.pl 60
.mb 3
.po 0
.op

THE PLOT EDITOR

USERS' HANDBOOK

Release 1
November 1989

This document has been prepared by:
L.Chiappetti, IFCTR

The software described in this document has been contributed by the following persons :

L.Chiappetti IFCTR, CNR Milano

.he$PAGThe Plot Editor Handbook (November 1989)
Contents
.pa
.pn 1
.foContents
Page #

Table of content

.cp50

Amendment history

February 1988 v 0.0 preliminary unofficial software release (without documentation)

November 1989 v 1.0 official release 1 of software and associated documentation

1. Introduction

The Plot Editor (**PLOTED**) is a program designed to handle complex plots with a sophisticated layout control, allowing to (edit) change them and generate new variants very easily in an interactive way, allowing at the same time to keep a compact re-usable record of the plot layout.

As an example, this program has been succesfully used to test and produce a 40-frame plot (combining light curves in 5 different energy bands for 8 observations of a BL Lac), or to produce standard "summary plots" for spectral fitting analysis.

1.1 Definitions

We start first with some useful definitions :

A plot is what appears on a page. As such it has some page-associated characteristics, namely the size (A4 or A3) and the orientation (landscape or portrait) of the page.

A plot consists of objects called frames and strings.

A frame is a rectangular area, generally bounded by solid lines, containing a cartesian plot, usually related to a given dataset. The boundaries of the frame correspond to the axes in user units. The annotations (numbers and caption) of the axes are outside the frame boundaries, but associated to it.

A string is a free-standing text (plot title, or comment of whatsoever nature), not immediately associated with a frame.

A frame has a number of associated characteristics. The first one is the frame type, which describes the content, i.e. the type of plot : currently four types are defined (the first three in accordance with the usage in the PABLO plotting program, see the relevant Users'

Handbook) namely :

Plots are cartesian plots of one set of data (the y-values) versus another (the x-values)

Histograms are cartesian plots of the frequency distribution (on y) of one set of data (reported on x).

Functions are cartesian plots of the values of an analytic function y=f(x), generated internally by a plotting program, without any external dataset (file).

Contours are contour maps of the levels of a quantity z versus x and y. The z-values are taken from an image-type file.

.cp4

The remaining frame characteristics may be typeless or type-related. A typeless characteristic is one that applies to all frames independently of the type (like e.g. the origin of the frame on the page, the length of the axes, the scales in user units, etc.). A type-related characteristic applies instead only to one particular type of frame (e.g. the data file name does not apply to functions, the way to connect data points applies to normal plots only, the level thresholds apply to contours only, etc.).

A string has a simpler set of characteristics (like the position, the text, the character size, the colour etc.)

Still concerning frames, it has to be noted that more frames may be superimposed. A simple case is that of insets, that is a small frame plotted in a free space within the boundaries of a bigger frame. However there are also cases of fully overtraced frames : this occurs in a simple case when one wants to plot more than one dataset in the same physical frame, or when one wants to plot an analytic function over a set of points. From the point of view of the Plot Editor each data set or curve shall be formally considered an independent frame.

However overtracing is used also in other cases. Dummy frames are allowed for the purpose of adding extra axis scale labels (e.g. a scale in keV and one in V). A dummy frame is a frame defined for the purpose of axis annotation only, but without any associated data file to be plotted (an "empty plot").

A further definition concerns axis sets. A frame may have a number of axis sets associated, which is not necessary 2 (it may be none if no axes have to be annotated, which is typical for overtraced frames). For each set of tics you want to put on an axis, there is one axis set : therefore if you want to have tics on the top, bottom, left and right frame boundaries, this makes 4 axis sets; and if on any one axis you want two sets of tics (say big tics every 10 units and smaller tics every 5 units), this makes 2 axis sets.

Axis sets also have associated characteristics (the tic spacing, the format of the numeric labels, their orientation, etc.).

1.2 What does the Plot Editor do ?

It is important to understand what the Plot Editor does, in order to use it taking maximum advantage of its capabilities. Befor using the Plot Editor it is important you plan the layout of your plot.

You should take a piece of paper and plan how many frames you want, and where you want them located on the page, what data files they will use, and all nuances of the plot. When you have done this you may use the program (and after that you will find very easy to change the layout of your plot, either to correct bugs in the design, or simply to produce new plots).

.cp5

You should realize that the Plot Editor does NOT plot anything directly. It creates and mantains (edits) a plot descriptor file, which contains the description of your plot in the more general terms as possible (general means here also independent of the particular plotting program used).

Additionally, when the editing phase is finished, the Plot Editor generates the command files for the various plotting programs supported, and, if requested, also runs such programs to produce the plot. Of course the programs supported are installation dependent (in our case such programs are PABLO, MCONT and LABLER).

The editing of the descriptor file is screen-oriented : for each frame or string you are shown a form on the screen, and you may move the cursor around and change

the fields (characteristics) you want to change. You also have a set of simple mnemonic commands to do global changes on the file (like e.g. changing the axis length in all frames, or translating/scaling a frame).

1.3 The current implementation

This section is of interest to system-oriented people only. Other readers should jump directly to section 2.

The currently available implementation of the Plot Editor is running on HP-1000 computers under RTE-6 VM with FMGR. Screen handling supports a range of HP terminals of the 26xx and 23xx series. Special support for plotting is given to the HP 7550 HP-GL plotter.

The following considerations about program portability has been made :

The plot descriptor file format has been designed as general as possible, and as such it should allow porting of the program to other systems very easily (although this does not imply exchange of descriptor files among heterogeneous systems). However the descriptor file contains some dependencies on the local plotting programs supported (see below; these will be pointed out throughout the text).

The screen handling has been designed to be semi-portable, that is, it should be able to handle other terminal types driven by different escape sequences at the expense of a few changes to some library routines. See programmers' notes in section 14 below.

The plotting program selection is a local business (which may be accounted for by the replacement of an entire segment) : the program currently supported are the general plotting program PABLO (described in a separate document), the contouring program MCONT (described separately in the Exosat and IUE documentation), and the Labeller program (also described in a separate document) used to annotate both types of plots on the HP-7550 plotter.

Section 2

2. Instruction for use

A notice for first-time readers : the following sections should be absolutely read : 2.1, 2.2, 2.4, 4 and 6. Section 7 and 8 are intended for consultation and browsing. Sections 10 and 11 should then be absolutely read. In emergencies, read section 12 instead of panicking.

2.1 Getting started

The Plot Editor is called by the following command :

PLOTED,fname

where fname is the (optional) name (namr in HP terms) of the plot descriptor file. You may specify a full namr (name:sc:cr) or an incomplete namr (without security code and/or cartridge). The handling of incompletely specified names is described in 2.3 and 11.3 below, when closure of files is explained.

If you do not specify a name when calling the Plot Editor, you will be prompted in a manner explained below.

But first of all you should know on which terminals you are allowed to run the Plot Editor. The terminals currently supported are:

HP 2397 colour terminal (logical unit 42 in HP room)
HP 2393 graphic terminal
HP 2623 graphic terminal (logical unit 44 in HP room)

It is likely that support to HP 2327 or 2390 may be implemented easily if not already available. Support to the HP 2648 is already foreseen in the code, but is disabled because of a problem with the DC1/DC2 handshaking during cursor position reads.

HP 2621 dumb terminals are not supported.

Usage of the colour terminal is recommended whenever possible as the program uses colour coding for various messages in a manner clearer than on other terminals.

Once you have issued the command, the program starts and the screen is cleared. However before doing this the program tries to identify the terminal. If it finds a

terminal not supported ends immediately with an explanatory message. If it seems to get stuck, type a carriage return (this should occur only on 2621 dumb terminals which do not identify) and you will also terminate with a "terminal not supported" message.

If you have not specified a plot descriptor file (or if you get any error on such file at startup) you will be prompted : the prompt occurs in the form of an inverse video field (it will be in inverse green on a colour terminal) which you should fill :

.cp5

note this is an alphanumeric field (alphanumeric fields are explained in 6.2) or better a set of three automatically parsed alphanumeric fields :

first type the name, up to 6 characters. Note that carriage return does not work as expected but just as a blank. At the end of the 6 characters, a colon (:) is issued by the program, and you input the next field. If your name is shorter than 6 characters, instead of typing the necessary blanks or returns, you may type <control D> or <control T> to jump to the next field.

The next field is the security code, 2 characters, handled in the same way.

The next and last field is the cartridge, 2 characters.

If you are getting repeated errors accessing a file, you are continuosly asked for a new name. Try changing the name. If the problem persists, issue a <control Q>. This should terminate the program cleanly. If the problem still persist walk to another terminal and break the program (breaking is explained in the "troubles" section, 12 below).

If the file is OK the program opens it, displays the global characteristics (page size and orientation, number of frames and strings, date of last change). If the file exists, it may take a while (you will get a message saying that the program is "ordering" the file). Ordering and in general the ways files are handled is explained in 2.3 below.

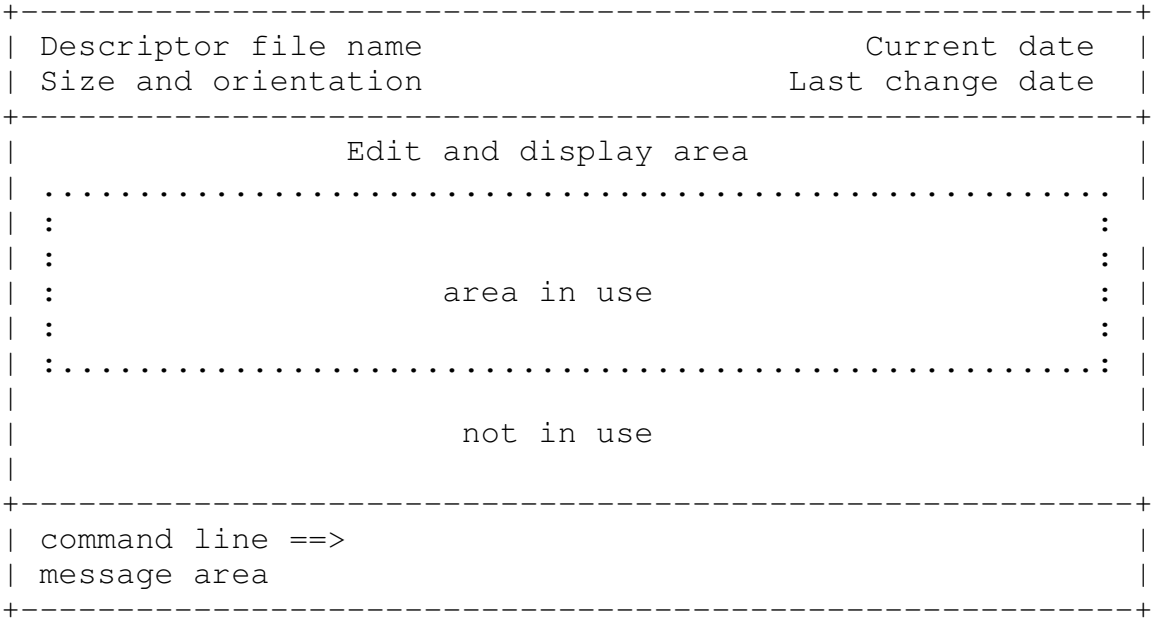
When the program is ready, it displays its typical screen layout and a frame directory (if any), and a command line. It also shows the current definition of the user keys. The screen layout and the usage of the keys are

described respectively in 2.2 and 2.4 below.

