UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 12

March 1987

Title:

EASYGRAPH

Author: Nick Stroud

Contact: Your Support Team

Software Support Category: See Note 15

Synopsis

EASYGRAPH is a program which draws graphs and histograms. It reads in its data in free format, and provides a wide range of control parameters giving great flexibility in such items as scaling, axes, titles, colour, and additional text. Any number of graphs can be overlaid on the same axes, and any number of separate graphs can be plotted (at any angle) on a single sheet of paper. As well as reading in its data, EASYGRAPH can generate data from a mathematical equation, for example enabling you to overlay a theoretical curve on experimental data points.

EASYGRAPH's default output format is acceptable for publication in many journals, and can be produced on paper or acetate (the latter for use on overhead projectors). A chatty front-end program called EASYMENU simplifies the preparation of EASYGRAPH instructions for the beginner. EASYGRAPH can also be called from a program to mechanize graph-production.

Keywords

EASYGRAPH, GIMMS fonts, graphs, histograms, plotting, EASYMENU, DRAWPICTURE, graphics, overhead transparencies, curvefitting, mathematical functions.

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1. How EASYGRAPH Works

EASYGRAPH requires data - the numbers to be plotted - and control instructions which tell it how to present this data on a graph. The most satisfactory way to use EASYGRAPH is to put the data into a file, and all the control instructions into a second file: EASYGRAPH reads from files just as easily as from the terminal. Small changes can then be made to the graph simply by editing the control file.

Having read in your data and instructions, EASYGRAPH creates the graph in a special type of computer file, known as an RCO PLOTFILE, which is ready to be sent to a plotter, or graphics terminal or printer. It uses the name **T#PLOTFL** for this file by default, though you can choose your own name with EASYGRAPH's **FILE** instruction. Note that you *cannot* get the output by LISTing this file to a plotter: you *must* use the EMAS command **GPLIST**. If wanting the picture on a graphics terminal or printer you use **DRAWPICTURE**. Both of these commands are described in the on-line help system, and in User Notes from your Support Team: they are used thus:

```
Command:GPLIST T#PLOTFL , .GP15
Command:DRAWPICTURE T#PLOTFL , .OUT
Command:DRAWPICTURE T#PLOTFL , .LP15
```

where **.GP15** is one of the ERCC plotters - see User Note 17; **.OUT** signifies your terminal - use command **TERMINALTYPE** first!; and **.LP15** is one of the few line printers on the network which can accept plotfiles.

The control instructions take the form of some 70 instruction keywords, some qualified by 'subcommands', and most of them taking one or more parameters. This User Note observes the following conventions: the word "Command" is reserved for EMAS commands; EASYGRAPH's control words are called "Instructions"; the subcommands which modify the effect of the instructions are called "Qualifiers"; and the numerical or textual information required by the instructions are "Parameters".

Qualifiers are attached to Instructions with the underline character _ with *no intervening spaces*. A complete EASYGRAPH instruction therefore takes the general form:

```
INSTRUCTION_QUALIFIER  PARAMETER(S)
```

For example:

```
XRANGE_LOG 1,100
```

says that the X-axis is to cover the range from 1 to 100 with logarithmic tickmarks and labelling.

Incidentally, wherever an EMAS 2900 command is depicted in this Note the NOBRACKETS option is assumed: if you still use brackets on EMAS or BUSH, you should read - for example - **HELP EASYGRAPH** as **HELP(EASYGRAPH)**.

One more convention is worth noting at the outset: the Note refers throughout to "X" data and "Y" data: "X" signifies the horizontal axis and the independent variable; "Y" signifies the vertical axis and the dependent variable.

1.2. Access to EASYGRAPH

Before you can use EASYGRAPH, you need to connect it into your process. To do this you need the once-only commands:

```
Command:OPTION SEARCHDIR=CONLIB.GRAPHICS          {on BUSH or EMAS}
Command:OPTION SEARCHDIR=ERCLIB.GRAPHICS

Command:SEARCHDIR ERCLIB:GRAPHICS                 {on EMAS-A}
```

The EASYGRAPH program itself can be run either interactively - called without parameters - or non-interactively, the latter optionally sending output straight to a plotter, terminal or printer. Thus the three forms of call are:

```
Command:EASYGRAPH

Command:EASYGRAPH <filename>

Command:EASYGRAPH <filename> , <device>
```

where <filename> is a file containing a list of EASYGRAPH instructions; and <device> is a plotter such as .GP15 or .GP25 , a terminal such as .OUT or .X5A , or a graphics printer such as .LP15 or .LP41.

1.3. Getting started

The best way to start with EASYGRAPH is to type your data into a file - and to keep things simple at first you should start either with a set of (X,Y) coordinate pairs or with a set of Y values. Having got your data into a file, the next Chapter shows how to get EASYGRAPH's basic "default" graph from it; or you could run the **EASYMENU** program to set up a control file of EASYGRAPH instructions for a more elaborate graph. EASYMENU asks a lot of questions about what the graph is to look like, constructing a file of control instructions which EASYGRAPH can then read - for example as in the non-interactive forms of the command indicated above. Note however that EASYMENU does not give access to all EASYGRAPH's facilities: it is only intended as an aid for getting started.

To use EASYMENU:

```
Command:EASYMENU
```

and answer the questions!

1.4. This User Note

The next chapter shows how to get your data into EASYGRAPH and get a simple graph out. Chapter 3 then describes all the EASYGRAPH instructions in detail, with a brief summary to help you find the ones you need to achieve a particular effect. Chapter 4 concentrates on the construction of axes, since there are so many ways of modifying the appearance of these; and Chapter 5 discusses text strings and the various "@" commands which can enhance these.

1:5. Examples

There are several EASYGRAPH example files which demonstrate most of the main features of the program, and which are available on-line to all users for copying and modifying. Many people find these the quickest way to begin using EASYGRAPH: the examples produce many different sorts of graph, and by looking at the descriptions of their instructions in Chapter 3 of this Note you should quickly discover how to produce your own graph. For more information about these on-line examples use:

```
Command:HELP EXAMPLES
```

and select the **EASYGRAPH** section for an explanation.

There is some on-line help information for EASYGRAPH, accessible via:

```
Command:HELP EASYGRAPH
```

or

```
Command:EASYGRAPH
Graph command :HELP    or    HELP <topic>
```

CHAPTER 2
BASIC GRAPHS

All EASYGRAPH's parameters have default settings, and if you do not modify any of these it will produce its default graph. To get this you merely need to supply your data, read it into EASYGRAPH with the **DATA** or **READ** instructions, then call **PLOT** to draw the graph, and **STOP** to stop EASYGRAPH.

So you can get a graph with just three instructions: **DATA**, **PLOT** and **STOP** ; or **READ**, **PLOT** and **STOP**.

The two different instructions which read in the data cope with the two possible formats of the data: (X,Y) coordinate pairs, or a single set of (X) or (Y) points. **DATA** reads in the pairs, and **READ** accepts the single set - which is then plotted either against another set of numbers read in by **READ**, or against data generated automatically by EASYGRAPH. These two instructions are fundamental in EASYGRAPH's operation, and are demonstrated in the two basic examples in this chapter.

The two examples will use the same data - being a set of temperature readings taken at The King's Buildings in April 1984. Both **DATA** and **READ** will accept as their parameter either the numbers themselves or the name of a file containing the numbers: to demonstrate this the DATA example will read from a file and the READ example from the terminal.

The two examples both show the input sequence - the user's dialogue with EASYGRAPH - and the graph obtained from that (the two graphs are identical of course, since the input data is the same). The data consist of X values from 1 to 30, and Y values from 6 to 20. EASYGRAPH discovers these ranges for itself and works out suitable axis-scaling to accommodate them - erring on the safe side by rounding up the ranges to "preferred values". This results in reasonable margins of blank space around the graph itself - as here, where 1-30 has been rounded to 0-48, and 6-20 to 5-29.

2.1. Reading in (X,Y) pairs: the DATA Instruction

In this example the temperature readings have been typed into a file called APRIL, in which each entry consists of the day of the month followed by the temperature on that day:

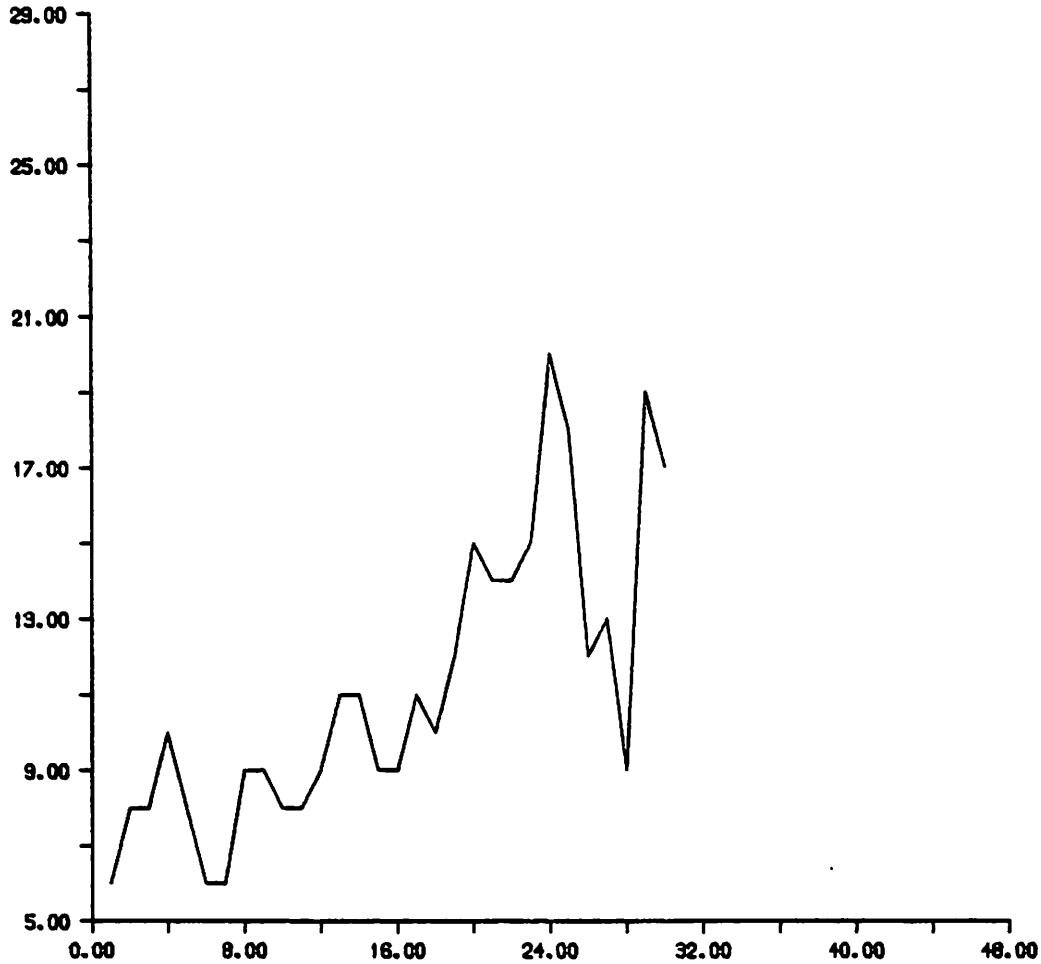
```
Command:LIST APRIL
1 6
2 8
3 8
4 10
 :
 :
 :
28 9
29 19
30 17
```

These numbers are read into EASYGRAPH by its **DATA** instruction, which in this case is simply followed by the name of the file - APRIL. The instruction **PLOT** is then all that is required to construct the graph, and **STOP** returns from EASYGRAPH to EMAS Command: level. The graph itself has been put into an RCO plotfile called **T#PLOTFL** by default, and this is listed to a plotter by the **GPLIST** command. The complete dialogue is as follows (user-input in **bold** print):

```
Command:EASYGRAPH
EASYGRAPH graph plotting program   Version 2.40  (6th March 1987)
Graph command :DATA APRIL
Graph command :PLOT

 ***(CHANNEL 80) File EKLD98:T#PLOTFL initialised.***
Graph command :STOP

 ***(CHANNEL 80) 44 records written to plotter file EKLD98:T#PLOTFL.***
 ** Drawing area is 19.24 cms. wide by 13.42 cms. high **
**
CAUTION! - picture file "T#PLOTFL" will be destroyed when you log off.. **

Command:GPLIST T#PLOTFL,.GP15

Command:
```

The resultant graph is shown on the next page.

Note that the **GPLIST** command *must* be used to send a plotfile to a plotter: if you use LIST the output will be in the wrong format for plotting and will therefore be thrown away! **.GP15** is one of the plotters on the Edinburgh network: for a complete list please see User Note 17, free from your Support Team.


29.00
25.00
21.00
17.00
13.00
9.00
5.00
0.00
8.00
16.00
24.00
32.00
40.00
48.00

2.2. Reading in just (Y) values: the READ Instruction

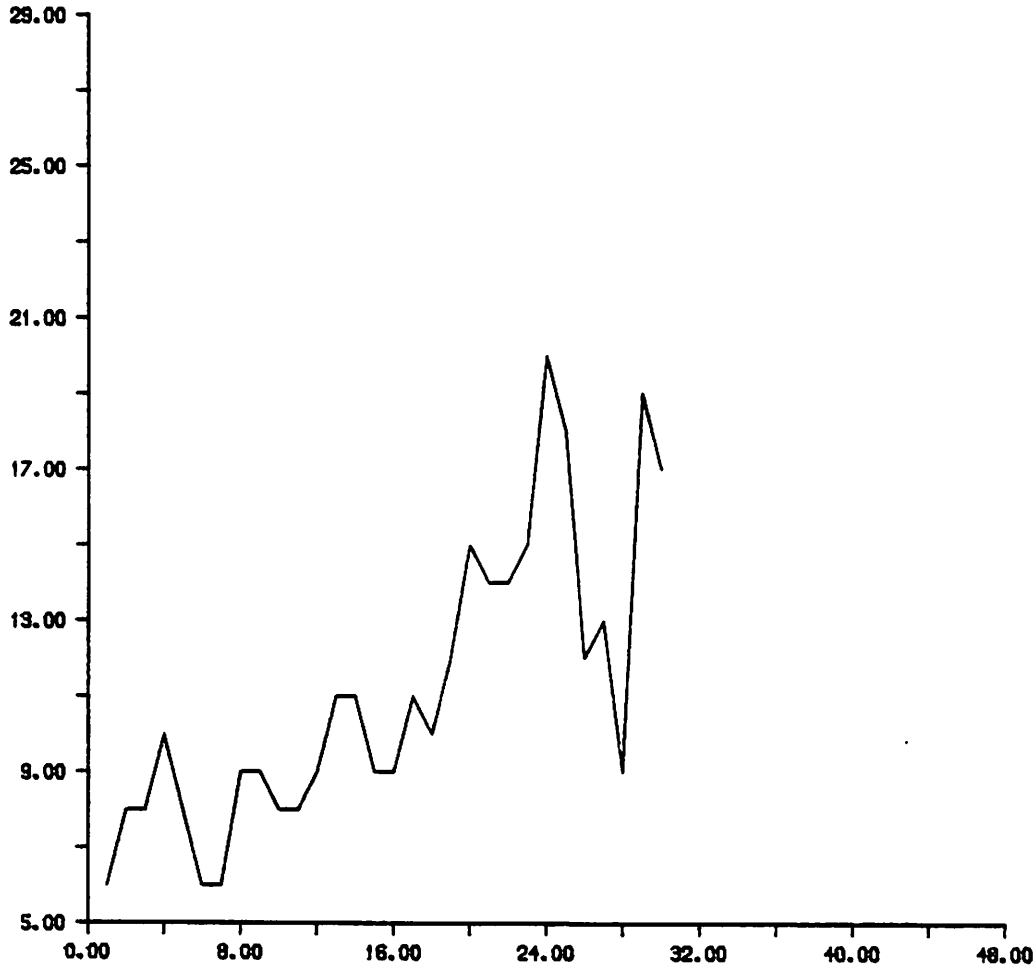
In this example we are going to type in the data from the terminal instead of reading it from a file; and the data itself consists of just the temperature each day, not the date and the temperature as in the previous example. Thus in EASYGRAPH terminology the data consists of a single set of (Y) values. These are handled slightly differently to the coordinate pairs read in by **DATA**, in that the numbers are assigned to a named storage area within EASYGRAPH by which they are subsequently identified. By this mechanism EASYGRAPH allows you to read in any number of these single-value "datasets", and plot any of them against any other. The actual pair to be plotted are given as parameters to the **PLOT** instruction.

In this example we only have the (Y) values - the temperatures - and want EASYGRAPH to construct a set of (X) numbers to act as the dates. As you will see in the following, this is achieved simply by using the name X in the PLOT instruction: if you have not created a dataset called X, EASYGRAPH automatically generates one for you.

