Laser-Scan Ltd.

FPP Plotters

User Guide

Science Park, Milton Road, Cambridge, England CB4 4FY tel: +44 (0) 1223 420414

Document "Plotters User Guide"		Category "User"
Document Issue 3.5	Clarke Brunt	16-December-1992
Document Issue 3.6	Clarke Brunt	26-July-1993
Document Issue 3.7	Clarke Brunt	13-September-1993
Document Issue 3.8	Clarke Brunt	18-January-1994
Document Issue 3.9	Clarke Brunt	10-March-1994
Document Issue 4.0	Clarke Brunt	18-July-1994
Document Issue 4.1	Clarke Brunt	22-November-1994
Document Issue 4.2	Clarke Brunt	25-September-1996
Document Issue 4.3	Clarke Brunt	10-December-1997

CONTENTS

CHAPTER 1

Introduction

This document describes the interfaces between Laser-Scan's vector plotting program, FPP, and the various plotters which it is capable of driving. Features of FPP not specific to a particular plotter are described in the FPP Reference Manual.

FPP is available in several versions, each driving one or two different plotters. The executable image names are FPP followed by something descriptive of the particular plotter used by this version, thus FPPCAL907 produces Calcomp 907 output suitable for various models of Calcomp plotters.

The FPP command WORKSTATION is used to pass on certain information to the device dependent parts of the program. Its use is described below for FPP versions in which it is relevant.

CHAPTER 2

AEG flatbed plotter

The image FPPAEG.EXE produces output suitable for AEG (and similar) photo plotters.

2.1 Mode of operation

The program is not designed to drive the plotter directly, but to produce a disc file of plot instructions. This is an ASCII file that can be edited using a text editor. It can be converted to ICL magnetic tape format by the program ASC2ICL.

It also produces a log of the plot giving the number of plot instructions generated, and the size of the plot.

2.2 File specifications

The file specification for the file of plot instructions is derived as follows :

- If the logical name LSL$AEG is not set up, the file specification is 'PLOTFILE.AEG', in the current directory.
- If the logical name LSL$AEG is set up, then any part of its logical translation that is missing as a valid file specification, is taken from the default 'PLOTFILE.AEG'.

The file specification for the log file is derived in a similar manner, using the logical name LSL$AEG_INFO and the default file name 'AEG_INFO.DAT'

Note that LSL$AEG_INFO can be assigned to the line printer (usually device LP:) or to the users terminal (TT:).

2.3 Hardware

The AEG family of plotters have a disc of "slugs" that are used for drawing lines and flashing as symbols. The information on which slug to be used for drawing a line or flashing as a symbol is extracted from the FRT being used by

FPP. Reference should be made to the FRT users guide for information on setting up a suitable FRT for use with such devices.

The plotters have the facility of tangential control of the light head, but do not have any possibility of magnifying the projected image, so any magnification defined in the FRT is ignored in FPPAEG.

The AEG version of FPP supports the use of hardware circles and curves, and hardware patterned lines. These are enabled by flag bits in the FRT (see FRT users guide). Any pattern which cannot be plotted by the hardware will still be plotted by software, even if the flag bit is set. The patterns supported are just major and minor pecks (no symbols). The only pattern flag bit used is the 'phase restore at each vertex' bit - all others (in particular scaling of the pattern) are ignored.

2.4 **Workstation Command**

FPPAEG recognises three different plotter types; which one is being used is determined by any WORKSTATION command that is given. The three plotters are:

1. The AEG 3012
   This has a maximum plotting area of 1.5m x 1.2m, and is selected by the command:
   WORKSTATION 3012

2. The ARISTO 306
   This has a maximum plotting area of 1.8m x 1.7m, and is selected by the command:
   WORKSTATION 306

3. The KONGSBURG 1825
   This has a maximum plotting area of 2.5m x 1.78m, and is selected by the command:
   WORKSTATION 1825

The default is to plot assuming a AEG 3012 plotter.

NOTE

FPPAEG has no knowledge of where the origin of the plotter has been set. If it has not been set to the extreme lower left corner the actual plotting area will be reduced from the figures given above and the plotters may hit the end stops when plots larger than this reduced plotting area are being produced.

For a similar reason it is probably desirable to use the FPP command

- CENTRE

to make FPP plot in the lower left corner of the film.

CHAPTER 3

Benson pen plotters

The image FPPBEN1600.EXE produces output suitable for Benson 1600 series pen plotters. Image FPPBEN is in use at some sites. This uses an older plotter data format. FPPHCBS may be used with shareable images HCBSBEN or HCBSBEN1600 to produce output equivalent to FPPBEN and FPPBEN1600.

3.1 **Paper size**

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Benson, the sheet is assumed to be 1.189x0.841 metres (A0 size). The initial pen position (the origin) should be at the lower right of the plotter when standing in front of it (minimum x, minimum y). The older FPPBEN program requires the initial pen position to be at the lower left of the plotter when standing in front of it (minimum x, maximum y). It is normally advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

The default behaviour of the program is to assume that the plotter is roll fed, and to advance to clean paper after a plot. If sheet feed is used, the WORKSTATION 101 command should be given before the map is plotted. After the plot, the pen will the return to the origin. The default behaviour is WORKSTATION 100 (102 for FPPBEN).

The full specification of the workstation type given in a WORKSTATION command is: 100 - origin at min x, min y (right hand side of plotter), return there after plot. To this add 1 to cause the paper to advance after a plot, and 2 if the initial origin is at max x, min y (left hand side of plotter). The latter option is useful on any models of plotter which leave the pen at the left hand side when initialised.

3.2 **Plotter interface**

The plotter will be connected to the computer by a serial line. The baud rate is normally set to 9600 since there is probably nothing to be gained by higher rates. The only other non-default characteristics which are normally set are /NOTYPEAHEAD/NOBROADCAST.

3.3 **Producing plots**

The program sends its output to logical name LSL$BENSON. This may be assigned either directly to the plotter or to a disc file, thus for example

```
$ Define LSL$BENSON TXA0:
```

This will send output directly to the plotter on serial line TXA0:, or

```
$ Define LSL$BENSON BENSON.PLT
```

This will send output to the disc file BENSON.PLT in the current directory. The file may subsequently be plotted by

```
$ COPY BENSON.PLT TXA0: (assuming plotter is on TXA0:)
```

If desired, the plotter may be set as a spooled device with an associated print queue. Another common approach to ensure that only one user attempts to access the plotter simultaneously is to run all plot jobs in a batch queue with a job limit of 1. Note that output from the older FPPBEN program cannot be sent to a file.

CHAPTER 4

Benson electrostatic plotters

The Benson 3000 series electrostatic plotters may be used by Laser-Scan programs FPPBEN3000 (package PLOTTING), and ROVERBEN3000 (package TVES).

4.1 Colour table

The Benson 3000 series colour electrostatic plotters differ from normal pen plotters in that the colours produced by a particular colour index (from an FRT file, for instance) may be defined by the user. In this respect, the plotter resembles a colour display terminal. The Laser-Scan programs initially define the colours by reading a disc file on logical name LSL$BENSON_COLOUR. The format of this file is the same as that used for the VAXstation and Tektronix displays, and the same file may be used, although it will probably be found that changes are necessary to get the desired colours on the Benson. Also white should not normally be used on the Benson, as it will be invisible against the white paper background. FPP will use the colours defined by the file on logical name LSL$BENSON_COLOUR for the plot, while ROVER always reads in another table subsequently (LSL$LOOKUP:ROVER5.DAT by default, or others by use of the PLANES command, or explicitly by use of the LUT command). An example colour table file is given in Appendix A.

4.2 Area filling

The maximum number of points allowed in a filled area is 8192 by default, but may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for hatched fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

4.3 Paper size and pixel size

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Benson, the sheet is assumed by default to be 1.189x0.841 metres (A0 size). It is normally advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

ROVER always fills the whole sheet, which is assumed to consist of 1448x1024 pixels for the purpose of displaying a raster image. Since this may produce a plot which is too large, a means is provided to change the sheet size, and the number of pixels.

Logical name LSL$BENSON_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, thus to give the same effect as the defaults, the file would have the single line:
1.189 0.841 1448 1024
The pixel numbers are not relevant to FPP, but control the number of pixels assumed to cover the area for ROVER and LITES2. LITES2 sets the number of pixels and their size itself, but will never draw more than the specified number of pixels across a plot. The number may be altered within LITES2 using the PLOT PIXELS command.

The maximum number of pixels which may be specified in either direction is 16384. Programs may impose other limits of their own e.g. 10000 for ROVER. The limit for LITES2 depends on the format of the raster data. For DTI files, the limit is 32768 for byte data, 16384 for word data, and 8192 for long data. There is no limit for LSI or LSR format files. All of these limits (except ROVER's 10000) are proportional to the maximum number of points in filled areas and hence may be increased using logical name LSL$FILL_POINTSMAX. The values given here are for the default of 8192.