.cp50

2.2 The screen

The typical screen of the Plot Editor appears partitioned in various areas as shown below :



.cp3

The top part of the screen contains information relevant to the whole plot descriptor file. The current time appears in the top right corner and is updated after each edit. The top area should always remain on the screen. Two fields in the top area are worth noticing, that is the "size" and "orientation" field : this field may be edited by the RESIZE command (described below in section 9).

The middle (and major) part of the screen is reserved for editing or display. The frame and string directories are shown here. Also all the fields related to a single frame or string to be edited are shown here and are editable moving the cursor there (see below 6.1). Note that the lower part of the display area may be unused for edit (in this case it is "outside of the edit area" and moving the cursor there will trigger a "termination of edit" request.

The bottom part of the screen includes a command line

(where the cursor is positioned when giving commands), marked by an arrow (==>). Below the command line there is a message area, where the Plot Editor displays messages, warnings and error messages (in increasing order of severity).

The first time reader should skip the next section and go directly to section 2.4 from here.

.cp6

2.3 File handling

The Plot Editor tries to do its best to avoid unwanted deletion of a plot descriptor file. At the same time it has to use direct access files to allow fast editing. The way to overcome this is to work on a copy of the original file, replacing it only at the very last minute.

If your descriptor file already exists, the Plot Editor copies it (and reorders it) into a work file at startup. The original file is then closed, and the program goes on working on the original file.

If your file does not exist the Plot Editor does not create it, but creates a work file instead. The name you supply is remembered.

The name of the work file is **PLOT**nn**::X2**, where the value of your terminal logical unit nn is used to avoid naming conflicts and designating a file in an unique way.

If the program crashes you may recover any edits done so far, by renaming the **PLOT**nn file to a name of your choice, and restarting working on that. Do the renaming before calling the Plot Editor again, as the work files are otherwise overwritten each time !

At the end of an editing session you are presented with three choices (if you have used a text editor, these concepts should sound familiar to you) : file, save or quit.

Quitting is the simplest case : you discard all edits done, the work file is closed, and the original file remains untouched (if it did exist; it is not created otherwise).

Saving is also a simple business : it means that you want

to save your edits to a file different from the original (you will of course be prompted for a name). The Plot Editor therefore copies its work file to the designated file (provided this is not somehow forbidden). The descriptor file is ordered during the copy, and this means here also that all deleted frames and strings are not copied (up to this time it is still possible to undelete them, from now onwards they are lost). The work file is then closed. The original file remains untouched (if it did exist, and it is not even created otherwise : this in case you changed your mind on the name you wanted to assign to a new file).

Filing into the original file is more tricky. As the program as to reorder the descriptor file before filing (inclusive of removing deleted frames and strings, see above), it has to copy it somewhere. However copying it directly to the original file may not always be possible (the file may be protected or there may be not enough space on the disk). To avoid corruption of the original file, the copy is first attempted to another work file, named this time **TEMP**nn:sc:cr, where nn is the usual terminal logical unit number, but the security code sc and the cartridge cr are derived (as far as they are known to the program) from those of the original file. If and only if the copy is succesful, the original file (if existing) is purged (deleted), and the **WORK**nn file renamed with the original (remembered) name.

This way, in case of errors, the original file is not altered, and you are allowed to cure the error (the most frequent will be that you have forgotten to give the right security code), or, as ultima ratio to save the file with a different name.

For details on the way file names, security codes and cartridges are handled by the program see section 11 below.

2.4 Giving commands

After the startup, the Plot Editor shows you the typical screen, possibly with a frame directory (if there are frames in the descriptor file), and displays the command line.

Commands may be entered on the command line, or also using the soft keys labelled f1 to f8. On most terminals you will see the labels for the soft keys appearing

automatically after startup.

There are nine top-level commands, of which the first eight are assigned to a soft key also. To issue a command, the first letter (or hitting the corresponding soft key) is enough (not even a carriage return is required !). The commands are listed here, together with their mnemonic form :

Frame directory displays the current frame directory (no further parameters required), see section 3

String directory displays the current string directory (no further parameters reuired), see section 3

Delete controls deletion and recovery of frames and strings (it will prompt for a frame or string number), see section 5

Edit opens a frame or string for edit (it will prompt for a frame or string number), see section 4

Add opens a new frame or string for edit (which can be modelled on a pre-existing frame or string, whose number will be prompted), see section 4

Resize allows editing of the page size and orientation fields in sequential mode (see 6.1 for modes), see section 9

Global edit allows to edit the characteristics of more than one frame or string, as well as to translate or scale whole groups of frames or strings (it will prompt for further actions), see section 10

.cp5

Terminate passes to the termination phase (see 11), where command files are created, plotting programs are run, and the descriptor file is saved to disk (this command is protected : you have to type it twice, otherwise it will return to asking for a new command)

Quit terminates the program immediately, without saving any change to disk. Use in emergency. This command is not assigned to a soft key. This command is protected similarly to the Termination command.

If you enter any other character than a valid command you will get an error message. All messages appear below the command line, and are cleared when you issue the next command. See 2.5 for colour coding of messages and severity.

The main commands are described below in sections 3 to 11 : have a look at section 4 at first.

Please note the following convention will be used describing command input modalities :

boldface will indicate a command keyword as you type it

normal typeface underlined will indicate what the computer types in reply (trying to complete the word, where the command is typically the initial)

italic indicates a variable field you should enter with the proper value

2.5 Colour codes

If you run the Plot Editor on a colour terminal, you will notice that different fields on the screen appear in different colours and enhancements. Such coding is not possible on other terminals, however it is replaced somehow by the use of enhancements like underline, half-bright, inverse video and blink.

All fields which are just fixed parts of the "form" appearing on the screen (informative sections) are in plain white.

.cp3

Fields which appear in yellow are variable fields under control of the program (not editable), which are displayed this way to make them noticeable (like the number of frames and strings in a file).

Fields which appear in green are editable fields. They will appear in inverse video green when they are being edited. This usage of the inverse video applies also to black-and-white terminals.

.cp5

Alphanumeric fields (which may be of variable length) use a

double-colour coding when being edited : inverse video green indicates a pre-existing non-blank text, while inverse video yellow indicates space free for a possible extension.

Red and blink are freely use to give emphasis to some particularly noticeable fields (e.g. the command line).

Concerning messages, the following coding applies :

Informative messages appear in magenta

Warnings, and messages concerning a selection of a range of allowed characters, appear in cyan

Error messages appear in red

Moreover, blink, terminal delay and the bell are freely used to emphasize important messages.

Finally colour coding is employed in the display of frame and string directories to show the status of the frame/string :

green indicating a normal (active) frame or string
cyan indicating an inactive frame or string
blue indicating a deleted frame or string

Frame and string status are explained below in sections 3 and 5.

.foThe PLOTED Handbook (release 1)
Section 3
.cp7

3. The directories

A frame or string directory is arranged on the screen on several columns. There are four major columns, so that frames (or strings) 1 to 4 fit on the first line, 5 to 8 on the second line and so on. The maximum number of frames or strings which fit on a screen in the display area is 72, which is also the maximum number of editable frames or strings.

Each frame or string is indicated by a sequence number, a one-letter status flag (alias disposition flag) and a 16-character identifier.
The meaning of the sequence number is obvious, this is also what you use to address a specific string or frame in all commands.

The identifier is a name of your choice you assign to a frame or string to tell it from the others. The name is assigned when creating/editing the frame or string, as any other characteristics (although it is not used for plotting). Frames with a null or blank identifier, will have it displayed with the word <blank>.

The status flag is also associated (on colour terminals) with colour coding (see 2.5 above). It may assume three values :

.cp3

a blank flag, with associated green colour, indicates a normal frame or string, ready to be edited.

an **I** flag, with associated cyan colour, indicates an inactive frame or string. This frame cannot currently be edited until is reactivated again, and will not be included in the command files for the plotting programs, however it will remain in the descriptor file.

a **D** flag, with associated blue colour, indicates a deleted frame or string. This cannot be edited, unless it is undeleted, which is always possible before termination. The frame or string will be actually deleted when the descriptor file is saved to disk.

.cp7

4. Editing and adding

The only difference between editing and adding is that editing applies to a pre-existing frame or string, while adding applies to a new frame or string.

In both cases you have first to specify, according to the request of the program in the message area, whether you want to edit/add a **F**rame or a **S**tring (the initial, without carriage return, is enough).

After that, in the case of editing, you have to supply the number of the frame or string you want to edit (this number should correspond to an existing and active frame or string).

.cp4

In the case of adding instead, the number of the new frame or string is supplied by the program. However you are asked whether you want to copy the layout of the frame or string from an existing one (in which case you specify its number as above) or create a blank one (in which case you reply with a **0** (zero)).

The sort of messages you will see on the command line are like, respectively :

Edit **F**rame nn

Add **S**tring mm copied from nn

Note that the program expects a two-digit frame or string number nn, if your number is 0-9 type a blank or a carriage return after the number.

If you get an error message, you are requested to supply the correct input. If by any chance you want to abort the current operation (typically if you have requested **E**dit instead of **A**dd), you may return to the empty command line just typing a <control Q>.

Once you have selected the frame or string to edit, a screenful of information with all the frame and string characteristics is displayed. This is the current content of the frame or string (if **E**diting), or the content of the template frame or string from which you are copying (if **A**dding copying from a preexisting object), or a "blank" string (if **A**dding a string "copying from zero"). See section 6 below for hints on how to edit the various fields.

If you are instead **A**dding a "blank" frame, you are first prompted to supply a frame type (types are specified in 1.1 above): just type a single letter (**P** for Plots, **F** for Functions, **H** for Histograms and **C** for Contours). Once you have selected a type, the appropriate "blank" frame screen is shown.

Once the screen is filled, you may edit the individual fields by moving the cursor there, or jumping sequentially through the fields, as explained below in section 6.

When you have finished editing, you should wish to save the new arrangement for the frame or string (to the work

disk file). To do this you move the cursor outside the edit area (either on top or on the bottom of the screen), and press a key. You are then prompted for one of the following actions (the initial, without carriage return, is enough):

Write to save the edits (make them permanent, you will note that the frame or string counter will increase by one : if the program crashes before current edits are saved they are irrecoverably lost)

Quit to abort all edits done (the frame or string will remain as before you started editing; if new it will not even be created)

Continue if you have changed your mind and want to go on editing

As a further facility you may also terminate editing while in the edit area, using a control sequence. Type <control W> (mnemonic : control WRITE) to save the edits, and <control Q> (mnemonic : control QUIT) to abort all edits.