Once again only three instructions are needed to plot the graph, but in this case they require more elaborate parameters. As well as the modified form of the **PLOT** instruction to cope with datasets created by **READ**, note in the following that the actual data - the temperatures - do not have to be given in any particular format: EASYGRAPH is very flexible in reading data. Note too that no special marker is required to terminate the input of data: the first character which does not belong to a number - here the "P" of "PLOT" - is sufficient to signal the end of the data.

```
Command:EASYGRAPH
EASYGRAPH graph plotting program   Version 2.40  (6th March 1987)
Graph command :READ TEMP
Data :6 8 8 10 8 6 6 9 9 8 8 9 11
Data :11 9 9 11 10 12 15 14 14 15 20 18 12 13 9 19 17
Data :PLOT X,TEMP

 ***(CHANNEL 79) File EKLD98:T#PLOTFL initialised.***
Graph command :STOP

 ***(CHANNEL 79) 45 records written to plotter file EKLD98:T#PLOTFL .***
 ** Drawing area is 19.24 cms. wide by 13.42 cms. high **
**
CAUTION! - picture file "T#PLOTFL" will be destroyed when you log off.. **

Command:GPLIST T#PLOTFL,.GP15

Command:
```

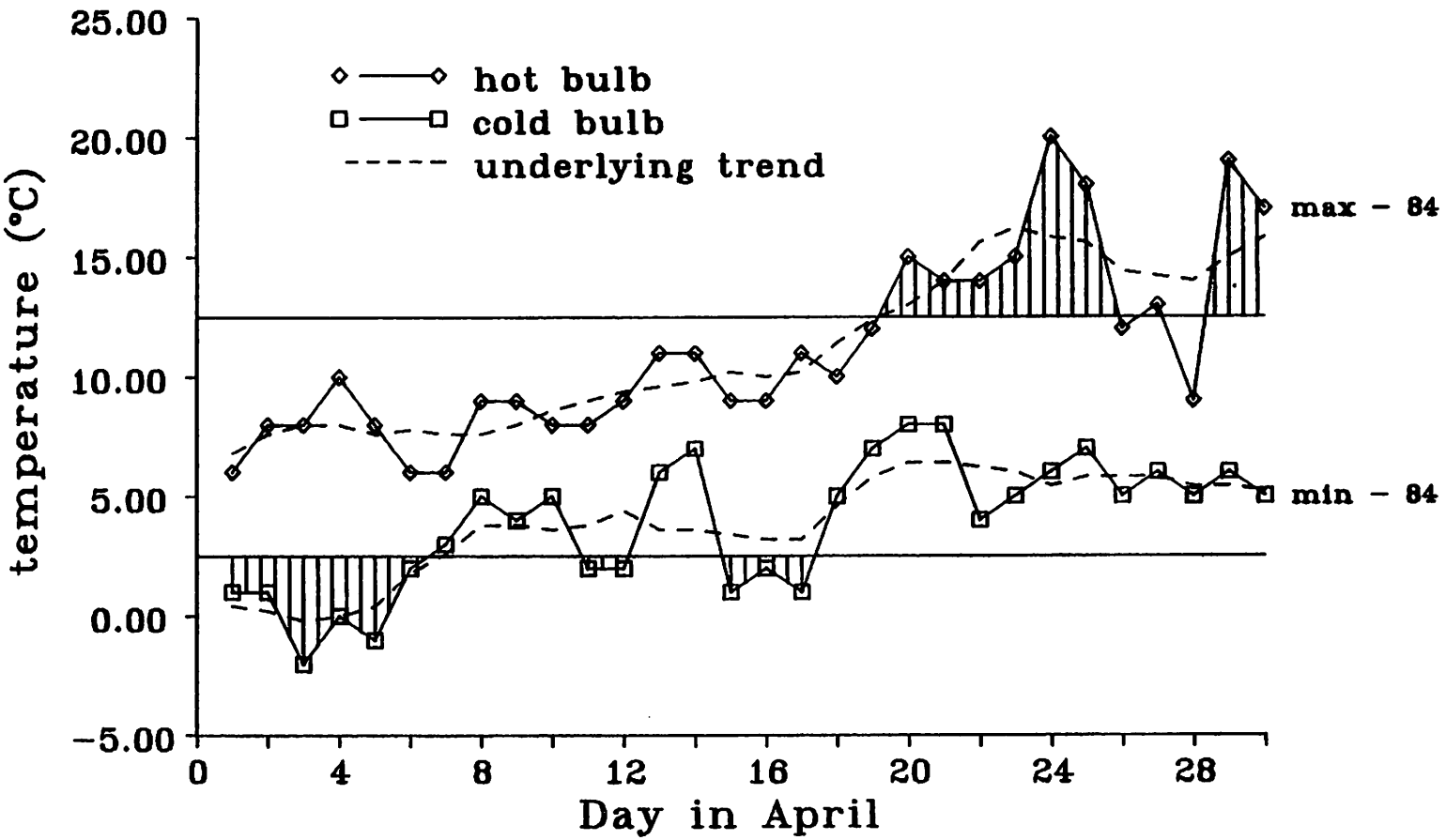
The graph obtained is the same as the previous one (as it should be - it was drawn from the same data!):


29.00
25.00
21.00
17.00
13.00
9.00
5.00
0.00
8.00
16.00
24.00
32.00
40.00
48.00

These "default" graphs are sufficient for some purposes, but you will usually want to modify them in some way - particularly, for example, to fix the scaling as you would like it. To give a flavour of a more elaborate EASYGRAPH production, the following instructions produce the graph shown on the next page (though sadly the colours are not reproduced!):

```
title "Air conditioning required in@BButterfly House - April 1984"
move 17,4
rotate 90
file aprilgraph
read day|max84|min84|max83|min83 aprilpd_apriltemps
xaxis 15,"Day in April"
xrange 0,30
xformat integer
size_xnumbers 0.2
yaxis 10,"temperature (@so@nC)"
yrange_0/5d -5,25
size_ynumbers 0.2
numberaxes_y/1
font 14
colour_graph red
colour_needles red
needles_+ 12.5,0.2
annotate " max - 84"
linetype line+points
symbol diamond
plot day,max84
overlay
linetype line,0.2,0.2
smoothe 5/1
plot day,max84
overlay
linetype line+points
symbol square
annotate " min - 84"
colour_graph blue
colour_needles blue
needles_- 2.5,0.2
plot day,min84
overlay
linetype line,0.2,0.2
smoothe 5/1
plot day,min84
key 2,9
@C4@4@p(1)@4 @clhot bulb
@C2@3@p(1)@3 @clcold bulb
@p(1.4,0.15,0.15) underlying trend
endkey
stop
```

Air conditioning required in Butterfly House – April 1984



CHAPTER 3
THE EASYGRAPH INSTRUCTIONS

3.1. Introduction

All the EASYGRAPH instructions consist of a single command word which is terminated by a space or newline. The instructions may be given in upper or lower case or a mixture of the two.

Many of the basic instructions can be modified by a "Qualifier" which modifies their effect in some way. For example all the axis-drawing instructions accept an X or Y as a qualifier to confine their effect to just one of the axes. These qualifiers consist of letters, or in some cases letters and/or numbers: as with the instruction words, the qualifiers can be in upper and/or lower case and are terminated by a space or newline. They are attached to the instruction word they are modifying by the _ underline character, and there must be no embedded spaces from the first letter of the instruction word to the last letter of the qualifier!

Most of the instructions take at least one parameter which specify their precise effect: for example the axis RANGE commands require the starting and finishing data values for the axis, and the AXIS commands themselves can optionally accept a text string to serve as an axis title. These parameters follow the instruction word - either on the same line, with an intervening space, or on the following line. Where there are several parameters, these are separated from each other by spaces, commas or newlines. The parameters are either numerical values, or text strings - the latter signified by being enclosed in single or double quotes. If you do not use quotes EASYGRAPH tries to interpret the string as its next instruction. Further, text strings cannot extend over more than one line - the end of the line is treated automatically as the end of the string: the method for plotting more than one line is described in Chapter 5 (starting on page 49).

You can have more than one instruction per line of input to EASYGRAPH - separated by spaces - provided the result is not ambiguous.

In the detailed descriptions of the EASYGRAPH instructions in this chapter, the instructions appear in alphabetical order following the next - summary - section. The name of the instruction is followed by its qualifier if it takes one, then its parameters. Optional qualifiers and parameters are enclosed in square brackets [ ].

This chapter is not intended as compulsory bedtime.reading! It is a reference section in which you can look up the precise descriptions of the instructions you know you are interested in from the following summary or from the on-line EASYGRAPH examples.

3.2. Summary of EASYGRAPH Instructions by Effect

Acquiring the DATA

DATA	reads in data in (X,Y) pairs.
READ	reads a series of (X) or (Y) values into an internal dataset.
FUNCTION	creates data from an equation whose curve is to be plotted.
TRANSFORM	modifies the data - e.g. square it or apply an arithmetic constant.

The GRAPH itself

LINETYPE changes the line that is drawn between points.
SYMBOL chooses a symbol to be drawn at each point.
HEIGHT governs the size of the SYMBOLs.
COLOUR selects colours other than black.
XLIMITS,YLIMITS
restricts the values in the X,Y data which may be plotted.
ERRORBARS adds horizontal or vertical error bars to the graph.
SMOOTHE applies a 'moving average filter' to smoothe the data.
NEEDLES draws vertical lines between some baseline and the graph.
SHADE shades histograms with vertical, horizontal or diagonal lines.

The AXES

XAXIS,YAXIS specify length, title and position for axis.
XFORMAT,YFORMAT
specify numbering format, as integer, real or scientific.
XLABELS,YLABELS
specify text labels instead of numbers for the tickmarks.
FLATNUMBER choose vertical X-numbers/labels (horizontal by default).
AXISBOX make a complete box of axes.
NODRAWAXES
omit the axes from the diagram.
NONUMBER to have axes with no numbering.
TICKIN/TICKOUT/TICKACROSS
govern the form of the axis tick marks.
COLOUR specify a colour for the axes.

Adding TEXT to the diagram

TITLE for a title to the whole diagram.
XAXIS/YAXIS to put titles on the axes.
XLABELS,YLABELS
to label the tickmarks rather than number them.
ANNOTATE to label individual curves or points.
KEY to add a key, or any text anywhere on the diagram.
KEYANGLE to set an angle for the KEY text (horizontal by default).
FONT to select a different font for some or all text.
COLOUR to choose colours for any or all text components.

Scaling and positioning the diagram

XRANGE,YRANGE
specify start and end values for axes.
XAXIS,YAXIS specify axis lengths, and thereby overall size of graph (in cm).
SIZE specify the size of the text and tickmarks.
MOVE position the graph on the paper.
ROTATE rotate the entire graph by any angle (usually 90°).
FACTOR scale the size of the diagram up or down, in the X or Y directions or both.

OVERPLOTTING two or more curves on one graph

OVERLAY to draw next graph on top of current one using same scaling.
SAMEPLOT to draw next graph on top of current one, re-scaling it in Y and drawing a new Y-axis to the right of the diagram.

VITAL!

PLOT plots the data (read in by DATA or READ or created by FUNCTION) into the output file, according to all previous instructions. Can also plot a number of curves in one operation, and/or plot their mean and standard deviation.

Miscellaneous

USE file obeys the EASYGRAPH commands in 'file'.
FILE specifies a file or plotter for the graph output.
PLOTTER specifies the plotter for the output - .GP15 by default.
NEW starts a new graph in the same output file - i.e. on the same paper.
VIEW draws the graph on your terminal - best on a graphics terminal.
INITPARMS stores various EASYGRAPH settings in permanent profile.
QUERYPARMS
displays the current and default settings.
DEFAULTPARMS
resets all EASYGRAPH parameters to their defaults.
DELETE removes a dataset previously created by READ.
HELP enters the EMAS VIEW system for access to help information about individual commands and general aspects.
STOP/END/QUIT/Q
closes plotter file and returns to EMAS *Command:* level.

There now follows a complete list of all the EASYGRAPH instructions in alphabetical order:

ABORT

This instruction is only useful when you are running EASYGRAPH interactively, and decide you want to abandon the run without sending anything to the plotter. Thus it is only really needed when you have used `FILE PLOTTERNAME` already, so that STOP/QUIT/etc. would cause the output to be sent automatically.

ANNOTATE [<disp>] , <annotation> , [<Xpoint>]

This instruction puts the text string `<annotation>` beside a data point on the graph. `<disp>` is optional; if present it is a vertical displacement of the annotation in character heights, positive upwards, and is used when the annotations on two separate curves would otherwise overlap. While you can use `<disp>` to shift the text vertically, you can shift it horizontally to the right by including leading spaces in the string. The parameter `<Xpoint>` specifies the X coordinate of the graph alongside which the annotation is to appear: if omitted, the last (right-hand) point of the graph is assumed (the Y value is picked up automatically from the data). A value of `<Xpoint>` below the minimum is interpreted as the minimum, and above the maximum as the maximum.

The text is horizontal by default - i.e. parallel to the X-axis, but can be set to any angle by the **KEYANGLE** instruction.

The ANNOTATE instruction *must* precede the **PLOT** instruction which draws the line or point to be annotated. Note also that ANNOTATE can only be used *once* per curve: if used more than once before PLOT, the most recent one prevails (use **KEY** to annotate more than one point).

Examples:

```
ANNOTATE 0.5 " Data points"
ANNOTATE "Fitted line" 79
annotate -1.5,'@gl@r-threshold',0.005
```

(See also the instruction **KEY** by which you can plot any amount of text wherever you like; the instructions **COLOUR**, **SIZE** and **FONT**, by which you can alter the appearance of the annotation text; and also the note on TEXT STRINGS and the "@" instructions, in Chapter 5, for example to change colour or throw a newline within the annotation.)

AXISBOX or **AXISBOX_NO**

This instruction tells EASYGRAPH to draw in the top and right-hand axes to make a box of axes around your graph. These extra axes take the same colour and tickstyle as the usual axes, but are not labelled, numbered or titled.

The instruction is cancelled by appending `NO` to it as a qualifier - that is:

```
AXISBOX_NO                    WITHOUT ANY SPACES!
```

`NO` is the default - so you only need the cancelling form of the instruction if you change your mind having once called AXISBOX in your current run of EASYGRAPH.

COLOUR_graphpart colour

The COLOUR instruction sets the colour for various components of the graph. <graphpart> is a Qualifier, attached to COLOUR by an underline character _ and can be any of:

GRAPH	(the graph itself)	**XAXIS,YAXIS**	(the actual axes, only)
ERRORBARS		**XNUMBERS,YNUMBERS**	(the axis numbers/labels)
NEEDLES		**XTITLE,YTITLE**	(the axis titles)
KEY	(see also text notes)	**TITLE**	(the title of the whole graph)
ANNOTATION		**AXES**	(everything on both axes)
ALL	(everything)		

<colour> is a colourname or pen number: on the HP A3 plotters such as .GP15 and .GP25 these are BLACK, BLUE, GREEN, RED, PURPLE, ORANGE, BROWN and THICKBLACK, or 1-8 respectively; on other plotters such as .GP23 only the first four are available. (See the section on STRINGS, in Chapter 5, for mention of the instruction "@Cn", which changes to colour 'n' within a text string.)

Examples:

```
COLOUR_ALL GREEN
COLOUR_AXES BLUE
```

COMMENT or *

This instruction may be used to insert comments into EASYGRAPH input files. It has the effect of skipping all input up to the end of the present line. The single character * may also be used as a comment instruction. This facility is provided to enable you to include comments in a large file of EASYGRAPH instructions, to remind you what each part does.

Examples:

```
COMMENT This is a comment line and will be ignored
*    This is also a comment and will be totally ignored
```

DATA[_SKIPmREADn] [<filename>]

The DATA instruction reads in the data for your graph in (x,y) coordinate-pairs. If followed by a filename it reads the data from that file; otherwise it expects to find the data following the DATA instruction itself (i.e. in the control file or from the keyboard). In either case the data input is terminated by the first non-number, which is usually the beginning of the next EASYGRAPH instruction. When reading from a file, the end of the file signals the end of the data. Thus you do not have to indicate the end of the data in any specific way.

The number-pairs must be in the form <x> <y> or <x>,<y>. They are in free format - that is, any valid IMP or FORTRAN form of number is acceptable. Thus for example 0.00176 could be given as 0.00176 or 1.76@-3 or 1.76D-3 or 1.76E-3 or 1.76Q-3. The numbers are separated by spaces and/or commas and/or newlines, and may appear anywhere on the line (i.e. all spaces are simply ignored). EASYGRAPH has no mechanism for jumping over parts of lines - for example to skip alphabetic characters. It does however allow you to skip any number of whole lines before starting to read in data, and also to limit the number of data pairs you take. This is achieved by the qualifier SKIPmREADn - given without spaces - which means "skip m lines, then read n data pairs".

The maximum number of points currently accepted by EASYGRAPH is 2000 (see Section 6.2). If you have more than this please see the description of the instruction **OVERLAY**, below, for a method of getting them all on to the same graph (or picture).

If you have data which is not in the form of (x,y) pairs you will have to use the **READ** instruction to get it into EASYGRAPH.

FORTRAN USERS! By default a Fortran program produces a DATA file which EASYGRAPH cannot read from directly. To get output in a character file: append ",,C" to your DEFINE command (e.g. `DEFINE 6,RESFILE,,C`); or specify `FILETYPE='C'` in your OPEN statement; or use the EMAS command CONVERT to change your data file into a character file - with `CONVERT DATAFILE,CHARFILE`.

Examples:

```
DATA
1.0,2.1
1.1  2.3
1    3E1
2E3 , 3@1
3.2Q2 1@-1
PLOT
```

```
data_skip2read52
*RESULTS FOR 1985-6
WEEK YIELD
1    33.7
2    41.94
3    39.82
etc.    etc.
plot
```

```
DATA_SKIP5 GRAPHDATA
DATA_READ365 RES1985
```

DEFAULTPARMS

This instruction restores all EASYGRAPH parameters to their default settings and stores them in your EMAS profile file - SS#PROFILE - for future use. You can see the parameters which are affected by using **QUERYPARMS**, and can store different settings for future use with **INITPARMS**.

DEFAULTPARMS takes no parameter or qualifier, and must be specified without spaces. It may be abbreviated to **DEFAULTS** or simply **DP**.

DELETE <dsn>

The DELETE instruction tells EASYGRAPH to forget a dataset of name <dsn> which has previously been stored by the **READ** instruction. It is only required if you want to READ in a new set of data to be known by the same name as an existing set: EASYGRAPH automatically deletes all datasets at the end of each session. If you have used READ's multiple-set facility, you can delete any contiguous sequence of sets with `DELETE <DSNm>...<DSNn>` or `DELETE <DSNm> - <DSNn>`.

Example:

```
DELETE YSET
delete y1-y7
```

DRAWAXES[_X or _Y]

Without a qualifier, DRAWAXES tells EASYGRAPH to draw both axes - which is the default anyway, so would only be used to counteract a previous use of **NODRAWAXES.**

The recognized qualifiers are **X** or **Y**, telling EASYGRAPH to draw the corresponding axis *only.* There is no parameter.

Examples:

```
NODRAWAXES . . . . . whoops! DRAWAXES
DRAWAXES_X     - draw only the X-axis
DRAWAXES_Y     - draw only the Y-axis
```

Note that you can still use instructions **XAXIS** and **YAXIS** to set the size of your graph, even if the axis itself is not being drawn. Similarly **XRANGE** and **YRANGE** are still used to adjust the scaling of the graph.

END or **STOP** or **QUIT** or **Q**

These are the four instructions which stop EASYGRAPH, although the precise effect depends on the context: if encountered in a file which is being USEd they would return control to wherever the USE came from - the keyboard or another file; but if typed in they stop EASYGRAPH and return you to EMAS *Command:* level. They are redundant at the end of a file, since hitting the end of the file has precisely the same effect.

If any plotting has been done the plotterfile is closed by this instruction, and sent to a plotter if FILE was used to specify a plotter.

