to half the line width.

Default: CAP = 'LONG'.

L N J O I N

The routine LNJOIN sets the current line join parameter.

The call is:	CALL LNJOIN (CJOIN)	level 1, 2, 3
or:	void lnjoin (char *cjoin);	

CJOIN — is a character string containing the the line join.

- = 'SHARP' defines sharp corners between path segments.
- = 'TRUNC' defines truncated corners between path segments.

Default: CJOIN = 'TRUNC'.

L N M L T

The routine LNMLT sets the current miter limit parameter. This routine can be useful if the line join is set to 'SHARP'.

The call is:	CALL LNMLT (XFC)	level 1, 2, 3
or:	void lnmlt (float xfc);	

XFC — is a floatingpoint number where XFC * line width will be used as the miter limit. The miter length is the distance between the inner and outside edge of a path corner.

Default: XFC = 2.

6.9 Shading

S H D P A T

SHDPAT selects shading patterns used by routines such as SHDCRV and AREAF.

The call is:	CALL SHDPAT (IPAT)	level 1, 2, 3
or:	void shdpat (long ipat);	

IPAT — is an integer between 0 and 17. The predefined patterns are shown in appendix B.

M Y P A T

MYPAT defines a global shading pattern.

The call is: CALL MYPAT (IANGLE, ITYPE, IDENS, ICROSS) level 1, 2, 3

or: void mypat (int iangle, int itype, int idens, int icross);

IANGLE is the angle of shading lines (0 - 179).

ITYPE defines the type of shading lines:

= 0	no shading lines.
= 1	equidistant lines.
= 2	double shading lines.
= 3	triple shading lines.
= 4	thick shading lines.
= 5	dotted lines.
= 6	dashed lines.
= 7	dashed-dotted lines.

IDENS defines the distance between shading lines (0: small distance, 9: big distance).

ICROSS indicates whether shading lines are hatched (0: not hatched, 1: hatched).

Examples: The following calls to MYPAT show the predefined shading patterns used by SHDPAT:

IPAT = 0:	CALL MYPAT (0, 0, 0, 0)
IPAT = 1:	CALL MYPAT (45, 1, 5, 0)
IPAT = 2:	CALL MYPAT (150, 4, 5, 0)
IPAT = 3:	CALL MYPAT (135, 1, 5, 0)
IPAT = 4:	CALL MYPAT (45, 4, 5, 0)
IPAT = 5:	CALL MYPAT (45, 1, 5, 1)
IPAT = 6:	CALL MYPAT (135, 2, 1, 0)
IPAT = 7:	CALL MYPAT (45, 4, 5, 1)
IPAT = 8:	CALL MYPAT (30, 1, 4, 0)
IPAT = 9:	CALL MYPAT (45, 2, 1, 1)
IPAT = 10:	CALL MYPAT (0, 1, 5, 1)
IPAT = 11:	CALL MYPAT (45, 3, 1, 0)
IPAT = 12:	CALL MYPAT (70, 4, 7, 0)
IPAT = 13:	CALL MYPAT (45, 3, 1, 1)
IPAT = 14:	CALL MYPAT (0, 4, 5, 1)
IPAT = 15:	CALL MYPAT (45, 2, 1, 0)
IPAT = 16:	CALL MYPAT (0, 1, 0, 0)
IPAT = 17:	CALL MYPAT (0, 5, 5, 0)

N O A R L N

With the routine NOARLN the outlines of shaded regions can be suppressed.

The call is: CALL NOARLN level 1, 2, 3

or: void noarln ();

6.10 Attribute Cycles

The attributes line style, colour and shading pattern can be changed automatically by routines such as CURVE, SHDCRV, BARS and PIEGRF according to a predefined cycle.
The cycles are:

Line styles:	SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL.
Colours:	WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN and MAGENTA.
Shading:	Pattern numbers from 0 to 17.

The following subroutines allow the redefining of cycles.

L I N C Y C

LINCYC changes the line style cycle.

The call is:	CALL LINCYC (INDEX, ITYP)	level 1, 2, 3
or:	void lincyc (int index, int ityp);	
INDEX	is an index between 1 and 30.	
ITYP	is an integer between 0 and 7 containing the line style (0 = SOLID, 1 = DOT, 2 = DASH, 3 = CHNDSH, 4 = CHNDOT, 5 = DASHM, 6 = DOTL, 7 = DASHL).	

C L R C Y C

CLRCYC changes the colour cycle.

The call is:	CALL CLRCYC (INDEX, ICLR)	level 1, 2, 3
or:	void clrcyc (int index, int iclr);	
INDEX	is an index between 1 and 30.	
ICLR	is a colour number (see SETCLR).	

P A T C Y C

PATCYC changes the shading pattern cycle.

The call is:	CALL PATCYC (INDEX, IPAT)	level 1, 2, 3
or:	void patcyc (int index, long ipat);	
INDEX	is an index between 1 and 30.	
IPAT	is a pattern number between 0 and 17 or is determined by the formula IANGLE * 1000 + ITYPE * 100 + IDENS * 10 + ICROSS with the parameters described in MYPAT.	

6.11 Base Transformations

The following subroutines create a transformation matrix that affects plot vectors contained within page borders. Vectors may be scaled, shifted and rotated and the transformations can be combined in any order.

T R F S H F

TRFSHF affects the shifting of plot vectors.

The call is: CALL TRFSHF (NXSHFT, NYSHFT) level 1, 2, 3

or: void trfshf (int nxshft, int nyshft);

NXSHFT, NYSHFT are plot coordinates that define the magnitude of shifting in the X- and Y-direction.