BEWARE: if you type <control Q> you are NOT asked any confirmation !

BEWARE also: if you type <control W> while a field is open for editing (displayed in inverse video), any edits done to such field are not saved ; please remember to close the field before issuing the <control W>.

Note that <control Q> and <control W> have a different meaning when the cursor is in the axis area (when editing frames), see 6.4 below.

.cp10

5. Deleting

The Delete command controls the frame/string status (i.e. is used to delete and undelete. In both cases you have first to specify, according to the request of the program in the message area, whether you want to delete a **F**rame or a **S**tring (the initial, without carriage return, is enough).

After that you have to supply the number of the frame or

string you want to operate upon (this number should correspond to an existing frame or string).

.cp4

Finally you are requested to supply the action (again the initial letter is enough) as one of :

Delete (mark for deferred deletion)
Inactivate
Undelete

The sort of messages you will see on the command line are like the following :

Delete **F**rame nn to be **U**ndeleted

Note that the program expects a two-digit frame or string number nn, if your number is 0-9 type a blank or a carriage return after the number.

If you get an error message, you are requested to supply the correct input. If by any chance you want to abort the current operation, you may return to the empty command line just typing a <control Q>.

If a frame or string directory is currently on the screen, it will be refreshed to indicate the occurred change of status.

Note that deletion of a frame/string implies just its status flag (see section 3) is set to "marked for deferred deletion" (blue on directory display). The frame will be actually deleted only when saving the descriptor file to disk.

Inactivation of a frame/string means it remains in the file, but is not used to generate a plot command file (hence it is not plotted).

The Undelete command brings a frame/string back to the active, normal status, irrespective of the previous status (i.e. it is used to recover both inactive and deleted frames/strings).

.cp10

6. How to edit on the screen

Most of the edits using the Plot Editor are done in a screen oriented way (the exception are "global edits" see section 10). On a typical screen you have many fields (on a colour terminal they will be displayed in green) which can be edited. You have generally to select a field (that is open it for editing) and then you can input your new value. An open field is always displayed in inverse video.

The way you move around on the screen and actually edit the different types of fields is described below.

6.1 Moving the cursor

There are, from this respect, two types of field: truly screen-oriented fields and sequentially editable fields.

Screen-oriented fields may be edited in any order. To select a field you move the cursor (with the arrow keys) to the field you are interested in (or even anywhere in the comment area close to it), and then hit any key (typically use carriage return, however note that any key you hit when a field is not selected will not be written to the screen, it just signals the program you want to edit a field). The selected field is displayed then in inverse video, and you can proceed as in 6.2 below.

When editing on a field is terminated, you are positioned to the "next" field by default (which however you may choose to ignore, and move the cursor elsewhere).

Sequentially editable fields are instead selected automatically by the program. The cursor is positioned on the first field of a family and this field is automatically open (appears in inverse video). When you have edited the field (see 6.2), the cursor is moved to the next field, which is in turn opened, etc. You are not allowed to go back to a previous field, but have to edit all fields of the family in the sequence.

6.2 Types of fields

Fields are classified also according to their content as numeric and alphanumeric fields. The way these fields are

edited is actually different. However there are some common characteristics :

A field is displayed with a given length (number of characters). If you type exactly that many characters, you do not need to provide a terminating carriage return (the program "reads" your value and jumps to the next field).

If you have typed less characters than the maximum lenght, you may terminate the input issuing a <control D> (always) or a carriage return (for numeric fields only).

In no case it is permitted to use the line-editing keys (insert char and delete char).

A numeric field contains only numbers. It is written in a format selected by the program (integer, floating point or exponential), and accepts input in free format (however the underlying variable is typed, in Fortran sense, therefore only integer values are allowed for integer variables (it should be obvious when a field is integer, as it is displayed as such).

You will always be sure about what you have typed, as the program redisplays the value in its own format once you have terminated the input (either typing all the requested characters, or hitting return, or hitting <control D>).

Note that you are not allowed to change (line-edit) what you are typing, that is you cannot use the arrow keys, nor the backspace key. If you have typed something wrong, the easiest way out is to generate a format error (e.g. typing two consecutive periods (..)).

Numeric fields are subject to error checking. Some errors are normal format errors, other errors are range errors, which are field- and context-dependent (the value should be in a given range). In all cases an erroneous field is cleared on the screen, and you are asked to re-enter a correct value.

Of course numeric fields keep defaults : if you type carriage return, the current value will be preserved (unless it is an out-of-range value, set by some other means like global editing to an illegal value).

An alphanumeric field contains a string. As such one wishes to keep as a default the current value of the string, and be able to edit it character by character. Limited line editing capabilities are provided :

Any printable character may be used to overwrite a pre-existing character at the cursor position. Non-printable characters are ignored with the exception given below.

The space bar is used to input destructive blanks, i.e. the typed spaces will overwrite existing characters.

.cp3

The carriage return is used as a non-destructive blank, i.e. as a way to move the cursor right by one character at a time. If the previous character is not displayed, it will appear typing return.
Remember that usage of the arrow keys is not allowed.

Consequently carriage return cannot be used as input terminator; to terminate input at the current place (and leave the rest of the string on the right of the cursor unchanged) use <control D>.

.cp2

Backspace is used to move the cursor left, and go back editing previous characters. Do not use the arrow keys for this.

<control T> (mnemonic <control TRUNCATE>) will terminate the input, and truncate the string at the current cursor position (all following characters set to blank)

<control Q> and <control W> are recognised in a context dependent manner. They generally terminate the input, losing all edits done to the string, and initiate some special action.

Note that there are no ways to insert characters in a string, you will have to retype it all over.

Error checking for alphanumeric fields is limited to keywords assuming only a restricted range of values (e.g. P,F,H,C for the frame type code).

6.3 Editing a frame

The typical frame screen consists of several fields, located in different screen regions. The majority of the fields correspond to frame characteristics (parameters) see section 7 for help.

The following regions may be individuated :

The frame identifier (title) which is not a frame characteristic used for plotting (see section 7), which is an editable alphanumeric field, and is on top of all.

The frame type (a one-character alphanumeric code), which is editable only the first time the frame is created as blank, and appears just below (see 1.1 for types).

The majority of the frame characteristics (see section 7), which occupy the middle part of the screen. Fields of various types, editable in screen-oriented way. Some fields are however conditionally editable, in the sense that edit is allowed only sometimes, in dependence of the content of other fields. If edit is not allowed the cursor cannot go there, and the fields cannot be selected. Examples are the regression parameters (not editable is the regression flag is set to NO), the axes of a contour frame (which are always linear and cannot switch to logarithmic), and a few other fields in contour frames.

The axis area in the bottom part of the edit area. As said in 1.1 one frame may have more than one axis sets. Only one at a time is shown in this area. The first line of the area indicates both the total number of axis sets, and the number of the current one. The cursor in the axis area will start axis editing. Axis editing is described in 6.4 below.

.cp4
Editing a frame is terminated as usually either by moving the cursor outside the edit area and selecting an action (W Q C see section 4), or by issuing a <control W> or <control Q> from anywhere outside the axis area.

6.4 Editing an axis

One is allowed to edit only one axis at a time. To start editing one moves the cursor to the beginning of the axis

area and selects it typing any character.

One is then requested to select the axis number. A carriage return selects the current axis, i.e. the one displayed on the screen. A number within the range, select an existing axis. A number equal to n+1 where n is the total number of axes select a new axis.

As for frames, a new axis can be copied from an existing one (you are prompted for the number) or be created as blank (you reply **0** to such prompt).

An axis shall be selected for editing also in order to be deleted. Deletion is explained below, and takes place immediately.

Once an axis has been selected, its characteristics are displayed in various fields. By default a blank axis will be an X-axis. The axis fields are sequentially editable, i.e. you have to edit all of them in order.

You may jump from one field to the next (leaving the previous unchanged) using <control D>, or carriage return for numeric fields. When you have edited the last field you are prompted for a terminating action, in reply to the question "is the axis OK", to be chosen among :

Yes axis is OK, save edits in memory and go back to axis menu
No axis is not OK, edit it again from the beginning
Quit axis is not OK, abort all edits and go back to axis menu

You may also terminate axis editing in advance, going back to the axis menu, typing at any time one of the two sequences (which take here a different action than when editing a frame) :

<control W> (control WRITE), to save the edits
<control Q> (control QUIT), to abort all edits

.cp2
Note that the content of the currently open field will not be saved when using control WRITE.

The following notes apply to the fields of an axis.

.cp5

The axis type field may assume one of the (one-character) values :

X	a main X axis on the bottom of the frame
Y	a main Y axis on the left of the frame
Bottom	a secondary X axis on the bottom of the frame
Top	a secondary X axis on the top of the frame
Left	a secondary Y axis on the left of the frame
Right	a secondary Y axis on the right of the frame
Delete	to delete the current axis

If you want to delete an axis, you are asked for confirmation (type **D** again). The axis is immediately removed, and the axis counter decremented (axes are renumbered). If you remove the last axis, it may occur you get an (innocuous) error 921 message.

The choice of axis types is relevant to the local implementation of plotting programs. X and Y axes are used directly in the command files for the PABLO and MCONT programs, while the other four cathegories (of obvious meaning for what their location is concerned) applies to the Labeller program only (X and Y axes apply also to the Labeller, and are handled as Bottom and Left axes respectively).

The fields controlling tic size, tic spacing, character size and orientation apply to the Labeller program only (see the relevant documentation). Note that PABLO and MCONT use the character size defined in the frame for all axes. Also note that tic spacing is not relevant to logarithmic axes.

Set a zero character size for axes you do not want to annotate.

The format field also applies to the Labeller only and determines the way numeric labels at tics are written. The documentation of the latter program describes the allowed format codes (essentially Fortran format codes, plus an "hour" format, for linear axes, and EXP or LOG for logarithmic axes). If you do not want any numeric label, you should type the word **<NONE>**, exactly as shown here, in the format field. Note that the format field is alphanumeric.

The last field is the label text for the axis caption. The program tells you the maximum allowed length for this field (when you have many axes you may run short of space,

otherwise you have typically up to 76 characters). The part currently in use is displayed in green, the allowed extension in yellow. The field is an alphanumeric field. You typically terminate editing with <control D> or exceptionally with <control T> (TRUNCATE).

Do not use "all blanks" to clear a label, nor do use <control T> to truncate it to zero-length. This may cause the program to crash. The correct way to clear a label, is to type as first characters the word **<NONE>**. The program will reply with a (red) "deleted" message. The space taken by the old label is not released in the file.

.cp4

Note that **<NONE>** shall be typed in capital letters. Use the CAPS key as necessary. The label text may be a mixture of upper and lower case letters, and also include Labeller escape sequences (see relevant documentation).

The Plot Editor converts the label to upper case, stripping the escape sequences, and truncating to 30 characters, when generating a command file for PABLO or MCONT. This way the plot is still legible.