ERRORBARS[<dataset>(V+)][,<dataset>(V-)][,<dataset>(H+)][<dataset(H-)])

This instruction tells EASYGRAPH to plot an errorbar at each graph point. The size of each errorbar must be provided in a dataset which has been created before the ERRORBARS instruction is called, by the **READ** instruction. The errorbars can be vertical and/or horizontal, symmetrical or asymmetrical, as specified in the parameter - which consists of the name of a dataset followed by the type of errorbar to be produced from it, in brackets. The four types of errorbar are:

(V+) for the positive half of a Vertical errorbar
(V-) for the negative half of a Vertical errorbar
(H+) for the positive half of a Horizontal errorbar
(H-) for the negative half of a Horizontal errorbar

You will usually need at least two of these, since a complete errorbar is plotted in two halves: the parameter can be given altogether on one line, as for example:

```
READ E1-E2 ERRFILE
ERRORBARS E1(H-),E2(H+)
```

or in separate calls of ERRORBARS. As usual the instruction has no effect until **PLOT** is called, so if you make a mistake with the parameters just type in the instruction again. (Omit the <dataset> name to cancel the instruction, as for example ERRORBARS (V+),(V-), which means "no vertical errorbars".)

The errorbar is drawn in the form of a capital I, with short lines across the end of the errorline. The size of these endlines is the same as the size of the graph symbol (though this symbol is only plotted if you specify a LINETYPE which includes POINTS):

thus the default length of the endlines is 0.2 cm, and can be changed with the **HEIGHT** instruction.

Once read in, the datasets which are going to be used for the errorbars can be manipulated with the **TRANSFORM** instruction: so for example if the Y-data is to be converted into LOGs, the vertical errorbars can be also. And it would even be possible to use the **FUNCTION** instruction to manipulate datasets to create new ones which could themselves be used for errorbars. The size of each errorbar must be positive, even if it is pointing in a 'negative' direction, i.e. H- or V-. If your 'negative' errors are negative numbers you have to transform them before you can use them as error sizes, i.e. simply `TRANSFORM dataset, *-1`.

See also the instruction **COLOUR**, which can be used to change the colour of the errorbars.

WARNING!! There is nothing statistically fancy about this EASYGRAPH facility: the errorbar is plotted with the size you specify, and nothing is assumed about Standard Deviations or anything like that!

Examples:

```
READ RES1-RES2, RESFILE
READ ERR1-ERR4, ERRFILE
TRANSFORM RES2,LOG
TRANSFORM ERR3,LOG
TRANSFORM ERR4,LOG
ERRORBARS ERR1(H-),ERR2(H+)
ERRORBARS ERR3(V-),ERR4(V+)
LINETYPE LINE+POINTS
SYMBOL DIAMOND
HEIGHT 0.3
PLOT RES1,RES2
```

```
SYMBOL 4 HEIGHT 0.3
read x|y|e
1 17 0.7
2 13 0.45
3 14 0.66
etc.
ERRORBARS E(V+)
LINETYPE HISTOGRAM
PLOT X,Y
```

```
data
1 14
2 17
3 16
4 18
etc.
plot
READ EP|EN,ERRFILE
errorbars ep(v+),en(v-)
```

FACTOR[_KEEP] <xfactor>, <yfactor>

This instruction sets scaling factors to expand or shrink the entire diagram in the X and/or Y directions. The factors can be the same or different - they are assumed to be the same if only one is given (which means that if you want them to be different you must specify both factors together on the same line). Thus for example if a graph has both axes the default 12 cm long, the instruction `FACTOR 1,0.5` will change its physical size to 12x6 cm. Any factor less than or equal to zero results in no

alteration in that factor. The instruction only takes effect if given before the first plot or a **NEW** plot.

If you want to change the size of the graph but not alter the size of the text, specify a qualifier of KEEP - as for example

```
FACTOR_KEEP 0.67,0.67
```

FACTOR affects all dimensions except the parameters to **MOVE**, which are always in full centimetres. So for example to reduce an A3 picture to A4, you need both FACTOR 0.67 (at the beginning) and to multiply any and all MOVE parameters by 0.67 also.

Examples:

```
FACTOR 2,0.5            {double all X measurements, and halve all Y}
FACTOR -1,1             {leave X factor as previously set, restore Y to unity}
factor_keep 0.67,0.67       {shrink graph but retain text-size}
```

(The FACTOR instruction is quite involved, and can have undesirable side-effects. If you simply want to change the size and tick-separation of the axes it is much better to use the **AXIS**, **RANGE** and **NUMBERAXES** instructions.)

FILE <filename>

This instruction allows you to select the destination of the graphical output from EASYGRAPH. <filename> is the name of a file in which the output is to be stored; or the name of a graph plotter (including the dot - e.g. .GP15). Output goes by default to a file called T#PLOTFL - a special type of EMAS file known as a TEMPORARY file, which is destroyed when you log off. If you want to be sure of keeping your picture over more than one log-on session, use FILE to specify a 'permanent' filename - i.e. one that does not begin "T#...". In the case of **FILE <plottername>**, output goes first to the temporary file T#PLOTFL, and is then GPLISTed to the chosen plotter automatically.. (With this latter form of the instruction you do not need to use instruction **PLOTTER**.)

Where used, the FILE instruction must precede the first **PLOT** instruction, or it will be ignored.

Examples:

```
FILE GRAFOUT
file .gp15
```

(A full list of plotters is published in User Note 17, from your Support Team.)

FLATNUMBER[_NO]

This instruction provides a (rather crude) method of setting the orientation of the numbers (or labels) along the X-axis: these are printed horizontally by default, but since some journals require them vertically oriented EASYGRAPH can plot them this way too. Use **FLATNUMBER_NO** to get vertical numbering, or **FLATNUMBER** to put them horizontal again. (This vertical numbering is illustrated on page 46.)

(See also **XFORMAT**, **YFORMAT**, **SIZE**, **FONT** and **COLOUR** to change the appearance of numbers.)

FONT[_item] <fontnumber>

This instruction requests the use of a special lettering font for some or all of the text in the graph. <item> is a qualifier which specifies which part of the graph is to be lettered with font <fontnumber>, and must be one of:

XTITLE, YTITLE	XLABELS, YLABELS			
XNUMBERS, YNUMBERS	KEY	TITLE	AXES	ALL

<fontnumber> is a number in the range 1-99, as indicated in the tables in Appendix II (page 59): not all of these numbers represent a font, however, and you can see the current ones if you ask for **FONT ?**.

Examples:

```
FONT_TITLE 14
font_axes 26
font 44                    (EVERYTHING in GREEK!)
FONT ?                     ("what fonts are there?")
```

FUNCTION[_<filename>] <y=f(x)> (CR) <xmin, xmax, xinc, y-dec-places>

This amazing instruction generates data from an equation you provide, ready for plotting as a graph. Before using this instruction, ensure that your EMAS process has a User Stack size greater than 256. If you do not know the equation you need, the CURVEFIT program may be able to provide one: this fits a polynomial or exponential curve to raw data points and produces coefficients for an equation which can be handled by EASYGRAPH. CURVEFIT is described in User Note 11, obtainable from your Support Team.

Having requested a function, you are prompted for the equation, which is of Y (the dependent variable) as a function of X (the independent variable), and must be in a form which would be acceptable in a FORTRAN or IMP program. Thus for example brackets must be matched, but in addition you can use functions such as SQRT and LOG. You are then prompted for the X-limits for the curve, and two 'precision parameters': the distance between values of X at which the function is to be evaluated, and the number of decimal places which are to be computed. If the generated curve is too angular, try reducing the former and increasing the latter (and/or try plotting them with **LINETYPE SPLINE**).

Having obtained the function and parameters, EASYGRAPH evaluates the function over the range of the independent variable, X, and writes the resulting pairs of coordinate points either into the file you name in the qualifier, or into an internal storage area. In either case this data is then accessible to EASYGRAPH and will be plotted if you use instruction **PLOT** next (but not otherwise!).

The function may include the names of datasets previously created with the **READ** instruction: up to 6 datasets may be included, and are specified by the name enclosed in angle brackets < >. The independent variable may, but need not, appear in the function, but will in any case continue to govern the amount of data generated. For example you can read in two datasets, Y1 and Y2, and use FUNCTION to compute their sum, thus:

```
FUNCTION Y=<Y1>+<Y2>
```

or their ratio thus:

```
FUNCTION Y=<Y1>/<Y2>
```

You still provide the X-range over which the function is evaluated, even though X does not appear therein: this will determine the maximum amount of data generated (the actual amount generated will be less than this if any of the datasets "run out" first), and also produce the X-values against which the generated data is plotted. The output thus still consists of coordinate pairs of X and Y, ready to be PLOTted.

The preceding description has shown X as the independent variable and Y as the dependent variable, but these roles may be reversed simply by giving a function which begins "X=...". EASYGRAPH then prompts for the Y range over which this is to be evaluated, though again Y need not appear explicitly in the function.

The ability to store the calculated data in a file permits the generation of quite elaborate curves. For example, the following sequence could be used to plot a graph of LOG(X) against X-squared:

```
FUNCTION_FILE1 Y=LOG(X)
1 10 1 4
FUNCTION_FILE2 Y=X**2
1 5 0.5 4
READ X1|LOGX,FILE1
READ X2|XSQ FILE2
PLOT XSQ,LOGX
```

Furthermore, data stored in file(s) can be read back into EASYGRAPH datasets for inclusion in new FUNCTIONs!!

Examples

```
Y=X**3
Y=17*SIN(X/180*PI)-39*COS(X/180*PI)
Y=0.127*X**3 - 1.729*X**2 + 0.59*X - 3.726

X=Y**2 - 14*Y
X=<res1>*<res2>/<res3>
```

Names which can be used in your function

The following names can be included in the function:

```
SIN(X)      COS(X)      TAN(X)
ARCSIN(X)   ARCCOS(X)   ARCTAN(X)
```

ABS(X)	(for the absolute value of X - i.e. nothing negative)
EXP(X)	(gives 'e' raised to power X)
MOD(X)	(as for ABS(X))
INT(X)	(for nearest integer to X)
INTPT(X)	(for nearest integer *not greater than* X)
LOG(X)	(generates LOG to base 'e' of X)
LOGTEN(X)	(generates LOG to base 10 of X)
PI	(yields value of PI (3.141592653589793))
SQRT(X)	(gives the square root of X)

HEIGHT <height>

This instruction sets the height (in cm) of the symbols plotted at the data points of your graph. The default is 0.2 cm. If you are putting errorbars on the graph, the HEIGHT setting determines the width of their endlines.

Example

```
HEIGHT 0.5
```

See instructions **SYMBOL** (to choose which of 15 symbols to use), and **ERRORBARS**.

HELP <topic>

Uses the EMAS VIEW system to scrutinize EASYGRAPH's on-line help information. This information is intended only as an aide-memoire, not as a replacement for the User Note. If you want a reminder of the parameters to a particular instruction, just try

```
HELP "instruction"
```

(See User Note 9 for information on how to use VIEW: for example to get a listing of a topic description on .LP15, go into HELP, find the topic, then use F<.LP15>)

IDENTIFICATION <ident>

This instruction causes the text string ident to be plotted as delivery information at the start of the plotter output. It may be abbreviated to ID or IDENT. The instruction is optional: if it is not encountered before the first **PLOT** instruction, EASYGRAPH uses your normal delivery information.

Example

```
IDENTIFICATION "W. Watson   Mol.Biol, KB"
```

INITPARMS or **IP**

This instruction allows you to change the default form of the basic graph: that is, it causes EASYGRAPH to 'remember' some of the features which you like in your graph, so that you do not have to re-specify them each time you start on a new one. The parameter settings are stored in a file called SS#PROFILE, using the EMAS Profile scheme (which is described in User Note 83), and affect such things as size and numbering of axes, the colour and linetype of the graph, and your delivery information. The way to use this instruction is to run EASYGRAPH, use any of the instructions from the list below whose settings you would like to store, then type **INITPARMS**. If it can, EASYGRAPH will store away all these settings and pick them up automatically in subsequent calls.

The instructions which can be stored are: AXISCOLOUR, FLATNUMBER, HEIGHT, IDENT, LINETYPE, NODRAWAXES, NONUMBER, NOTICK, PEN, PLOTTER, SYMBOL, TICKACROSS, TICKIN, TICKOUT, XAXIS length, XFORMAT, YAXIS length, YFORMAT, and the sizes of the various texts (instruction SIZE). The stored settings will remain until you change them (specifically by name - i.e. to change one you do not have to change them all) or until you lose the file SS#PROFILE.

See also the instructions **DEFAULTPARMS** which restores all parameters to their default settings, and **QUERYPARMS** which tells you what the current settings are.

INTERVAL <int>

This instruction supplies a value for the 'evaluation interval', which is used when you ask for `LINETYPE CURVE` or `LINETYPE CURVE+POINTS` to determine the smoothness of the curve. (The curve is generated by fitting a third-order polynomial to your data points taken in sets of four.) The default setting for INTERVAL is 20 (per cm), but can be any positive number.

Example

```
INTERVAL 10
```

KEY <Xstart> <Ystart>

This instruction allows you to place extra text anywhere you like on the diagram. It can only be used *after* **PLOT** has been called at least once to open the output plotfile. It takes 2 parameters - the X and Y coordinates of the point at which you want the text to begin, measured from the origin of the current graph. These values are in centimetres by default, but can be followed by any of the following units letters - 'c' (centimetres), 'd' or 'u' (your data units) or 'i' (inches). The text to be plotted is then whatever follows the KEY instruction, until terminated by the word `ENDKEY` (up to a maximum limit of 100 lines). *Take care not to forget the* `ENDKEY` - otherwise all your subsequent input to EASYGRAPH will be plotted as text! Once at least one graph has been plotted, you can call KEY as many times as you like to plot blocks of text anywhere on the diagram.

To facilitate the creation specifically of a KEY to the graph, and the different SYMBOLs and COLOURs you may have used in this, you can include in your text `@n` to obtain symbol number `n` (see description of **SYMBOL** for values of `n`, and see **HEIGHT** to change the size of the symbol in the key); or `@Cp` to switch into colour number `p` (see **COLOUR**). There is also a special command to draw a solid or dashed line of any length, at the current position and in the current colour: for a solid line include in the text `@P(l)`, and for dashed, `@P(l,d,g)` - where `l` is the length and `d` and `g` the dashlength and gaplength respectively, all in centimetres. These and other text enhancements are described in Chapter 5, on page 49.

(Various ways of using KEY are illustrated in the EXAMPLES file, members EASYGRAPH1 and EASYGRAPH2.)

Examples:

```
DATA RESULT1                    data result1
PLOT                            plot
KEY 15,12                       font_key 14
Key                             key 15c,4.4u
---                             KEY
@4 - 1901-1925                  ***
@6 - 1926-1950                  @c4@1@c1 1983
@7 - 1951-1975                  @c3@3@c1 1984
ENDKEY                          @c2@7@c1 1985
                                endkey
```

KEY could also be used to create *only* text - i.e. with no graph - though a 'dummy' graph would have to be drawn: the following would achieve this:

```
NODRAWAXES
XRANGE 0,10 YRANGE 0,10 DATA * * * * PLOT
KEY 0u,10u
etc. etc. etc.
ENDKEY
```

(See also instructions **COLOUR**, **FONT** and **SIZE** to change the appearance of key text, and **KEYANGLE** to change its orientation. Note that these must be called *before* KEY, since the key is plotted immediately the KEY instruction is received - with the colour, font, size and angle as currently set. See also **ANNOTATE** for including text at one specific point on the graph.)

KEYANGLE <angle>

This instruction sets the angle at which KEY and ANNOTATE text is to be plotted, in degrees (positive anticlockwise, negative clockwise). The default angle is zero - for horizontal text (i.e. parallel to the X-axis). The instruction is particularly useful when KEY is being used to label close-packed points along a graph.

KEY and KEYANGLE can also be used to write labels along straight lines on the graph - for example showing the equations or parameter-settings which the lines represent.

Example:

```
KEYANGLE 75
```

LINETYPE <style> [<dash> , <gap>]

This instruction is used to indicate the style of line required in your graph. <style> is an integer or mnemonic chosen from the following table:

CURVE+POINTS	— 1 —	curved plot with points marked
CURVE	— 2 —	curved plot only
LINE+POINTS	— 3 —	line plot with points marked
LINE	— 4 —	line plot only
POINTS	— 5 —	points only
HISTOGRAM	— 6 —	a histogram
TOPS	— 7 —	just the tops of a histogram
SPLINE+POINTS	- 8 —	simple spline plot with points marked
SPLINE	— 9 —	spline curve only

If the LINETYPE instruction is not used, EASYGRAPH takes style 4 as the default (straight lines connecting the data points with no points marked). The <dash> and <gap> parameters are optional, and are used to specify a DASHED line: <dash> and <gap> are in centimetres. When specifying a LINETYPE which includes points (e.g. STYLE+POINTS), there must be *no spaces* around the +. For more details of each of the four main types of curve see the individual sections below.

Examples:

```
LINETYPE 1
linetype curve+points
linetype histogram
linetype spline,0.2,0.1
```

(See also **SYMBOL**, **COLOUR**, **NEEDLES** and **SHADE** to enhance the appearance of the graph.)

CURVE

With this LINETYPE, EASYGRAPH draws a smooth curve through the series of data points. The method used takes 4 points at a time, evaluates a third-degree polynomial through them, draws the section of this curve from point 2 to point 3, then moves along one point and repeats the entire process. Two stringent conditions must be satisfied: there must be at least four points; and the X values must be in ascending order - not even two the same, but all bigger than the last! (See LINETYPE SPLINE, below, if your data does not meet this latter requirement.)

The smoothness of the resulting curve is governed by the EASYGRAPH instruction **INTERVAL**: increase this to get a smoother curve (the default value is 20).

Symbols are drawn at each data point if `LINETYPE CURVE+POINTS` is used.

(For the fullest description please see Chapter 5 of the ERCC Graphics Manual.)

LINE

With this LINETYPE, EASYGRAPH simply draws straight lines from point to point, starting with the first you specified and finishing at the last. There is no constraint on the relative values in the data - which can go up or down, back and forth in any order you like.

Use `LINETYPE LINE+POINTS` to add a symbol at each data point.

(For the fullest description please see Chapter 5 of the ERCC Graphics Manual.)

HISTOGRAM

This LINETYPE draws a simple old-fashioned histogram - i.e. with vertical bars. The best way to draw a histogram with EASYGRAPH is to read the Y-data into a dataset, and get EASYGRAPH to generate the X-data, as for example:

```
READ Y,DATAFILE
LINETYPE HISTOGRAM
PLOT X1,Y
```

Use **XAXIS** and **XRANGE** to get the output looking right - and see the example in EXAMPLES_EASYGRAPH4 if you would like some guidance.

