```
.pl 60
.mb 3
.op
.po 0
.. fare ^OR 72
```

# The PABLO Handbook

Release 1
December 1988

This document has been compiled by L.Chiappetti (IFCTR)

The software herein described has been produced by P.Giommi, with contributions by L.Chiappetti and L.Stella, as described in section 1.2 of the present document.

.pl 60
.mb 3
.po 0
.he$PAGThe PABLO Handbook Dec 88
Page 0-#
.foThe PABLO Handbook
Contents

**Table of contents**

.cp50
.pn 1
.he$PAGThe PABLO Handbook Dec 88
Page 1-#
.foThe PABLO Handbook
Section 1

# 1. Introduction

## 1.1 Foreword

If you are already familiar with PABLO, you have certainly often complained about the absolute lack of documentation about it : the time to fill such gap has come ! If you are not familiar with PABLO, this manual will tell you everything about it (well, at least everything I could bury out of some 2000 Fortran lines of code). Previous PABLO users are invited to look in the manual and find tricks they did not know about. Beginners are urged to read the manual, instead of coming and asking the undersigned or some other scapegoat.

The manual will start telling you what PABLO is and how it (he?) was born. It will then proceed to show you how to use it in the most basic and in the most complicated ways. A section for programmers who ever want to put their hands into PABLO or mantain it is also included.

PABLO related programs, of various origin, are not described here, but in separate documents.

## 1.2 PABLO: a versatile plotting program

Actually, I'd better titled this section "a versatile and user-friendly plotting program", but I guess somebody would have argued about it. Bah ! It's just matter of a bit of time and exercise !

PABLO is an evolution of an ancestor called MOSTRO, which lived on a Data General M600 at CFA under the auspices of Paolo Giommi and Tommaso Maccacaro. PABLO in (almost) his present form was born in a cold and quiet German January (back in 198..3 ?) at ESOC, Darmstadt. The main author of PABLO is Paolo Giommi, while contributions were provided by Lucio Chiappetti (like the segmentation scheme for the HP RTE operating system) and by Luigi Stella (like the wonderful PARabolic INterpolation routine), all of them at the time at the Exosat Observatory. Since then (there was not even Fortran 77 ... so do not blame us for some programming archaisms) PABLO had several improvements, and I am being told a MicroVAX version is running in ESTEC.

PABLO was layered over a device-independent Calcomp-style graphics library, to which all low-level graphics tasks are deferred. For all details see the Programmers' Notes below (Section 7).

This document makes reference to PABLO in the version currently running on the HP at IFCTR in Milano, which has been developed in a partially independent way by the undersigned.

Well, I guess after all this bit of history, you are eager to know ...

What $#%@* does PABLO do ?

then ....please turn to the next page

.cp5

PABLO is a general purpose plotting program, useful to produce scientific plots (not business graphics, actually its ancestor MOSTRO was born to make quasar statistics ... which is why sometimes a data point is still called a "source" by PABLO), therefore, as everybody may understand after the Discours sur la Methode, cartesian plots.

You will find no pie charts here, nor you will be able to make mechanical drawings or other CAD stuff. Just

the following type of plots:

cartesian plots of one variable versus another, with a choice of log and lin scales, error bars and diamonds, scattergrams with a variety of symbols, solid or dashed curves, etc. etc. In PABLO these are called data plots.

frequency distributions of one variable, with a selection of options, called histograms in PABLO

cartesian plots of a choice of analytic functions (these are generated within the program without need of an external data file), just called functions

you will also be able to annotate your plots with text

to perform linear regression analysis on data plots

and also to plot celestial coordinates with the Aitoff projection

there was in the past the idea of inserting in PABLO an option to draw contour maps, but this is not available (use programs like **CONTR** or **MCONT** for that)

With PABLO you can draw more plots on the same frame (like data and the best fit through it), and arrange as many frames as you like in one page. According to the installation, you will be able to access a variety of devices, from b/w terminal screens to colour TVs to hardcopy devices and pen plotters.

You will be able to keep your data in plain ASCII files with a free format arrangement. You will also be able to keep a trace of a PABLO session in a command file, which will allow you to redo the plot correcting any errors until you are satisfied, and repeat it later whenever you need it. Separate tools are available to generate and handle complex command files for multi-frame plots (the Plot Editor), to draw nice annotations on an HP7550 plotter (the Labeller). There is also a library by which you may call PABLO from within your program, and there are a number of utilities to operate on PABLO data files.

.pa
.pn 1
.he$PAGThe PABLO Handbook Dec 88

## 2. Usages of PABLO

You may use PABLO in a variety of ways, which are listed here from the simplest to the most complex (which may also be the most efficient for an experienced user).

### 2.1 For a single manual plot

If you want to produce a single data plot (with or without linear regression), or a single histogram plot, or a function plot, you can just do the following:

a Edit the main parameter file, which is essentially a menu of the plot options, and fill in the relevant sections. The parameter file is described in detail in section 3 below. The default parameter file is called **PLT.PF**. The file is essentially free format : although most versions contain extensive comments, these are inessential and may be dropped or overwritten by user comments.

Note that it is preferred to have a private copy of PLT.PF on a private user cartridge. This will ensure you may run your copy of PABLO at all times. If instead you are using the default PLT.PF on a system/group cartridge, sometimes you may get an error 508 (lock error) when somebody else is accessing the file. It is therefore recommended to make a private copy of PLT.PF.

At IFCTR the default master parameter file is PLT.PF:LC:TB

b Once you have filled the parameter file, you run PABLO (just type the program name) and answer manually to a few simple questions:

b1 The plotter logical unit, to select the device for your plot. The logical unit assignments are site dependent. A listing of available devices for IFCTR is presented in Appendix B.

b2 You may then select the type of plot: enter **1** for a data plot, **2** for an histogram, **3** for a function and **4** for text (see 2.4 below for text plotting). Do not use option 5, it is not operative.

b3 If everything is set up correctly, you should get some messages, and your plot. In some cases you'll

have to answer a few self-explanatory questions (see section 4).

b3b For data plots only you are prompted for linear regression analysis. Reply **YE**s or **NO**.

b4 You are then asked what you want to do next. Type **0** to stop.

.cp5

## 2.2 For multiple plots in manual mode

You can also produce multiple plots (more data in the same frame, or more frames in a page; even mix data plots, histograms, functions and text), still running PABLO manually.

a You fill in your parameter file as above, providing full information for the first plot. Note that generally some sections of the parameter file are not used (e.g. the function section is not used for data plots etc.). If you plan to do later a plot of a type different than the first, you may fill the relevant sections of the parameter file from the very beginning.

b You than run PABLO, and do your first plot exactly as above, until point b4 (excluded).

b4 When you are asked what to do next, you are presented with the usual menu options (there are a few variants, depending on what you have done before: e.g. if you have plotted data, you're asked "do you want to plot other data"), and may select again 1 to 4 (or 0 to stop). If you type in an illegal value (out of range; also note sometimes you are not allowed to use all options) you are prompted again.

b5 You are then asked if you want to change something in the parameter file. Most of the times you may want to do so (reply **YE**s). However it is possible you want to reply **NO**, e.g. to plot another column of data (see section 5 for data file structure) within the same frame and with all the same characteristics (if you want to change a single thing, like the curve colours, you have to reply YEs).

c If you requested to modify the parameter file (note you will be modifying a temporary internal copy used by PABLO, and not the PLT.PF on disk, which remains intact), you enter

a loop where you have to type:

c1 the line number in PLT.PF corresponding to the parameter you want to modify (keep a listing with line numbers on hand)

c2 and then the new value of the parameter

c3 this will continue (c1-c2) until you type a line number of 0 (stop edit)

c1b if you do not have a listing of PLT.PF on hand, or if you got confused about what you have done, you may get a display of the current parameter settings typing a line number of -1, then resume editing at point c1 as above. Note comments are not shown in the listings for modified entries.