6.5 Editing a string

Editing a string is simpler than editing a frame, as you have less fields. Typically you have a string identifier (not plotted, see section 8), the main string characteristics, and the string text.

Editing is screen-oriented, and you may edit fields in any order. Note however the edit area is shorter. You terminate edit as usual (cfr. 6.3) putting the cursor outside the edit area, or with <control W> and <control Q>.

The string text is split on the screen on several lines, which you edit one at a time. However it will be plotted as a single long string. Note that the entire string (trailing blanks removed) can be up to 232 characters. However if you want to edit the command files, note that the HP EDIT allows editing only up to 150 characters.

Here too the current part of the string is displayed in green, and the possible extension in yellow. You terminate input with <control D> or <control T> (TRUNCATE), and do this once for each line.

Use the CAPS key as necessary. The label text may be a mixture of upper and lower case letters, and also include Labeller escape sequences (see relevant documentation).

The Plot Editor converts the label to upper case, stripping the escape sequences, when generating a command file for PABLO or MCONT. This way the plot is still legible.

.pa
.fi ploted2.hp

7. Frame characteristics

The listing below describes all frame characteristics (see 1.1), as close as possible to the order in which they appear on the screen. For each frame it indicates the type of the corresponding field, and the allowed range of values, plus any hint for editing.

In addition a four-character code for a longer mnemonic name is given in boldface: this is the code that shall be used to change a characteristic in more than one frame in global edit mode (see section 10).

7.1 Typeless characteristics

Frame identifier

This is not a plottable characteristic. It is a 16-char alphanumeric fields, which contains an user selected title, displayed in directories. It cannot be changed via global edit.

Frame type

A one-character code (chosen among P,F,H,C), set once when the frame is created, and not modifiable afterwards.

Subflag or overtrace flag

It is an integer field, assuming the values 0,1 or 2. It cannot be changed via global edit. It controls the way frames are overtraced (PABLO users will recognize it is very close to the "flag for new frame" used there). A value of **0** indicates a self-standing frame : in this case the axes, tics, etc. will be plotted (by PABLO or MCONT). A value of **1** indicates an overtraced frame : no new axes will be drawn, and all information about axis scales (origin, length, start, end) will be ignored and taken from the "previous" frame (used when plotting more datasets exactly in the same frame). A value of **2** indicates also an overtraced frame : in this case drawing of new axes is inhibited, but the scale is changed (used when plotting datasets with different scale/units in

the same physical frame, or for "dummy" frames).

X-axis origin **XORI**gin
Y-axis origin **YORI**gin

The location of the frame origin in "outer" units (i.e. as used by the plotting programs, which means pseudo-inches for PABLO and MCONT). A numeric real field, whose value should be in the range allowed for a page and orientation as currently selected (A4 is about 7*10 and A3 is about 10*15 inches, or rotated according to orientation). Note that the A3 size should be suitable for plots going to a terminal or Ramtek CDU using the full, square screen. See PABLO handbook appendixes for screen sizes on all devices.

.cp5
X-axis length **XLENgth**
Y-axis length **YLENgth**

The length of the axes in "outer" units (inches). Another numeric real field, which is also checked to be in the range allowed by the page size and orientation.

X-axis start and end **XSTA**rt and **XEND**
Y-axis start and end **YSTA**rt and **YEND**

These are in user units (kg, parsecs, number of pears, etc.), and are numeric real fields without any limitation.

X-axis linear or logarithmic **XLIN**log
Y-axis linear or logarithmic **YLIN**log

Alphanumeric fields assuming the values **LIN** or **LOG** (for programmers : only the two last letters are preserved internally). Note that only linear axes are allowed for contour frames (edit inhibited).

Frame colour **FRCO**lour

The colour for frame is an integer field between 0 and 256. The way colour is handled is device dependent (see also the PABLO Handbook), typically a 0 colour will cause no plotting.

Colour **COLO**ur

The colour for plots, also ranges 0-256. For PABLO plots consult relevant note below.

Character size **CHSI**ze

The size of the characters for annotations in "outer" units (inches). In the current implementation applies to PABLO and MCONT plots only.

Number of axes

This is accessed only indirectly by means of the "axis edit" capabilities (see 6.4).

.cp12

7.2 Plot frame characteristics

Connect flag **CONN**ect

This is implementation-dependent. Currently is the PABLO connect flag (see relevant manual for all details) which assumes values 0 to 6 :

0	data points not connected
1	data points connected by a broken line
2	as 1, but "also first and last"
3	data points connected by a smooth line (beware !)
4-6	pseudo-histogram modes (see PABLO handbook)

Error bar flag **ERRB**ars

This determines the way error bars are drawn. Currently is identical with the PABLO flag of similar usage, a value between 0 and 4

0	no error bars
1-2	error bars on X (or Y) only
3	error bars on both X and Y
4	error diamonds drawn

Symbol size **SYMS**ize
Symbol number **SYMB**ol
Colour **COLO**ur

The size and shape of the symbol to be drawn at each data point. Sizes are in "outer" units (inches). Use size 0.0 to draw no symbols. The symbol shapes are coded as a number, and are implementation dependent. Symbols used by PABLO are

described in Appendix A of the PABLO handbook.

BEWARE of a peculiar implementation of the **SYMB**ol field. This is defined as a two character alphanumeric field. You may enter a numeric code, OR the two-letter code **CC** (programmers: kept internally as -1) to exploit the "colour column" feature (see PABLO Handbook for details).

If you enable the "colour column" the **COLO**ur field assumes a different meaning, it is a number 1-9 indicating a column in the data file containing the symbols and colours in which each point has to be drawn.

Upper limit flag **ULFL**ag
Upper limit colour **ULCO**lour1

Implementation dependent. See PABLO Handbook for details.

Regression flag **REGR**ession
Minimum x **MINX**
Maximum x **MAXX**

Implementation dependent. See PABLO Handbook for details. The regression flag assumes values **NO** (default) or **YES**. The editing of the other two fields is allowed only in the latter case, inhibited otherwise.

.cp10
Data file name **FILE**name
Error flag **ERRF**lag
Header lines **HEAD**er
First point **FIRS**t
Last point **LAST**
X column **XCOL**umn
Y column **YCOL**umn

The characteristics of the data file (in the format required by PABLO, (see relevant documentation). Most of them shall be obvious. Some parameters are limit checked (i.e. no more than 9 columns in a file, last point cannot be less than first point).

Filename is a 12-character alphanumeric field. If blank or null (as for dummy frames) it appears as <undefined>. To remove it just replace it with a blank string.

Error flag is implemented as in PABLO (i.e. 0 for a file "with

errors" and 1 for a file "without errors"). Errors are in the same column as data, with alternated records value-error, value-error etc.

7.3 Histogram frame characteristics

Reverse mode **REVM**ode
Dash size ("trat length") **DASH**size

Implementation dependent. Control dashed lines according to usage in PABLO. See relevant manual.

Upper limit flag **ULFL**ag
Upper limit colour **ULCO**lour

Implementation dependent. See PABLO Handbook for details.

Data file name **FILE**name
Error flag **ERRF**lag
Header lines **HEAD**er
First point **FIRS**t
Last point **LAST**
X column **XCOL**umn
Percentage flag **PERC**flag

The characteristics of the data file, similar to those indicated for Plot frames in 7.3 above. The only peculiarity for histograms is that the Y-column is not used, replaced by the percentage flag (0 or 1 as described in PABLO Handbook, to select absolute number frequency or percentage frequency).

Number of bins **NBIN**
Histogram start x **HSTA**rt
Bin width **BINW**idth

Numeric parameters of self-explanatory meaning (see also PABLO Handbook).

.cp 10

7.4 Function frame characteristics

Function identifier **IDEN**tifier

A numeric code for the different supported function. Implementation dependent (1-20 for PABLO; see Appendix C of the PABLO Handbook for locally implemented functions).

.cp7

Function parameters **PAR1** to **PAR5**

Up to five real parameters, whose meaning depends on the function identifier (see Appendix C of PABLO Handbook where they are labelled however P_0 to P_4).

Number of points **NPTS**

Minimum X **MINX**

Maximum X **MAXX**

Numeric parameters of self-explanatory mening (see also PABLO Handbook).

7.5 Contour frame characteristics

Number of levels **NLEV**els

The number of wished levels. Integer range 1-256

Level code **LEVC**ode

Implementation dependent. Currently uses the MCONT level codes (see Exosat and IUE system handbooks):

0	automatic selection of thresholds
1	linear-spaced thresholds between minimum and maximum
2	log-spaced thresholds between minimum and maximum
3	user supplied thresholds
4	thresholds read from a file

The setting of this code conditions the editing of some further fields. Note that only limited support is given to code 3 (a warning is issued). As such mode requires the user to supply a (variable) list of thresholds, which cannot be kept in the descriptor file, the Plot Editor writes a message in the command file generated for MCONT. It is care of the users to replace such messages with the required threshold values. The edited command file can be saved for future use.

Otherwise for reproducibility in automatic way use level code 4 (to create a level file use program DISPS, see Exosat

and IUE system manuals).

.cp5

Minimum level **MINL**evel

Maximum level **MAXL**evel

Integer values, range -32767 to 32767, according to image file content. Editing is allowed only for level codes 1 and 2.

Scale-by-pixel-size **SCAL**e

Implementation dependent. See MCONT documentation quoted above. The normal value should be **YES** (but **NO** is also possible, though discouraged).

Data file name **FILE**name

The name of an image file in EXOSAT format (this format is described in the Exosat system programmers' reference), as required by MCONT.

Level file name **LEVF**ile

The name of a file (created by DISPS) containing the level thresholds. Editing allowed only for level code 4.

7.6 Axis characteristics

Note that all parameters, unless explicitly stated, do not have a mnemonic name, as they are not globally editable (they can only be edited in the individual axis).

Axis type

One of X,Y,Top,Bottom,Left or Right. See 6.4

Tic size **TICS**ize

The tic size in "outer" units (inches). Tics are always orientated to the top (x-axis) or to the right (y-axis). For Top and Right axes, or anyhow to invert tic direction, code a negative size. This parameter is globally editable.

Tic step

In user units, the distance between tics. Not used for log

scales.

Character size for label **LBSI**ze

The size of characters used for the annotations of a particular axis (in "outer" units, i.e. inches). In current implementations applies to Labeller only (PABLO and MCONT use the frame characteristic). This parameter is globally editable.

Format

An alphanumeric field, containing a format in the way required by the Labeller (see relevant handbook, under command NUMBER).

.cp4
Orientation

A one-character code, specifying **N**ormal or **R**otated. As described in the Labeller Handbook, controls the orientation of numeric labels. Note that centering of numeric labels, and offset of axis caption take the default values (see Labeller Handbook under command NUMBER).

Label text

An alphanumeric field, eventually with imbedded escape sequences.
.cp7

8. String characteristics

The listing below describes all string characteristics (see 1.1), as close as possible to the order in which they appear on the screen. For each string it indicates the type of the corresponding field, and the allowed range of values, plus any hint for editing.