A related facility is `LINETYPE TOPS`, which draws just the tops of the bars.

SPLINE

This LINETYPE results in a polynomial curve drawn through the points, but unlike `LINETYPE CURVE` there is no restriction on the values in the data: specifically, the X points do not need to be in strictly ascending order.

(For the fullest description please see the ERCC Newsletter of May 1986.)

MOVE <x> , <y>

This instruction allows you to position the graph wherever you want on the paper. The parameters <x> and <y> are displacements in centimetres from the angle-marker in the bottom left-hand corner of the output which marks the absolute origin of the output. <x> is the horizontal displacement, <y> the vertical.

MOVE not only allows you to place the graph wherever you like - for example to get plenty of blank space on the paper all around it, but also enables you, in conjunction with the **NEW** instruction, to put as many graphs as you like on one sheet of paper (for an illustration of this have a look at the file EXAMPLES_EASYGRAPH1). And if you want to rotate the graph (see instruction **ROTATE** below) you will *have* to use MOVE to position the origin sensibly or the graph will be rotated right off the paper! (An illustration of MOVE with ROTATE is provided in EXAMPLES_EASYGRAPH2.)

Examples:

```
move 2,15      (graph in top-left-hand segment of paper)
MOVE 20,3      (graph in bottom-right-hand segment of A3 page)
```

NEEDLES[_+/-] <baseline> <separation>

This instruction tells EASYGRAPH to plot "needles" on your graph: needles are vertical lines running from some baseline (set by yourself) to the graph-curve (the effect is like looking at park railings through the inside of your graph).

The NEEDLES instruction reads values for the baseline and for the horizontal separation of the needles, and also accepts a qualifier of '+' or '-' indicating that you want only the positive or negative needles (respectively). <baseline> is in your data units; <separation> is in centimetres by default but can be in inches or data units if followed by 'i' or 'd' or 'u' respectively. If you want a "solid infill" set it to zero. <separation> can also be the letter P instead of a distance - in which case the needles are drawn from the baseline to the actual data points.

Examples:

```
needles -25,0.4u
NEEDLES_- 0,0          (e.g. to emphasize sub-zero temperatures)
Needles_+ 50,p
```

(See also **COLOUR** and **SHADE** instructions. And there was an example of needles in the third example in Chapter 2, on page 9.)

NEW

This instruction resets the EASYGRAPH parameters ready for the production of a new graph - in the same output file. It should be followed by a **MOVE** instruction to position the new graph, otherwise this may come out over the first graph. If you want all the graphs to line up exactly, in rows or columns, you will also have to use MOVE to position the first - to give you a precise position against which to line up the subsequent ones.

There are three parameters which are not reset by NEW: **COLOUR**, **FONT** and **SIZE**. If you want to retain other settings for the new graph you should either use the EASYGRAPH instruction **INITPARMS**, or edit a list of instructions into a file and **USE** that file for each new graph.

Example

```
NEW
MOVE 2,16
USE EGCMNDS
```

(See also the examples in the file EXAMPLES_EASYGRAPH1)

NODRAWAXES[_x/y]

If given without a qualifier, this instruction tells EASYGRAPH not to plot either the X or the Y axis - thus for example allowing you to produce a diagram having no resemblance whatever to a graph (User Note 47 contains a map of Scotland produced in this way, for example).

If you only want one axis, you include in the qualifier the name (X or Y) of the one you *do not* want -

NODRAWAXES_Y	(only the X-axis will be drawn)
NODRAWAXES_X	(only the Y-axis..)

(See the instruction **DRAWAXES** for the opposite effect.)

NONUMBER[_X/Y]

This instruction suppresses the numbering of axes, leaving just the axis, tickmarks and axis-title (if requested). The optional qualifier suppresses numbering of just *one* of the axes -

NONUMBER_X	(number only the Y-axis)
NONUMBER_Y	(number only the X-axis)

(See the instruction **NUMBERAXES** for the opposite effect.)

NOSCISSOR

This is the default mode of operation for EASYGRAPH, in which any plotting (of lines or text) which goes outside the reserved plotting area is treated and flagged as an error: and EASYGRAPH aborts if there are too many of these errors. This is usually not a problem, since EASYGRAPH works out how much room it needs and scales your data automatically to fit in this. But if you override the automatic scaling and get it wrong, or if you plot a second graph with a much wider data range on top of the first, then the plot will inevitably go outside the reserved area and if excessive will cause EASYGRAPH to halt prematurely.

There are occasions however when it would be useful to let EASYGRAPH run to completion, simply "scissoring off" any plotting which goes out of area. The EASYGRAPH **SCISSOR** instruction is provided for just this purpose - and is reversed (to restore the default situation of aborting excessive out-of-area plotting) by the NOSCISSOR instruction.

```
Graph command : NOSCISSOR        (no parameter)
```

NOTICK[_X,Y]

This instruction causes the X and Y axes to be drawn without tickmarks. Note that the usual numbering will be added unless you also call **NONUMBER**. The effect can be limited to just one axis by including its name (X or Y) as a qualifier. Otherwise NOTICK takes no parameter. Its effect can be reversed before the first call of **PLOT** by any of the instructions **TICKIN**, **TICKOUT** or **TICKACROSS**.

Example:

```
NOTICK_X
TICKACROSS_Y      (no ticks on X-axis, cross-ticks on Y)
```

NOTRACE

This instruction switches off EASYGRAPH's progress-reporting, which is its default state: see **TRACE**.

NUMBERAXES[_x or _y] or NUMBERAXES[_z/n]

This is the default mode for EASYGRAPH and indicates that numbering is to be included along the axes. The instruction may be abbreviated to NUMBER, and takes an optional qualifier of X or Y, enabling you to number one axis but not the other: that is, **NUMBER_X** draws the X-axis numbers but not the Y, and **NUMBER_Y** draws the Y-numbers but not the X.

An alternative form of the qualifier allows you to specify which ticks on one or both axes are to be numbered: in this form you supply the name of the axis (which can be omitted if both are to be the same), then a slash / followed by an integer which indicates every how-many ticks to number (1 means every tick, 2 means alternate ticks, etc.). Used in conjunction with the **XRANGE** and **YRANGE** instructions this gives great flexibility in determining axis-style.

Note the subtle difference between the two forms of qualifier! The first means "number the specified axis *only*"; the second means "number the specified axis as indicated and leave the other axis as it is".

These instructions are both overridden by the **XLABELS** or **YLABELS** instructions, and also by subsequent calls of **NUMBER** or **NONUMBER** - everything can be changed as much as you like until you call **PLOT**.

Examples:

```
NUMBER_x  nonumber_Y     (same effect - number X, not Y)
NUMBER_Y/1               (put numbers at every Y-tick, leave X as is)
numberaxes_/4            (number every fourth tick on both axes)
```

See Chapter 4, below, for more examples and illustrations.

OVERLAY[_STACK]

OVERLAY tells EASYGRAPH to plot another graph on the same axes as the first, and with the same scaling. Having set up the data and parameters for the first graph and called PLOT, you call OVERLAY and then supply the fresh data - with **DATA**, **FUNCTION** or **READ** instructions (though any datasets you may have created with an earlier READ

instruction will still be available for plotting). EASYGRAPH then uses the same scaling as for the first graph to plot the second - so if you leave EASYGRAPH to do the scaling automatically make sure you use the data with the widest range first, and overlay the smaller one(s) on that!

You can use OVERLAY as often as you like, but you must use **PLOT** for each graph individually, or the new data will overwrite the old.

The optional qualifier is the word STACK, which will cause the new data to be added on to the first before plotting, cumulatively. This is particularly useful for histograms (LINETYPE HISTOGRAM), but could also be used for example in showing a company's total financial expenditure month-by-month, as the sum of the expenditures of its individual departments. (See the example in EXAMPLES_EASYGRAPH4.)

If you want to plot a second set of data over a first but with different Y-scaling, use instruction **SAMEPLOT** - not OVERLAY. And for a completely new graph, on new axes, see instruction **NEW**.

Examples:

```
DATA FILE1                    read y1|y2 datafile
PLOT                          linetype histogram
OVERLAY                       plot x1,y1
DATA FILE2                    overlay_stack
PLOT                          plot x1,y2
```

OVERLAY can be used to plot graphs with more points than EASYGRAPH can accept at one go (this limit is currently 2000) - by plotting the graph in sections. In the following examples, 5000 points are plotted - in one case using the DATA instruction to read in (x,y) pairs, in the second using READ to acquire the 'y' values and getting EASYGRAPH to generate x-values from 1 to 5000. In both cases the data is in a file called 'RAWDATA' and the qualifiers SKIP and READ are used to extract sections from this:

```
DATA_SKIP0READ2000 RAWDATA
PLOT
OVERLAY
DATA_SKIP1999READ2000 RAWDATA
PLOT
OVERLAY
DATA_SKIP3999READ1001 RAWDATA
PLOT

READ_READ2000 Y1,RAWDATA
READ_SKIP1999READ2000 Y2,RAWDATA
READ_SKIP3999READ1001 Y3,RAWDATA
PLOT X1,Y1
OVERLAY
PLOT X2000,Y2
OVERLAY
PLOT X4000,Y3
```

(Note in both cases that you would have to use **XRANGE** and **YRANGE** to scale the axes, since EASYGRAPH does not have all the data for the first plot and would therefore get the automatic scaling wrong. The reason for skipping 1999 points instead of 2000, for example, is that this causes the separate sections of the graph to join up: there would otherwise be a small gap.)

PAPER <length>

This instruction specifies how much paper (in centimetres) your entire plot will require, and is only needed if your diagram will be more than 100 cm long (in which case the only plotter big enough to take it would be .GP23).

Example:

```
PAPER 250
```

PLOT [<dsn1> , <dsn2>]

This is the instruction which draws the graph from data read in by the DATA or READ instructions or generated by the **FUNCTION** instruction. It is the only instruction which actually produces graphical output, so any other EASYGRAPH instruction can be used any number of times beforehand.

If the data was supplied by the DATA or FUNCTION instructions then PLOT takes no parameters, picking up the data automatically from within EASYGRAPH. If you used READ to create some datasets, you must tell PLOT which one to plot against which: in this case <dsn1> is the X-variate and <dsn2> the Y-variate. If there are different numbers of points in <dsn1> and <dsn2> EASYGRAPH prints a warning message, then does the best it can. As examples the following are the simplest possible calls on EASYGRAPH:

```
DATA filename
PLOT
STOP
```

and:

```
READ X,file1
READ Y,file2
PLOT X,Y
STOP
```

If you only have one set of values, which is to be the Y-variate, you can get EASYGRAPH to generate a set of X-values to plot it against, by specifying <dsn1> as:

```
Xf/g
```

where f is the initial value and g the increment (which can be negative). For example X1950/0 would produce 1950, 1960, 1970... - which could be used as years. One such "X-value" is generated for each Y-value in your dataset. f and g may be omitted: both are 1 by default, producing a sequence 1,2,3,4,....

If you have used READ to read in a number of datasets with names in a series (e.g. Y1 to Y10), you can get PLOT to draw all graphs with one call, without having to use OVERLAY - as follows:

```
READ Y1-Y10 datafile
PLOT X0/1 Y1-Y10
```

(where datafile contains 10 columns of Y-numbers). If you have Y-data in multiple sets like this EASYGRAPH can give a rough idea of its mean and standard deviation - you get these by adding the qualifier MEAN or MEAN+SD to the basic PLOT instruction, as for example:

```
READ Y1-Y10 datafile
PLOT_MEAN+SD X0/1,Y1-Y10
```

This form of the PLOT instruction does not show the raw data: if you want this as well you would have to plot this first, then call OVERLAY, then plot the mean (perhaps in a different colour!).

Note that where PLOT accepts a series of Y-datasets, as in the last two cases here, these Y-sets do not all have to be created by the same call of READ - just so long as their names fit the series - i.e. begin with the same letters (up to 3) and end with a number - with no more than 50 in all.

NB! It is this instruction - PLOT - which actually draws the picture into the plotterfile. Once it has been given you cannot change any part of the diagram previously specified: and conversely, *until* it has been given you can change everything!

(See instructions DATA, FUNCTION and READ for information on how to set up data ready to plot, and OVERLAY for examples of how to plot more than 2000 points on a single graph.)

PLOTTER <type>

This instruction determines which plotter-type the output is to be generated for. If used, it must precede the first PLOT instruction - otherwise the plotter-type defaults to Hewlett-Packard 7220T (of which the .GP15 plotter in room 3210 of JCMB is one). Since these are A3 plotters you *must* use this instruction when creating a bigger drawing. <type> should be chosen from one of the supported network devices (listed in User Note 17) or from the following table - by number or name:

```
 1  —  .GP23
 6  —  .GP16
 7  —  .GP15 or .GP33
 8  —  .GPSRSX
 9  —  .GP64
10  —  .GP34 (narrow)
11  —  .GP34 (wide)
12  —  .GPENG
13  —  .GPENGTR
14  —  .GP15TR or .GP33TR
15  —  .GPENGHTR
16  —  .GP15HTR or .GP33HTR
```

In general, whenever a new plotter is added to the master-list (and thus recognized by GPLIST) EASYGRAPH will be able to recognize both its number and its name.

Examples:

```
plotter 1
PLOTTER .GP23
```

The main function of this instruction is to tell EASYGRAPH the maximum size of paper available for the graph: if your plot will fit on an A3 page - 42 cm by 29.7 cm - there is no need to change from the default, which accommodates this. If you want a larger diagram than this however, you must use PLOTTER to tell EASYGRAPH how much space is available.

See also instruction FILE, which can not only specify a plotter, but also send output straight to it.

QUERYPARMS or **QP**

This instruction - which may be abbreviated to QP - displays the current and default settings of the major EASYGRAPH parameters - such as axis lengths and text-sizes. It is useful both to remind yourself of the settings you have given to parameters, and also as a method of displaying the settings stored by a previous call of **INITPARMS** (for the latter, use QUERYPARMS immediately after running EASYGRAPH). The parameters are stored in your file SS#PROFILE for future use by the. INITPARMS instruction.

(See also **INITPARMS** and **DEFAULTPARMS.**)

READ[_SKIPmREADnSERIAL] <dsn-name(s)> [<file>]

READ and **DATA** are the two key instructions by which EASYGRAPH reads in the data for the graph. DATA reads in (x,y) pairs, but READ reads in UNIVARIATE data - i.e. a set of X-numbers *or* a set of Y-numbers. See Section 6.2 for details of limits on the number of values which may be read. The simplest form of the READ instruction is:

```
READ <dsn>        or        READ <dsn> <filename>
```

In the first case the data itself follows the READ instruction - in the control file or from the keyboard - and is simply terminated by a non-numeric character (usually the next EASYGRAPH instruction); in the second case the data comes from <filename> and is terminated by a non-numeric character or simply by the end of the file. In both cases EASYGRAPH sets up an internal storage area for the numbers and gives it the name dsn - from which it can be accessed by the **PLOT** instruction.

Examples:

```
READ XSET 1, 2, 3, 4, 5

READ YSET YFILE

READ TIME,TIMEFILE
READ TEMP,TEMPFILE
PLOT TIME , TEMP
```

EASYGRAPH stops if you try and overwrite an existing dataset of the same name - should you want to do this use the **DELETE** instruction first. Otherwise all dataset names are entirely local to EASYGRAPH, and are lost automatically when you return to EMAS *Command:* level.

You can read in more than one dataset with one call of READ, if the data is in suitable format. For this the instruction is:

```
READ dsn1|dsn2|dsn3|...
```

or

```
READ dsn1-dsnn
```

In either case the maximum number of datasets is 50. In the first case each <dsn> is an entirely separate name (of up to 8 characters). In the second case EASYGRAPH constructs the names, as an ordered sequence between the limiting names given - which must both have the same "root" (of up to three letters), followed by a number. In the example shown here EASYGRAPH would construct n names, starting at 1: dsn1, dsn2, dsn3, ... ,dsnn. Any range could be used - for example PIG189-PIG207 - providing there were no more than 50 altogether.

In all cases the names are followed by the data, or by the name of a file which

contains the data.

Examples:

```
READ PIGS|COWS|HORSES|RABBITS filename
read pop1-pop8              (meaning pop1, pop2, pop3, . . pop8)
```

In either case these names can be used individually in the PLOT instruction.

Where a number of sets are being read in at once there is a choice of data format - 'serial' or 'parallel'. In the serial case all the numbers for the first dataset come first, then all those for the second set, then the third, and so forth. To allow different amounts of data in the various sets EASYGRAPH needs to be told where each set ends - indicated in the file by an exclamation mark !. It also needs to be told that the data is serial rather than parallel, as follows:

```
READ_SERIAL Y1-Y3
1 2 3 4!
2 3 4 5!
3 4 6 8!
```

by which Y1 contains numbers 1 2 3 & 4; Y2 2 3 4 & 5; and Y3 3 4 6 & 8.

The parallel case is simpler, and is the default for 'multiple reads'. Here the first number in the input becomes the first number in <dsn1>, the second becomes the first in <dsn2>, the 3rd the first in <dsn3>, and so on until each dataset contains one number. The next input number is then taken to be the second in <dsn1>, and so forth. In brief: the serial form reads ROWS into datasets; the parallel form reads COLUMNS.

Example:

```
READ YY1-YY4
10 100 1000
2 20 200 2000
3 30 300 3000
4 40 400 4000
```

(this is a parallel read, by which:

YY1 ends up as 1,2,3,4,..
YY2 as 10,20,30,40,..
YY3 as 100,200,300,400,..
and YY4 as 1000,2000,3000,4000,..)

The READ instruction can be qualified with the keywords SKIP and READ to tell EASYGRAPH to skip some number of lines before beginning to read data, and/or to limit the number of data values it is to take. The general form of this instruction is:

```
READ_SKIPmREADn <dsname>
```

(followed as ever by the name of a file which contains the data, or by the data itself). m and n are integers, and this instruction means "skip m lines of the input, then read n values into the dataset dsname". There must be *no spaces* anywhere in the SKIP or the READ!

Example:

```
READ_skip2read20 y1,resfile
```

There is a further example in the description of **OVERLAY**, above.

As with DATA, output from a Fortran program must be converted into character-file

format if necessary.