Once you have finished modifying the parameter file (c3), or skipped this phase altogether (replying NO at b5), the program continues asking things and plotting as usual for the options you selected in b4.

.cp2

You then go back to point b4 (type a **0** to terminate PABLO).

This way you may do plots as complicated as you like. If they are too complicated however, the possibility you may a mistake increases. When you are plotting on a screen you may recover some mistakes (like replotting over a wrong plot with colour 0 to erase it), but on an hardcopy plotter you have to do it all over again. It is recommended to use command files (see below) for such cases, and in all instances you may need to repeat a long sequence of commands.

## 2.3 Using a customized parameter file

As said above, the default parameter file is PLT.PF (the first one with such name available in the cartridge list). This means that for all plots you have to edit and change it. This may be annoying if later you want to do again a plot you've already done ... but your PLT.PF is no longer corresponding to your plot and you'll have to ripristinate the old one. You may wish to keep a number of "standard" variants of PLT.PF for each "family" of plots you may often repeat.

One way out is to run independently of the current PLT.PF, by plotting a "dummy text" (see below 2.7 for details) as first

option (this will make sure the internal parameter file is set up), and then editing the internal parameter file as described above (points c1-c3), filling it in full. This is quite annoying if you operate manually (although it is used by some programs, see 2.8 below).

A better way is that you prepare customized parameter files for all kinds of plots you may frequently do. This is most useful if you have to repeat quite often plots of similar format and layout with only a few differences (typically changing the data files).

You just make a copy of PLT.PF into another file pffile (you can use any name, although for documentation it is suggested to use names of the format aaa.**PF**. You can then edit your private parameter file as usual, the only thing is that you have to tell PABLO to use your file instead of PLT.PF. To do this, when running PABLO, use the syntax :

**PABLO**,,pffile

Note the use of a double comma in front of the file name. The reason for this will be explained immediately after the next section below.

## 2.4 Plotting text

The logics of plotting text is slightly different from other plots. The fact is that text has its own parameter file, called **TEX.PF**. If you like you could fill in TEX.PF with the characteristics of the first text string you want to plot, and then "edit" it for further strings. If you do that you are urged to make your private copy of TEX.PF with the same name on your private cartridge. However ...

...you are strongly recommended not to do that. You just leave the default TEX.PF unchanged, which contains instructions to plot a blank (dummy) string. Then, when running PABLO and asking for text (option 4), the dummy text will be plotted and you will be asked "do you want to add more text", then reply **YE**s and follow the instructions to edit the parameter file, and do that as often as you need more strings.

When you leave text mode, replying **NO** to the above question, the current modifications of the parameter

file are lost. Next time you enter text mode, the first string plotted will be again that in TEX.PF : that's why it is important to leave TEX.PF blank.

It is anyhow suggested to draw all text strings sequentially in one go.

As a final thing concerning text, note that PABLO allows text of any colour and size, with software generated characters. However text is allowed only horizontally, and only upper case characters are defined. Also the characters are quite rough (in particular the letter O is awful, use number 0 instead). For better results on the plotter you should use the Labeller.

## 2.5 Using a command file

The use of a customized parameter file is a good solution to make a template for a single plot , where you just change a few things like the file name. If your plot has however more frames, you still have to repeat manually all the sequence of commands. If the sequence is long it is likely to make mistakes.

.cp3

The solution to avoid mistakes (or to be able to correct them) is the use of command files. In a command file you just type all the replies you will normally give to PABLO when prompted, one on each line. Of course you should know what kind of questions PABLO is going to ask. The PABLO "operator dialogue" is described in section 4 below. With a bit of experience you will learn that by heart, however to start with the suggestion is to run PABLO manually the first time, writing down all the replies you type in (or making a screen hardcopy later, for lazy people like me). Then you can edit those replies in the command file.

Note you can type as many comments you like on the same line as the reply; PABLO will read the reply needed and ignore the rest.

By convention (by no means compulsory) the command file name starts with an "at" (**@**) sign.

To run PABLO with a command file, means PABLO will take all input from the command file instead that from the

terminal keyboard. The terminal is actually a special kind of command file (file **1**). The syntax used to run PABLO with command file cfile is :

**PABLO,**cfile

.cp4
(that's what the first comma above was holding the place for !). The default command file is the terminal. You get the same result if you type **PABLO** tout court or **PABLO,1.** (Incidentally you could use any valid logical unit number as command file also, although I cannot see much use of having PABLO taking input from a tape or similar).

Of course the command file must exist and be a valid sequential file (not a type 1 or 2 file). In case of any error (this includes the file being locked to another program, error 508), after an error message, the control is given back to the terminal.

Note that, when running with a command file, there is no echo on the terminal of the dialogue between ..PABLO and the file. Only PABLO's warning messages are issued to the terminal. In particular PABLO is able to trap some command file errors, like any illegal options (out of the range 0-4 used to specify respectively stop, data, histogram, function or text) : in such case it will terminate telling you the number of the offending line. Any other error in command files (typically a noL.Chiappetti at ESOC to emulate the WKS-Plot package in use at Werkgroep Kosmische Straling in Leiden. Only simple things are possible, like setting up an entire frame for a data plot, for a function plot and for plotting text, with a few limitations. All details are presented below in Appendix D.

Note that LNPAB does not do any plotting. It does only write the command file for PABLO (it may also write the data if they are not in PABLO format) and runs PABLO. This is generally inefficient, or less efficient than using a proper plotting package. However it is handier than using a primitive plotting package like a Calcomp-style one (which does all plotting in inches and leaves all the scaling to you), specially when a program like PABLO already contains the code for scaling once for ever.

## 2.9 Using the Plot Editor

The most recent addition, still partially under way, to

the PABLO family is the Plot Editor. This is described in detail in separate documentation. The Plot Editor is suitable for use when you are wanting to do complex plots with tens of frames, also mixing texts and contour maps, and when you may want to change their layout quite often.

For the Plot Editor a full plot consists of frames (PABLO data, histogram and function plots and MCONT contour maps) and strings. The Plot Editor keeps the description of all these items in a descriptor file, which is the only permanent file required, and allows to create and modify it.