In addition a four-character code for a longer mnemonic name is given in boldface: this is the code that shall be used to change a characteristic in more than one string in global edit mode (see section 10)

X position **XPOS**ition
Y position **YPOS**ition

The position of the string in "outer program" units (in our case pseudo-inches, see 7.1 above, under XORIgin).

Colour **COLO**ur

An integer 0-256, device dependent. See 7.1 above, under COLOur.

Slant **SLAN**t
Orientation **ORIE**ntation

Real values in degrees (0.0-360.0). No slant is 0 degrees. Horizontal direction is also 0 degrees.

Character size **CHSI**ze

The size for characters in "outer" units (inches).

String text

This field is not globally editable. It is a long alphanumeric field, split on the screen on several lines. Note that the way the string is divided on the screen is for pure convenience and has nothing to do with the way the string appears on the plot. Normally the string appears on a single line (be careful when editing a Labeller command file, as the string can be longer than the 150 characters allowed by EDIT). However you may use the slash (**/**) to cause division of the string on the screen into several lines when plotted.

.cp20
.foThe PLOTED Handbook (release 1)
Section 9

9. Resizing

The **R**esize command enters a short sequential editing of two fields (which appear close to the top of the screen) :

the page size (alphanumeric) : A4 and A3 allowed

the orientation (alphanumeric one-character codes) : **P**ortrait or **L**andscape

These fields affects the limits used for checking the xy axis origin and length. For square or rectangular screens (10*10 inches as Ramtek or bigger as some terminals), use A3 size in order to access the entire screen.

10. Global changes

The **G**lobal edit command enters a "dialogue" on the command line. This command is used to apply edit changes to more than one frame or string. You first specify, with one-letter codes, what you want to edit (i.e. **F**rames, **S**trings or **B**oth), then the range of frames or strings, and finally the action and the relevant parameters.

Note that if you select to global-edit **B**oth frames and strings, you implicitly select **ALL** frames and strings. Therefore range selection is bypassed.

The allowed actions are (1-letter codes): applying a **T**ranslation to one or more objects, applying a **S**caling to one or more objects, or **C**hanging one or more parameters in one or more objects. You are also allowed to **R**epeat the last action performed for a different range of frames/string. In the case you select **B**oth frames and strings (ALL), you are allowed only to translate and scale.

<Control Q> aborts the global-edit, except if used during a **C**hange operation (see below).

10.1 Defining a range

A range specification is entered as an alphanumeric field containing a list of sub-ranges of frames or strings. As generally the list is shorter than the space provided, you terminate it with a <control D>.

You specify a frame or string range as a list of one or more of the following items, separated by commas or blanks :

The keyword **ALL** to select all frames or strings (if this keyword appears anywhere in the list, it overrides all other selections)

The number of a frame or string

A sub-range of frames or strings, in the form nn**-**mm, to indicate all objects between nn and mm (included).

.cp3
Hence, to apply a change to frames 1,4,6,7,8,11,14,15,16 you may code a range list like :

Global Frames 1,4,6-8,11,14-16

If you specify any illegal frames or strings, they are ignored (a warning is issued).

10.2 Translating

A translation is specified separately in X and Y. The extent of the translation is specified in "outer" units (inches) and is equivalent to move the frame x- and y-origin (or the string x- and y-position) by the specified extent.

You are requested to enter the wished extent (a numeric field). The default is no translation (0.0).

The sort of dialogue appearing on the command line is like :

Global Frame range **Translate in x by** xx.x

(a further message about y appears on the same line, overwriting the one about x, when done)

Note that no checks are done while translating that the resulting position will fall in a legal place in the current page size and orientation !!

10.3 Scaling

A scaling is an omothetic transformation specified separately in X and Y. The extent of the transformation is specified as a scaling factor with respect to an origin in "outer" units (inches) and is equivalent to the following transformations :

$$x' = a(x-x_0)+x_0$$
$$l' = al$$

where x,x' is typically a position (x or y frame origin or string position), and l,l' a length (axis length) or size

(character size).

You are requested to enter the scaling factor a (a real number) and the origin x_0 separately in x and y. The default is no transformation (a=1.0, x_0=0.0).

You are also requested to specify with 1-letter codes whether the scaling applies to the frame or string **P**osition, to the **C**haracter sizes only, or to **B**oth of them.

.cp4
The sort of dialogue appearing on the command line is like :

Global **F**rame range **S**cale in x by a from x0

(a further message about y appears on the same line, overwriting the one about x, followed by one about positions, characters or both).

Note that no checks are done while scaling that the resulting position will fall in a legal place in the current page size and orientation !!

10.4 Changing parameters

With this option you are allowed to do things like "change the x-origin of frames 1,4,7,10 and 13 to 5.5 inches", or "change the tic size in all frames to 0.01 inches", or "change the colour in frames 7 and 9 to red", etc.

To save time, once you have selected a range, and specified as action **C**hange, you are repeatedly asked what you want to change and into which new value you want to change it. This way you may specify an entire list of parameters, and pass only once through the list of frames or strings, doing all the changes simultaneously.

You specify what to change as a four-letter code of the relevant frame characteristics. Such codes are given in the titles of the subsections in 7.1 to 7.6 and 8, as well as in Appendix B.

You specify the value into which to change any previous values for the selected parameter as a number or string (according to the type of field). If you do not want to change a parameter, but keep the default (e.g. you have typed the wrong

parameter name), just enter a carriage return or a <control Q>. The <control Q> is necessary if you have already started typing the new value and find it is wrong. In both cases a flashing "unchanged" message appears.

As you are typing sequences of what and into what, you may suddenly realize you have requested to change a parameter which should not be changed. In such case just retype the parameter name and a <control Q>.
The parameter will be reset as "not to be changed".

When you have given all your list of parameters, you enter either the keyword **GO** or a <control W>. This will start processing all frames or strings in range and doing the changes.

Note that global changes are not protected in the sense the new values you specify may be totally illegal, and the program will not complain (it will next time you try to edit such a field individually).

On the other hand the program is clever enough to know which parameter apply to frames and strings, and also to know to which type of frames a parameter applies : therefore if you asked to change **LEVC**ode (a contour parameters) in frames 1,7 and 9, and frame 7 is an histogram frame, the latter will remain intact.

Examples of command line dialogue in the cases quoted at the beginning of this section are :

Global Frame 1,4,7,10,13 Change XORI 5.5

Global Frame ALL Change TICS 0.01

Global Frame 7,9 Change COLO 1

.cp7
.foThe PLOTED Handbook (release 1)
Section 11

11. Terminating and plotting

11.1 Generating command files

When you have requested termination, you are asked (in the

current implementation) three questions about the (currently supported) plotting programs LABLER, PABLO and MCONT, namely :

whether you want to generate a command file for the indicated program (may reply **YES** or **NO**, which is the default)

what is the name of the command file to be created (this is an alphanumeric field in the form name:sc:cr, which is then parsed into name, security code and cartridge and redisplayed by the program). The name is saved as default name in the descriptor file (next time it will be remembered).

Typing a <control Q> anywhere in this phase, will jump to the "file saving" phase immediately (see 11.3 below).

The next question concern the identification of the plotting device to be indicated in the command files. This is currently the logical unit number of the device (see PABLO Handbook appendixes for local implementations). If you type a wrong number you will get a list of allowed logical units (note that "logical unit 1" to indicate your own terminal is not allowed, use the system logical unit number).

The program then takes a while to write the command files.

11.2 Plotting

The next question is whether you want the Plot Editor to schedule the plotting programs immediately. If you reply **YES**, the upper part of the screen is frozen, and the messages from the plotting programs scroll in the lower part.

You are not allowed to run a plotting program on the same terminal, while the Plot Editor is still running.

.cp5
If you have selected the HP 7550 plotter (locally unit 38), you will get extra messages about paper loading. Moreover the paper orientation will be set automatically by the Plot Editor.

.cp4
In the current implementation, the Labeller is run first, followed by other plotting programs. As the current release of

the Labeller does not unload paper at the end, this allows to use the plotter in auto-feed mode (if a single program follows the Labeller).

If you are instead running PABLO followed by MCONT, you should still have auto-feed disabled, and reload the paper manually between programs, when prompted by the Plot Editor.

11.3 File saving

At the end (after plotting if requested), you are asked to dispose of the edited plot descriptor file (the command files are already on disk and remain there until overwritten or deleted).

You may (see 2.3) **Q**uit, **F**ile or **S**ave to another file. At the end the name of the output file as used by the program is displayed (you can check whether it did put it where you intended it to go, in the case you did not supply enough information).

At this stage you may get an error message typically with one of the following FMP or Fortran codes :

6	506	a work file not found (ignore this message)
7	507	wrong security code (see below for recovery)
2	502	file already exists
32	532	wrong cartridge specified
33	533	cartridge is full
14	514	the cartridge directory is full
8	508	file is in use by another program and locked
15	515	file name has illegal syntax

In most cases, and if you persist getting errors, the easier way out is to issue a **S**ave command changing the output file name. A different action may be taken by the program in the case you get a -7 error while trying to **F**ile (most likely you forgot to specify the security code, or gave a wrong one). In such case :

Give the correct security code when prompted
Issue a **F**ile request again

Note that if you do not specify the cartridge in a file name, the file will go the default cartridge (the first one it finds, generally your private cartridge). The following cases are possible :

.cp5
a You entered the Plot Editor giving a full name:sc:cr of an existing file, then

a1 You will be able to **F**ile it OK
a2 You can always save it specifying a full namr other:os:oc
a3 If you save it to other, without specifying security code and cartridge, they will default to sc and cr
.cp2
a4 If you save it to other::oc, you actually force the security code to none
a5 If you save it to other:os:, you force it to the first cartridge found.

b You entered the Plot Editor giving a full namr, with the wrong security code or without it, you will get a -7 error when filing. See above for recovery.

c You entered the Plot Editor without specifying the cartridge

c1 when filing, the file on the original cartridge will be deleted, and you will unexpectedly find your file has been written on the first cartridge ! You have been warned !
c2 saves occur to the first cartridge if none specified

d You entered the Plot Editor with a full namr of a non-existing file, then you will always be able to file or save it.

e You entered the Plot Editor with a namr like name::cr for a non-existing file, the file will be created with no security code.

f You entered the Plot Editor without specifying the cartridge for a non-existing file, it will be filed or saved to the first cartridge

.cp7

12. In case of trouble

If you are getting a minor problem, remember that one of the following actions may in general put you back working, possibly losing just the last edits :

issue a <control Q>, this will "quit" what you are doing (the exact details depend on the part of the program you are in)

issuing a <control W> is often identical to a <control Q>, except that in some cases (see rest of documentation) the edits are saved

otherwise walk to another terminal, log in under your user name and issue a break request to your copy of the Plot Editor (the copy will be called **PLO**nn, where nn is the terminal number); the syntax of the break request is :

SYBR,PLOnn

.cp5

(if you break during startup, you terminate immediately; if you break while editing a frame/string, you jump back to command mode; if you break while in command mode, you jump to the "termination" segment; if you break while writing command files, such files are closed as they are, and you jump to the scheduling of the plotting programs)

A seemingly major problem may occur if you type too quickly, and you get an **S=nn COMMAND ?** system message. The fact is that most likely the keyboard will be locked by the program and you cannot type anything. In this case do a soft reset of the terminal. This will unlock the keyboard. You can then type a return to clear the system message, and then go on typing.