T R F S C L

TRFSCL affects the scaling of plot vectors.

The call is: CALL TRFSCL (XSCL, YSCL) level 1, 2, 3

or: void trfscl (float xscl, float yscl);

XSCL, YSCL are scaling factors for the X- and Y-direction.

T R F R O T

TRFROT affects the rotation of plot vectors around a point.

The call is: CALL TRFROT (XANG, NX, NY) level 1, 2, 3

or: void trfrot (float xang, int nx, int ny);

XANG is the rotation angle measured in degrees in a counter-clockwise direction.

NX, NY are the plot coordinates of the rotation point.

T R F R E S

TRFRES resets base transformations.

The call is: CALL TRFRES level 1, 2, 3

or: void trfres ();

6.12 Shielded Regions

This section describes how to protect regions from being overwritten. Shielded regions can be defined automatically by DISLIN or explicitly by the user. Shielded regions are stored in a buffer which can then be manipulated by the user.

S H I E L D

SHIELD selects shielded regions which are set automatically by DISLIN.

The call is: CALL SHIELD (CAREA, CMODE) level 1, 2, 3

or: void shield (char *carea, char *cmode);

CAREA is a character string defining the regions:

= 'MESSAG' is used for text and numbers plotted by MESSAG and NUMBER.

= 'SYMBOL' will shield symbols.

= 'BARS' will shield bars plotted by BARS.

= 'PIE' will shield pie segments plotted by PIEGRF.

= 'LEGEND' will protect legends. All legend attributes should be set before calling CURVE because the shielded region of a legend is defined by CURVE. If there is no legend position defined with LEGPOS, CURVE assumes that the legend lies in the upper right corner of the axis system.

CMODE	is a character string defining a status:
= 'ON'	means that the regions defined above will be written to the shielding buffer and are protected.
= 'OFF'	means that regions will not be written to the shielding buffer. Regions that are still stored in the buffer will be shielded.
= 'DELETE'	removes regions from the shielding buffer.
= 'RESET'	is a combination of 'OFF' and 'DELETE'. Regions are removed from and will not be written to the shielding buffer. To save computing time, this command should always be used when shielding is no longer needed.
= 'NOVIS'	The shielding of regions held in the shielding buffer is disabled. This is not valid for regions newly written to the buffer.
= 'VIS'	Disabled regions will be protected. This is the default value for regions newly written to the buffer.

The following routines set user-defined regions:

The calls are:	CALL SHLREC	(NX, NY, NW, NH)	for rectangles
	CALL SHLRCT	(NX, NY, NW, NH, THETA)	for rotated rectangles
	CALL SHLCIR	(NX, NY, NR)	for circles
	CALL SHLELL	(NX, NY, NA, NB, THETA)	for rotated ellipses
	CALL SHLPIE	(NX, NY, NR, ALPHA, BETA)	for pie segments
	CALL SHLPOL	(NXRAY, NYRAY, N)	for polygons.

NX, NY	are plot coordinates of the upper left corner or the centre point.
NW, NH	are the width and height of rectangles.
NR, NA, NB	are radii in plot coordinates.
THETA	is a rotation angle measured in degrees in a counter-clockwise direction.
ALPHA, BETA	are starting and ending angles for pie segments measured in degrees in a counter-clockwise direction.
NXRAY, NYRAY	are arrays of the dimension N containing the corner points of a polygon.

S H L I N D

The index of shielded regions in the buffer can be requested with SHLIND. It returns the index of the region last written to the buffer.

The call is:	CALL SHLIND (ID)	level 1, 2, 3
or:	int shlind ();	
ID	is the returned index.	

S H L D E L

SHLDEL removes entries from the shielding buffer.

The call is:	CALL SHLDEL (ID)	level 1, 2, 3
or:	void shldel (int id);	
ID	is the index of a shielded region. If ID is 0, all regions defined by the user will be deleted.	

S H L R E S

SHLRES deletes regions last written to the shielding buffer.

The call is:	CALL SHLRES (N)	level 1, 2, 3
or:	void shlres (int n);	

N is the number of regions to delete.

S H L V I S

SHLVIS disables or enables shielded regions. Disabled regions are no longer protected but are still held in the shielding buffer.

The call is:	CALL SHLVIS (ID, CMODE)	level 1, 2, 3
or:	void shlvis (int id, char *cmode);	

ID is the index of a shielded region. If ID is 0, all entries are disabled or enabled.

CMODE = 'ON' enables shielded regions. This is the default value for regions newly written to the buffer.

= 'OFF' disables shielded regions.

Additional notes:

- A frame is plotted around regions defined by the user. The thickness of frames can be set with FRAME. Regions defined automatically by DISLIN are not enclosed by a frame but frames plotted by MESSAG after using FRMESS and shielded regions defined by MESSAG are identical.
- Shielded regions can overlap each other.
- The statement CALL RESET ('SHIELD') resets shielding. All regions defined by DISLIN and the user are removed from the shielding buffer and no new regions will be written to the buffer.
- The number of shielded regions is limited to the size of the shielding buffer which is set to 1000 words. The number of words used by regions are: SHLREC = 6, SHLRCT = 7, SHLCIR = 5, SHLELL = 7, SHLPIE = 7 and SHLPOL = 2*N+3.
- Shielding of regions is computer intensive. Therefore, shielding should be used very carefully and shielded regions should be deleted from the buffer when no longer needed.
- Base transformations do not affect the position of shielded regions.
- SHLPOL can be used between the routines GRFINI and GRFFIN. The shielded region will be projected into 3-D space. This is not valid for other shielded regions.