The Plot Editor is also an interface able to generate command files for PABLO, MCONT and LABLER (the Labeller), and to run all such programs. It will automatically include the text and labelling commands either in the PABLO or in the LABLER command file, according to your requests. All command files can then be deleted, as they can be generated again when desired. Editing the layout of a plot with the Plot Editor is easier than modifying all relevant items in a PABLO or LABLER command file, moreover the Plot Editor ensures the consistency of PABLO, MCONT and LABLER command files.

## 2.10 Using the Labeller

You may ask what the Labeller is. It is a program used to annotate a plot with text, axis labels etc. on an HP 7550 plotter. The reason to do that is that it can do nicer things that PABLO does not do, like using nicer firmware fonts, lower case letters, symbols, italics, putting tics at intervals other than one inch, putting more sets of tics on one axis, labelling an axis in hours etc.

The usage and the commands of the Labeller are described in a separate documentation. Here we give only a few hints for PABLO users.

The idea is to use PABLO to draw frames and do plots, and let the Labeller do text and axis labels (tics included). To do this (if you use the Plot Editor, the latter does it for you) you should disable text in PABLO, which is simply done using a zero or negative character size in
the parameter file.

.cp4
A zero character size will cause all text to be drawn as

infinitesimal dots (in the IFCTR PABLO Version 2 text in axis labels is actually suppressed). Also tics will drawn of size zero (or respectively not drawn at all).

A negative character size will still allow text outside labels to be drawn (as it is controlled by TEX.PF), but will reduce to zero the size of text in axis labels. The tics are however drawn normally. Note that this applies to all axes in IFCTR PABLO Version 2 only (only to x-axis for original PABLO).

If you are satisfied with PABLO tics use the second alternative (negative size), otherwise (if you do the tics with LABLER) use the first one. The Plot Editor uses zero size, not only for labels but also for text. Text plotting should be put at the end of the command file (like the Plot Editor does) so that it can be deleted or disabled.
.pa
.fi pablo-1.hp

## 3. The parameter file

> We give here a line-by-line description of the content of the PABLO parameter file. Note that a nice and tidy parameter file keeps all parameter values on the left in column one, and uses the rightmost columns for comments. The comments are however optionals, and may be dropped (not recommended) or replaced by user comments (encouraged). The format of the values and of the comments is free. The order is instead compulsory : a given parameter must be at a given line number.

Here we explain the meaning of the various parameters and indicate also to which kind of plot are they relevant.

Line 1 Input file : this is the name of the data file used for data and histogram plots. Use the full HP namr (name:sc:cr) or just the name. If you specify the cartridge you are sure that PABLO picks up the right file, and also it runs faster. The format of the PABLO files is described below in section 5.

Line 2 error file flag : it can be **0** if the file is "with errors" or **1** if the file is "without errors". See section 5.

Line 3 number of header lines : a PABLO file (see section 5) can have any number (including 0) lines of free-format text header at the beginning.

Line 4 first point : this is the order number of the first point (or "source") to be plotted. All previous points are skipped. See section 5 for definition of a point (it may be a single line, if the file is without errors, or a couple of lines if a file is with errors)

Line 5 last point : self-explanatory. Note that PABLO is able to plot only 1000 points at a time (at IFCTR, the value is site dependent). Therefore the difference between first and last point cannot exceed 999. See 6.2 below for hints on plotting more than 1000 points.

Line 6 plotter logical unit : this is normally not used, as the logical unit is specified manually or in the command file. See however beginning of section 4 below.

.cp7
Line 7 colour column : this is 0 if no colour column is wished (all points in a data plot are plotted in the same colour and with the same symbol). If it is non-zero, the symbol and colour used to plot each one point may be different, and are contained in the column indicated in the data file : the symbol in the value line, and the colour in the error line.

Line 8 start values for x and y respectively. In user units. They Line 9 apply to a linear axis only (separately for x and y) for all kind of plots. As an exception (only in IFCTR PABLO Version 2) they apply also to logarithmic scales if number of decades (lines 30 or 31) is coded as negative.

Line 10 step for x and y respectively. In user units. Same as start Line 11 values (only linear scales, with exception for log scales in Version 2). It specifies the step in user units for one inch on the plot. It may be computed from the wished stop value, considering that stop = start + step * axislength.

Line 12 auto scale flag : applies to data, histogram and function plots. If it is **1** the scale is arranged as specified in the parameter file. If it is **0** the scale is not taken from start and step at lines 8-11, but is calculated automatically from the data (good for a first trial, but axis values are not "tidy", better refined manually).

Line 13 Upper limit flag : **0** means upper or lower limits in the data file are ignored. Upper and lower limits are identified by a conventional negative error of -1 and -2 respectively. Set flag to **1** to include them.

Line 14 error bar flag : ignored if file is without errors. Controls error bar rendition as follows.

0 no error bars drawn
1 only x error bar drawn
2 only y error bar drawn
3 both x and y error bars drawn
4 an error diamond is drawn

see below 6.3 for clipping of error bars

Line 15 Connect flag : it can assume the following values

0 data points are not connected

1 data points are connected by segments. If horizontal error bars are drawn, their extrema are connected instead of the central point (histogram-like plot). It is users' care to ensure points are ordered (ascending or descending; only ascending in the case of horizontal error bars)

2 "also first and last". similar to previous mode, only the first and last point or bar extrema are connected with the x-axis by a vertical segment

3 "smooth". data points are connected by a continuous line based on a parabolic interpolation. Points shall be ordered. Works nicely if point distribution is nearly continuous, may behave oddly if there are edges, discontinuities, or points at same coordinates

4 the connect modes 4 to 6 are available only in IFCTR to PABLO Version 2. These modes are used only for files 6 "without errors" to generate binned plots (pseudo-histograms) If a file has errors they default to connect mode 1 (also error bar handling is affected) which give similar renditions.

4 from one data point to the next, a vertical segment is drawn at the x-coordinate of first point, followed by an horizontal segment at the y-coordinate of second point

5 similar to 4, but horizontal segment is drawn first at height of first point, than vertical segment reaches up to next point

.cp7

6 "pseudo error bar". an horizontal segment, at the level of the first point, reaches the middle between the two points, than a vertical segment goes up to the level of the second point, finally a further horizontal segment reaches the destination

Line 16 more-than-one-frame flag : this value shall be set to

**1** whenever a new frame has to be drawn. If it is **0** no axes are drawn for the current frame, the origin and length of the axes remain unchanged (lines 17-20 ignored) and also the start and step values (lines 8-11) are ignored. In PABLO Version 2 only a value of **2** is allowed: no new axes are drawn, but the scale of the plot is updated (for late plot overtrace).

Line 17 axis length for x and y respectively. In inches or
Line 18 pseudoinches according to device. See Appendix B for (site dependent) maximum plot size on different devices

Line 19 axis origin for x and y respectively. In inches or
Line 20 pseudoinches as above.

Line 21 character size for numbers and labels. Also linked to tic size. In inches. Set to zero for no labels, no tics (use in conjunction with the Labeller). Set to negative value to suppress labels (tics are drawn using absolute value as reference size). Note negative values applies to all axes in Version 2 only (only to x-axis in original PABLO). Characters have a fixed x-y aspect ratio

Line 22 symbol size in inches. used when symbols are drawn for isolated (not connected) points. Set to zero if symbols are not desired.