If you get error messages here, check the CAPS key (maybe the program wants upper case, and the reset changed into lower case).

It is also possible that the screen is altered. If the S=nn message remains on the screen, ignore it, it is harmless now. If the screen is shifted, scroll it back to the original position.

If you used an hard reset by mistake, you have cleared the screen. This way you have typically lost the top part of the screen (never mind !), but you will get the rest back when you do something.

If you are puzzled, close your edit with a <control W> or <control Q> and restart, and you should get the screen back all right. It is unlikely that your files have been damaged.

If by any chance the program crashes, issue a soft reset if the keyboard is locked. You may recover what you have done by renaming the work file **PLOT**nn (see 2.3)

.cp 7
.foThe PLOTED Handbook (release 1)
Section 13

13. Future improvements

Possible future improvements not yet implemented concern :

the use of the cursor to select a frame or string to be edited directly from the directory display;

the use of insert character and delete character when editing alphanumeric fields

the possibility of deleting entire ranges of frames or strings

the possibility of inserting "variable" parameters for file names and other fields, which can be replaced in one instance (to have true template descriptor files).

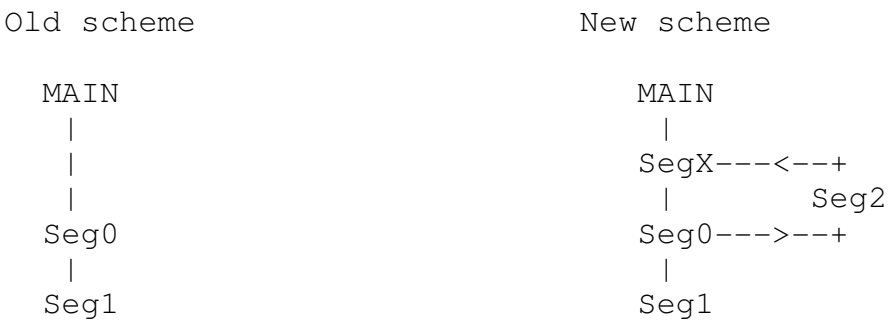
.pa
.foThe PLOTED Handbook (release 1)
Section 14

14. Programmers' notes

This section is in lieu of formal program sheet and file sheet for the plot descriptor files.

14.1 Segmentation scheme

The current implementation of the Plot Editor on the HP-1000 F under RTE 6-VM (with FMGR) is based on a segmented program. The linear segmentation scheme (with SEGLD calling a "dummy main" followed by a call to a subroutine relocated with the segment) used in the previous unofficial release has now been changed to insert the "global edit" segment. Therefore a short "switching segment" has been introduced :



The MAIN root takes care of startup, segment 0 is the command segment, and segment 1 is the termination segment (which takes care of command file writing, file disposition and stops). Segment 2 is the new global edit segment, and Segment X is just there to allow switching back to command segment.

14.2 Loading instructions

The source of the program is split among the following files (all of them on cartridge LC):

&PLTED	the PLOTED main
&PLTE0	the segment 0 (command segment)
&PLTE1	the segment 1 (termination segment)
&PLTE2	the segment 2 (global edit segment)
&PLTEX	the segment X (switching segment)
&PLELB	a library of common Fortran routines
PLTCOM	include file for the main common block
PLTCO1	include file used by segment mains only
PLTCO2	include file used by segment 2

A transfer file used to compile all source files, generate an indexed library out of &PLELB, and link everything using the linker commands contained in #PLTED is provided, and may be invoked as:

TR,*PLTED

In case of maintenance, avoid deleting all the relocatables, and make a copy of the transfer file which compiles only the segments on which you are working.

.cp5

14.3 Descriptor file layout

The schemes in Appendix C describe the structure of the plot descriptor files used by the Plot Editor. Such files implement a hierarchy made of the following

components :

a header record
a set of pointer records
a set of descriptor records

There are separate pointer and descriptor records for frames and strings. Essentially what happens is that :

The file is a type-2 binary direct access file, with record length 256 bytes.

The header record is the first record, and contains the information relevant to the whole plot (page size and orientation, command file names, date of last change, number of frames and strings). It also points to the first frame and string pointer records.

.cp2

Each pointer record of one type (frame or string), autonomously points to the next one (if any). It also contains information about up to 14 frames/strings, namely the frame/string identifier, and a pointer to the frame/string descriptor record

Strings have a single descriptor record, frames have at least one descriptor record, optionally followed by one extension record

The layout of the frame descriptor record varies somewhat depending on frame type, with an attempt to kept orderly equivalences between fields of similar usage. The last part of the primary record, and eventually the extension record contain a variable structure with the axis information.

All these records may occur in any order or position in the file, as they are always accessible via suitable pointers. While editing a lot of new records need to be created, which always occurs by appending at the end of a file. When a frame or string is deleted some of them are rendered inaccessible.

The "ordering" process which occurs during startup and at file disposition, recovers lost space and rearranges the pointers such that:

The header record comes first

followed immediately by all frame and string pointer records in sequence

followed by all frames and string descriptors in order as pointed, moreover the extension record (if any) follows immediately the frame descriptor of which it is an extension.

.cp50
.fi ploted3.hp

Appendix A
Error codes

This is a listing of all error messages produced by the Plot Editor, together with some explanation (when the message itself is not self-explanatory). Errors produced may be classified as :

Error code is negative	FMP errors, see HP quick reference
Error code < 100	FMP and other HP errors (see HP documents)
Error code 400-700	HP Fortran runtime errors (see table in Fortran manual appendix)
	all above codes are indicated below for simplicity as fmp
Error code 800-1000	errors detected by the program, see below

Here follows a complete listing of all error messages :

Error fmp

the following are normal FMP or runtime errors which may occur opening files, when improper names are used

opening filename
opening apurging work file filename
or a plot commandcreating work file filename
deleting work file
opening new file filename
deleting temp file filename
creating temporary file filename
deleting old copy of filename
renaming oldname as newname

these errors occur when descriptor file file these errors occur during the QUIT or FILE terminating phase (generally an improper file name or a wrong security code)

the following are normal FMP or runtime errors

	(i/o on descriptor files) which should occur only if the file is somewhat corrupted
reading from disk descriptor file	trouble reading
reading descriptor from disk	idem
reading extension record	idem
reading from input file	trouble when reordering
copying to work file	idem
initializing new file header desc file	trouble opening a new
initializing frame pointers	idem
initializing string pointers	idem
in updating work file descriptor	trouble writing edited
writing descriptor record	idem
writing header record file header	trouble updating desc
reading pointers from disk	self explanatory
Loading base segment errors, and	these are scheduling
reloading base segment loading segment n	should never occur
RP'ing program name errors for	these are scheduling
offing program name which maybe already engaged	the plotting programs,
in session number trying to get the terminal	this is an error in LOGLU session number and should never occur
	the following are format errors
illegal numeric value when entering data	this is a standard format error (49x), indicating a wrong format has been used
range specifier ignored when giving a frame or string range formatting output field routine REALX	this is a format error in global edit commissioning error,
Error nnn	internally-numbered errors

801 zero axis length in frame n — self explanatory (zero inches)

802 zero or negative value for log at zero axis in frame n — a log axis cannot start or negative user units

815 negative level with log scale on frame n — for contours only

890 schedule on same terminal not terminalallowed — cannot plot on same

891 plot LU locked : program not otherscheduled — plotter unit is in use to

900 There are no frames or strings was madeto edit (no strings — an edit or delete request with an empty file

900 There are no frames or strings to delete — and no frames)

901 There is nothing to edit in the it isdirectory — similar to the above, but the case of no strings or no

901 There is nothing to delete in the directory — frames

910 attempt to read nonexisting errordescriptor — commissioning

911 record or wrong type instead of errorpointers — commissioning

912 illegal pointer record indicated — commissioning error

915 file name cannot be parsed — the file name you specified cannot be parsed in name, security code and cartridge !

921 axis number out of range — commissioning error

922 axis extension area overflow — commissioning error

930 no room for more axes — the space for axes is limited !

950 editing deleted/inactive frames — self explanatoryor strings

960 no more frame/strings can be each !added — there is a maximum of 72

970 end of pointer area reached — commissioning error

980 impossible frame/string number — commissioning error

981 numeric value out of range — this is the typical case of a limit checking, the values input are outside the limits

982 some parameter changes inhibited — while global changing some parameters for frames of different types share the same area (sse appendix C) and cannot be changed at same time

.cp55
.foThe PLOTED Handbook (release 1)

Appendix B
Mnemonic codes for parameters

When doing a global edit change, use the following (four letter) mnemonic codes to indicate a parameter.