4.4 **Plotting mode**

The Benson 3000 series plotter by default plots in transparent mode, which means that where parts of the plot overlap, the inks will be combined, generally giving darker colours. It also has the ability to plot in opaque mode, where anything drawn replaces what is there already (underlying colours do not show through).

Several mechanisms are provided to select the drawing mode, to satisfy the needs of the programs FPP, ROVER, and LITES2.

- By default FPPBEN3000 plots in transparent mode. The command ESCAPE 1 will cause opaque mode to be used for subsequent plotting, while ESCAPE 2 will revert to transparent mode. As an alternative, giving the command WORKSTATION 1000 before beginning a plot will select opaque mode initially.

- By default, ROVERBEN3000 draws the vector overlay using opaque mode, so that the colours replace any underlying colour. In order to allow the overlay to be transparent, define the logical name LSL$BENSON_TRANSPARENT (as anything).

- By default, LITES2 will draw any data in overlays whose base plane is greater than zero in opaque mode. If it is required to draw everything in transparent mode, then either define the logical name LSL$BENSON_TRANSPARENT (as anything), or give the command PLOT TYPE 1000 (before PLOT OPEN). Any number other than 1000 will revert to the original behaviour.

As an alternative to the above mechanisms, the PLOT ESCAPE command may be used, in the same way as for FPPBEN3000. The command PLOT ESCAPE 1 will switch to opaque plotting, while PLOT ESCAPE 2 will revert to transparent plotting.

4.5 **Plotter interface**

The plotter will be connected to the computer by a parallel interface, probably with device name PLA0:, using the Benson supplied driver PLDRIVER.

4.6 **Producing plots**

The plotting programs send their output to Fortran unit number 7, which results in the file FOR007.DAT being created in the current directory. Alternatively logical name FOR007 may be defined to be a file name, for example
$ Define FOR007 ROVER.BIF
This will send output to the disc file ROVER.BIF in the current directory.

Once a disc file has been created, it is sent to the Benson using the Benson supplied program SENDBIF. This must be set up as a foreign command, e.g.
$ SENDBIF :== LSLEXE:SENDBIF
and sends its output to logical name SYS$PLOT, which must be defined to be the plotter device, e.g.
$ Define SYS$PLOT PLA0:
The command to plot a file is then e.g.
$ SENDBIF ROVER
which will plot the file ROVER.BIF, or
$ SENDBIF FPP.PLT
which will plot the file FPP.PLT.

SENDBIF counts up the number of buffers sent to the plotter. Each buffer is 4096 bytes, or 8 disc blocks. If using SENDBIF at a hardcopy terminal or in a batch job, it is best to assign SYS$OUTPUT to NL: to avoid the incrementing block count wasting paper.

4.7 **Use of ROVER**

ROVER was originally designed for interactive use on colour graphics screens. When used on a hardcopy plotter no attempt should be made to clear parts of the picture, or to use the editing facilities. The aim should be to produce the desired picture using the minimum number of commands.

CHAPTER 5

Calcomp electrostatic plotters

The Calcomp 5800 series electrostatic plotters may be used by Laser-Scan programs FPPCAL5800 (package PLOTTING), ROVERCAL5800 (package TVES), and LITES2MOTIF (package LITES2).

5.1 Colour table

The Calcomp 5800 series colour electrostatic plotters differ from normal pen plotters in that the colours produced by a particular colour index (from an FRT file, for instance) may be defined by the user. In this respect, the plotter resembles a colour display terminal. The Laser-Scan programs initially define the colours by reading a disc file on logical name LSL$CALCOMP_COLOUR. The format of this file is the same as that used for the VAXstation and Tektronix displays, and the same file may be used, although it will probably be found that changes are necessary to get the desired colours on the Calcomp. Also white should not normally be used on the Calcomp, as it will be invisible against the white paper background. FPP and LITES2 will use the colours defined by the file on logical name LSL$CALCOMP_COLOUR for the plot, while ROVER always reads in another table subsequently (LSL$LOOKUP:ROVER5.DAT by default, or others by use of the PLANES command, or explicitly by use of the LUT command). An example colour table file is given in Appendix A.

5.2 Area filling

The Calcomp's hardware fill is used for solid areas with up to 4095 points, unless logical name LSL$CALCOMP_HW_FILL is defined as 0, in which case it is never used. Otherwise software filling consisting of close hatch lines is used, but this cannot guarantee to render all colours identically to the hardware fill. The maximum number of points allowed in a filled area is 8192 by default, but may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for software fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

5.3 **Paper size and pixel size**

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Calcomp, the sheet is assumed by default to be 1.189x0.841 metres (A0 size). It is normally advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

ROVER always fills the whole sheet, which is assumed to consist of 1448x1024 pixels for the purpose of displaying a raster image. Since this may produce a plot which is too large, a means is provided to change the sheet size, and the number of pixels.

Logical name LSL$CALCOMP_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, thus to give the same effect as the defaults, the file would have the single line:
1.189 0.841 1448 1024
The pixel numbers are not relevant to FPP, but control the number of pixels assumed to cover the area for ROVER and LITES2. LITES2 sets the number of pixels and their size itself, but will never draw more than the specified number of pixels across a plot. The number may be altered within LITES2 using the PLOT PIXELS command.

The maximum number of pixels which may be specified in either direction is 16384. Programs may impose other limits of their own e.g. 10000 for ROVER. The limit for LITES2 depends on the format of the raster data. For DTI files, the limit is 32768 for byte data, 16384 for word data, and 8192 for long data. There is no limit for LSI or LSR format files. All of these limits (except ROVER's 10000) are proportional to the maximum number of points in filled areas and hence may be increased using logical name LSL$FILL_POINTSMAX. The values given here are for the default of 8192.

5.4 **Calcomp specific plotting**

By default, FPPCAL5800 assumes that the plotter is set to a step size of 2032 steps/inch (800 steps/cm). If it is not, then plots will be the wrong size. This particularly applies to the small format plotters, whose step size is equal to the resolution (200 or 300 steps/inch). These plotters may however be set to scale the plot to the correct size. The step size expected by FPPCAL5800 may be altered by defining logical name LSL$CALCOMP_STEPS to be the number of steps per inch. For example, by defining LSL$CALCOMP_STEPS to be 400, and setting the plotter to match (perhaps by the command STEPSIZE=400 at the local terminal) the amount of plot data will be reduced without degrading the plot, since the resolution of the plotter is 400 dots per inch in any case.

Logical name LSL$CALCOMP_SCALE has a similar function. It may be defined to be an integer in the range 0-32. 0 is the default and has no effect (the same as 1). Other numbers cause the plotter moves to be scaled down by the given number, but instructs the plotter to scale them back up again. This can again reduce the amount of data which needs to be transmitted to the plotter. Too large a number will cause the plotted lines to be inaccurate, but on a 400 dpi plotter, numbers up to 5 should not produce any visible difference (since 400 is approximately 5 times smaller than the default step size of 2032 steps/inch).

The Calcomp 5800 by default plots in transparent mode, which means that where parts of the plot overlap, the inks will be combined, generally giving darker colours. It also has the ability to put plot data in up to 255 'levels', each of which, in this implementation, is plotted in order and in an opaque mode (underlying colours do not show through).

Several mechanisms are provided to select the drawing mode, to satisfy the needs of the programs FPP, ROVER, and LITES2.

- By default FPPCAL5800 plots in transparent mode. The command ESCAPE 1 will cause the next higher opaque level to be used for subsequent plotting, while ESCAPE 2 will revert to putting data in the bottom transparent level. ESCAPE 3 begins a new transparent layer above all the layers used so far. Thus, to draw using several opaque levels, repeat the ESCAPE 1 command before making selections and plotting each level of data.

- By default, ROVERCAL5800 draws the vector overlay using opaque mode, so that the colours replace any underlying colour. In order to allow the overlay to be transparent, define the logical name LSL$CALCOMP_TRANSPARENT (as anything).

- By default, LITES2 will draw any data in overlays whose base plane is greater than zero in opaque mode. If it is required to draw everything in transparent mode, then either define the logical name LSL$CALCOMP_TRANSPARENT (as anything), or give the command PLOT TYPE 1000 (before PLOT OPEN). Any number other than 1000 will revert to the original behaviour.

  As an alternative to the above mechanisms, the PLOT ESCAPE command may be used, in the same way as for FPPCAL5800. The command PLOT ESCAPE 1 will cause the next higher opaque level to be used for subsequent plotting, while PLOT ESCAPE 2 will revert to putting data in the bottom transparent level. PLOT ESCAPE 3 begins a new transparent layer above all the layers used so far. Thus, to draw using several opaque levels, repeat the PLOT ESCAPE 1 command before making selections and drawing each level of data.