Line 23 symbol number in the range 0-89 (normally use 0-16). specifies the kind of symbol to be drawn for isolated points. See figure in appendix A for available symbols.

Line 24 frame colour : device dependent colour number for axis frame. see Appendix B for colour numbers and ranges available for various (site-dependent) devices. A non-zero colour will generally cause axes to be drawn. A zero colour is often the background colour (usually black) used to "undraw" axes.

.cp4
Line 25 symbol colour : used for isolated data points and curves in data plots (unless a colour column is specified at line 7). Not used for upper and lower limits. See above for more info.

Line 26 upper limit colour : used for upper and lower limits in data plots, if enabled at line 13. Also in histograms (see 6.6).

Line 27 log scale flag : applies to data and function plots. May assume one of the following values :

| | | |
|---|---|---|
| 0 | log-log | both axes are logarithmic |
| 1 | log-lin | x axis is logarithmic, y is linear |
| 2 | lin-log | x axis is linear, y is logarithmic |
| 3 | lin-lin | both axes are linear |
| 4 | special case for Aitoff's projection (see 6.4) | |

Line 28 start decade for x and y respectively. It is the logarithm of
LIne 29 the start value in user units (used, instead of the values in lines 8-11, only if the relevant axis is log), hence code -2 to start at $10^{-2}$, 0 to start at 1, 2 to start at 100, etc. Only integer values (start at powers of 10) allowed in PABLO basic version. Real values are allowed in PABLO Version 2 at IFCTR.

Line 30 number of decades for x and y respectively. In PABLO basic
Line 31 version only an integral number of decades may be drawn. This is actually expanded by a log-2 extension (e.g. if you have asked two decades starting at -1, your log scale will go from 0.1 to 20) to provide a nice-looking margin.

In IFCTR PABLO Version 2 it is allowed to specify the number of decades as a real number (difference of the log values of start and end), allowing for fractionary decades. Note that, for compatibility with previous version, a log-2 extension is still applied (if you wanted an axis from 0.1 to 4, you would have coded a start exponent of -1 and 1.602 decades, but you will get a scale from 0.1 to 8)

As a further facility, in Version 2 only, if you code a negative value as number of decades (any value), the start decade and number of decades in lines 28-31 are ignored, and the values are computed from start and end values implied by lines 8-11 (conventionally assuming end = start + step * axislength) This way you get exactly the scale you want (the log-2 factor is eliminated internally).

Line 32 percentage flag : for histograms. Determines the y-scale for histogram plots. **0** gives a scale in percentage frequency, while **1** gives the number of occurrences.

.cp4
Line 33 upper limit flag : for histograms. Determines if upper limits are considered in the distribution. **0** to analyse only normal points (called "detections"), **2** to analyse upper limits only, **1** to analyse both. See 6.6 for details, also in conjunction with reverse mode flag.

.cp4
Line 34 dashed line interval or so called trat length (historical anglo-milanese name), used to control the style of dashed lines (provided by software)

Line 35 histogram start : in user units, determines the start of the x-axis for histograms (lines 8 10 not used, line 17 used)

Line 36 number of bins on x-axis for histograms. It determines the end value for x-axis in conjunction with start and bin width (lines 35,38) and axis length (line 17) and also the range of values to be analysed

Line 37 auto bin width flag : a value of **1** means the histogram scale is set up using parameter file lines 35-38, a value of **0** it is automatically set up from the data (good for first trials)

Line 38 bin width : in user units. The width of a bin in the histogram, if not disabled by flag at line 37. Used in conjunction with lines 35-36 to set up x-scale

Line 39 histogram label : max 20 characters, the label for the histogram y-axis. The x-axis is one of the data labels.

Line 40 label for column 1 to column 9 : max 20 characters, the labels to associated to each column of the data file, used for x and y
Line 48 axes for data plots, or x-axis for histograms. If a column is not present, or you want not to see it displayed on the screen, just code a label of **NN**.

Line 49 reverse mode flag : **0** for normal plots (solid lines) or **1** for dashed lines (used in conjunction with line 34). See 6.2 and 6.6 for usage in data and histogram plots respectively.

Line 50 function colour : the colour (see above for info) used to draw curves in function plots

Line 51 axis label for x and y axis respectively, used for function Line 52 plots only

Line 53 function pf flag : if **1** the function is described by the identifier and parameters in lines 54-58 and 62 of the parameter file. If **0** they are asked interactively.

Line 54 function parameters par0 to par4. Each function is to characterised by up to 5 parameters, according to the function
Line 58 identifier (see below line 62)

.cp3
Line 59 function extrema respectively maximum and minimum x-values,
Line 60 delimiting the range where the function has to be computed. Note that the x-axis extrema are set as usual. These extrema determine a different range (possibly included in the x-axis scale)

.cp4
Line 61 number of points : for function. The function is evaluated at
particularly if you are skipping a number of points.

At the end you get either a message "Data reading completed" or a warning message telling you less points than expected have been found (do not care, it is common practice to tell PABLO to go from point 1 to 1000 and let him find out how many points are in the file, machines are there for that !).

If you get a runtime error while reading, it means there is something wrong in your file (format error, non-numeric field instead of numeric, or file corrupted by non-printable characters, or, if you get an EOF error, you are most likely telling PABLO the file has errors while it has not, or you have given the wrong number of header lines).

You may also get a message telling you negative or zero points have been eliminated (in case of log scales).

You then get a message "Plotting on LU ..." and a message "Plot done" : you hopefully also get a plot in between.

In PABLO Version 2 you may get a message regarding the number of points clipped (see 6.3 below for clipping).

You are then prompted for linear regression analysis : a common error in command file is to forget that you always have to reply to this question, either YE or NO.

You are finally prompted what you want to do next. As usual you have options 0 to 4 (5 is unused). Note that for option 1 you are prompted "do you want to plot other data", which may be the case of two other columns in the same data file, or also in a different file.

When running from terminal an out-of-range option will cause you to be prompted again. When running from command file you will get an error message and PABLO will terminate.

In all cases you are prompted whether you want to change something in the parameter file. If you reply YE you go to the parameter file change segment, otherwise you go directly to the segment corresponding to the option you selected.

### 4.2 If you select option 2 (plot histogram)

You enter the histogram plotting segment, and PABLO displays a message with a list of all column labels which do not contain an NN as for the data plotting segment.

You are then prompted to select which column you want to analyse (the prompt is "which variable"). This will be on the x axis. Select by number one of the columns listed above.

You do not get the message "Reading data ...", but at the end you get either a message "Data reading completed" or a warning message telling you less points than expected have been found.

You then get a message "Plotting on LU ..." and a message "Plot done" : you hopefully also get a plot in between.

You are finally prompted what you want to do next. As usual you have options 0 to 4 (5 is unused), subject to verification as said above.
Note that for option 2 you are prompted for "another histogram".

.cp4
In all cases you are prompted whether you want to change something in the parameter file, and you either go to the parameter file change segment, or directly to the segment corresponding to the option you selected.

### 4.3 If you select option 3 (plot function)

You enter the function plotting segment.