If SKIP and READ are to be used in conjunction with a serial READ which is creating several datasets, the word SERIAL is added after the READn - all without spaces; i.e.:

```
READ_SKIPmREADnSERIAL <dsname>
```

Once a set of numbers has been read into a named dataset in EASYGRAPH there are various ways of using it: it can be plotted against another, read in at the same time, or against data generated by EASYGRAPH; or all the sets can be plotted at once with or without their mean and standard deviation (see **PLOT**); it can be TRANSFORMed - for example to take its LOGs or to SQUARE it; it can be included in a FUNCTION to generate new data; and it can be used as the magnitude of errorbars. (See instructions **PLOT**, **TRANSFORM**, **FUNCTION** and **ERRORBARS**, respectively.)

ROTATE <angle>

This instruction rotates your whole graph through <angle> - which is in degrees, positive anticlockwise. Rotation is precisely all this instruction does, and you will also need to use the **MOVE** instruction to relocate the graph-origin, otherwise the graph may be rotated right off the paper!

Example:

```
rotate 90
```

There are various illustrations of the use of this instruction in files EXAMPLES_EASYGRAPH1 and EXAMPLES_EASYGRAPH2.

SAMEPLOT

This instruction is used to indicate that another graph is to be drawn over the previous one (i.e. on the same axes), using the same X-axis scaling, but recalculating the Y-axis scaling from the new data (cf. **OVERLAY**, which does not recalculate the Y-scaling). A second Y-axis is added automatically at the right-hand side of the graph to indicate the new scaling unless suppressed with **NODRAWAXES**. Indeed any number of additional axes can be added, one for each SAMEPLOT, and located anywhere along the X-axis by use of the 3rd parameter to the **YAXIS** instruction. Use instruction **YRANGE** or **YSCALE** to override the automatic scaling-calculation; and use any/all of YAXIS, TICK instructions, **COLOUR**, **FONT**, **NUMBERAXES**, **YLABELS** and **YFORMAT** to modify the appearance of the second axis.

SAMEPLOT takes no parameters.

Example:

```
DATA RESULTS01
PLOT
SAMEPLOT
DATA RESULTS02
PLOT
```

SCISSOR

With this mode selected, EASYGRAPH will cope with any amount of plotting which goes outside the bounds of the reserved plotting area - as a result of the scaling specified by yourself or as calculated from your data. Such "out-of-area" plotting occurs usually because of a mistake in an **XRANGE,YRANGE** or **XSCALE,YSCALE** instruction; or because a second graph overlaid on a first has a much bigger range in its data. Both of these would normally be regarded as faults which required correction, so the SCISSOR instruction is a tool which assists in the development of a new graph - it should not be needed once the graph is entirely correct.

SCISSOR chops a graph at the boundary of the physical paper: the **XLIMITS** and **YLIMITS** instructions enable you to define any window you like outwith which the graph is to be chopped or data points excluded.

SCISSOR takes no parameter: it is cancelled by **NOSCISSOR**.

SHADE[_style] <sepn>

This instruction tells EASYGRAPH to shade a histogram - plotted as **LINETYPE HISTOGRAM**. Shading is by straight lines in horizontal, vertical or diagonal direction, or any combination of these - the default being diagonal from lower-left to upper-right. The direction of the shade-lines is specified by the qualifier, by using one of the following characters to represent the style you want:

```
/  \  |  -  +  x  *
```

The parameter <sepn> specifies the separation in centimetres you want between the shade-lines. If you have shaded something and want to overlay it with an UNSHADED plot, use **SHADE OFF**.

Examples:

```
shade_x 0.25      (cross-hatching, lines separated by 0.25 cm)
SHADE_\ 0           (solid infill)
```

(For further illustration see the examples in the EXAMPLES file, members EASYGRAPH1 and EASYGRAPH4 in particular.)

SIZE_<component> <size>

The SIZE instruction is used to override the default sizes assumed for texts and ticks on the graph drawing. Note that the actual height of characters is `1.75*<size>`, and total width occupied `1.5*<size>`. <size> is a number (in centimetres) and <component> is one of the following:

GRAPHTITLE	or **GT**	Initial size 0.30 cm
XTITLE	or **XT**	0.24
YTITLE	or **YT**	0.24
XNUMBER	or **XNUM** or **XN**	0.10
YNUMBER	or **YNUM** or **YN**	0.10
ANNOTATION	or **ANN**	0.15
KEYTEXT	or **KT**	0.20

These values are included in the **QUERYPARMS** information. See also **FONT** and **COLOUR** to modify the appearance of text.

SIZE is also the instruction to set ticklengths for the X- and Y-axes, in which case <component> is XTICKS or YTICKS, and <size> is again a number of centimetres. The default length for ticks is 0.15 cm - or 0.3 cm centred for ticks which cross the axis (see TICKACROSS, below). One interesting use of the SIZE instruction is to set ticks to the same length as the other axis, and set TICKIN for inward-pointing ticks - which results in a "grid" on the graph. (See the LOG graph in EXAMPLES_EASYGRAPH1 for an illustration.)

Note for both text and ticks that <component> is separated from the word SIZE by an underline character *without spaces*, and *not* a hyphen (which would be ignored !).

Examples:

```
SIZE_XNUMBER 0.15
SIZE_ANN 0.05

tickin
size_xticks 12
size_yticks 16          (draws a "grid" of ticklines on the graph)
```

SMOOTHE <n> / <a>

This instruction runs a 'moving-average filter' over your data to smoothe out a spiky graph. The general form of the instruction is: SMOOTHE n/a which means that EASYGRAPH will take the average of the first <n> points, then advance by <a> points and take the average of the next <n>, repeating until it gets to the end of the data. If you omit <a> it defaults to 1, and you end up with the same number of points as you started with. The larger <n> and <a> are, the coarser is the smoothed result, tending towards a straight line as <n> approaches the total number of data points. When you first use this instruction try starting with an odd value of <n> around 5%-10% of the total number of points, and <a> less than <n> - probably 1. (The effect of setting <a> greater than 1 is to compress the data, since the number of points is reduced to 1/a of the original number.)

There is nothing sophisticated about this technique, but it can give a useful, qualitative view of a trend in widely-varying data.

Example:

```
smoothe 6/2
```

SYMBOL <sym>

This instruction is used to select a character (see following table) for plotting at the data points on the graph with linetypes POINTS, LINE+POINTS, CURVE+POINTS or SPLINE+POINTS. The default symbol is CIRCLE.

```
CROSS       -- 1 -- cross
PLUS        -- 2 -- plus
SQUARE      -- 3 -- square
DIAMOND     -- 4 -- diamond
CIRCLE      - - 5 - - circle - the default
TRIANGLE    -- 6 -- triangle
DELTA       -- 7 -- inverted triangle
```

```
RIGHTARROW      -- 8 -- right arrowhead
LEFTARROW       -- 9 -- left arrowhead
ASTERISK        -- 10 -- asterisk
DOT             -- 11 -- dot
MINUS           -- 12 -- minus
VERTICALLINE    -- 13 -- vertical line      (or VLINE)
SLASH           -- 14 -- slash
BACKSLASH       -- 15 -- backslash
```

See the HEIGHT instruction for adjusting the size of the symbol.

Examples:

```
symbol 12
SYMBOL CIRCLE
```

TICKACROSS , TICKIN , TICKOUT

TICKACROSS causes axis tickmarks to be drawn symmetrically on either side of the axis or axes, hence are twice the length specified by SIZE.

TICKIN causes the axis tickmarks to point upwards from the X-axis or rightwards from the Y-axis, i.e. pointing in towards the graph.

TICKOUT causes the axis tickmarks to point downwards from the X-axis or leftwards from the Y-axis, i.e. away from the graph.

Ticks are 0.15 cm long by default: use SIZE_XTICKS or SIZE_YTICKS to modify this. Also, all three instructions can take a qualifier of X or Y meaning "this tickstyle on specified axis only, others unchanged". And as usual in EASYGRAPH, you can change your mind about tick-style as often as you like until you call PLOT.

Examples:

```
TICKIN_X            tickacross_y
```

TITLE <graph title>

Use this instruction to put a title along the top of the graph area. `<graph title>` is the text and must be enclosed in single or double quotes, and it must be given all on one line: EASYGRAPH treats the end of the line as the closing quote mark. If you want more than one line in the title put `@B` in your text string wherever there is to be a line break (for example "`line 1@bline2`"). Each line of the title is left-justified, but can be centred or right-justified by the inclusion of spaces (see second example below). If you want to include the quotes delimiter in the text, repeat it: for example `TITLE 'Fingal''s Cave'` will be plotted as `Fingal's Cave`. See instructions COLOUR, SIZE and FONT to modify the appearance of the title - and also the notes on text strings below.

If you do not like the way EASYGRAPH positions the title you can use the KEY instruction to put any text anywhere you like on the graph.

If you do not use TITLE (or KEY), no title is given to the graph.

Examples:

```
TITLE "Graph of Y vs X"
title "Distribution of Godwit@B          (Portobello, 1986)
```

TRACE

This instruction switches on a rudimentary 'tracing' facility in EASYGRAPH, causing the program to print out occasional messages to let you know how it is getting on. This may help in pinpointing the cause of out-of-area warning messages - for example to indicate whether "PLOTSTRING number 24" is referring to axis-labelling or a key.

TRACE takes no parameters, and is switched off by **NOTRACE**.

TRANSFORM <setname> , <transform>

This instruction takes an existing dataset <setname>, transforms it according to the transformation you request <transform>, and stores the transformed values back in the same dataset. The currently-recognized transforms are:

EXPONENTIAL	-- e**X (can abbreviate to **EXP**)
LOGE	-- natural logarithm
LOG or LOG10	-- logarithm (base 10)
SQUARE-ROOT	-- square root
SQRT	-- square root
DIFF	-- replace the 'absolute' data with its DIFFERENCES
ACC	-- ACCUMULATE the data - i.e. add each to previous value
****C**	-- raise to power C (a constant)
+C	-- add const C to all values
-C	-- subtract const C from all values
C-	-- subtract all values from const C
***C**	-- multiply all values by const C
/C	-- divide all values by const C
C/	-- divide const C by all values

If you read in your data with the **DATA** instruction rather than with **READ** you will not have any datasets. You can still transform data though, referring to the X-points as "X" and Y-points as "Y".

Examples:

```
TRANSFORM X-VARIATE LOG

TRANSFORM Y, SQRT

data resfile
transform y,**2          (square all the Y-values..)
```

(See instruction **FUNCTION** for a method of combining datasets.)

USE <filename>

This instruction tells EASYGRAPH to read its instructions from <filename>. This is definitely the best way to use EASYGRAPH - to edit a set of instructions into file <filename>, run EASYGRAPH, then USE <filename>. EASYGRAPH looks for further instructions, and possibly data, in <filename> until it reaches the physical end-of-file or the word END (at which point the program will return control to the keyboard or a previous instruction file - whichever the USE came from).

Note that a file being USEd may also USE other files.

It is more flexible to run EASYGRAPH, then USE <filename>, than it is to call EASYGRAPH with <filename> as a parameter - which is the alternative way of getting EASYGRAPH to read input from a file. You can have as many instruction files as you like, and USE one after another - for example with axis-formats in one, title formats in a second and data in a third. This certainly simplifies calling EASYGRAPH from a program (see Section 6.7, page 54). And when EASYGRAPH is used interactively this method allows you to look at the result immediately with the VIEW instruction. Then if it is not quite right you simply have to make corrections in the instruction file, then USE the file again, rather than having to type all the instructions in again from scratch.

Example:

```
USE COMMANDFILE
VIEW
```

VIEW

This parameterless instruction tells EASYGRAPH to stop plotting and display the result on your terminal. To do this it calls the DRAWPICTURE command, which in turn requires that you have previously used the EMAS command TERMINALTYPE to tell the system what type of terminal you are using. This does not have to be a graphics terminal - if it is not you will just get a rough idea of the graph, drawn in asterisks.

Example:

```
TERMINALTYPE 27        (a Datatype X5A)
EASYGRAPH
USE CMDFILE
VIEW
```

Note if using a BBC microcomputer with terminal-emulator that you must either

- run XTALK in the BBC and use TERMINALTYPE 29 on EMAS

or

- run the Sussex Workstation on the BBC, choosing menu option 2 (Tektronix 4010), and use TERMINALTYPE 17 on EMAS.

If you do not have the BBC and EMAS set for the same terminaltype you will get streams of text on the screen, not a picture!

If you forget to call TERMINALTYPE before running EASYGRAPH, you can call it from inside EASYGRAPH before calling VIEW as follows:

```
Graph command :USE CMDFILE
Graph command :!TERMINALTYPE 27
Graph command :VIEW
```

XAXIS or **YAXIS** **<length> , <title> , <position>**

The XAXIS and YAXIS instructions allow you to specify the LENGTH, TITLE and POSITION of the corresponding axis. **<length>** is the length you want the axis to be - in centimetres - and is the only way to change the width or height of the graph. **<title>** is the text string you want plotted as a title along the axis. (The Y-axis title runs vertically, along the axis: there is a suggestion on page 47 for creating a horizontal Y-title.) **<position>** is the data value on the other axis at which you want *this* axis to cross it: by default the axes cross in the bottom-left-hand corner of the diagram, but with this instruction you can get them to cross anywhere. For example if your Y-axis scaling goes from -10 to +10 and you want the X-axis at Y=0, use XAXIS with **<position>**=0 - meaning "start the X-axis at Y=0". If you move the X-axis up to the top of the Y-axis, or the Y-axis across to the right-hand end of the X-axis, the tickmarks and labelling will automatically be moved round to be on the side of the axis away from the graph.

If omitted, **<length>** defaults to 12 cm, **<title>** to nothing, and **<position>** to "bottom left-hand corner". Note though that you must specify a length and/or a title if you want to change **<position>** - since otherwise **<position>** might be read as **<length>**. All the usual text-modifying instructions can be used with **<title>**: for example, see **FONT**, **COLOUR** and **SIZE**, and the string instructions in Chapter 5; and the title can be moved to right or left by the inclusion of leading or trailing spaces (it is centred by default). Use "@b" to produce newlines (for example **"line1@Bline2"**).

There is only one X-axis per graph, since X-scaling must always be the same for all graphs. However, EASYGRAPH's **SAMEPLOT** instruction allows a new graph to be plotted on existing axes with new Y-scaling, and therefore with a new Y-axis. Any number of graphs can be added in this way, and any number of Y-axes! With judicious use of colour (**COLOUR_GRAPH** and **COLOUR_YAXIS**), **TICKIN** & **TICKOUT**, and **SIZE_YTICKS**, one can achieve a reasonable - if unusual - effect.

Examples:

```
XAXIS 12,"Xaxis title"

XAXIS "Xaxis title"

xaxis 22,"          Month1985-6"

yaxis "",0          (start Y-axis at X=0; no title, 12 cm long)

YAXIS 24,"A@t267@n concentration",50
```

XFORMAT or **YFORMAT** **<m> , <n>**

These instructions control the format of numbers plotted along the axes: they specify the number of places before **m** and after **n** the decimal point.

Specifically: if m>0, n=0 the number is in INTEGER format.
m>0, n>0 REAL
m=0, n>0 EXPONENTIAL or SCIENTIFIC

Any other combination of **m** and **n** is ignored.

To save having to remember numbers, you can use the words INTEGER, REAL or SCIENTIFIC as parameters to the FORMAT instructi---

The default values for m and n are 2 and 2 respectively, for REAL numbering.

Examples:

```
XFORMAT 0,3              yformat 0,5
XFORMAT 5,0              yformat integer
xformat 2,1
```

(See also **FONT**, **COLOUR**, **SIZE**, **FLATNUMBER**, **XLABEL** and **YLABEL** to alter the appearance of the axis-numbering.)

XLABEL or **YLABEL <label1/label2/label3/...>**

It is sometimes useful to be able to plot text labels along an axis rather than numbers, and these instructions enable you to do this. The parameter is a list of labels, separated by a slash /, and all on the same line. The current limits are 6 characters per label, and 50 labels altogether. The first label is then placed at the first tick (by default at the origin), the second at the next and so forth. To leave a tick unlabelled simply place no text between its slashes - e.g. "tick1//tick3/..".

Note that if you use these instructions you will probably also have to use the **RANGE** or **SCALE** instructions to get the scaling right - so that the labels relate correctly to the data (for example EASYGRAPH will not stop you labelling graph-data values of 1,2,3,.. as 300,400,500,....!). Note further that EASYGRAPH does not check the labels - for example to ensure that there are not too few or too many.

See **SIZE**, **COLOUR** and **FONT** to alter the style of the labelling characters; **KEY** for another way of adding text; and **NUMBERAXES** to override text-labelling and restore ordinary numbering.

Example:

```
XLABELS  /Jan/Feb/March/Apr/May/Jun/Jul/Aug/Sept/Oct/Nov/Dec
```

Note that if using I/.../ in EDIT to insert your instructions into a file that you type two slashes together to get one into the file: a single one would terminate the insert! For example:

```
Edit: I/XLABELS //Jan//Feb//...../
```

XLIMITS or **YLIMITS[_CUT] <low> , <high>**

These instructions are used to exclude data points which lie outside a chosen area - this area being defined by the <low> and <high> parameters. If there is no qualifier, points lying outside the area are *removed* from the data before the graph is drawn. If the qualifier CUT is included the effect is to place a 'window' on to the graph: no data points are ignored, but wherever the graph goes outside your specifed area the line becomes invisible until it comes back into the area.

Thus the former method ignores data outside the area; and the latter indicates the presence of these points but does not abort the graph as it otherwise would because of excessive out-of-area drawing.

<low> and <high> set the actual limits, in your data units. Either can be specified as * to mean the edge of the graph, or ? to mean the edge of the whole drawing area (i.e. "no limit").

Examples:

```
XLIMITS 40,60
YLIMITS 10,15

xlimits *,*
ylimits_cut *,30

XLIMITS ?,100
YLIMITS *,100

YLIMITS_CUT 10,15

xlimits_cut 40,60
```

XRANGE or **YRANGE[_LOG** or **_p/q] <low> , <high>**

These instructions set the data range which the axis is to cover, and optionally the style and positioning of the tickmarks along the axes. **<low>** is the value for the left-hand end of the X-axis or bottom of the Y-axis, and **<high>** is the value for the other end. **<high>** does not have to be greater than **<low>**: that is, the range can go in a 'negative' direction along the axis, as for example **XRANGE 10,0**.

Thus these instructions determine the scaling of the graph. They are optional, since EASYGRAPH by default works out the range from your data - though in doing so it rounds out the scaling to its idea of neatness, which the RANGE instructions allow you to override. You will also need one or both RANGE instructions if you are going to overlay several sets of data and the first (from which the automatic scaling would be calculated) does not have the biggest range.