A Calcomp colour electrostatic plotter usually has four coloured toners, yellow, magenta, cyan, and black. Any dot on the paper is either toned, or left blank - the toners cannot be applied at less than full intensity. This means that colours which are not fully saturated can only be achieved using a screen pattern with some dots on and some dots off. The particular screen pattern used is chosen by the Calcomp library. These patterns are used for filled areas. For lines, a dot pattern is laid out along the length of the line, which can give the appearance of a line pattern for some colours. For solid filled areas, the default is to use 4x4 dot patterns to represent colours, so 16 gradations of each toner are available. Logical name LSL$CALCOMP_DOTSIZE may be assigned to be an integer (2, 4, 8, or 16) to specify the size of dot pattern to use instead. For instance, a setting of 8 will give 8x8 = 64 gradations of each toner, at the expense of using larger dots.

Lines up to 16 nibs wide (i.e. up to 1mm on a 400dpi plotter) are drawn using hardware thick lines. Thicker lines switch to using software thick lines, where each line segment is drawn as a filled area. The hardware thick lines always have ends which are squared off parallel to the sides of the paper, whereas the

software lines have the ability to use different join and cap styles. Logical name LSL$CALCOMP_HWTHK may be set to be an integer, denoting the maximum width in nibs for which hardware lines will be used, thus 0 always uses software lines, while a large number will always use hardware lines. The default is 16.

For software thick lines, the setting of the hardware field in the FRT is used to indicate what style of line caps and joins are required. The CAP style is a number in the range 0-2. 0 means butt caps (square ends), 1 means round caps, and 2 means extended caps (square but extended by half the line thickness. The JOIN style is also a number in the range 0-2. 0 means mitre joins (the sides of the thick lines extended to meet), 1 means round joins, and 2 means bevel joins (each segment has a square end, but the missing triangle is filled in). Mitre joins are changed to bevel if the resulting spike would be 'too long' - this is taken to be when there is an acute corner in the line with an angle of less than about 11 degrees. Closed line features will use the join style at the start/end point, and will not have a cap at all. The FRT hardware field is built up as 10*JOIN + CAP, so for instance 21 would mean JOIN = 2 = bevel, CAP = 1 = round. If omitted, the hardware field defaults to 0, meaning mitre joins and butt caps. Remember that the hardware field in the SCT entry (if present) will override that in the FRT.

A side effect of using software thick lines is that the area colour is used. The colours of hardware thick lines are not as versatile, and sometimes have difficulty in producing the same colour as for areas of the same colour index. This sometimes leads to unexpected colour changes in solid area fill when the maximum number of points for hardware fill is reached and close hatching is used instead. Using software thick lines throughout (LSL$CALCOMP_HWTHK set to 0) may well avoid this problem. Very thin lines may produce unexpected effects if the area colours are used, such as if a 1 nib horizontal or vertical line completely misses the toned dots in the area pattern. To avoid this, either give all lines a sensible thickness in the FRT (rather than 0), or define LSL$CALCOMP_HWTHK to a number such as 3, so that the thinnest lines still use hardware lines.

Note that the use of software thick lines is likely to increase the plot file size by a factor of around 4, and so it should only be used if high quality thick lines are essential.

Since hardware thick lines are specified to the plotter in terms of its resolution, they will only be the correct thickness if the plot is drawn on a plotter of the same resolution as intended by the software. The default is 400 dots per inch, but logical name LSL$CALCOMP_DPI is provided to specify an alternative resolution in dots per inch.

5.5 **Plotter interface**

The plotter will be connected to the computer by a serial line. The baud rate should be set as high as possible on plotter and computer for fastest data transfer (e.g. 19200 baud). The plotter will usually have a terminal attached. If the terminal line has the TYPEAHEAD characteristic, it will be possible to log on to the VAX on this terminal, though it is not expected that users will normally do so.

When powered up, after initialising, the terminal will be communicating with the VAX. Hitting carriage return will produce a login prompt. Use CTRL/P to switch to local communication with the plotter (the terminal should normally be left in

this mode). The local Calcomp prompt is "$" (without a space after it), as opposed to "$ " from the VAX. Give the command EXIT to return to VAX communication. See the Calcomp manuals for the various commands which may be issued locally to the plotter.

5.6 **Producing plots**

The plotting programs send their output to logical name LSL$CAL5800. This may be assigned either directly to the plotter or to a disc file, thus for example
$ Define LSL$CAL5800 TXA0:
This will send output directly to the plotter on serial line TXA0:, or
$ Define LSL$CAL5800 CALCOMP.PLT
This will send output to the disc file CALCOMP.PLT in the current directory. The file may subsequently be plotted by
$ COPY CALCOMP.PLT TXA0: (assuming plotter is on TXA0:)
or
$ TYPE CALCOMP.PLT if you are logged on to the Calcomp local terminal.

5.7 **Use of ROVER**

ROVER was originally designed for interactive use on colour graphics screens. When used on a hardcopy plotter no attempt should be made to clear parts of the picture, or to use the editing facilities. The aim should be to produce the desired picture using the minimum number of commands.

CHAPTER 6

Calcomp pen plotters

The FPP image for Calcomp pen plotters is FPPCAL907.EXE. This image behaves identically to FPPHCBS (see section on this FPP version) using shareable image HCBSCAL907 on logical name LSL$HCBSSHARE. Other versions of FPP producing Calcomp output for plotters set up differently to the Laser-Scan standard are in use at some sites. These are FPPCALAIDU and FPPCALGEOG.

6.1 Mode of operation

The program produces Calcomp 907 output on logical name LSL$CALCOMP. The is usually assigned directly to the plotter serial line, but may under some circumstances be assigned to a file for transmission to the plotter later. For example

```
$ Define LSL$CALCOMP TXA1:
or
$ Define LSL$CALCOMP CALCOMP.DAT
```

The plotter serial line must be set to the correct baud rate for the plotter, but otherwise the default characteristics are satisfactory.

6.2 Paper size

FPP always has the concept of a sheet, in which it positions the plot according to the commands given by the user. On the Calcomp, the sheet is assumed to be 1.189x0.841 metres (A0 size). The initial pen position (the origin) should be at the lower left of the plotting area (which is usually at the right hand side of the plotter). It is advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

The default behaviour of the program is to assume that the plotter is sheet fed, and to return to the origin after a plot. If roll feed is used, the WORKSTATION 100 command should be given before the map is plotted. After the plot, the pen will advance to a new origin 2cm beyond the current plot. (The default behaviour is WORKSTATION 101.)

The full specification of the workstation type given in a WORKSTATION command is: 100 - origin at min x, min y (right hand side of plotter), return there after plot. To this add 1 to cause the paper to advance after a plot, and 2 if the initial origin is at max x, min y (left hand side of plotter). The latter

option is useful on any models of plotter which leave the pen at the left hand side when initialised.

6.3 **Plotter interface**

There are several options which may be set on the Calcomp control panel affecting the computer interface. The settings required in the setup menu for the 1040 range of plotters are described - for other models the equivalent will be required.

```
Serial
Host baud 9600 (or whatever)
Mode 1=960, 0=PCI 0
Muting no
Cksum N (but see below)
Isochronous N
EOM char 03
Direct ctrl N
XON/XOFF Y (but see below)
Term baud 9600 (or whatever)
Duplex 1=half
Sync codes 1
Sync code value 002
Parity 0=no
Bits/char 8
Stop bits 1
Clock int
```

Two of the settings are optional:

1. Software handshaking. After each buffer of data sent, the computer prompts with ">". The plotter responds with "0" (data was OK) or "1" (data was bad - retransmit). In addition to data validation, this provides a method of synchronization if either the Calccomp controller or the computer does not use xon/xoff flow control. If enabled, then Calcomp output cannot be sent to a disc file, since the response messages are not sent. The plotter will use software handshake if XON/XOFF is set to N.

2. Checksum. The penultimate byte of each record sent to the plotter is a checksum of the preceding ones. Set Cksum Y for the plotter to expect checksum.

Old versions of FPP always used both software handshaking and checksum. The new version (since 18 July 1988) uses neither by default, but allow either to be used optionally. Laser-Scan prefer that the plotter be set not to expect software handshaking or checksum, but if this is not possible (for instance because other software must also drive the plotter) then either can be enabled.

Some plotters will not allow drawing outside of a 'sheet' area, even in successive plots unless an 'index command' is received. FPP does not send index commands by default, but this may be enabled.

All options are enabled by defining logical names to be "1". The absence of the logical name, or its being defined as anything other than "1" will disable the option.

```
$ Define LSL$CALCOMP_HANDSHAKE 1   enables software handshaking
$ Define LSL$CALCOMP_CHECKSUM 1    enables checksum
$ Define LSL$CALCOMP_NEWPLT 1      enables index commands
```

6.4 **Special versions**

FPPCALAIDU is used at AIDU. Always uses software handshake and never uses checksum. The SYNC character is 02 (as usual) but is doubled. The EOM character is 13 (CR). This version is equivalent to using FPPHCBS with shareable image HCBSCALAIDU on logical name LSL$HCBSSHARE.