Only if the relevant flag is set in the parameter file, you are prompted for a function identifier and the relevant parameters (always 5).
If the minimum x for function is less or equal to zero, you get a warning message and no plotting is done.

Otherwise you then get a message "Plotting on LU ..." and a message "Plot done" : you hopefully also get a plot in between.

You may also get a warning message telling you there were some points out of scale.

You are finally prompted what you want to do next. As usual you have options 0 to 4 (5 is unused), subject to verification as said above. Note that for option 3 you are prompted for "another function".

If you select another function, note that this implies temporary setting of the flag such that you are prompted interactively for the function parameters.

In all cases you are prompted whether you want to change something in the parameter file, and you either go to the parameter file change segment, or directly to the segment corresponding to the option you selected.

### 4.4 If you select option 4 (text)

You enter the text plotting segment, and immediately PABLO displays a message "Writing text .." and plots the text currently in TEX.PF.

You are then prompted whether you "want to add more text".

If you said YES you are presented with a list of the content of the current memory copy of TEX.PF, and prompted for what "you would like to change".

You may reply 0 to terminate the editing, -1 to see the list again, or the number of the line you want to change, in which case you are prompted for a value. All this in a loop until you terminate the editing.

You then go back to the beginning of the segment ("writing text" etc.)

.cp4
When you have replied NO to the question about more text, you are finally prompted what you want to do next. You have only options 0 to 3 (5 is unused), subject to verification as said above. You cannot specify option 4 again !!!

In all cases you are prompted whether you want to change something in the parameter file, and you either go to the parameter file change segment, or directly to the segment corresponding to the option you selected.

## 4.5 Messages in the regression segment

You will always (in interactive mode) get a message displaying the slope and intercept of the regression analysis and the associated correlation coefficient.

For the rest the messages and prompts you get are the same as those of the final part of the data plotting segment.

.cp5
## 4.6 Editing the parameter file

If you are running from a command file you just get a message "changing parameters" and no prompts. Beware of any runtime error here, which generally means you have mistyped something in the parameter file, or skipped one or more lines.

If you are running interactively instead you are prompted for what "you would like to change". The current setting of the parameters is not displayed by default.

You may reply 0 to terminate the editing, -1 to see the list of parameters and current values, or the number of the line you want to change, in which case you are prompted for a value. All this in a loop until you terminate the editing.

When you have finished editing you jump directly to the segment selected in the last option selection.

.pa
.pn 1

## 5. The PABLO data files

The current PABLO file format offers considerable liberty, and is an extension of an original more rigid format. Essentially a PABLO file is a simple ASCII free-format tabular file.

A PABLO file may be preceded by any number of header records (lines), containing whatever information the user may think useful to identify the file and its content. It is not used by PABLO, which needs only to know how many lines to skip. It is however strongly recommended that users identify what is in a file (otherwise they may forget). It is also suggested the last line(s) in the header contain column headings for the following data.

The data in a PABLO file are arranged in columns. There shall be at least one column (if the file is to be used for histograms only) or at least two (if it is to be used for data plots). Up to nine columns are supported by PABLO (actually I believe it could even read more in line of principle, but without a correct labelling). Data are written in free format, and columns do not need to be necessarily neatly aligned (although recommended). Columns shall be separated by at least one blank (recommended) or exactly one comma (use of commas discouraged).

See the file documentation sheets in the Exosat System Reference (The IFCTR Exosat Analysis System - Programmers' Reference) for more information on file layout and previous file formats, particularly the older (still accepted) n (1X,E10.4) and the previous (may give problems if the first column is preceded by a comma) n(',',E10.4).

The IFCTR version of PABLO does not support the nE11.5 format (with columns not separated by any separator).

Within each column data are arranged in points (called historically "sources"). For each point there is always a value. This value may have an associated error (the file is said to be "with errors"). If any value has errors, all values in all columns shall have an error. Note also that it is not allowed to have "missing points" in one column.

If a value is not defined in a given column, put any dummy value there (at your choice, a zero, a value out-of-range, or a value flagged as upper limit). If an error is not defined put a zero there.

Files without errors are arranged one record per point (with as many columns as required).
.cp4
Files with errors are arranged two records per point : the first record contains the values, and the second record contains the errors.
Therefore the errors are in the same column as the values, one record below. Note that PABLO, when skipping data, thinks in terms of points, not of records (i.e. for a file with errors, it skips a couple of records at a time).

There is no limit of the number of points in a file, however only a site-dependent number of points (1000 at IFCTR) can be plotted or analysed at any one time (see 6.2 below).

A file may contain some special data. One case is the colour column and the other is that of upper limits (and lower limits).

The colour column is an additional (optional) column used to specify a different symbol and colour for each data point. The value in the colour column is the number of the symbol associated to the point (see Appendix A for symbols). The value in the error field is the colour number (see Appendix B for colours). The colour column is enabled specifying a non-zero entry at line 7 of the parameter file.

Points may be flagged as upper or lower limits by coding an error of **-1** and **-2** respectively. There are some flags in the parameter file which may be used to include or exclude such points from the plotting or the histogram analysis. One additional usage of upper limits is to flag "bad" points with a -1 error to avoid them being plotted, while retaining the data in the file.

.cp3
You may include comments (other than header) in the data file if you wish, with the following caveats.

You are not allowed to include lines of comments in the middle of a file. If you include lines of comments at the end of the file (after the data), you are compelled to specify to

PABLO the exact number of points, to prevent the comments being read (this causes a crash with a format error). This approach is therefore discouraged.

You may include comments on the same line as data (not necessarily on all lines), provided you put those comments after all data columns, i.e. in a column which is never accessed. Similarly you may have "sparse" columns (columns with some blank entries) included in your file for documentation (or usage by programs other than PABLO), provided they are located after the last column ever accessed by PABLO. One usage of sparse columns could be to preserve in a file a copy of data flagged as "bad/no plot" by use of the upper limit feature.

.pa
.pn 1
.he$PAGThe PABLO Handbook Dec 88
Page 6-#
.foThe PABLO Handbook
Section 6

# 6. General hints

## 6.1 Drawing frames

PABLO keeps a "frame" counter, which is incremented after a data, histogram or function plot has been done (but not for text). This number appears in some PABLO messages, and is used in the plotting routine to override the "more-than-one-frame" flag (that is, the first frame has always the correct axis settings).

A new frame ("more-than-one-frame flag set to 1) implies both drawing the axes and updating the axis and scale settings. Remember that frames are always drawn, in linear scale, with tics every inch (axes length may however be fractional). There is sometimes limited control on the "neatness" of the axis labels (particularly when using the autoscale feature); for best results the use of the Labeller is suggested.

Note also that the usage of negative or zero character size is different in PABLO basic version and in Version 2, namely while a zero character size suppresses labels and tics in both cases, a negative character size suppresses labels only, but on both axes in Version 2

(for use with the Labeller) and on x-axis only in basic version (to allow plots to be adjacent in y).