You can share the ranging work with EASYGRAPH by specifying either **<low>** or **<high>** as ?, in which case EASYGRAPH will work out that value from the data and accept the other one as you give it.

Apart from the parameters (**<low>** and **<high>**), the RANGE instructions accept Qualifiers which can radically alter the appearance of the axis, by determining where the tickmarks are to be placed: these qualifiers are LOG for logarithmic ticks, or p/q for regular tickmarks starting at <p> and spaced every <q>. Take care in both cases not to forget the **<low>** and **<high>** parameters: it is easy to overlook these in the excitement of the qualifier but they must still be provided!

When qualifier LOG is used to position tickmarks logarithmically, **<low>** and **<high>** should both be (positive or negative) powers of 10 (EASYGRAPH will adjust them to be, if necessary). The word LOG can optionally be followed by /n to show that only every nth tick is actually to be drawn: n can be 1,2,3,5 or 10. The numbering of log-ticks is made INTEGER automatically: use X,YFORMAT instructions *after* X,YRANGE to override this.

The other form of the qualifier is p/q, where p is the position for the first tickmark (distance along the axis from the origin), and q is the interval for the subsequent ticks. Both or either accept a units-specifier - c for cm (the default), i for inches, or d or u for data units. An asterisk instead of a number for p is interpreted as "end of axis".

In both cases, note that if using I/.../ in EDIT to insert your instructions into a file that you type two slashes together to get one into the file: a single one would

terminate the insert! - e.g. `Edit: I/XRANGE_LOG//5 0,10/ .`

Examples:

```
XRANGE 0,100

yrange 0,?

XRANGE -1E7, 10@3

yrange_log/5 0.01 100

YRANGE_li/2i 0 100
```

You could plot your own log-log graphpaper by specifying:

```
TICKIN
SIZE_XTICKS <length of Y-axis>
SIZE_YTICKS <length of X-axis>
```

(see the log example in EXAMPLES_EASYGRAPH1)

There are various illustrations of the use of RANGE in the EASYGRAPH members of EXAMPLES.

(See also the **XSCALE** and **YSCALE** instructions for setting the axis-scales.)

XSCALE or **YSCALE <fv> , <dv>**

These instructions are alternatives to the **XRANGE** and **YRANGE** instructions for the specification of the exact scale of the axes. <fv> is the lowest value for the axis, and <dv> is the increment in data units per centimetres. By default, EASYGRAPH tries if possible to have <fv> as 0. Otherwise, both <fv> and <dv> are chosen to maximize use of the axis length and to be in the form i*10**j, where 'i' is an integer in the set {1, 2, 2.5, 4, 5, 8}, and 'j' the appropriate integer exponent.

If the word **AUTO** appears instead of numeric values, the axis will be scaled automatically by EASYGRAPH - which is the default situation anyway.

Examples:

```
XSCALE 0,1
YSCALE 5.5,2
XSCALE -10,-1
```

NB! - the SCALE instructions override any previous call to the corresponding RANGE.

CHAPTER 4
INSTRUCTIONS TO CHANGE THE AXES

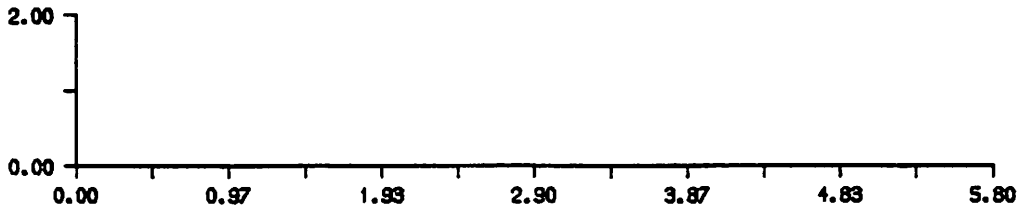
To a very large extent the form of the AXES governs the overall appearance and impact of the whole graph - by their length, position, scaling and labelling - even their presence or absence! When constructing a new graph it is usually best to begin by specifying how the axes should look, after which the rest of the diagram should fall easily into place. EASYGRAPH instruction files therefore commonly begin thus:

```
XAXIS length , "title"
XRANGE minvalue , maxvalue
XFORMAT INTEGER/REAL/SCIENTIFIC/m,n
YAXIS length , "title"
YRANGE minvalue , maxvalue
YFORMAT INTEGER/REAL/SCIENTIFIC/m,n
```

This chapter assists with the design of axes by indicating the use of various axis-changing instructions: hopefully by scanning through the illustrations you will quickly pick up the instructions you need. The EASYGRAPH instruction-file which produced the diagrams for this chapter is available on-line, as EXAMPLES_EASYGRAPH7 in CONLIB on BUSH or EMAS or ERCLIB on EMAS-A. A Y-axis is indicated in a few of the following diagrams, but is usually excluded to save space. However, all the X-instructions have Y-equivalents which achieve the same effect.

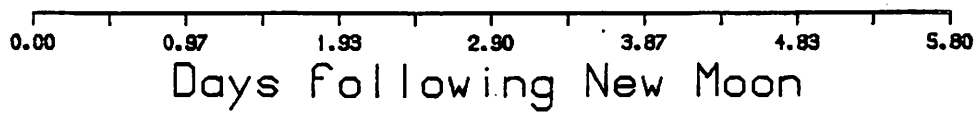
The default axes are 12 cm long, have no title, meet in the bottom left-hand corner of the diagram, and are scaled automatically by EASYGRAPH to fit your data. They might therefore look something like this (though here the Y-axis has been heavily truncated):

(default settings)



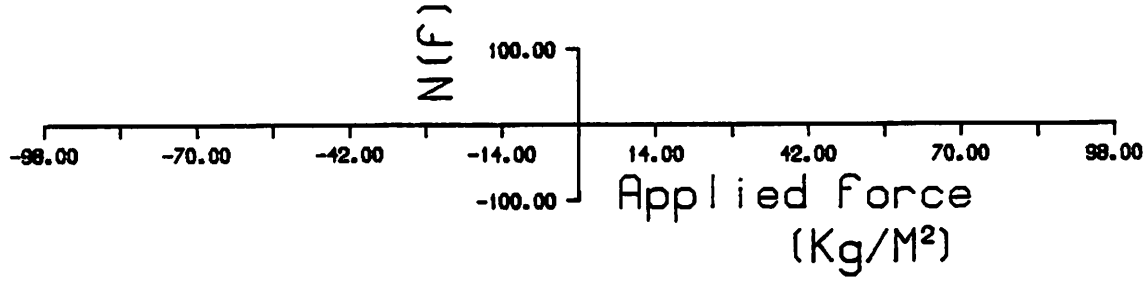
The XAXIS and YAXIS instructions control the length, title and relative positions of the two axes. To add a title to an axis, use XAXIS and/or YAXIS, followed by the text of the title enclosed in quotes, as for example:

```
XAXIS "Days following New Moon"
```



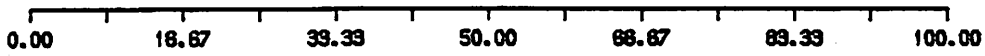
The XAXIS and YAXIS instructions can also be used to fix the lengths and relative positions of the two axes, as in the following example:

```
XAXIS 14,"        Applied Force@b          (Kg/M@s2@n)",0
YAXIS 2,"      N(f)",0
```



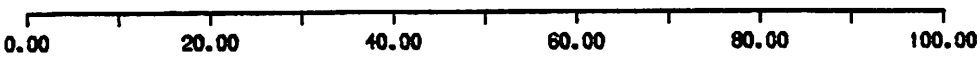
In the previous example the third parameter, 0, specifies where the two axes are to cross - the Y-axis at X=0 and the X-axis at Y=0. When EASYGRAPH is doing its own scaling the 0 values may not fall neatly on a tickmark, and it will usually be necessary to take over control of the scaling yourself to get a neat result - by using the XRANGE and YRANGE commands. These fix the starting and finishing values for the axes, as for example:

```
XRANGE 0,100
```



The RANGE commands can also be used to fix the position of the tickmarks. By default these are spaced every centimetre, with every second one numbered, as in the preceding examples. The RANGE qualifiers specify a starting point and interval for the ticks, in data units, centimetres or inches - thereby enabling you to avoid the curious numbering obtained - as in the previous example - when the default numbe· of tickmarks does not divide cleanly into the axis-range. For example:

```
XRANGE_0d/10d 0,100
```

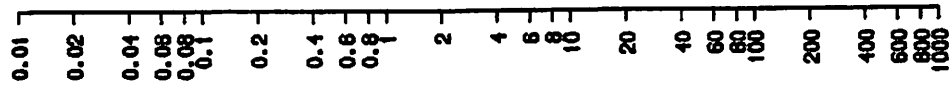


As well as regularly-spaced tickmarks, EASYGRAPH can draw logarithmic axes, and this style too is specifed by the qualifiers to the RANGE instructions. The word LOG, optionally followed by '/n' to draw every 'n'th tickmark in each decade - specifies an axis in log-format, with every tick labelled by default. (**NB!** specifying a log-axis does not automatically convert data into logs: this must be achieved separately, for example with TRANSFORM.)

```
XRANGE_LOG/5 1,100
```

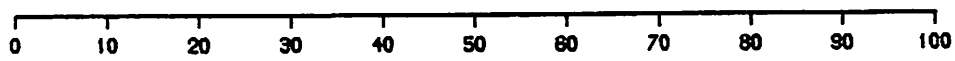
The minimum axis value for log-ticks can be any positive number down to zero, and it is possible that decimal numbers between 0 and 1 could overlap if the tickmarks come too close together - as indeed they could with closely-spaced regular tickmarks. One answer to this problem is to turn the X-numbers around 'sideways' - for which the instruction is FLATNUMBER_NO. (FLATNUMBER without the qualifier turns them back to horizontal; there is no equivalent for the Y-axis):

```
XRANGE_LOG 0.01,1000
FLATNUMBER_NO
```



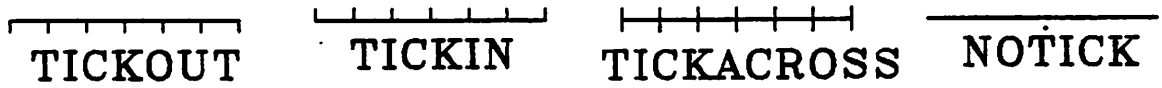
In all the preceding examples the axes have been numbered with real values to two decimal places. The XFORMAT and YFORMAT instructions enable you to specify the numbering format, either with 2 integers which specify the number of places before and after the decimal point, or with one of the keywords REAL, INTEGER or SCIENTIFIC. The following two examples illustrate INTEGER and SCIENTIFIC numbering, and in passing show that the NUMBERAXES command accepts a qualifier which specifies every how many tickmarks are to be numbered:

```
NUMBERAXES_X/1
XFORMAT INTEGER
```



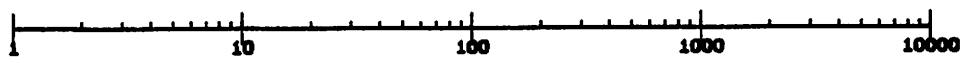

```
NUMBERAXES_/1
XFORMAT SCIENTIFIC
FLATNUMBER_NO
```

Whatever style of axis and number-format is adopted, the TICKMARKS can go in, out, across, or be excluded altogether. Their numbering can also be suppressed, as in these four examples, which show the four tickmark styles.



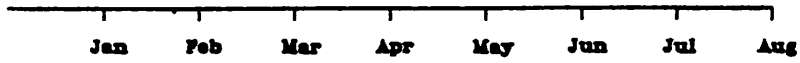
The tickmarks are 0.15 cm long by default: they can be changed with the SIZE command. All ticks along a single X or Y axis have to be of the same length, though special effects could be obtained by overlaying two graphs at the same position, as for example:

```
MOVE 3,3
XRANGE_LOG/10 1,10000
TICKACROSS
NEW
MOVE 3,3
NONUMBER
XRANGE_LOG/1 1,10000
TICKIN
SIZE_XTICKS 0.08
NONUMBER_X
```



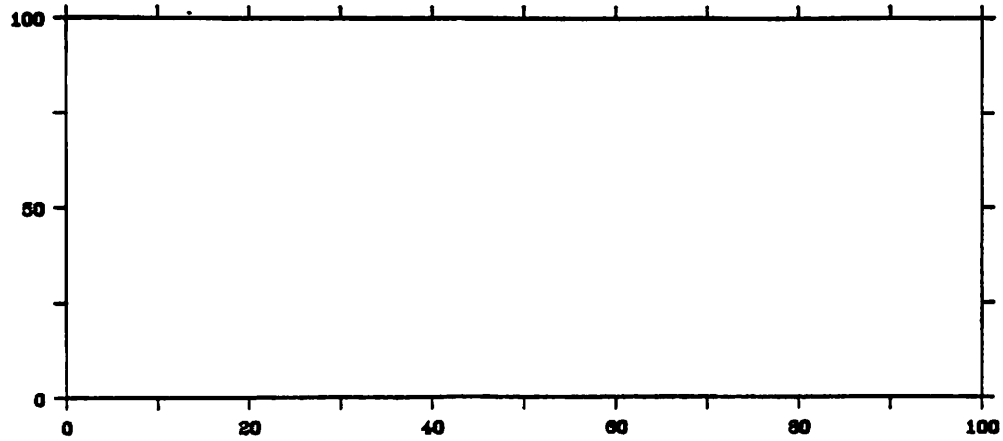
The tickmarks can be labelled rather than numbered, using the XLABELS and YLABELS instructions, which take as parameter a list of text labels separated by slashes /. At present the individual labels are restricted to 6 characters in length. For example:

```
XRANGE_1d/1d 0,12
XLABELS Jan/Feb/Mar/Apr/May/Jun/Jul/Aug
```



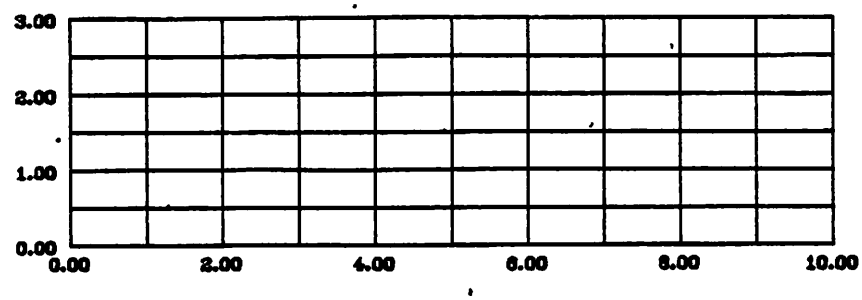
By default a graph only has two axes - X and Y - but the AXISBOX instruction enables you to add axes at the top and right of the diagram to complete a box of axes:

```
AXISBOX
```



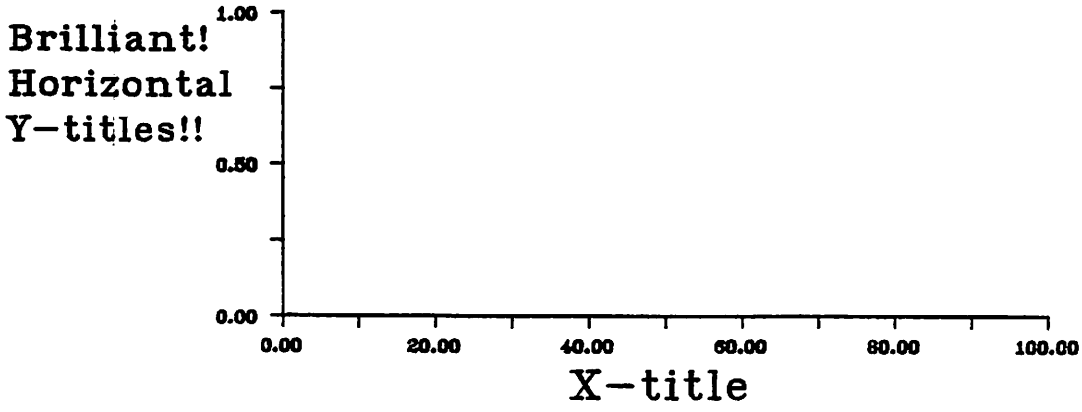
Incidentally, by choosing inward pointing tickmarks (TICKIN) and setting their length to the length of the axes (SIZE_XTICKS, SIZE_YTICKS) you can obtain a complete grid. (The example in EXAMPLES_EASYGRAPH1 includes such a grid on logarithmic axes.)

```
TICKIN
XAXIS 10
SIZE_YTICKS 10
YAXIS 3
YRANGE_0/0.5 0 3
SIZE_XTICKS 3
```



The SIZE instruction used in the previous example can also be used to change the sizes of all the text items. The other two instructions which can have a big impact on the appearance of the axes, but which are not illustrated here, are FONT (to change the font used for titles and numbering); and COLOUR (to change the colour of any of the axis-components).

The final example in this chapter merely shows that KEY can be used instead of YAXIS to produce a horizontal title for the Y-axis, instead of one which runs along the axis and which therefore has to be read 'sideways'. If you want to see exactly how this was achieved please look at EXAMPLES_EASYGRAPH7.


Brilliant!
Horizontal
Y-titles!!
1.00
0.50
0.00
0.00
20.00
40.00
60.00
80.00
100.00
X-title

CHAPTER 5
THINGS TO DO WITH TEXT STRINGS

5.1. Plotting strings

When an EASYGRAPH instruction requires a text string, the string is delimited either by quotes (") or by apostrophes ('). The characters of the string may be in upper and lower case, and must *not* include any newlines: the end of a line is taken as the end of the title, whether or not preceded by the closing quote delimiter. There is however a mechanism to generate newlines and certain control functions, using the reserved character '@': whenever this is encountered in the string of text, it is discarded and the next character is used to specify a control function, as follows:

Char	Feature
B	Insert a NEWLINE into the string at this point
S	Enter superscript mode (exit with @T or @N)
T	Enter subscript mode (exit with @S or @N)
N	Enter normal mode from subscript or superscript
I	Set italics mode ('unset' italics with @V)
V	Set vertical (normal) mode after italics
L	Translate all upper case letters to lower case
U	Do not translate upper case to lower case
G	Select a Greek character set
R	Select normal (English) character set after Greek
Cn	changes immediately to colour 'n' (see also COLOUR, Chapter 3 above)
n	for symbol 'n', 1<=n<=15 (see SYMBOL, Chapter 3 above)
F(n)	switches immediately to GIMMS font 'n'
Z(c)	plots symbol 'c' at current position - 'c' a number 0<=c<=127
P(l,d,g)	plots a line 'l' cm long, dashed by 'd' and 'g' if given.
@	Insert this character into the string (i.e. "@@" plots as @)
any other	IGNORED

Initially, the settings are equivalent to @N@V@U@R. Note that the Greek and English character sets do not correspond exactly, and you may need to refer to the ERCC Graphics manual for the Graphpack character codes (EASYGRAPH generates a Greek letter by adding 64 to the letter you give it. Thus to plot a particular Greek character, look it up in the Graphics Manual and track back four columns to the corresponding English character. This is equivalent to your subtracting 64 from the Greek character, and when EASYGRAPH finds this English letter it will add the 64 back on to get the Greek one. Thus for example if you present the string "@GR", you will get a capital sigma plotted.