in order of appearance

parameters for frames

XORIgin	x-axis origin in inches
YORIgin	y-axis origin in inches
XLENgth	x-axis length in inches
YLENgth	y-axis length in inches
XLINlog	x-axis is linear (LIN) or logarithmic (LOG)
YLINlog	y-axis is linear (LIN) or logarithmic (LOG)
XSTArt	x-axis start in user units
YSTArt	y-axis start in user units
XEND	x-axis end in user units
YEND	y-axis end in user units
FRCOlour	colour for frame
CHSIze	character size in inches used by PABLO
CONNect	connect flag for plots
IDENtifier	function identifier
NLEVels	number of levels for contour plots
ERRBars	error bar flag for plots
NPTS	number of points for functions
REVMode	reverse mode flag for histograms
LEVCode	level code for contour plots
SYMSize	symbol size for plots
DASHsize	dash size (so called TRATLENGTH) for histograms
MINLevel	minimum level threshold for contour plots
MAXLevel	maximum level threshold for contour plots
SYMBol	symbol number or CC for colour column (coded as -1)
COLOur	colour for data in all plots
ULFLag	upper limit flag for plots and histograms
SCALe	scale-by-picsize for contour plots (Y or N)
ULCOlour	colour for upper limits in plots
PAR1	parameter 1 for functions (1-5)
PAR2	parameter 2 for functions (1-5)
PAR3	parameter 3 for functions (1-5)
PAR4	parameter 4 for functions (1-5)
PAR5	parameter 5 for functions (1-5)
FILEname	name of data file (for plots, histograms, contours)
ERRFflag	flag for file with/without errors (plots and histo)
HEADer	number of header lines (plots and histo)

```
LEVFile    name of level file for contour plots
FIRSt      first point to be plotted         (plots and histo)
LAST       last  point to be plotted         (plots and histo)
XCOLumn    column for X                      (plots and histo)
YCOLumn    column for Y                      (plots only     )
PERCflag   percentage flag for histograms
REGRflag   regression flag for plots
NBIN       number of bins for histograms
MINX       minimum x for regression in plot or for functions
HSTArt     start x for histograms
MAXX       maximum x for regression in plot or for functions
BINWidth   bin width for histograms

parameters for axes

TICSize    size for tic in axes
LBSIze     size for character in axis label (used by LABLER)

parameters for strings

XPOSition   x-position of string in inches
YPOSition   y-position of string in inches
COLOur      colour for string
SLANt       slant in degrees for strings
ORIEntation orientation in degrees for strings
CHSIze      character size for strings


                        in alphabetic order

BINWidth   bin width for histograms
CHSIze     character size in inches used by PABLO
CHSIze     character size for strings
COLOur     colour for data in all plots
COLOur     colour for string
CONNect    connect flag for plots
DASHsize   dash size (so called TRATLENGTH) for histograms
ERRBars    error bar flag for plots
ERRFflag   flag for file with/without errors (plots and histo)
FILEname   name of data file (for plots, histograms, contours)
FIRSt      first point to be plotted         (plots and histo)
FRCOlour   colour for frame
HEADer     number of header lines            (plots and histo)
HSTArt     start x for histograms
IDENtifier function identifier
LAST       last  point to be plotted         (plots and histo)
LBSIze     size for character in axis label (used by LABLER)
LEVCode    level code for contour plots
LEVFile    name of level file for contour plots
```

```
MAXLevel   maximum level threshold for contour plots
MAXX       maximum x for regression in plot or for functions
MINLevel   minimum level threshold for contour plots
MINX       minimum x for regression in plot or for functions
NBIN       number of bins for histograms
NLEVels    number of levels for contour plots
NPTS       number of points for functions
ORIEntation orientation in degrees for strings
PAR1       parameter 1 for functions  (1-5)
PAR2       parameter 2 for functions  (1-5)
PAR3       parameter 3 for functions  (1-5)
PAR4       parameter 4 for functions  (1-5)
PAR5       parameter 5 for functions  (1-5)
PERCflag   percentage flag for histograms
REGRflag   regression flag for plots
REVMode    reverse mode flag for histograms
SCALe      scale-by-picsize for contour plots (Y or N)
SLANt      slant in degrees for strings
SYMBol     symbol number or CC for colour column (coded as -1)
SYMSize    symbol size for plots
TICSize    size for tic in axes
ULCOlour   colour for upper limits in plots
ULFLag     upper limit flag for plots and histograms
XCOLumn    column for X                      (plots and histo)
XEND       x-axis end   in user units
XLENgth    x-axis length in inches
XLINlog    x-axis is linear (LIN) or logarithmic (LOG)
XORIgin    x-axis origin in inches
XPOSition  x-position of string in inches
XSTArt     x-axis start in user units
YCOLumn    column for Y                      (plots only     )
YEND       y-axis end   in user units
YLENgth    y-axis length in inches
YLINlog    y-axis is linear (LIN) or logarithmic (LOG)
YORIgin    y-axis origin in inches
YPOSition  y-position of string in inches
YSTArt     y-axis start in user units

.cp55
.foThe PLOTED Handbook (release 1)
Appendix  C
```

Appendix C
Tables of descriptor file layout

File structure

```
**GRAPH
$X+150
$BOX1500,1510,6
$RECORD1;Header record
$BLA$ENDREC
$RECORD2;First frame  pointer record
$STREC$$$ENREC n
$BLA$ENDREC
$RECORD3;First string pointer record
$STREC**$ENREC m
$BLA$ENDREC
$RECORD4;
$BLA$ENDREC
$PUSH$GRAY1500,75,6$POP$Y+75
$RECORDp;(Frame) descriptor record$ENREC q
$BLA$ENDREC
$PUSH$GRAY1500,75,6$POP$Y+75
$RECORDq;optional extension record$ENREC 0
$BLA$ENDREC
$PUSH$GRAY1500,75,6$POP$Y+75
$RECORDn;frame pointer record
$STREC$$$ENREC..
$BLA$ENDREC
$PUSH$GRAY1500,75,6$POP$Y+75
$RECORDm;string pointer record
$STREC**$ENREC..
$BLA$ENDREC
$PUSH$GRAY1500,75,6$POP$Y+75
**ALPHA
```

The descriptor records occur where pointed by pointers contained in therelevant pointer record (see below). Frame descriptors (only) can havean optional extension record (pointed by an internal pointer). So faronly one extension is supported.

Pointer records are chained, in the sense an internal pointer points tothe next pointer record of the same type.

In line of principle all records can occur in any position after theheader; in practice new records are created appending at the end offile, and everything is reordered when filing, so that all pointerrecords come first (frames then strings) and all descriptor records comenext (frames than strings) in sequence number order (also each record isimmediately followed by its extension, if any).

.cp50

Structure of header record

Here and in all diagrams below, primary addresses are in bytes (startingfrom 1). 2-byte fields (when a secondary address is indicated) areINTEGER*2, and 4-byte fields are REAL*4. All other fields (including 2-byte fields with no secondary address) are CHARACTER*n.

```
**GRAPH
$MACRO111
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND111
$MACRO112
$Y+80$RULE100,1
$Y+40$RULE100,1
$Y+40$RULE100,1
$Y+40$RULE100,1
$Y+40$RULE100,1
$Y+40$RULE100,1
$MACEND112
$X+150
$SMAL
$BOX1500,1885,6
$BYTE  1;
$WORD  1;
$BORDER80;NFRAME  number of frames$ENDBOX80;
$BYTE  3;
$WORD  2;
$BORDER80;NSTRING number of strings$ENDBOX80;
$BYTE  5;
$WORD   ;
$BORDER80;SIZE    size of page (A3,A4)$ENDBOX80;
$BYTE  7;
$WORD  4;
$BORDER80;ORIENTATION (coded as 1=L,2=P)$ENDBOX80;
$BYTE  9;
$WORD  5;
$BORDER80;NBLOCKS number of blocks in file$ENDBOX80;
```

```
$BYTE 11;
$WORD  6;
$BORDER80;
FRAME_PTR1  pointer to first frame pointer record$ENDBOX80;
$BYTE 13;
$WORD  7;
..need to reset page number here !!
$BORDER80;
STRING_PTR1 pointer to first frame pointer record$ENDBOX80;
$BYTE 15;
$FNAME;
$BORDER315;
LAB_FILE name of Labler command file$ENDBOX315;
$BYTE 27;
$FNAME;
$BORDER315;
PAB_FILE name of PABLO  command file$ENDBOX315;
$BYTE 39;
$FNAME;
$BORDER315;
MCO_FILE name of MCONT  command file$ENDBOX315;
$BYTE 51;
$TIMES;
$BORDER280;
Date of last modification (JYEAR, ITIME) EXEC 11
format$ENDBOX280;
$PUSH$GRAYEND$POP$Y+75
$ELITE
**ALPHA
.pn 44
.cp 50
```

Structure of pointer records

```
**GRAPH
$MACRO113
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
```

```
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND113
$X+150
$SMAL
$BOX1500,1800,6
$BYTE  1;
$WORD   ;
$BORDER80;
Type-of-pointer keyword$ENDBOX80;
$POINTARRAY
$BYTE  3;
$WORD   ;
$BORDER80;
POINT_VALUE pointer to descriptor record or 0$ENDBOX80;
$BYTE  5;
$IDENT
IDENTIFIER 16-char frame or string identifier$ENDBOX415;
$POINTARRAY
$BYTE 21;
$WORD   ;
$BORDER80;
 next (second) element of pointer array$ENDBOX80;
$BYTE 23;
$IDENT
$ENDBOX415;
$PUSH$GRAY1500,155,7$POP$Y+155
$POINTARRAY
$BYTE237;
$WORD   ;
$BORDER80;
 last (14th ) element of pointer array$ENDBOX80;
$BYTE239;
$IDENT
$ENDBOX415;
$BYTE255;
.. need to reset page number here
$WORD   ;
$BORDER80;
NEXT_PTR_REC pointer to next pointer record or zero$ENDBOX80;
$Y+75
$ELITE
**ALPHA
```

Pointer records share the same structure for frames and strings, namelya keyword, an array of 14 pointer-structures, and a final pointer to thenext record.

The kewyord is equal to ASCII $$ for frames and ** for strings.

A pointer to descriptor record equal to zero, means there is nodescriptor for such frame/string (does not exist or has been deleted).

The pointer to the next pointer record is zero only for the last pointerrecord of the entire chain.

.pn 45
.cp 50

Structure of frame descriptor records

```
**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,2420,6
$BYTE  1;
$WORD  1;
$BORDER65;
EXTEND_PTR pointer to extension record or zero$ENDBOX65;
$BYTE  3;
$WORD  2;
$BORDER65;
DISP_FLAG  disposition flag$ENDBOX65;
$BYTE  4;
$REAL  2;
$BORDER115;
XORIGIN frame origin in inches$ENDBOX115;
$BYTE  9;
$REAL  3;
$BORDER115;
YORIGIN frame origin in inches$ENDBOX115;
$BYTE 13;
$REAL  4;
$BORDER115;
XLENGTH axis length  in inches$ENDBOX115;
$BYTE 17;
$REAL  5;
$BORDER115;
YLENGTH axis length  in inches$ENDBOX115;
$BYTE 21;
$WORD   ;
$BORDER65;
```

```
XLINLOG axis is lin (IN) or log (OG)$ENDBOX65;
$BYTE 23;
$WORD   ;
$BORDER65;
YLINLOG axis is lin (IN) or log (OG)$ENDBOX65;
$BYTE 25;
$REAL  7;
$BORDER115;
XSTART axis start in user units$ENDBOX115;
$BYTE 29;
$REAL  8;
$BORDER115;
YSTART axis start in user units$ENDBOX115;
$BYTE 33;
$REAL  9;
.. need to reset page number here !!!
$BORDER115;
XEND   axis end   in user units$ENDBOX115;
$BYTE 37;
$REAL 10;
$BORDER115;
YEND   axis end   in user units$ENDBOX115;
$BYTE 41;
$WORD 21;
$BORDER65;
FRCOLOUR frame colour$ENDBOX65;
$BYTE 43;
$WORD   ;
$GRAYWORD
$BYTE 45;
$REAL 12;
$BORDER115;
CHSIZE character size in inches$ENDBOX115;
$BYTE 49;
$BORDER400;
this area different for each plot type$ENDBOX400;
$BYTE 89;
$BORDER40;
Plot type (coded as P,F,H,C)$ENDBOX40;
$BYTE 90;
$GRAYBYTE
$BYTE 91;
$BORDER250;
this area different for each plot type$ENDBOX250;
$BYTE101;
$WORD 51;
$BORDER65;
NAXES number of axes in this frame$ENDBOX65;
$BYTE103;
```

```
$BORDER200;
This area used for axis descriptors
$ENDBOX200;
$Y-160
$BYTE   ;$XY+300,+30
eventually continued in extension record
$Y+275
$ELITE
**ALPHA
.pn 46
.cp 50
```