FPPCALGEOG is used at Cambridge University Geography Department. It is linked with a different HCBS to the others, and is not affected by the above logical names.

CHAPTER 7

HP-GL plotters

HP-GL is the Hewlett-Packard Graphics Language. It is understood by plotters manufactured by Hewlett-Packard (HP), and by plotters produced by various other manufacturers which claim compatibility with the HP-GL language.

There have been two versions of HP-GL. The original was designed for handling only pen plotters. The later version (HP-GL/2) is a superset of HP-GL, but has extensions suitable for handling some raster plotters.

The FPP image for HP-GL plotters is FPPHP.EXE.

The FPP image for HP-GL/2 plotters is FPPHPGL2.EXE. Support for HP-GL/2 in general is still under development by Laser-Scan. This interim version has been produced as a testbed to drive the NOVAjet A0 colour inkjet. It has not been tested on other HP-GL/2 plotters.

7.1 **Mode of operation**

The programs produce HP-GL output on logical name LSL$HP. The is usually assigned directly to the plotter serial line, but may under some circumstances be assigned to a file for transmission to the plotter later. For example

```
$ Define LSL$HP TXA1:
or
$ Define LSL$HP HP.DAT
```

Both programs can attempt to read back the paper size from the plotter. They do this by sending an 'OH;' command and then attempting to read back the reply on logical name LSL$HPINPUT. If you are sending output direct to the plotter, and the plotter supports the 'OH;' command, then assign LSL$HPINPUT to be the plotter line (the same as LSL$HP). For example

```
$ Define LSL$HPINPUT LSL$HP
```

The HP version requires this input. Therefore, if the plotter does not support the 'OH;' command, or you are sending output to a file, then LSL$HPINPUT must be assigned to a file containing a single line with the numbers that 'OH;' would have output. For example

```
$Define LSL$HPINPUT HPSIZE.DAT
```

where HPSIZE.DAT contains the single line

0 0 10900 7650

The 4 numbers are xmin ymin xmax ymax in plotter units which are assumed to be 0.025 mm.

The HP-GL/2 version does not require the OH; information. It will attempt to read back the size from an online plotter (i.e. both LSL$HP and LSL$HPINPUT are a terminal line) but if plotting is not online, then there is no need to define LSL$HPINPUT at all. Note that it is not usually necessary to set the paper size correctly. It is only needed if either auto-scaling, or FPP/LITES2 commands are used to locate the plot somewhere other than the bottom left of the paper, are used. See below for the HP-GL/2 method of determining the size.

The plotter serial line must be set to the correct baud rate for the plotter, but otherwise the default characteristics are usually satisfactory.

7.2 **HP-GL/2 Differences**

The data format used by the HP-GL/2 driver includes 8-bit character values in the range 191-254, in addition to printable characters in the range 32-126. It must be sent down a communication channel capable of transmitting these values unchanged.

The colours used for the picture are defined using a text file on logical name LSL$HP_COLOUR (if this is set up). See Appendix A for the format of this file. Colours 0 to 255 may be defined.

The HP-GL/2 driver is able to draw raster images when used via LITES2. These image are drawn as lines using a pen with width equal to the pixel size. Logical name LSL$HP_BACKGROUND may be defined as 0 or 1. The default is 1, which means to draw all colours in images, including the background colour 0. If the logical name is defined as 0, then only non-zero pixels are drawn.

There is no control available for whether graphics are opaque or transparent. As far as is known, on raster plotters all drawing is opaque, so anything drawn will obscure things drawn previously. Obviously, on a pen plotter, marks already made will remain.

LSL$HP_NOMLW may be defined to be a real number - the nominal line width in millimetres. The default is 0.1. The nominal line width is used for any features whose line width is specified as 0 in the FRT and also as the width and separation of the hatch lines used to render any solid filled area which does not use hardware fill.

LSL$HP_MAXPATH may be defined to be an integer. Solid filled areas with more points than this limit will be rendered by close hatch lines (see LSL$HP_NOMLW) rather than using hardware filling.

The maximum number of points allowed in a filled area is 8192 by default, but may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for hatched fill) may cut the polygon is 100 by default, but may be altered by defining

logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

LSL$HP_HW_FILL may be defined to be 0 or 1. By default, solid filled areas with less than LSL$HP_MAXPATH points will use hardware fill, while those with more will use close hatching. If the logical name is defined as 0, then all filling will use close hatching (thickness and spacing equal to LSL$HP_NOMLW).

Logical name LSL$HP_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, thus to give the same effect as the defaults, the file would have the single line:

1.189 0.841 1448 1024

Although all four numbers must be given, either pair may be set to zero, in which case they are ignored. The pixel numbers are not relevant to FPP, but control the number of pixels assumed to cover the area for LITES2 raster images. LITES2 is able to set the number of pixels and their size itself, but will never draw more than the specified number of pixels across a plot. The number may be altered within LITES2 using the PLOT PIXELS command, so there is little need to bother with the pixel numbers in the LSL$HP_SIZE file.

The maximum number of pixels which may be specified in either direction for LITES2 depends on the format of the raster data. For DTI files, the limit is 32768 for byte data, 16384 for word data, and 8192 for long data. There is no limit for LSI or LSR format files. All of these limits are proportional to the maximum number of points in filled areas and hence may be increased using logical name LSL$FILL_POINTSMAX. The values given here are for the default of 8192.

If an LSL$HP_SIZE file is specified, then the sizes obtained from it are sent to the plotter in a 'PS' command, and will therefore affect the amount of paper used on roll-feed plotters. According to the HP-GL/2 Reference Guide, a 'PS' with width greater than length will result in the plot being rotated so that the X-axis is across the width, rather than the length as usual. To counteract this, an 'RO90' command is added to return the axes to their normal position.

Logical name LSL$HP_ROTATE may be defined to be an integer from 0-3. It defines a number of right-angles by which to rotate the plot anti-clockwise. This amount is added to the 'RO90' mentioned above.

Logical name LSL$HP_SCALE may be defined to be an integer, either 1, 10, or 100. 1 is the default, and does not do anything. The other values cause the coordinate values sent to the plotter to be scaled up by this amount, but an 'SC' command is inserted at the start of the plot to scale the plot back to the correct size. The purpose of this is to improve the resolution of the coordinates if the final output device is capable of steps smaller than 40 per mm (the default device coordinates in HP-GL). There will be some increase in the amount of plot data generated if scale factors of 10 or 100 are used.

7.3 **HP-GL Commands Used**

The following table indicates which HP-GL instructions are used in the two drivers:

Instruction	HPGL?	HPGL2?	Comments
IN	x	x	Initialise
OH	x	x	Output hard-clip limits, see above
SPn	x	x	Select pen
PU	x	x	Pen up
PD	x		Pen down
PAx y...;	x		Plot absolute, multiple coordinates
PE		x	Polyline encoded
BP1,"Title"		x	Begin plot
PS		x	Plot size, see LSL$HP_SIZE above
RO		x	Rotate, see LSL$HP_ROTATE above
SC		x	Scale, see LSL$HP_SCALE above
NP256		x	Number of pens, always 256
PG		x	Advance full page
PM		x	Polygon mode, uses PM, PM1, and PM2
FP		x	Fill polygon, used after PM2
PC		x	Pen colour assignment
PW		x	Pen width, mm, 3 decimal places
LA		x	Line attributes, sets ends and joins
TR		x	Transparency, always off, TR0

Notes:

The HPGL driver includes the following instructions at the start to set the communications

<esc>.Y<esc>.I81;;17:<esc>.N;19:

and the following at the end

<esc>.Z

The 'PE' (polyline encoded) instructions used by the HP-GL/2 driver use base 64 relative coordinates only, with no fractional data. The only flags used are ':' (select pen) and '<' (pen up). This data format uses character values 191-254 in addition to printable characters in the range 63-126.

CHAPTER 8

IFF Primitive files and Raster output

The image FPPPRIM.EXE produces an IFF graphics primitive file for further processing to a raster plot format. An IFF primitive file is one in which all the feature representation lookup has already been done, leaving straight lines and solid fill areas as the only entities in the file.

The setting of the hardware field in the FRT is used to indicate in the primitive file what style of line caps and joins are required. The CAP style is a number in the range 0-2. 0 means butt caps (square ends), 1 means round caps, and 2 means extended caps (square but extended by half the line thickness. The JOIN style is also a number in the range 0-2. 0 means miter joins (the sides of the thick lines extended to meet), 1 means round joins, and 2 means bevel joins (each segment has a square end, but the missing triangle is filled in). Miter joins are changed to bevel if the resulting spike would be 'too long' - this is taken to be when there is an acute corner in the line with an angle of less than about 11 degrees. Closed line features will use the join style at the start/end point, and will not have a cap at all. The FRT hardware field is built up as 10*JOIN + CAP, so for instance 21 would mean JOIN = 2 = bevel, CAP = 1 = round. If omitted, the hardware field defaults to 0, meaning miter joins and butt caps. Remember that the hardware field in the SCT entry (if present) will override that in the FRT.