If you are using a GIMMS font and ask for Greek symbols with "@G", the font is switched automatically to one of the GIMMS Greek ones; in such a case "@R" will then switch back to the GIMMS font you were originally using.

Example:

```
data
0.1 0.5
3.2@-1 1.1
0.51 1.32
1.0,2.1
1.08 2.3
1.47 2.55
2    3
2.3, 3.1
annotate '@f(2)&@f(0) 100@z(37) (percent)'
plot
* The text string above uses a GIMMS font and an ISO symbol.
key 1.55u 1.85u
@9 Roses are @c4red@cl,@b   and violets are @c2blue@cl @8
endkey
* The text string above includes an arrowhead, a coloured word, a new line,
* spaces, a coloured word and an arrowhead.
key 0.2u 3.1u
@iRosacea v Violacea @vhybrid
endkey
* The text string above uses italic and normal mode.
```

CHAPTER 6
MISCELLANEOUS

6.1. The order of EASYGRAPH instructions

Instructions to EASYGRAPH may be given in any order, with one important exception: the PLOT instruction is the only one which actually causes graphical output to be written into your plotfile - that is, as soon as you call PLOT you start drawing, and once something is drawn it cannot be changed. With this one exception the EASYGRAPH instructions can be called as often as you like, and if you repeat one the most recent parameters will override the previous ones - that is, if you change your mind you simply repeat the instruction the way you want it.

Nothing starts to happen until you call PLOT.

6.2. How much data can I have

Easygraph provides very generous limits on the number of data points you can supply to it, but limits there are nevertheless.

The maximum number of (X, Y) pairs you can read in with the DATA instruction is 2000, i.e. 2000 X-values and 2000 Y-values. As implied above, a 2nd or subsequent DATA instruction will overwrite any data read with previous DATA instructions. Similarly FUNCTION will create a maximum of 2000 (X, Y) pairs *and will overwrite any data read with previous DATA instruction.* Data read with a subsequent DATA instruciton will also overwrite any values generated by FUNCTION.

READ also allows a maximum of 2000 values per named data set, but this is only achievable when reading into one or two data sets. In general, when READing N data sets you will be allowed 4000//N values per data set ('//' is the integer division operator). For example, when reading three data sets each may have 4000//3=1333 values; when reading four 4000//4=1000 values, and so on. This limit applies to each instance of the READ instruction, so using READ twice will allow a maximum of 8000 values. Using READ's SKIP and READ qualifiers you can even read from the same file twice!

N.B. A note for the curious: you may be wondering how many times you can use READ before it runs out of space and the answer is, of course, 'it depends.' DATA and FUNCTION use a data area of fixed size, internal to EASYGRAPH. The data sets generated by READ are stored in a temporary file, and use up 8 bytes for each value stored, so each call on READ can use up to 32000 bytes. You can keep on using READ until this temporary data set area reaches your maximum individual file size. Use the EMAS command FILES ,P to find out what this is. Thus with a 1Mbyte maximum individual file size you have space for approximately 128000 data values to be stored in 64 data sets. (This is ridiculously huge value - see the next paragraph.)

EASYGRAPH will let you use much more data than it is sensible normally to put on a graph - for guidance on producing legible and meaningful graphs see the reference given at the end of this Note. If you do need to use many points or segments on a

graph consider the following points before embarking on the project:

- will the points be distinguishable from each other, given the thickness of the the lines drawn by a graph plotter pen, or will they just merge into a blob?

- plotting many points on line segments in the same place can wear out both the graph plotter pen and the paper. Under the worst circumstances this will affect not only your plot but those of the people queued to use the plotter after you - the pen quality may deteriorate and torn paper can jam the plotter.

6.3. Getting the picture

By default, the graphical output from EASYGRAPH is put into a file called T#PLOTFL, which will be destroyed automatically when you log off. You can use EASYGRAPH's FILE instruction to choose your own filename for output if you want to keep a permanent copy. Whatever the filename there are two programs to get it to a graphics device so that you can see the result: GPLIST sends it to a plotter, and DRAWPICTURE sends it to any graphics device, including your terminal (whatever that is).

6.3.1. GPLIST

The form of this command is:

```
Command:GPLIST filename,device,copies,forms
```

'filename' is the name of your graphics file - "T#PLOTFL" if you have not changed it. 'device' is a plotter, specified as ".GPnn" where 'nn' is its number - the common ones are:

.GP15 - A3 plotter at JCMB rm 3210 .GP25 - A3 plotter in Appleton Tower
.GP23 - large (300 cm x 82 cm) plotter at JCMB, The King's Buildings.

'copies' is the number of copies you want (default 1), and 'forms' is only used for special output such as vertical/horizontal transparencies (see section on FORMS, below, for the complete list). Full details are published in User Note 17, or on-line in HELP PLOTTERS.

6.3.2. DRAWPICTURE

The form of this command is:

```
Command:DRAWPICTURE filename,device,copies,forms
```

where 'filename' is the graphics file (e.g. "T#PLOTFL"), and 'copies' and 'forms' are as for GPLIST - so usually omitted: in any case they only take effect when 'device' is a plotter or printer.

'device' is ".OUT" for your terminal (use the EMAS Command TERMINALTYPE first to say what your terminal is!)
".GPnn" for output to plotter 'nn'.
".LPnn" for output to PRINTRONIX printer 'nn' - e.g. .LP15.
".*" for a 'rough idea' on a non-graphical terminal.

DRAWPICTURE also has an interactive mode which enables you to scrutinize and/or plot sections of your picture. BEWARE!!! - if using this form of the command - you will only get the first frame of your plot when you say DRAW - it may look as though some of your output is missing!

6.3.3. FORMS QUEUES on .GP15

You can specify any of the following numbers as the fourth parameter to the GPLIST and DRAWPICTURE commands to get the indicated form of output. The immediate effect is to hold your output in a special queue where it will remain until the 'device' you specify is next producing the required form of output. This means that if you specify an invalid forms queue or a device which does not handle a particular queue, your output could be held up for ever! Even when you do specify a valid queue and device (the latter being .GP15 for all the following), it may still be several days before your output turns up because of the high demand for forms work - so please do not request special output until you know it will turn out right, and please be patient!

1: black liquid ink, 0.3mm pen, on quality paper.
2: black liquid ink, 0.3mm pen, on ordinary paper.
3: as 1, but 0.5mm pen.
4: as 2, but 0.5mm pen.
21: as usual but with all new pens: note though that pens wear rapidly on .GP15 and lines drawn by new pens are much finer than by older pens!
54: vertical transparency, 8 colours, 0.3mm pen.
55: horizontal transparency, 8 colours, 0.3mm pens.
56: as 54, with 0.6mm pens.
57: as 57, with 0.6mm pens.

6.4. EMAS commands

You can call an EMAS command from within EASYGRAPH by prefixing it with an exclamation mark ('!'). For example to edit a file called DATAINF, type:

```
Graph command :!EDIT DATAINF
```

NB! - After an EMAS command is detected, *all* the remainder of the line is taken to be the argument to the command, and is passed as such after the removal of all spaces. Note also that *no* brackets should enclose the parameter, if any, to the command.

6.5. Aborting an instruction or a run

An unwanted instruction may be aborted by simply typing a slash (/) in place of one its arguments either on the same line as the instruction request, or to one of its prompts. You will then get a message from EASYGRAPH informing you that the instruction has been aborted. For example, the line

```
Graph command :XSCALE 1 /
```

will be treated the same as the lines

```
Graph command :XSCALE
Graph command :1
Graph command :/
```

and produce the response:

```
** XSCALE has been aborted **
```

If you want to abort a whole run of EASYGRAPH (an interactive one), simply use the EASYGRAPH instruction ABORT - though unless you have asked for output to go straight to a plotter either STOP or QUIT would be adequate.

6.6. Progress of a run

It can sometimes be useful to know exactly how far processing has gone, especially when the instruction USE has been invoked. To this end, a user interrupt has been implemented. Press the **<CRTL+P>** and then **B** keys and, to the `Int:` prompt, reply **WH** (for WHere). EASYGRAPH should then tell you which file it is reading its instructions from, and what the last-executed instruction was.

6.7. Calling EASYGRAPH from a program

6.7.1. On EMAS 2900 (hosts EMAS or BUSH)

The parameterised call, in which EASYGRAPH is called with a string of instructions beginning with '&', is especially useful for calling EASYGRAPH from within an IMP program. The program must contain the declaration:

```
%externalroutinespec EASYGRAPH ( %string(255) PARAMETER )
```

and then build up a string starting with "&" and containing a sequence of EASYGRAPH instructions, or - better - write these instructions into a file and call EASYGRAPH with this filename.

FORTRAN programmers can use EASYGRAPH via the statement:

CALL EZGRAF (PARAM, LENGTH) or **CALL EZGRAF ('Quoted string', LENGTH)** Here `PARAM` contains a *continuous* set of characters, possibly read in using a **CHARACTER** variable, **CHARACTER*1** array, **LOGICAL*1** variable (using A1 formats), or **INTEGER** variable (using A4 formats). In any case, `LENGTH` must be the *exact* length of the parameter string.

For FORTRAN77 users *only*, there is another way of calling EASYGRAPH. **CALL EZGR77 (PARAM)** or **CALL EZGR77('Quoted string')** This routine uses information passed to describe the parameter and has only been shown to work for **CHARACTER*n** variables and quoted strings from FORTRAN77. FORTE users - *Do NOT try to use!!*

6.7.2. On EMAS-3 (host EMAS-A)

The process for calling EASYGRAPH is much simpler on EMAS-3 than on EMAS 2900, for both IMP and FORTRAN programmers.

IMP: include the statements:

```
%externalroutinespec EMAS3(%stringname COMMAND,PARAM,%integername FLAG)
:
EMAS3("EASYGRAPH","filename",FLAG)
```

FORTRAN: simply call EASYGRAPH, with:

```
CALL EMAS3('EASYGRAPH','filename',I)
```

Reference

EASYGRAPH has grown up entirely spontaneously, largely as a result of comments from its users. I have recently seen mention of a book which discusses aspects of drawing graphs in some detail. I have not had a chance to look at this book but would be most interested to hear from anyone who has. The book is:

"The elements of graphing data", by William S. Cleveland -
published by Wadsworth Publishing Co. Inc., Belmont, California, 1985.

The book concentrates on how to use graphs to communicate information - what to put in and what to leave out - as well as the sort of graph to construct, and points out that the rules and syntax which promote the construction of clear writing have their graphical equivalents.

Acknowledgements

EASYGRAPH was written by Willie Watson, of the University's Molecular Biology department (now at SDL), and Nick Stroud of Physics. It relies completely on the ERCC Graphpack, designed and implemented by Malcolm Brown of the ERCC. Malcolm and Helen Talbot have made many valuable suggestions to improve the program.

The lettering fonts available via EASYGRAPH's FONT command were originally mounted on EMAS for the GIMMS mapping package, by Tom Waugh of the University's Geography Department. I am most grateful to him for doing this, and for providing the font-tables (pages 62 to 65), and to Malcolm Brown for incorporating the fonts into the ERCC Graphpack. I am also very grateful to Gordon Hughes, now at Lattice Logic, then of the University's Computer Science department, both for general graphical help and specifically for the alternative fonts-tables on pages 60 and 61: These are copied from the Edwin 5 User Manual with all due acknowledgement to Gordon and to the Computer Science Department.

I am most grateful to Neil Hamilton-Smith of the ERCC, whose meticulous checking of both program and User Note greatly improved both. Particular thanks also go to Andrew McKendrick of the ERCC, Professor David Wallace of the Physics Department of the University, and Meg on the home front whose support and encouragement made this User Note possible.

Finally it is a pleasure to acknowledge all the many EASYGRAPH users who have suggested improvements to the program and put up with its deficiencies.

Request

We are always looking for ways in which we can improve EASYGRAPH - the program or the User Note or the on-line documentation. If you can suggest any improvements please let us know, preferably by electronic mail to ADVICE, or by writing to:

Nick Stroud,
Physics, JCMB,
The King's Buildings,
Mayfield Road,
Edinburgh EH9 3JZ.

APPENDICES

I. Complete examples

There are several complete example-files ready for you to try - if you ANALYSE the file CONLIB.EXAMPLES on BUSH or EMAS, or ERCLIB:EXAMPLES on EMAS-A, you will find a number of members called EASYGRAPHn, 'n' being 1,2,3... These each demonstrate a different aspect of EASYGRAPH, and each begin with a comment explaining what this is. Thus you may find something of direct interest by listing the first few lines of each of these files. To try one, run EASYGRAPH, then USE the filename - for example **USE CONLIB.EXAMPLES_EASYGRAPH1**, or **USE ERCLIB:EXAMPLES_EASYGRAPH7**. The result is put into the file T#PLOTFL which you can display on a graphics terminal (EASYGRAPH instruction VIEW) or send to a plotter or printer (with EMAS commands GPLIST and/or DRAWPICTURE). And if you LIST the input file to a printer you will see how it was done!

The following sections each contain a simple example of EASYGRAPH input:

I.I. Example 1

Plot data on graph with titles on the x- and y-axes

```
XAXIS "Wavelength  (nm)"
YAXIS "Absorbance"
DATA
250,0.11
255,0.092
260,0.201
265,0.476
270,0.492
275,0.317
280,0.115
285,0.09
PLOT
END
```

I.II. Example 2

Plot several sets of data on the same graph:

```
TITLE "Normalised time courses at A@t292"
XAXIS "Time  (seconds)"
YAXIS 15 "A@T292@N  (Normalised)"
FILE COURSES
PLOTTER .GP23
XRANGE 0,1000
DATA RUN1
PLOT
OVERLAY
DATA RUN2
PLOT
OVERLAY
DATA RUN3
PLOT
END
```

I.III. Example 3

Plot several sets of data using datasets:

```
TITLE " An example of EASYGRAPH "
READ X
1, 2, 3, 4, 5, 6
READ Y1
30.4 ,82.4 ,464.5 ,1145.2 ,1504.4 ,2588.5
READ Y2
10.2 ,97.1 ,282.5 ,831.4 ,981.3 ,2012.2
READ Y3 FILEY3              **** Read from file FILEY3
READ Y4
18.4 ,81.6 ,220.6 ,332.4 ,488.3 ,703.8
LINETYPE LINE+POINTS
XAXIS 12,"Time"
YAXIS 20,"Leaf Area/Stem"
PLOT X,Y1
COLOUR_GRAPH BLUE
OVERLAY
PLOT X,Y2
COLOUR_GRAPH GREEN
SYMBOL DIAMOND
OVERLAY
PLOT X,Y3
COLOUR_GRAPH RED
OVERLAY
COMMENT                     **** Re-define dataset X for the last plot
DELETE X
READ X
0.5, 1, 2, 2.5, 4, 5.5
LINETYPE LINE 0.2,0.05      **** Use a dashed line without symbols
PLOT X,Y4
STOP
```

II. Text-fonts for use in EASYGRAPH

These are the default characters used by EASYGRAPH - columns 0 to 7. '@G' in text switches from columns 4 to 7 to 8 to 11, and Greek characters. If you want a particular Greek character, look it up in the table and count 4 columns back to find the equivalent Roman one: then use this in your text. For example, get Σ with "@GR@R", βλ with "@bk@r", Ππ with "@GPp@R", and γω with "@Gc@Tx".