An useful feature present only in PABLO Version 2 is the possibility of overtracing frames with different scales. This is done setting the "more than one frame" flag to 2. This implies the axes are not redrawn, but the scale and position of the frame is recomputed. This is useful either to overtrace a frame which has already been abandoned (typically with the same scale as originally), or to overtrace the current frame with a plot with a different scale (if the data for some reasons are in different units because of a different intrinsic scaling factor).

## 6.2 Data plots

The kind of plots done by a function or histogram plots are quite simple and do not deserve further explanation (a few hints for peculiar behaviours are given below). The data plotting offers more varieties. First of all points may be drawn isolately, or be connected. Isolated points can be plotted as a symbol of user specified shape and size, or with separate x and y error bars, or as an error diamond.

Ordered points may be connected directly by a broken line, or by a smoother line. Also binned plots are possible, either joining the extrema of the x error bars (if data has errors, otherwise you may introduce in the file pseudo-errors equal to the bin width), or making use of the new "pseudo-histogram" features (in IFCTR Version 2 only).

It is possible to draw error bars and connect their centres with a smooth curve using connect mode 3.

.cp6
If there are more than 1000 (IFCTR value, site-dependent limit) points in a file, they may be plotted as follows :

first plot points 1 to 1000

then ask for a further data plot (preferably with the "flag for new frame" set to zero) and edit the parameter file, changing the "first and last point" e.g. to 1001 and 2000

and plot the same data columns again

and go on like this 1000 points at a time

.cp4
Note that a special call to generate the appropriate commands (CALL RPEAT) is available in the LNPAB library. Note also that, when using the Plot Editor, the appropriate commands are generated automatically (you just specify first and last point in the file, e.g. 1 to 2345).

There are some "secret" options in the modes of connection, also in connection with the reverse mode used for dashed lines, which are described here.

A negative connect mode (**-1** or **-2**) has the effect that only every other point is connected (i.e. only points 1 and 2, 3 and 4 etc. but not 2 and 3, 4 and 5 etc.)

A negative parabolic connect mode (**-3**) has the effect of enabling the reverse mode (it is the same as connect mode 3 and reverse mode 1), i.e. the connecting line is dashed. It appears to be working nicely.

Reverse mode is ignored for connect mode 0 (no connect) and for connect modes greater than 3 (pseudo-histograms in version2).
Reverse mode operates with connect modes 1 and 2, but the result may be screwed up. They work better when error bars extrema are joined, but still with problems.

## 6.3 Data clipping

In the basic PABLO version, all data points flagged as plottable (i.e. with the exception of upper limits or detections in accordance with the relevant flag) are always plotted (the only exception is the elimination of zero or negative points in log scales). This causes points to be plotted outside of frames, and sometimes also outside of the screen (however on some devices, e.g. Ramtek, a wrap-around may occur and points will be plotted at wrong, unexpected positions within the frame).

PABLO Version 2 (available at IFCTR) provides instead data clipping. Points which fall outside of a frame are clipped, i.e. not plotted, and a warning message is displayed. Note that, when plotting isolated points, such points are lost (you have no idea where they are, only how many the are). When connecting points, the connecting line will be instead

broken (beware of funny results when using parabolic interpolation), and you will be able to locate where points have been clipped (except if they are at the beginning or at the end).

With linear scale, when plotting error bars, it is still possible even in Version 2 to have error bars of a not clipped point going outside of a frame. With log scales, the basic PABLO version does only one kind of error bar clipping, that is, error bars on the minus side, which will lead to negative or zero values, are suppressed. PABLO Version 2 instead forces all error bars (on plus and minus side) and diamonds to remain within the frame.

Note that clipping may be used to produce "break points" in a connected plot. If you want to plot a series of connected lines, the usual approach would be to have separate data files. It is however possible, with clipping, to use a single file and introduce a dummy "out-of-range" value to force clipping, and interruption of the connecting line, which will normally resume at the next point.

### 6.4 Plotting data with Aitoff's projection

Plotting celestial coordinate data with Aitoff's projection is possible setting the scale flag (line 27 of parameter file) to 4. Note that in this mode no frame is drawn, and also (in Version 2) no clipping is made.

It is however possible to draw a "parallel-meridian" grid, using data stored in a file. As a sample a file FRAME::TB is provided. The command file @SKY::TB contains all necessary commands to plot such grid after the data have been plotted. The set-up for data at the beginning of @SKY may be freely changed, provided the part for the grid is not modified. A typical starting set-up in a parameter file is provided by parameter file SKY.PF.

Note that the correspondence between the limits of the "virtual" frame (x and y start and step in user units) and the position on the screen (axis origin and length in inches) are somewhat ambiguous. There is an hidden scaling factor of 2.2 in size, and an even more hidden shift. Contact Paolo Giommi for details, or do some experiments of your own.

### 6.5 Linear regression

Concerning linear regression note that if you specify equal values for the extrema of the best fit line in the parameter file (lines 63 and 64 equal), the extrema are adjusted to the extrema of the data used on x in the plot (not necessarily the full frame).

The best fit line is plotted in the current (symbol) colour, always as a solid line (irrespective of reverse mode setting).

.cp5
You shall use at least three points to produce a linear regression (a warning is issued if less are used).

### 6.6 Histograms

Concerning histograms, note that logarithmic scales are not allowed. Also note the autoscale flag at line 12 of parameter file applies to histograms as well. However note that flag to be used to automatically set the bin width is line 37 in parameter file.

There is an interaction between upper limit flag (line 33 of parameter file) and reverse mode flag in the case of histograms. The following hints are based on experiments more than on the interpretation of the source listing.

When analysing only detections (i.e. no upper limits), the reverse mode flag can be used to produce a dashed histogram of detections and upper limits in the current symbol colour (otherwise the histogram relevant to detections only is drawn as a solid line).

When analysing detections and upper limits together, the reverse mode flag shall be disabled. This way two separate distributions will be plotted: the detection histogram as solid, in the current symbol colour, while the upper limit histogram will be dashed, in the current upper limit colour. If reverse mode is enabled, a single, cumulative histogram for both detections and upper limits will be produced, dashed, in the current upper limit colour.

When analysing only upper limits (we always refer to line 33 of the parameter file), the reverse mode shall be disabled. This way a single solid histogram of upper limits is produced in the current symbol colour. If the reverse mode is enabled,

the histogram produced is still solid, and in the symbol colour, but is relative to detections and upper limits together.

### 6.7 Functions

Concerning functions, note that there is no protection for illegal function values (like negative points in log scale, or other arithmetic library errors, overflows etc.), except that the minimum x being negative gives a warning when x scale is log.

Reverse mode does not apply to functions.

The autoscale flag at line 12 of parameter file applies to functions as well. Note instead that, in order to have an automatic adjustment of the extrema for function to the full frame, you should set the xmin and xmax values (lines 59 and 60 of parameter file) equal (between them, to any value).

.cp4
Note that, after having plotted a function, you ask for another function, and do not want to change the parameter file, you are prompted for a function identifier and parameters, irrespective of the setting of the flag in parameter file line 53.