The image PRIM2RASTER.EXE takes an IFF graphics primitive file produced by FPPPRIM, and does vector to raster conversion, outputting in various raster formats (see PRIM2RASTER User Guide).

8.1 **Sheet size**

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. For FPPPRIM the sheet is assumed to be 1.12 x 2.47 metres. It is normally advisable to use FPP plot position 0 (POSITION 0 command) so as to position the plot in the bottom left corner of the sheet (otherwise PRIM2RASTER will rasterise the blank areas).

8.2 **Producing a Primitive file**

The spelling 'miter' is used, rather than 'mitre' so as to agree with X-Windows and PostScript documentation.

The FPPPRIM program sends its output to an IFF file of the same device, directory, and name as the first IFF input file, but with file extension ".PRIM", eg LSL$IF:name.PRIM.

8.3 **Primitive file format**

A primitive IFF file, while following IFF format, has some conventions intended specially for the PRIM2RASTER program. The MH entry contains the text 'PRIM' at the start, followed by a version number - PRIM2RASTER will not accept a file without this, or with the wrong version number. Features all contain a TH entry, whose value is the line thickness in microns (with 0 meaning minimum thickness). The first word of the FS entry (the feature code) will be 100 for solid area features, or the number from the FRT hardware field (built up from cap and join styles as above) for lines. This will be one of 0, 1, 2 ,10, 11, 12, 20, 21, 22. If the file is to be viewed in LITES2 or plotted using another FPP, an FRT with these codes must be provided. The third word of the FS entry (normally the process code) contains the colour index to be used for the feature.

CHAPTER 9

KERN GP1 flatbed plotter

Image FPPKERNGP1.EXE produces output suitable for a KERN GP1 flatbed plotter.

9.1 Mode of operation

The program produces output on logical name LSL$KERNGP1. This logical name is usually assigned directly to the plotter serial line, but may under some circumstances be assigned to a file for transmission to the plotter later. For example

```
$ Define LSL$KERNGP1 TXA1:
or
$ Define LSL$KERNGP1 KERN.DAT
```

If the logical name LSL$KERNGP1 is not set up, then the file KERNGP1.DAT is produced.

9.2 Plotting area

When operating on line, FPPKERNGP1 will only plot in the area above, and to the right of the currently set origin. It assumes that the size of paper available extends from the origin to the upper right corner of the currently set window. This is the area used by the POSITION command, so, for example, the command POSITION 0 will plot as close as possible to the currently set origin, while the command POSITION 8 will plot in the top right corner of the currently set window.

9.3 Number of pens

The Kern GP1 is usually supplied with a two pen drawing head. Features that have a colour index greater than 2 will be drawn with either pen 1 or 2 (depending on whether the index is odd or even).

To specify a different number of pens (for example if the GP1 is fitted with a four pen drawing head, or if mono plots are required) the logical name LSL$KERNGP1_PENS can be set to the number of pens available.
Features that have a colour index greater than this number will be drawn with

the corresponding pen in the available range; e.g. if LSL$KERNGP1_PENS has been set to 4, a feature with colour index 5 would be drawn with pen 1 while a feature with colour index 12 would be drawn with pen 4.

```
$ Define LSL$KERNGP1_PENS 4   allows four different pens to be used
$ Define LSL$KERNGP1_PENS 1   forces all plotting to be drawn with pen 1
```

9.4 **Messages**

This program is linked with a library supplied by KERN. This library is called GP1LVF, and on occasions it produces its own error messages. These messages are preceded by the name "GP1LVF", and are generally self-explanatory.

This library requires that the plotter is set up to use "Xon / Xoff" protocol. If the plotter has been set to use the "ACK / NACK" protocol then FPP will fail with the GP1LVF error message

GP1LVF - Please set GP1 to "XON-XOFF" protocoll (transmission-mode 2)

after the first IFF command has been given. In this case it will be necessary to set the GP1 to transmission mode 2 by selecting Test 0 on the GP1 hand controller. (See KERN GP1 Manual for more details)

CHAPTER 10

Kongsberg flatbed plotter

Image FPPKONG.EXE produces output suitable for a Kongsberg flatbed plotter.

10.1 Mode of operation

The program is not designed to drive the plotter directly, but to produce a magnetic tape of plot instructions. The program writes these instructions to a nine track magnetic tape on logical name LSL$KONGSBERGTAPE.

This logical name **must** be assigned to a tape drive. The tape will be written at 1600 bpi if the drive allows the density to be set by software.
$ Define LSL$KONGSBERGTAPE MSA0:

The tape must previously have been mounted as a foreign device, for example
$ MOUNT/FOREIGN MSA0:

The tape is always rewound on initialisation, i.e. after the first IFF command, and after any subsequent WORKSTATION commands.

The tape is written in ASCII by default. If logical name LSL$KONGSBERG_EBCDIC is defined (as anything) then it is written in EBCDIC. In this mode, a linefeed is also added at the end of each line. e.g.
$ Define LSL$KONGSBERG_EBCDIC true

10.2 Paper size

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Kongsberg, the sheet is assumed to be 1.6x1.2 metres.

CHAPTER 11

X-Windows display (MOTIF)

The image FPPMOTIF.EXE produces output on a workstation running X-Windows.

11.1 **Display**

Although the output from this version of FPP may in principle be directed to any workstation or terminal running X-Windows, Laser-Scan only support interaction via a DEC VMS workstation running DECwindows and the Motif window manager (mwm). FPPMOTIF may be run from a terminal emulating window on the display screen, or from a separate terminal, possibly on another network node, in which case an appropriate (DCL) SET DISPLAY command will be required to direct the output to the workstation.

By default, on an 8 plane system, colours 0-127 may be used for the plot (0 is the background).

On a 4 plane system, colours 0-7 may be used for the plot.

On a monochrome system, only colours 0 and 1 are available. In all cases, features specified with colours out of range will be drawn in colour 1.

The number of colours used may be modified to some extent by the use of the WORKSTATION command.
WORKSTATION 7000 is relevant to both 4 and 8 plane systems. The number of colours used is then allowed to reach the maximum for the particular number of planes (16 and 256 respectively), the defaults being determined from the number that appear to be free, often around 240 on the 8 plane display. If logical name LSL$DECW_MAX_COLOUR is defined to be an integer, then this number of colours will be used instead. This could be used for example to use the full 256 colours of an 8 plane display.

When attempting to use the full number of planes on a display, then there is inevitably contention for the available colours, as the window manager and other applications will have already allocated some cells, for example for window borders and banners. The default action is to use as many cells as possible in the default colour map. This means that the colours corresponding to the cells which were already allocated will be displayed incorrectly. The colours are used in such a way that the incorrect colours are the high-numbered ones, and hopefully the effect is not noticeable, or at least isn't serious. If you really must have more, or even all, colours displayed correctly, then logical name LSL$DECW_MAX_COLOUR may be set all the way up to 256.

Two other logical names control whether the number of free cells in the default colour map is considered adequate when LSL$DECW_MAX_COLOUR is not defined. These are LSL$DECW_MIN_COLOUR - the minimum number of cells which must be free (defaults to half the number requested, for example 128 when requesting all the cells on an 8 plane display), and LSL$DECW_FREE_COLOUR - the number of cells to leave free for subsequent use, for example for menus (defaults to 0).

If there are not enough free cells in the default colour map, then a private colour map must be created. Whenever this is done, it is likely that the colours of existing (e.g. terminal emulator) windows will be altered, and you might for example have to click in the banner of a window in order to get the window manager to install that window's colour map. The particular action needed to do this will depend on your window manager settings.

If logical name LSL$DECW_INVERT is defined (as anything) then the colours will be used in reverse order (this is sometimes done automatically in order to minimise contention over colours). The only obvious use of this is on a monochrome display with a read-only colour map - it may be used to switch round black and white between background and foreground.

Whenever a display window is created, some diagnostic information is printed regarding allocation of a colour map, for example:

```
...allocating 8 planes
...trying default colormap
...using colours 0-241
...except 236-239
...first allocated colour 0, plane offset 0
...colours used in inverted order
```

The interpretation of this information is: Command WORKSTATION 7000 has been used, so an attempt is being made to allocate all 8 planes, or 256 colours; The default colormap is tried first; There are enough free cells here to be able to use colours 0-235 and 240-241, so colours 236-239 and 242-255 will be displayed incorrectly; The 'first allocated colour' and 'plane offset' information gives which X-windows cells are being used - this need not concern the user; The colours are being used in inverted order to ensure that only high numbered ones are displayed incorrectly (again this need not concern the user).