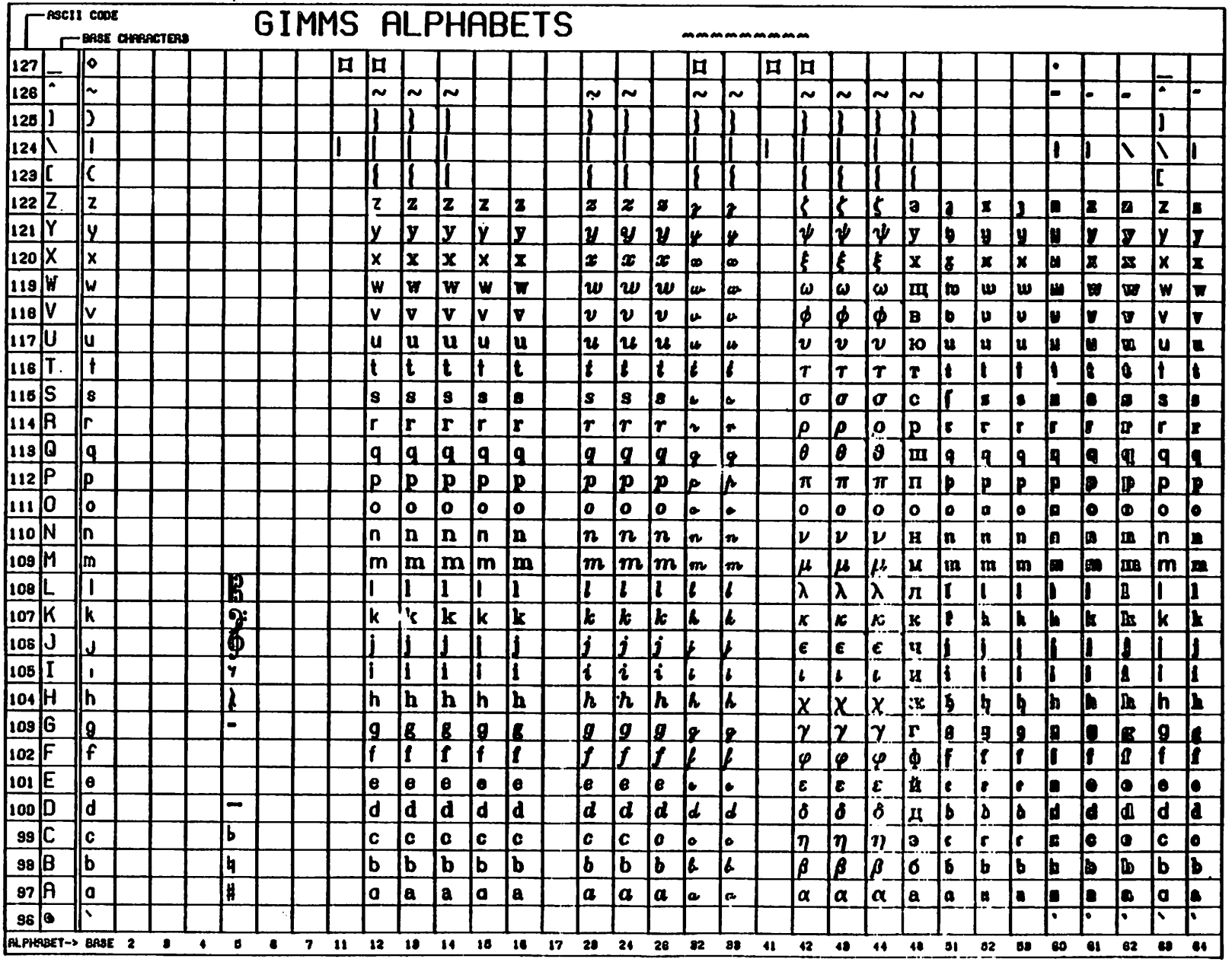
ISO SYMBOL SET AND SYMBOL VALUES FOR
IMP AND EDINBURGH FORTRAN GRAPH PLOTTING PROGRAMS

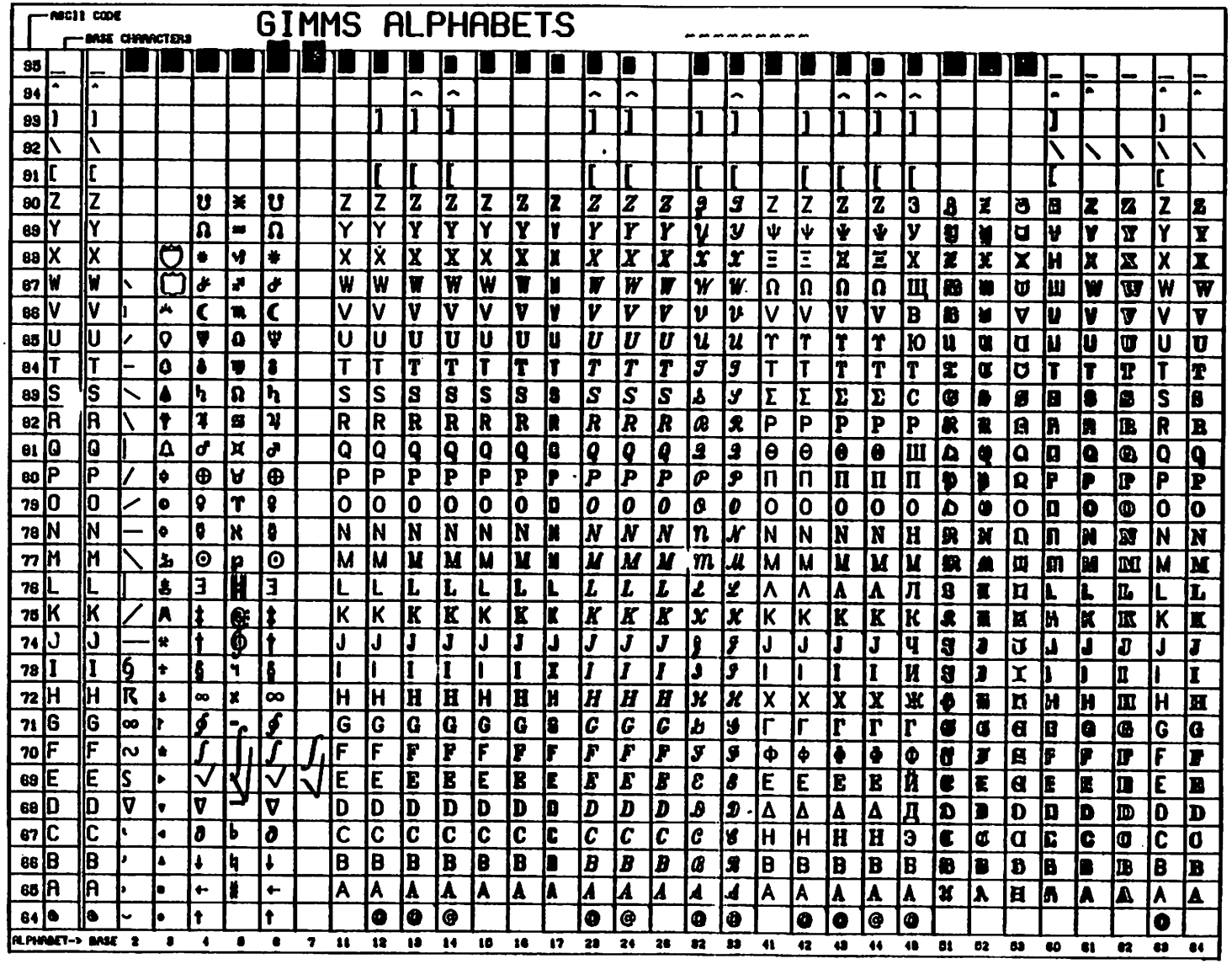
				b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1	COL / ROW	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	←	SP	0	@	P	`	p		Π		π			ION	▫
0	0	0	1	1	~	∧	!	1	A	Q	a	q	A	P	α	ρ			IOF	○
0	0	1	0	2	■	∨	"	2	B	R	b	r	B	Σ	β	σ	ς		↑	△
0	0	1	1	3	≠	⊃	#	3	C	S	c	s	Γ	Τ	γ	τ			¬	+
0	1	0	0	4	±	¢	£	4	D	T	d	t	Δ	Υ	δ	υ				×
0	1	0	1	5	+	×	%	5	E	U	e	u	Ε	Φ	ε	φ				◆
0	1	1	0	6	<	↓	&	6	F	V	f	v	Z	Χ	ζ	χ				♠
0	1	1	1	7	>	∣	'	7	G	W	g	w	H	Ψ	η	ψ				♥
1	0	0	0	8	BS	_	(	8	H	X	h	x	Θ	Ω	θ	ω				z
1	0	0	1	9	→	*	)	9	I	Y	i	y	Ι		ι					Y
1	0	1	0	10	NL	↕	*	:	J	Z	j	z	K	·	κ					♯
1	0	1	1	11	↑	‡	+	;	K	[	k	{	Λ		λ					*
1	1	0	0	12	FF	¯	,	<	L	\	l	|	M	,	μ					♯
1	1	0	1	13	CR	—	-	=	M	]	m	}	N		ν					∣
1	1	1	0	14	SUP	∫	.	>	N	^	n	~	Ξ	.	ξ					
1	1	1	1	15	SUB	∞	/	?	O	_	o	$	O	,	ο					

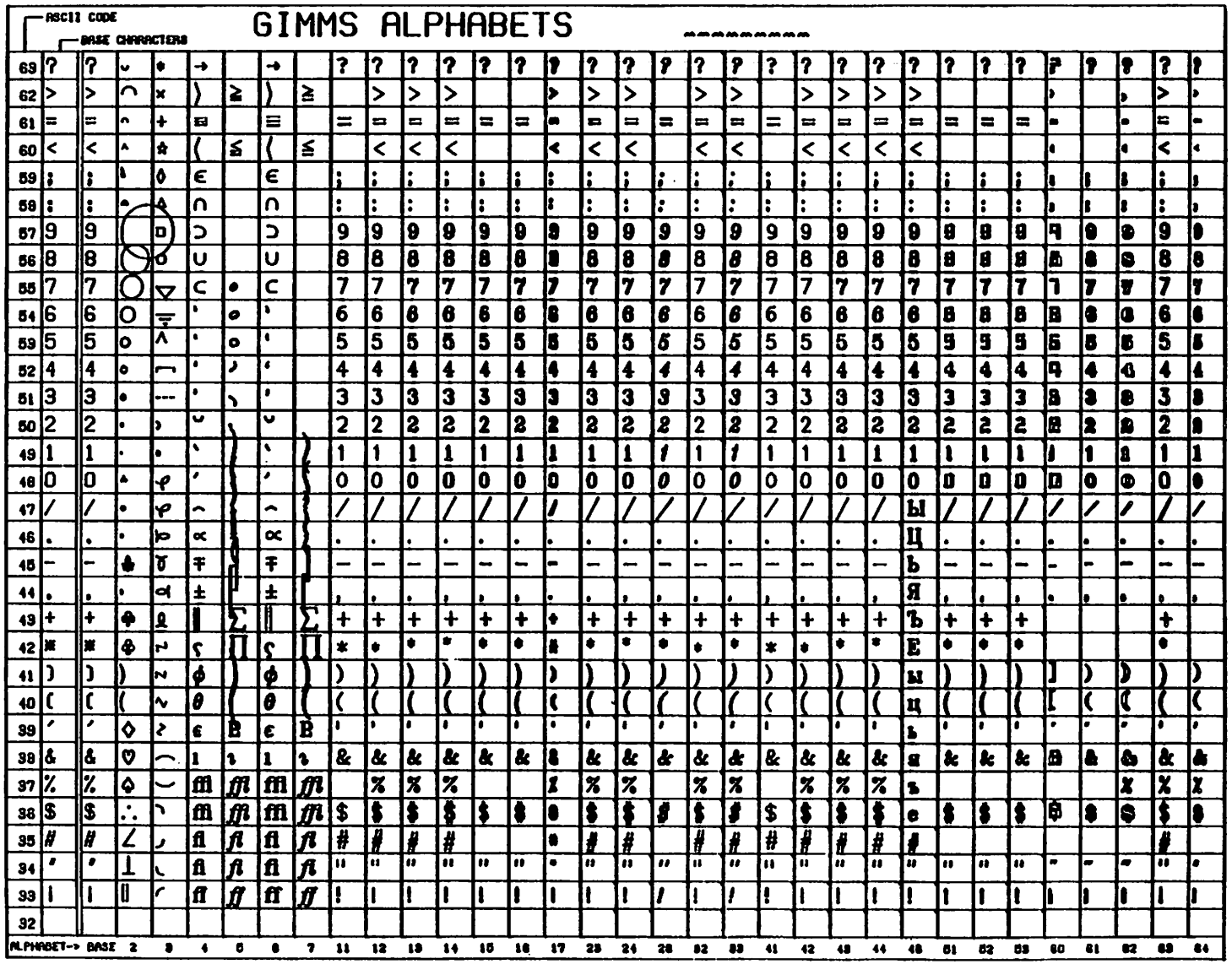
ROMAN GREEK

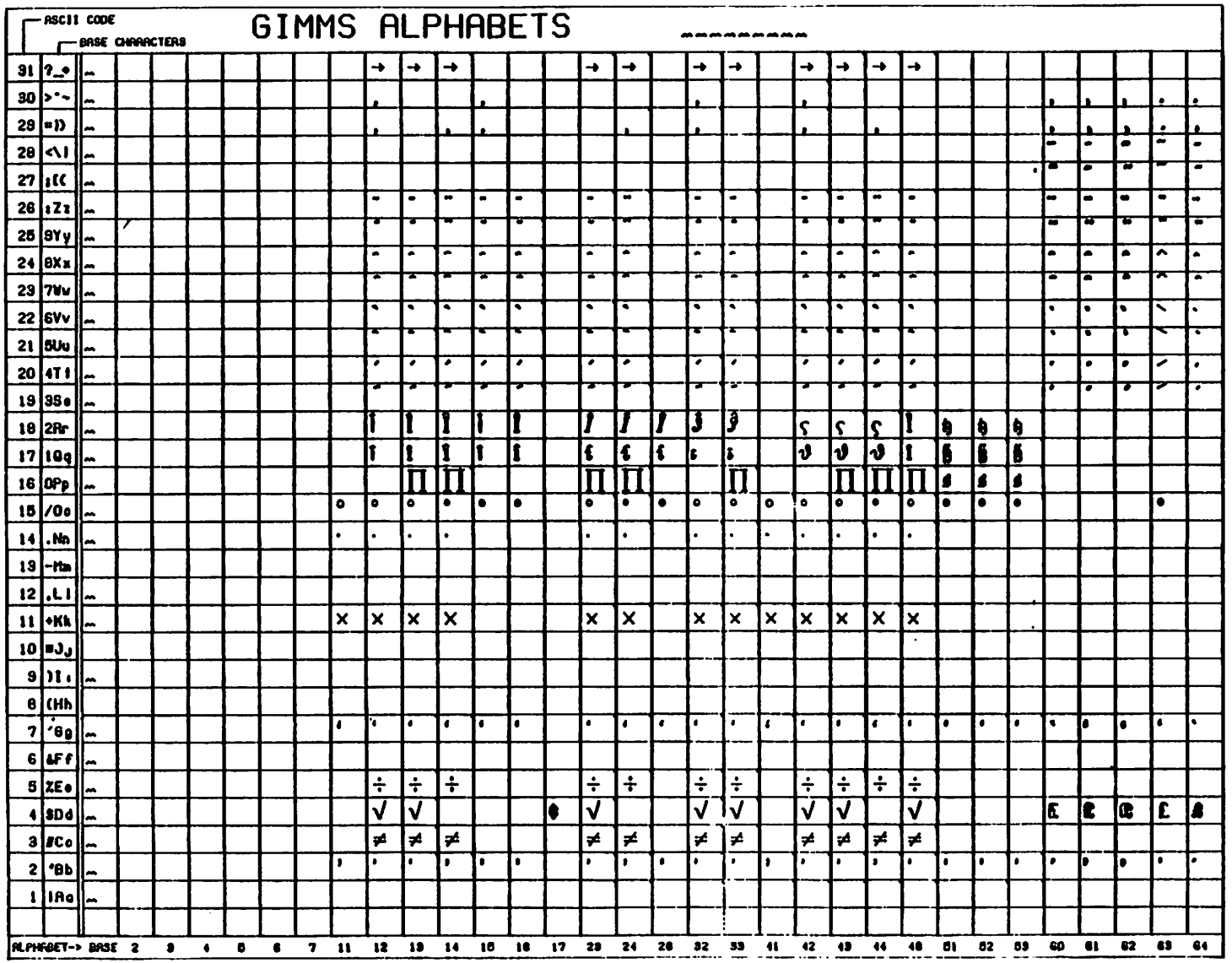

Set 64 :
Set 63 :
Set 62 :
Set 61 :
Set 60 :
Set 59 :
Set 58 :
Set 57 :
Set 56 :
Set 55 :
Set 54 :
Set 53 :
Set 52 :
Set 51 :
Set 50 :
Set 49 :
Set 48 :
Set 47 :
Set 46 :
Set 45 :
Set 44 :
Set 43 :
Set 42 :
Set 41 :
Set 40 :
Set 39 :
Set 38 :
Set 37 :
Set 36 :
Set 35 :
Set 34 :
Set 33 :

Set 32 : '≠√÷'x·°ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@*ABCDEFGHIJKLMNOPQRSTUVWXYZ*[]*abcdefghijklmnopqrstuvwxyz*{|}~¤
Set 31 :
Set 30 :
Set 29 :
Set 28 :
Set 27 :
Set 26 : "°ıȷ !"$&'()*+,-./*0123456789:;=?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz*
Set 25 :
Set 24 : '≠÷'x·°∏ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@*ABCDEFGHIJKLMNOPQRSTUVWXYZ*[]~*abcdefghijklmnopqrstuvwxyz*{|}~
Set 23 : '≠√÷'x·°∏ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@*ABCDEFGHIJKLMNOPQRSTUVWXYZ*[]~*abcdefghijklmnopqrstuvwxyz*{|}~
Set 22 :
Set 21 :
Set 20 :
Set 19 :
Set 18 :
Set 17 : ◆ !"#%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ
Set 16 : "°ıȷ̈ !"$&'()*+,-./0123456789:;=?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Set 15 : "°ıȷ̈ !"$&'()*+,-./0123456789:;=?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Set 14 : '≠÷'x·°∏ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]~abcdefghijklmnopqrstuvwxyz{|}~
Set 13 : '≠√÷'x·°∏ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]~abcdefghijklmnopqrstuvwxyz{|}~
Set 12 : '≠√÷'x·°ıȷ̈ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]abcdefghijklmnopqrstuvwxyz{|}~¤
Set 11 : "x·° !"#$&'()*+,-./0123456789:;=?ABCDEFGHIJKLMNOPQRSTUVWXYZ|¤
Set 10 :
Set 9 :
Set 8 :
Set 7 : ﬀﬁﬂﬃﬄıȷ()∏Σ[]{}∫≤≥√∫
Set 6 : ﬀﬁﬂﬃﬄıȷεθϕς‖±∓∝~^''''∪⊃∩∈(≡)→↑↓∂∇√∫∮∞§†‡∃⊙♀♂⊕ħΨ∉∗Ω℧
Set 5 : ﬀﬁﬂﬃﬄıȷ()∏Σ[]{}∫≤≥√∫∮∞§†‡∃⊙♀⊕ħΨ
Set 4 : ﬀﬁﬂﬃﬄıȷεθϕς‖±∓∝~^''''∪⊃∩∈(≡)→↑↓∂∇√∫∮∞§†‡∃⊙♀⊕♂ħΨ∉∗Ω℧
Set 3 :
Set 2 : ‖⊥∠∴◇▽◇◇○○○○
Set 1 :


GIMMS ALPHABETS
ASCII CODE
BASE CHARACTERS
ALPHABET-> BASE 2 3 4 5 6 7 11 12 13 14 15 16 17 20 24 26 32 33 41 42 43 44 48 51 52 58 60 61 62 63 64


ASCII CODE
BASE CHARACTERS
GIMMS ALPHABETS
ALPHABET-> BASE 2 3 4 5 6 7 11 12 13 14 15 16 17 20 24 26 32 33 41 42 43 44 48 51 52 53 60 61 62 69 84


GIMMS ALPHABETS
ASCII CODE
BASE CHARACTERS
ALPHABET-> BASE 2 3 4 5 6 7 11 12 13 14 15 16 17 23 24 26 32 33 41 42 43 44 48 51 52 53 60 61 62 63 64


ASCII CODE
BASE CHARACTERS
GIMMS ALPHABETS
ALPHABET-> BASE 2 3 4 5 6 7 11 12 13 14 15 16 17 20 24 28 32 33 41 42 43 44 48 61 62 69 60 61 62 63 64

This is EASYGRAPH output! There are 124 overlaid 'graphs' here. If you can supply the data, EASYGRAPH will plot it!!