.pa
.pn 1
.he$PAGThe PABLO Handbook Dec 88
Page 7-#
.foThe PABLO Handbook
Section 7

## 7. Programmers notes

The HP PABLO version is a segmented program, currently consisting of a main and 7 segments (the eigth segment, for contouring, is not enabled). External references, other than plotting routines, are mostly resolved within the PABLO library ($PBLIB).

Plotting routines shall be provided in a site-dependent Calcomp-style library. Only the following routine calls are used (the names should be customary for Calcomp-style calls, possibly abbreviated for consistency with previous HP conventions)i:

.cp4

one call to PLTLU to initialize the system
calls to PEN to change the pen colour
calls to PLOT(X,Y,pen) where X and Y are the coordinates of the next point in inches, and pen is either 2 or 3 (pen up or pen down) or 999 (end of plot)
calls to SYMB and NUMB for character labels and numbers
calls to SCALE (non-plotting) for the autoscale feature

Note that the AXIS routine generally available, but quite unsatisfactory, in most Calcomp-style packages, is not used. A modification of it is provided in $PBLIB.

## 7.1 Loading instructions (IFCTR)

To load PABLO the following steps are necessary :

If anything is changed in &PBLIB, the indexed library shall be regenerated. This may be skipped if the changes are only in the segments. The current version of the indexed relocatable $PBLIB is permanently kept on disk TB. To regenerate the library do:

**TR,*CLIB,&PBLIB,B,PG,TB**

If anything is changed in a segment, that segment shall be recompiled. As the relocatables of the segments are not kept permanently on disk, all segments shall be recompiled all the times, even when changes are in the library only. To compile all segments and the main and link them, use the procedure :

**TR,*PABLO**

Note this procedure deletes the relocatable. An exception to the use of this procedure is if you are doing some development, in which case you may want to keep the relocatables of the segments you are not changing on disk for some time, and recompile only the others (use **TR,*F**). At any time you may then link PABLO with

**LINK,#PABLO**

The libraries required, other than $PBLIB and the plotting library $MPTLB, are $LBINT and $EXLIB.

.cp5
It is recommended, when modifying PABLO, to rename the current **PABLO:PG:E6** to something else like OPABLO (old PABLO) before starting changing things, so that the original working

version remains available.

## 7.2 Segmentation scheme

The current (HP RTE-6) segmentation scheme for PABLO uses SEGLD to transfer control from one segment to the other. Segments are structured as pieces of main program (no SEGRT calls used, nor dummy mains). The usage of the segments is as follows:

The main program (source in **&PABLO**) just selects the plotter logical units, initializes it and opens the parameter file. Control is then transferred to segment 1.

The source of segment n is in file **&PABL**n

Segment 1 is entered only from main and from segment 5. It reads the current (original or working copy) of the parameter file. The first time only (when called from main) it asks the plotting option. On further calls the option had already been selected. Control is then transferred to segments 2,3,4, or 7 as specified by the option.

All plotting segments (2,3,4, and 7) may transfer control to any other plotting segment (if no parameter file change is requested) or to segment 5 for parameter file change.

Segment 2 is the data plotting segment. All plotting is done in routine PADAT. This segment may also transfer control to segment 6 for linear regression. Otherwise see above.

Segment 3 is the histogram segment. All plotting is done in routine PAHIS. Control transfer as usual.

Segment 4 is the function plotting segment. Routine PFUN is used to generate the data, and PTFU to plot them. Control transfer as usual.

Segment 5 is the parameter file editing segment. At the end control is always transferred to segment 1, which re-reads the parameter file and then transfer to the appropriate data plotting segment.

Segment 6 is the regression segment, entered by segment 2

only. It behaves as any other plotting segment, and may transfer control to segment 5 or to any plotting segment, except itself.

Segment 7 is the text segment. Does its own parameter file change, and does control transfer as usual.

.cp3
In the case repetition of the action of the current plotting segment is required, this occurs directly (via a GOTO) if no parameter file change is required. Otherwise it goes via segments 5 and 1.

All segments and main are able to terminate PABLO, which is normally done via the STOPP routine, which gets rid of all scratch working files.

.cp7

## 7.3 Future developments

Proposals for possible future improvements are the following :

cleanup and streamlining, removing some unreferenced variables or replacing unstructured code, adding i/o error protection, simplifying the scratch file usage etc.

enable segment for contours

allow functions to be defined by user in an external file, corresponding to a full main program. PABLO shall verify whether such program is linked or compile and/or link it as appropriate, then schedule it passing the array of x, and retrieving the array of y.

add further statistical/graphic stuff, like polynomial fits, splines, general curve fitting (CURFIT) etc.

This documentation is kept in
files PABLO.HP and PABLO-1.HP
Appendix is in file PABLO-A.HP
L.Chiappetti 17 august 1988
Released in December 1988

.pa
.fi pablo-a.hp

## Appendix A : table of symbols

The following table contains a listing of symbols by number, as plotted by the current routine SYMB. To find a symbol by number look at the appropriate x and y coordinate (number = 10*y + x, e.g. symbol 24 is at the intersection of y=20 with x=4 etc.).



## Appendix B : device characteristics for IFCTR

The following table lists the plotting devices available at IFCTR, together with the relevant logical unit number, the maximum size of x and y axes in pseudo-inches and the characteristics of the colours available. Pseudo-inches are conventional units mapped on a screen, while coincide with true inches for hardcopy devices (plotter and laser printer).

| LU | Device | Max x size | Max y size | Colour availability |
|---|---|---|---|---|
| 0<br>47 | dummy (bit bucket) | any | any | irrelevant, no plot is performed |
| 1 | graphic terminal | see below | see below | see below for specific terminals |
| 38 | HP 7550 plotter with | | | colours 1-8 according to pens mounted; colour 0 to put pen away; when pen not mounted see plotter manual. |
| | A4 paper rotate 0 | 10.x | 7.x | |
| | A4 paper rotate 90 | 7.x | 10.x | |
| | A3 paper rotate 0 | 15.x | 10.x | |
| | A3 paper rotate 90 | 10.x | 15.x | |
| 41 | HP 2648 terminal | 20.x | 10.x | B&W terminal; Undraw |

| | | | | |
|---|---|---|---|---|
| | resolution 720x360 | | | with colour 0 not allowed. |
| 42 | HP 2397 terminal | | | colours 0-7 according to current palette. |
| | normal 512x390 | 13.x | 10.x | colour 0 may be used |
| | altern 640x400 | 13.x | 10.x | to undraw. |
| HP 2393 terminal | | | | B&W terminal; colours not relevant. Undraw |
| | normal 512x390 | 14.x | 10.x | with colour 0 not |
| | altern 640x400 | 17.x | 10.x | allowed. |
| 44 | HP 2623 terminal | 13.x | 10.x | B&W terminal; Undraw |
| | resolution 512x390 | | | with colour 0 not allowed. |
| 46 | Ramtek 9030 display | 10.x | 10.x | allowed range 0-1023 |
| | resolution 512x512 | | | according to current colour table. Colour 0 is background (used to undraw). |
| 48 | Laserjet II printer | 7.x | 10.x | B&W hardcopy device |
| | A4 portrait only | | | |

It is suggested that only colours 1-8 are actually used (for compatibility with the plotter). On the Ramtek the colour tables >PLOT and >PLUT provide 8-colour tables, with 0 as black. Refer to the Exosat system documentation for use of colour tables.