The setting of the hardware field in the FRT is used to indicate what style of line caps and joins are required. The CAP style is a number in the range 0-2. 0 means butt caps (square ends), 1 means round caps, and 2 means extended caps (square but extended by half the line thickness. The JOIN style is also a number in the range 0-2. 0 means miter joins (the sides of the thick lines extended to meet), 1 means round joins, and 2 means bevel joins (each segment has a square end, but the missing triangle is filled in). Miter joins are changed to bevel if the resulting spike would be 'too long' - this is taken to be when there is an acute corner in the line with an angle of less than about 11 degrees. Closed features will use the join style at the start/end point, and will not have a cap at all. The FRT hardware field is built up as 10*JOIN + CAP, so for instance 21 would mean JOIN = 2 = bevel, CAP = 1 = round. If omitted, the hardware field defaults to 0, meaning miter joins and butt caps.

The spelling 'miter' is used, rather than 'mitre' so as to agree with X-Windows and PostScript documentation.

Remember that the hardware field in the SCT entry (if present) will override that in the FRT.

FPPMOTIF provides support for hardware text, if the option is enabled and the appropriate bit is set in the flags entry of the FRT file. By default, X-Windows text is used. X-Windows text is not very versatile - it can only be drawn horizontal and at a limited range of sizes. The intention is that Display PostScript is used, but not all X-Servers support it. To attempt to use Display PostScript, define the logical name LSL$DECW_DPS as 1. If you do this, but the X-Server does not support Display PostScript, then a message to that effect will be produced. Logical name LSL$DPS_FONTLIST is used to define which PostScript fonts are to be used. It need not be specified if hardware text is not used. The logical name should be defined to point to a file (default filespec LSL$FRT:*.PSFONTLIST) containing directives defining the fonts. The format of this file is documented in the FRT User Guide (MAPPING package). Previous releases of FPPMOTIF allowed the logical name to be defined as a search list, specifying the fonts. This still works, but the new mechanism is preferred - it allows several facilities not previously available. Note that attempts to italicise a font using a negative font number in the FRT does not work for hardware text. If Display PostScript is used, then the character shapes and widths in the TRI file are ignored - a TRI file must still be supplied, but it need not contain any fonts.

11.2 **Colour table**

The colours used for the picture are defined using a text file on logical name LSL$DECW_COLOUR. See Appendix A for the format of this file.

11.3 **Control options**

Several logical names are available to control the way in which drawing on the X-Windows workstation takes place.

Hardware area filling facilities are used for solid fill areas by default. If logical name LSL$DECW_HW_FILL is defined as 0, or if there are too many points for hardware fill (the limit is inquired from the X-server but is normally at least 4092), close horizontal hatch lines generated by the software are used. Hardware filling may well be faster than the software fill. Filled areas are limited to 8192 points by default. The limit may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for software fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

If logical name LSL$DECW_SYNC is defined (as anything), then all X-Windows operations will be completed immediately, rather than being buffered up. This can be very inefficient, and is only intended as a debugging aid.

Logical name LSL$DECW_MODE and LSL$DECW_UPDATE are used to control the method used to keep the screen picture up to date. The current picture is kept in backing store (called a 'PIXMAP') and is repainted from here when the window is uncovered, or de-iconised etc. By default, vectors are drawn only into the

PIXMAP, which is painted onto the screen after 1000 vectors have been drawn (in order that the user sees signs of progress) and also when drawing has been completed. This default state corresponds to LSL$DECW_MODE being defined as 0, and LSL$DECW_UPDATE as 1000. By defining LSL$DECW_UPDATE to be a different number, the screen updates can be made more or less frequent. If they are too frequent then drawing will take longer, while if they are too infrequent, the user if left wondering what is happening. If LSL$DECW_MODE is defined as 1, then all drawing is performed twice - once into the PIXMAP and once onto the screen. The picture is now always up to date and LSL$DECW_UPDATE is irrelevant, but at the expense of everything being drawn twice. The intention is that the settings of these logical names be adjusted to produce the fastest, or most pleasing result.

LSL$COMPOSITE_CHARACTERS may be defined to be an integer in the range 0-1. It specifies whether to attempt to interpret composite characters in text strings. 1 enables the mechanism, 0 (or not defined) disables it. Composite characters are made up by superimposing two or more existing characters in a font. They are defined in the AFM file, specified in the PSFONTLIST file. All characters in the font (including the composites) have a name, such as Zcaron. If enabled, then a composite character is plotted by including its name, surrounded by braces (e.g. {Zcaron}) in a text string. An opening brace must then be represented by two opening braces. Any character may be included by this mechanism, even non-composite ones. This is of little benefit for ordinary characters (e.g. {A} is the same as A), but may be of benefit for a character encoded in the range 128-255 as an alternative to the '$' escape mechanism. If the name inside the braces is not recognised, or an attempt is made to plot a composite character when TRI text fonts are in use, then the first letter of the name will be plotted instead (e.g. {Zcaron} plots as Z). If the mechanism is disabled, then braces are treated as normal characters.

CHAPTER 12

Precision Image C448 electrostatic plotter

The Precision Image C448 electrostatic plotter may be used by Laser-Scan programs FPPPI448 (package PLOTTING), and ROVERPI448 (package TVES).

12.1 Colour table

The Precision Image C448 colour electrostatic plotter differs from normal pen plotters in that the colours produced by a particular colour index (from an FRT file, for instance) may be defined by the user. In this respect, the plotter resembles a colour display terminal. The Laser-Scan programs initially define the colours by reading a disc file on logical name LSL$PI448_COLOUR. The format of this file is the same as that used for the VAXstation and Tektronix displays, and the same file may be used, although it will probably be found that changes are necessary to get the desired colours on the Precision Image plotter. Also white should not normally be used (except perhaps as an overlay colour in ROVER), as it will be invisible against the white paper background. FPP will use the colours defined by the file on logical name LSL$PI448_COLOUR for the plot, while ROVER always reads in another table subsequently (LSL$LOOKUP:ROVER5.DAT by default, or others by use of the PLANES command, or explicitly by use of the LUT command). An example colour table file is given in Appendix A.

12.2 Paper size and pixel size

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Precision Image plotter, the sheet is assumed by default to be 1.1704x0.8636 metres.

ROVER always fills the whole sheet, which is assumed to consist of 1152x850 pixels for the purpose of displaying a raster image. Since this may produce a plot which is too large, a means is provided to change the sheet size, and the number of pixels.

Logical name LSL$PI448_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, thus to give the same effect as the defaults, the file would have the single line:
1.1704 0.8636 1152 850

12.3 **Plotting mode**

The PI C448 plotter by default plots in transparent mode, which means that where parts of the plot overlap, the inks will be combined, generally giving darker colours. It also has the ability to plot in opaque mode, where anything drawn replaces what is there already (underlying colours do not show through).

Several mechanisms are provided to select the drawing mode, to satisfy the needs of the programs FPP, ROVER, and LITES2.

- By default FPPPI448 plots in transparent mode. The command ESCAPE 1 will cause opaque mode to be used for subsequent plotting, while ESCAPE 2 will revert to transparent mode. As an alternative, giving the command WORKSTATION 1000 before beginning a plot will select opaque mode initially.

- By default, ROVERPI448 draws the vector overlay using opaque mode, so that the colours replace any underlying colour. In order to allow the overlay to be transparent, define the logical name LSL$PI448_TRANSPARENT (as anything).

- By default, LITES2 will draw any data in overlays whose base plane is greater than zero in opaque mode. If it is required to draw everything in transparent mode, then either define the logical name LSL$PI448_TRANSPARENT (as anything), or give the command PLOT TYPE 1000 (before PLOT OPEN). Any number other than 1000 will revert to the original behaviour.

  As an alternative to the above mechanisms, the PLOT ESCAPE command may be used, in the same way as for FPPPI448. The command PLOT ESCAPE 1 will switch to opaque plotting, while PLOT ESCAPE 2 will revert to transparent plotting.

12.4 **Plotter interface**

If the plotter is connected to the computer by a serial line, then the baud rate should be set as high as possible on plotter and computer for fastest data transfer (e.g. 19200 baud).

12.5 **Producing plots**

The plotting programs send their output to to a file with logical name LSL$PI448. This may be assigned either directly to the plotter or to a disc file, thus for example
$ Define LSL$PI448 TXA0:
This will send output directly to the plotter on serial line TXA0:, or
$ Define LSL$PI448 PI448.PLT
This will send output to the disc file PI448.PLT in the current directory. The file may subsequently be plotted by
$ COPY PI448.PLT TXA0: (assuming plotter is on TXA0:)
or possibly using the PRINT command if an appropriate queue has been set up. If logical name LSL$PI448 is not defined, then a file LSL$PI448.DAT will be created

in the current directory.

12.6 **Use of ROVER**

ROVER was originally designed for interactive use on colour graphics screens. When used on a hardcopy plotter no attempt should be made to clear parts of the picture, or to use the editing facilities. The aim should be to produce the desired picture using the minimum number of commands.