Plot frame dependent part

```
**GRAPH
$MACRO111
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND111
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,1585,6
$BYTE 49;
$WORD 25;
$BORDER65;
CONNECT  Pablo connect flag$ENDBOX65;
$BYTE 51;
$WORD 26;
$BORDER65;
ERRORBAR Error bar flag$ENDBOX65;
$BYTE 53;
$REAL 14;
$BORDER115;
SYMSIZE  Symbol size$ENDBOX115;
```

```
$BYTE 57;
$WORD 29;
$BORDER65;
SYMBNO   Symbol number (-1 if CC)$ENDBOX65;
$BYTE 59;
$WORD 30;
$BORDER65;
COLOUR   Colour value or column$ENDBOX65;
$BYTE 61;
$WORD 31;
$BORDER65;
ULFLAG   Upper limit flag (0-1)$ENDBOX65;
$BYTE 63;
$WORD 32;
$BORDER65;
ULCOLOR  Upper limit colour$ENDBOX65;
.. reset page number here !!!
$BYTE 65;
$FNAME;
$BORDER315;
DAT_FILE name of data file$ENDBOX315;
$BYTE 77;
$WORD 39;
$BORDER65;
IFERR    flag for errors (0,1)$ENDBOX65;
$BYTE 79;
$WORD 40;
$BORDER65;
HDRLINES number of header lines$ENDBOX65;
$BYTE 81;
$WORD 41;
$BORDER65;
PBEGIN   first point to be plotted$ENDBOX65;
$BYTE 83;
$WORD 42;
$BORDER65;
PLAST    last  point to be plotted$ENDBOX65;
$BYTE 85;
$WORD 43;
$BORDER65;
XCOLUMN  column for x$ENDBOX65;
$BYTE 87;
$WORD 45;
$BORDER65;
YCOLUMN  column for y$ENDBOX65;
$BYTE 89;
$BORDER40;
Plot type (always P)$ENDBOX40;
$BYTE 90;
```

```
$GRAYBYTE
$BYTE 91;
$WORD 47;
$BORDER65;
REGRESS regression flag$ENDBOX65;
$BYTE 93;
$REAL 24;
$BORDER115;
REGMINX minimum x for regression plot$ENDBOX115;
$BYTE 97;
$REAL 25;
$BORDER115;
REGMAXX maximum x for regression plot$ENDBOX115;
$Y+275
$ELITE
**ALPHA
.pn 47
.cp 50
```

Histogram frame dependent part

```
**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,1585,6
$BYTE 49;
$GRAYWORD
$BYTE 51;
$WORD 26;
$BORDER65;
REVMODE  Reverse mode flag$ENDBOX65;
$BYTE 53;
$REAL 14;
$BORDER115;
TRAT_LEN dashed line length$ENDBOX115;
$BYTE 57;
$GRAYWORD
$BYTE 59;
$WORD 30;
$BORDER65;
COLOUR   Colour value or column$ENDBOX65;
$BYTE 61;
$WORD 31;
$BORDER65;
```

```
ULFLAG   Upper limit flag (0-1)$ENDBOX65;
$BYTE 63;
$GRAYWORD
$BYTE 65;
$FNAME;
$BORDER315;
DAT_FILE name of data file$ENDBOX315;
$BYTE 77;
$WORD 39;
$BORDER65;
IFERR    flag for errors (0,1)$ENDBOX65;
$BYTE 79;
$WORD 40;
$BORDER65;
HDRLINES number of header lines$ENDBOX65;
$BYTE 81;
$WORD 41;
$BORDER65;
PBEGIN   first point to be plotted$ENDBOX65;
$BYTE 83;
$WORD 42;
$BORDER65;
PLAST    last  point to be plotted$ENDBOX65;
.. rese tpage number here !!!
$BYTE 85;
$WORD 43;
$BORDER65;
XCOLUMN  column for x$ENDBOX65;
$BYTE 87;
$WORD 45;
$BORDER65;
PERC_OR_ABS percentage flag (0,1)$ENDBOX65;
$BYTE 89;
$BORDER40;
Plot type (always H)$ENDBOX40;
$BYTE 90;
$GRAYBYTE
$BYTE 91;
$WORD 47;
$BORDER65;
NBIN number of bins in histogram$ENDBOX65;
$BYTE 93;
$REAL 24;
$BORDER115;
START_HISTO histogram start$ENDBOX115;
$BYTE 97;
$REAL 25;
$BORDER115;
BIN_WIDTH   bin width$ENDBOX115;
```

```
$Y+275
$ELITE
**ALPHA
.pn 48
.cp 50
```

Function frame dependent part

```
**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,1600,6
$BYTE 49;
$WORD 25;
$BORDER65;
FUNCTION_ID  function identifier (1-20)$ENDBOX65;
$BYTE 51;
$WORD 26;
$BORDER65;
NPTS number of points in function$ENDBOX65;
$BYTE 53;
$GRAYWORD
$BYTE   ;
$GRAYWORD
$BYTE 57;
$GRAYWORD
$BYTE 59;
$WORD 30;
$BORDER65;
COLOUR   Colour value or column$ENDBOX65;
$BYTE 61;
$REAL 16;
$BORDER115;
PARAMS(1) parameter array start$ENDBOX115;
$BYTE 65;
$REAL 17;
$BORDER115;
PARAMS(2) $ENDBOX115;
$BYTE 69;
$REAL 18;
$BORDER115;
PARAMS(3) $ENDBOX115;
$BYTE 73;
$REAL 19;
```

```
$BORDER115;
PARAMS(4) $ENDBOX115;
$BYTE 77;
$REAL 20;
$BORDER115;
PARAMS(5) parameter array end$ENDBOX115;
$BYTE 81;
$GRAYWORD
$BYTE 83;
$GRAYWORD
.. rese tpage number here !!!
$BYTE 85;
$GRAYWORD
$BYTE 87;
$GRAYWORD
$BYTE 89;
$BORDER40;
Plot type (always F)$ENDBOX40;
$BYTE 90;
$GRAYBYTE
$BYTE 91;
$GRAYWORD
$BYTE 93;
$REAL 24;
$BORDER115;
REGMINX minimum x for function plot$ENDBOX115;
$BYTE 97;
$REAL 25;
$BORDER115;
REGMAXX maximum x for function plot$ENDBOX115;
$Y+275
$ELITE
**ALPHA
.pn 49
.cp 50
```

Contour frame dependent part

```
**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,1555,6
$BYTE 49;
$WORD 25;
```

```
$BORDER65;
NLEV     number of levels  $ENDBOX65;
$BYTE 51;
$WORD 26;
$BORDER65;
LEVCODE  level code (1-4)$ENDBOX65;
$BYTE 53;
$WORD 27;
$BORDER65;
MINLEV   minimum level$ENDBOX65;
$BYTE 55;
$WORD 28;
$BORDER65;
MAXLEV   maximum level$ENDBOX65;
$BYTE 57;
$GRAYWORD
$BYTE 59;
$WORD 30;
$BORDER65;
COLOUR   Colour value or column$ENDBOX65;
$BYTE 61;
$WORD 31;
$BORDER65;
SCALE_BY_PICSIZE flag (0-1)$ENDBOX65;
$BYTE 63;
$GRAYWORD
$BYTE 65;
$FNAME;
$BORDER315;
DAT_FILE name of data  file$ENDBOX315;
$BYTE 77;
$FNAME;
$BORDER315;
LEV_FILE name of level file$ENDBOX315;
$BYTE 89;
$BORDER40;
Plot type (always C)$ENDBOX40;
$BYTE 90;
$GRAYBYTE
$BYTE 91;
.. reset page number here !!!
$GRAYWORD
$BYTE 93;
$GRAYWORD
$BYTE   ;
$GRAYWORD
$BYTE 97;
$GRAYWORD
$BYTE   ;
```

$GRAYWORD
$Y+275
$ELITE
**ALPHA
.pn 50
.cp 50

Structure of extension record for frame descriptors

**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$X+150
$SMAL
$BOX1500,665,6
$BYTE 1;
$BORDER600;
continuation of axis descriptors$ENDBOX600;
$BYTE255;
$WORD128;
$BORDER65;
FURTHER_EXT pointer to further extension always 0$ENDBOX65;
$Y+275
$ELITE
**ALPHA

Axis descriptor structure

Note that addresses here are relative to the beginning of the axisdescriptor. The first axis descriptor starts at byte 103 of the framedescriptor record, the second one just after the end of the first and soon. As they are variable-length, 16-bit and 32-bit boundary alignment isnot guaranteed, therefore word and doubleword addresses are not given.

**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
$MACRO115
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1

```
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND115
$X+150
$SMAL
$BOX1500,1040,6
$BYTE  1;
$WORD   ;
$BORDER65;
AXFLAG  axis type flag (1-6)$ENDBOX65;
$BYTE  3;
.. need to reset page number here too !!
$REAL   ;
$BORDER115;
TICSIZE tic size inches$ENDBOX115;
$BYTE  7;
$REAL   ;
$BORDER115;
STEP    tic step user units$ENDBOX115;
$BYTE 11;
$REAL   ;
$BORDER115;
LBSIZE  character size inches$ENDBOX115;
$BYTE 15;
$FORMAT;
$BORDER165;
FORMAT (6 chars)$ENDBOX165;
$BYTE 21;
$WORD   ;
$BORDER65;
LBLENGTH label length l (in chars)$ENDBOX65;
$BYTE 23;
$BORDER400;
text of label (l chars)$ENDBOX400;
$Y+275
$ELITE
**ALPHA
.pn 51
.cp 50
                structure of string descriptor record

**GRAPH
$MACRO114
$Y+40$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$Y+25$RULE100,1
$MACEND114
```

```
$X+150
$SMAL
$BOX1500,1435,6
$BYTE  1;
$WORD  1;
$BORDER65;
EXTEND_PTR always zero$ENDBOX65;
$BYTE  3;
$WORD  2;
$BORDER65;
DISP_FLAG  disposition flag$ENDBOX65;
$BYTE  5;
$REAL  2;
$BORDER115;
XPOSITN string origin in inches$ENDBOX115;
$BYTE  9;
$REAL  3;
$BORDER115;
YPOSITN string origin in inches$ENDBOX115;
$BYTE 13;
$WORD  7;
$BORDER65;
SCOLOUR string colour$ENDBOX65;
$BYTE 15;
$WORD  8;
$BORDER65;
SLENGTH string length n (in chars)$ENDBOX65;
$BYTE 17;
$WORD  9;
$BORDER65;
SLANT   text slant  in degrees$ENDBOX65;
$BYTE 19;
$WORD 10;
$BORDER65;
SORIENT orientation in degrees$ENDBOX65;
$BYTE 21;
$REAL  6;
$BORDER115;
SCSIZE  character size inches$ENDBOX115;
$BYTE 25;
$BORDER700;
text area n chars (up to 232)$ENDBOX700;
$Y+275
$ELITE
**ALPHA
.. need last reset of page number here !!
```

The string descriptor record has <u>never</u> extension records