On the 2397 colour terminal, the default palette has 0=black 1=red 2=green 3=yellow 4=blue 5=magenta 6=cyan 7=white. See terminal manual about selection of other palettes.

The behaviour of the plotter when a not-mounted pen is requested is random (see plotter manual for details). If pen 0 is called, the current pen is put away.

On black and white devices colour 0 will result in something being drawn as with any other colour.

The resolution of the various device in pixels is given for screens. Note that the resolution is configurable for HP 239x terminals (refer to terminal manual). The reason of the different pseudo-inches rendition between 2393 and 2397 (at

normal resolution and at alternate resolution, where the 2397 has a 13x10 centred frame) is unexplained.

Note that the resolution of the hardcopy Thinkjet printer attached to a 239x may be configured (refer to terminal and printer manuals).



## Appendix C : PABLO functions available at IFCTR

This table will be provided after a suitable collegial choice has been made about which functions to include.



## The LNPAB library calls

Refer to the Library Documentation (IFCTR Software Documentation) pag D-30 for general information. Here we list only the calls to the various routines, in the order they should generally occur. Refer to the library listing (file &LNPAB::LC) for more details.

### Routine **OPNPL**

This routine is always called first to open a PABLO command file cmdfile and select the plotting logical unit plotlu. The calling sequence is:

**CALL OPNPL (**plotlu,cmdfile**)**

where cmdfile is a 12-character namr kept in a 6-element INTEGER*2 array (or Hollerith variable), and plotlu is an INTEGER variable. This routine opens cmdfile and writes to that the commands for a "dummy text" (see 2.7 and 2.8).

### Routine **FRAME**

This routine is called to generate the commands to modify the parameter file for the next frame to be plotted. It is mainly intended for data plots, but shall be called also for other types of plots (followed by suitable modification calls). The list of arguments is very long (arguments are of mixed type, most INTEGER, some REAL and some character Hollerith string held in INTEGER array), and is not given here (refer to the source listing), as they correspond to setting almost all lines in the parameter file, namely :

lines 1 to 5 (data file characteristics)
lines 7 to 11 (see below for specification of start/end x/y)
lines 14 to 25 (see note for lines 12,13,26)
lines 27 to 31 (for log scales, only integral decades supported)
lines 40 to 41 (sets labels of columns 1 and 2)

Lines 13 and 26 are set to plot no upper limits. Line 12 is set to disable autoscale. All arguments shall be provided all the times (arguments not needed could be set to any value). Note that for linear scales, instead of giving start and step in user coordinates you give start and end (more friendly) in an array, and the program computes start and step given the axis length.

Note that the routine sets up the commands to do a data plot of column 1 along x and column 2 along y.

.cp6

Routine **RPEAT**

This routine is called to plot more than 1000 points. First call FRAME to plot points 1 to 1000 (or whatever) then :

**CALL RPEAT (**start,end,step**)**

to generate the commands necessary to make the number of further plots necessary to exhaust all points from start to end plotting step points at a time (max 1000). All arguments are INTEGER.

Also this routine sets up the commands to do a data plot of columns 1 and 2 (similarly to FRAME).

Routine **ADLAB**

This routine shall be called after FRAME to generate the commands to set in the parameter file the label for the nth column (1-9). The routine is called once for each column as:

**CALL ADLAB (**label,n**)**

where label is a 20-character string in an Hollerith variable or a 10-element INTEGER*2 array, and n is INTEGER. The routine deletes the last commands generated by FRAME, inserts the new "parameter file edit" commands, and ripristinates the commands necessary to plot column 1 and 2 on x and y.

Routine **OTHER**

This routine shall be called after FRAME to generate the commands to set in the parameter file some parameter not set by FRAME. It has to be called once for each parameter as:

**CALL OTHER (**linenumber,value**)**

in order to set the value for parameter file line linenumber. Both arguments are INTEGER, therefore only integer parameters may be set (intended for line 12,13,26). The routine deletes the last commands generated by FRAME, inserts the new "parameter file edit" commands, and ripristinates the commands necessary to plot column 1 and 2 on x and y.

Note that ADLAB and OTHER may be called as many times as wished after FRAME in any order and combination.

Routine **COLUM**

This routine shall be called after FRAME (and eventually ADLAB and OTHER) in order to plot columns other than 1 and 2 on x and y. It is not necessary to call it for columns 1 and 2, as the relevant commands are generated by default by FRAME. The calling sequence is :

**CALL COLUM (**ix,iy**)**

to plot column number ix on x and iy on y. Both arguments are INTEGER. Labels for such columns shall have been set by ADLAB. The relevant commands are replaced in the file.

Routine **PLFUN**

This routine (see the source listing for argument details, as the list of arguments is quite long, similarly to FRAME, although shorter) generates the commands to set the parameter file for a function plot. It allows setting only parameter file lines 50-52 and 54-62. Line 53 is set to 1 (function characteristics taken from parameter file, as just set).

FRAME shall be called first, possibly plotting a data set, or plotting dummy data (at least one point, clipped, or in the origin). This will allow plotting the frame. Then PLFUN will set the "more-than-one-frame flag" to 0, and include the commands to plot a function. (Do not forget to reset the "more-than-one-frame flag").

Routine **PLSTR**

This routine is used to generate the commands to plot a string of text. It is called once for every string, one string after the other, as follows:

**CALL PLSTR (**colour,x,y,text,chsize,rptflag**)**

where colour (INTEGER) is the string colour, x and y (REAL) are the coordinates in inches, text is the text string (Hollerith or INTEGER*2 array, only first 60 characters used), and chsize (REAL) is the character size in inches.

The INTEGER rptflag is used to determine whether the text is isolated, or is the first, last or central of a sequence (this is a concept which stems from the arabic alphabet), as the relevant commands are different (see 4.4) as follows :

0 only this text plotted
1 first text string of a sequence
2 continuation of a sequence (not last)
3 last text string of a sequence

Routine PLDAT

This routine was intended to generate a data file in the older fixed PABLO format in the original WKS emulation package. With the newer free format this call should not be necessary (use any Fortran instead, if file is not pre-existing).

However see listing for details.

Routine **CLOSP**

This is always the last call, it closes the command file, and schedules PABLO. It is called as :

**CALL CLOSP (**plotlu, schedflag, cfile)

where plotlu and cfile are the same used in OPNPL. The INTEGER schedflag may assume the values :

1 to close the command file without further action. PABLO is not scheduled (do it later manually as **PABLO,**cfile)

0 close the command file and schedule PABLO without wait. The calling program continues executing. Not recommended because of interferences with other terminal i/o.

1 close the command file and schedule PABLO with wait. The calling program waits for PABLO to finish. Recommended.

Summarising

Always start with a CALL OPNPL.
For each data plot, include a CALL FRAME (followed by some CALL ADLAB and one CALL COLUM if not using columns 1 and 2).
After all your data frames, include any CALL PLSTR you need for text.
Finish with a CALL CLOSP.