CHAPTER 13

PostScript

The image FPPPOSTSCRIPT.EXE produces output using the Adobe PostScript page description language. This is usually stored on disc for subsequent transmission to a device which supports PostScript, though it may be possible to transmit direct to the device. As part of the PLOTTING package, the program PSEXPORT is provided to convert Laser-Scan PostScript files into other formats, such as Adobe Illustrator. See the PSEXPORT User Guide for details.

13.1 Features available

FPPPOSTSCRIPT offers most of the hardware facilities currently available in the FRT system (circles, curves, text, and pecked lines). If the flag values in the FRT do not request the hardware facility, then it will be emulated in software, which gives more control at the expense of efficiency. For example, the various TOLERANCE commands will not affect hardware circles and curves, and hardware text will use a built in text font, rather than using the TRI file. Note that unless an AFM file is used (see below), the TRI file must still contain all the fonts and characters used, but merely for the character widths to calculate the lengths of strings - the line detail is irrelevant if hardware text is used. Even the character widths are irrelevant if all text is at location point 0, 1, or 2.

The setting of the hardware field in the FRT is used to indicate what style of line caps and joins are required. The CAP style is a number in the range 0-2. 0 means butt caps (square ends), 1 means round caps, and 2 means extended caps (square but extended by half the line thickness. The JOIN style is also a number in the range 0-2. 0 means miter joins (the sides of the thick lines extended to meet), 1 means round joins, and 2 means bevel joins (each segment has a square end, but the missing triangle is filled in). Miter joins are changed to bevel if the resulting spike would be 'too long' - this is taken to be when there is an acute corner in the line with an angle of less than about 11 degrees (this could be changed by using the PostScript 'setmiterlimit' operator in a header file - the default is 10). Boundaries of area features will use the join style at the start/end point, and will not have a cap at all (provided that they can be drawn as a single path - see below). The FRT hardware field is built up as 10*JOIN + CAP, so for instance 21 would mean JOIN = 2 = bevel, CAP = 1 = round. If omitted, the hardware field defaults to 0, meaning miter joins

---

The spelling 'miter' is used, rather than 'mitre' so as to agree with X-Windows and PostScript documentation.

and butt caps. Remember that the hardware field in the SCT entry (if present) will override that in the FRT.

Note that all drawing using PostScript is opaque. Any marks on the paper are completely hidden by anything drawn subsequently, independent of colour.

Raster images are supported when GKSPOSTSCRIPTSHR is used from within LITES2. By default these are rendered in greyscale, but may optionally use colour (see below).

13.2 **Logical names**

Several logical names are used to control the operation of FPPPOSTSCRIPT.

LSL$PS should be defined to be the output file or device to which the PostScript is to be written. If it is not defined, then a file PLOTFILE.PS in the current directory will be produced.

If it is desired to plot raster images in colour on a colour PostScript plotter, then logical name LSL$PS_COLORIMAGE (note the spelling) should be defined as 1. If the resulting plot is sent to a device which does not support the 'colorimage' operator, then the 'image' operator is used instead and the output will still be greyscale. If the logical name is not defined, or is defined as 0, then 'image' will be used, and the output will be greyscale. If a PostScript file has been produced in greyscale, and it is desired to plot it in colour, then either edit the file, or prepend another file which defines 'USECOLOR' as 'true' using the sequence of PostScript "/USECOLOR true def". Note the the above applies only to raster images - vector data will always be drawn in colour if plotted on suitable hardware.

LSL$PS_NOMLW may be defined to be a real number - the nominal line width in millimetres. The default is 0.0635 (1/400 inch). The nominal line width is used for any features whose line width is specified as 0 in the FRT and also as the width and separation of the hatch lines used to render any solid filled area which does not use PostScript fill.

LSL$PS_MAXPATH may be defined to be an integer. This is the maximum number of points that will be put into a PostScript path before it is rendered onto the paper. Each PostScript interpreter will have a limit for this which may not be exceeded. The default value is 1000. Any attempt to set a value less than 100 will result in 100 being used. Ordinary lines are broken automatically at this limit, which may produce an untidy join if thick lines are used, though attempts are made to minimise this effect. Solid filled areas with more points than this limit will be rendered by close hatch lines (see LSL$PS_NOMLW) rather than using PostScript filling. The maximum number of points allowed in a filled area is 8192 by default, but may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for hatched fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

LSL$PS_HW_FILL may be defined to be 0 or 1. By default, solid filled areas with less than LSL$PS_MAXPATH points will use PostScript fill, while those with more will use close hatching. If the logical name is defined as 0, then all filling

will use close hatching (thickness and spacing equal to LSL$PS_NOMLW).

LSL$PS_FILL_INVIS may be defined to be 0 or 1. If PostScript solid fill is used, then by default all lines forming the boundary of a polygon (including any flagged as invisible in the IFF file) will be passed to PostScript. If the logical name is defined as 1, then the invisible segments will be omitted.

LSL$PS_SIGFIG may be defined to be an integer - the number of significant figures to be used for all numbers written to the PostScript file. The default is 6. The value may be in the range 1-8. Smaller values will produce smaller files, while too small a value will degrade the resolution. The sensible value depends on the size and resolution of the output device.

LSL$PS_FONTLIST is used to define which PostScript fonts are to be used. It need not be specified if hardware text is not used. The logical name should be defined to point to a file (default filespec LSL$FRT:*.PSFONTLIST) containing directives defining the fonts. The format of this file is documented in the FRT User Guide (MAPPING package). Previous releases of FPP allowed the logical name to be defined as a search list, specifying the fonts. This still works, but the new mechanism is preferred - it allows several facilities not previously available. Note that attempts to italicise a font using a negative font number in the FRT does not work for hardware text. If hardware text is used, then the character shapes in the TRI file are ignored. The widths from the TRI file will still be used for justification unless an AFM file (containing character widths) is specified in the .PSFONTLIST file, in which case the TRI is not used at all - it must still be supplied, but need not contain any fonts. Note that attempts to italicise a font using a negative font number in the FRT does not work for hardware text.

Instead of using the logical name LSL$PS_FONTLIST, it is possible to define the fonts in a PostScript header file (see below), but this is not recommended except in special circumstances.

LSL$PS_DOCUMENTFONTS is used supply a value for the DocumentFonts header comment in the PostScript file. It may be omitted, but if present should be a list (separated by space) of the fonts used in the file. For example:

```
$ Define LSL$PS_DOCUMENTFONTS "Times-Roman Helvetica"
```

LSL$PS_COMPRESS may be defined to be an integer in the range 0-1. It specifies how image data is to be encoded in the PostScript file. 1, the default, uses run length encoding, whilst 0 causes each pixel value to be stored. The former will result in a smaller plot file if the data has large areas of the same colour. The plot file may be up to 128 (16 for bit data) times smaller if the data was all one colour, but up to 2 (8 for bit data) times larger if every pixel was different from its neighbours.

LSL$COMPOSITE_CHARACTERS may be defined to be an integer in the range 0-1. It specifies whether to attempt to interpret composite characters in text strings. 1 enables the mechanism, 0 (or not defined) disables it. Composite characters are made up by superimposing two or more existing characters in a font. They are defined in the AFM file, specified in the PSFONTLIST file. All characters in the font (including the composites) have a name, such as Zcaron. If enabled, then a composite character is plotted by including its name, surrounded by braces (e.g. {Zcaron}) in a text string. An opening brace must then be represented by two opening braces. Any character may be included by this mechanism, even non-composite ones. This is of little benefit for ordinary

characters (e.g. {A} is the same as A), but may be of benefit for a character encoded in the range 128-255 as an alternative to the '$' escape mechanism. If the name inside the braces is not recognised, or an attempt is made to plot a composite character when TRI text fonts are in use, then the first letter of the name will be plotted instead (e.g. {Zcaron} plots as Z). If the mechanism is disabled, then braces are treated as normal characters.

13.3 **Colour table**

The colours used for the picture are defined using a text file on logical name LSL$PS_COLOUR (if this is set up). See Appendix A for the format of this file. Colours 0 to 255 may be defined. If the colour table is not set up, or contains insufficient entries, then any undefined colours will appear in black. Instead of using the colour table, it is possible to define the colours in a PostScript header file (see below).

13.4 **Paper size and pixel size**

FPP always has the concept of a sheet, in which it positions the plot according to the commands given by the user. When using FPPPOSTSCRIPT, the sheet is assumed by default to be 1.189x0.841 metres (A0 size). It is normally advisable to use FPP plot position 0 (POSITION command) to draw into the bottom left of this area. Some Postscript printers or plotters may be unable to plot near to the edge of the paper. In this case the OFFSET command (for example) should be used in FPP to move into the usable area. A means is provided to change the assumed paper size, but there should be no need to use this unless other plot positions (for examples POSITION 4, the default, which means centred) are to be used.

Logical name LSL$PS_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, thus to give the same effect as the defaults, the file would have the single line:

```
1.189 0.841 1448 1024
```

The pixel numbers are not relevant to FPP, but control the number of pixels assumed to cover the area for LITES2. LITES2 sets the number of pixels and their size itself, but will never draw more than the specified number of pixels across a plot. The number may be altered within LITES2 using the PLOT PIXELS command.

The maximum number of pixels which may be specified in either direction for LITES2 depends on the format of the raster data. For DTI files, the limit is 32768 for byte data, 16384 for word data, and 8192 for long data. There is no limit for LSI or LSR format files. All of these limits (except ROVER's 10000) are proportional to the maximum number of points in filled areas and hence may be increased using logical name LSL$FILL_POINTSMAX. The values given here are for the default of 8192.

13.5 **Header file**

Logical name LSL$PS_HEADER may be defined to be a file of PostScript commands. The contents of this file will be copied into the output PostScript file in the Setup section, before the start of the first page. There is no restriction on what the commands in the file may do - though it is of course most important that they are valid PostScript. The intention is that the various defaults in the PostScript Graphics State (such as mitre limit) can be altered (this is not changed by FPPPOSTSCRIPT). #copies could be defined to cause multiple copies of each page to be printed. Other possibilities are to define a clipping path, or to change the CTM (current transformation matrix) so as to scale or rotate the plot. Another possibility is to change the filling rule for polygons. The default is to use the even/odd rule fill. FPPPOSTSCRIPT uses "FL" as an abbreviation for "eofill", so to change to non-zero winding rule fill, then define FL as fill. At the point in the PostScript file that the header is included, the user coordinate system is in its default state, assumed to be 1/72 inch units with the origin at bottom left. The header file could even print pages of output itself.

If desired, the colours and fonts may be defined in the header file, instead of using the table and font list. To define FRT colour N, you need to put a procedure which sets this colour into the Nth element of array COLTAB, for example:

```
COLTAB 3 {0.0 1.0 0.0 setrgbcolor} put
```

will define colour 3 to be green. There is no reason why the other colour setting operators, sethsbcolor or setgray, should not be used instead.

To define FRT font N, you need to define a name F<N> which corresponds to a procedure which finds the desired font. The name should preferably be defined in the dictionary FNTDICT, for example:

```
FNTDICT begin /F3 {
/Times-Roman findfont
} def end
```

will define font 3 to be Times-Roman.

13.6 **Filled areas**

Any areas described as hatched in the FRT file use a software algorithm to generate the hatch lines - this is limited to 8192 points, and a maximum of 100 intersections of any hatch line with the polygon. Note that solid filling can be achieved by specifying equal hatch thickness and spacing in the FRT.

For solid filled areas, it is best to take advantage of PostScript's own filling algorithm if possible, but the maximum number of points which PostScript will allow to be solid filled may be somewhat limited for some applications (see LSL$PS_MAXPATH above). By default, PostScript fill will be used if there are less than LSL$PS_MAXPATH points, and close hatching (thickness and spacing LSL$PS_NOMLW) if there are more. Note that if hatching is used, then the quality of the PostScript output becomes dependent on the resolution of the output device used. By defining LSL$PS_HW_FILL as 0, it is possible to prevent any use of PostScript fill.

Complex polygons (e.g. with holes) should be defined using identical pairs of 'connecting lines' to attach the components. These lines may be flagged as 'invisible' in the IFF file. For hatched areas, this has no effect on the filling algorithm (which is designed not to draw anything when a hatch line crosses an identical pair of lines), but it affects whether the segments are drawn if the area is outlined. Solid areas are never outlined, but it has been observed that PostScript tends to produce a visible line of filling along pairs of identical lines when these lie outside the polygon. In order to overcome this problem, if LSL$PS_FILL_INVIS is defined as 1, then segments flagged as invisible will not be inserted into the PostScript path to be filled. This has not been made the default because of the way that PostScript fills a composite area. Each sub-path (delimited by invisible segments) will be closed before filling, which will only give the correct result for some arrangements of invisible segments - those in which all invisible segments connecting to a component, are attached to the same point on that component.

13.7 **Features and groups**

The PostScript output produced by LITES2 or FPP contains markers to delimit the start and end of IFF features. These markers ('u' and 'U') are defined to do nothing if the file is plotted on a PostScript printer, but are useful if the PSEXPORT program is used to export the file to another format. They allow features to be identified as objects in the output.

These same markers may be used to delimit groups of features. The FPP command ESCAPE 1, or the LITES2 command PLOT ESCAPE 1, cause a new group to be started. Used 'manually', this is only useful if the plot is produced in several passes, each with different feature selections, beginning a new group for each. If the 'sort by priority' mechanism is used, then FPP and LITES2 can automatically perform the ESCAPE 1 function between each level of priority. FPP will do this by default, unless turned off by a SEPARATOR ESCAPE 0 commands, whereas LITES2 requires the command PLOT SEPARATOR ESCAPE 1 to turn it on. (This same mechanism is used in some of the other Laser-Scan plotter drivers to begin a new opaque layer, but this is not needed in PostScript because everything is opaque anyway.)

13.8 **CMYK separations**

The PostScript plot files produced by FPPPOSTSCRIPT includes a facility to plot CMYK (cyan, magenta, yellow, and black) separations, instead of a full colour plot. This facility is not enabled by default. In order to plot a separation, you must define the PostScript name /SEPARATION to be one of /C, /M, /Y, or /K. The fragment of PostScript required to do this is e.g. (to plot the magenta separation)

```
/SEPARATION /M def
```

The best way to achieve this depends on the mechanism for transmitting the plot file to the plotter, but a typical method is to create a series of one-line files containing lines like the example above, and to concatenate one of these with the actual plot file (with a '+' sign) in the PRINT or COPY command.

For the benefit of those with a knowledge of PostScript, this separation mechanism tests whether the PostScript interpreter supports the 'currentcmykcolor' operator. If it does, then this is used to derive the separation colours, otherwise the behaviour of the operator is emulated. When plotting a separation, the colour is eventually set using the 'setgray' operator, with 0 representing the maximum amount of the separation ink, and 1 representing none. If 'currentcmykcolor', a PostScript Level 2 operator, has to be emulated, then the behaviour of 'blackgeneration' and 'undercolorremoval' are emulated by calls to procedures BG and UCR respectively. These are by default null procedures, but appropriate ones may be defined at the same time as defining /SEPARATION.

13.9 **Spot Colour separations**

It is possible to produce PostScript output in which parts of the plot are drawn into named colour separations (for use with PostScript Level 2 printers capable of producing separations, or with separating software which understands Level 2 PostScript). The user must choose which objects to draw into each separation, and make the plot in several passes selecting just the appropriate groups of objects.

Within a separation, the amount of colour (the 'tint') is calculated from the grey-level of the appropriate numbered colour in the colour table. A means is provided to set a new colour table for each separation. Whilst it may be easier to create a table consisting entirely of grey shades, this is not compulsory - the shade of grey corresponding to a given RGB value will be calculated.

The FPP command:

ESCAPE 10 filename

or the LITES2 command:

PLOT ESCAPE 10 filename

cause a new colour table to be read from the given filename or logical name. The format is the same as the table read initially (see above), as described in Appendix A.

The FPP command:

ESCAPE 11 name overprint model colour

or the LITES2 command:

PLOT ESCAPE 11 name overprint model colour

cause the new named separation to be started. If no arguments are given after 'ESCAPE 11', then plotting reverts to normal composite colours.

The arguments to this command are:

- name - the name of the separation to be produced, enclosed in double-quotes if it contains spaces.

overprint - the value 0 or 1. 0 means that drawing into this separation will erase the corresponding area in other separations. 1 means to 'overprint', i.e. other separations are not affected. Whether this option has any effect depends on the output device in use.

model - either RGB or CMYK, the colour model of the colour to be used to represent this spot-colour, if separations are **not** being produced.

colour - 3 (RGB) or 4 (CMYK) numbers in the range 0-1 describing the colour of this spot-colour (at maximum tint) to be used as a backup if separations are **not** being produced.

CHAPTER 14

VAXstation display (UIS)

The image FPPUIS.EXE produces output on a VAXstation screen, or optionally creates a UIS file for subsequent display on a variety of devices. Note that DEC workstations are more likely to be running DECWindows than the older UIS software for which this chapter is relevant. See the "X-Windows display (MOTIF)" chapter if running DECWindows.

14.1 **Display**

FPPUIS may be run from a terminal emulating window on the VAXstation screen, or from a separate terminal. The graphics display appears in a window on the VAXstation screen, which is accessed on logical name SYS$WORKSTATION (which is normally set up system wide).

On an 8 plane system, colours 0-63 may be used for the plot (0 is the background).

On a 4 plane system, colours 0-7 may be used for the plot.

On a monochrome system, only colours 0 and 1 are available. In all cases, features specified with colours out of range will be drawn in colour 1.

The number of colours used may be modified to some extent by the use of the WORKSTATION command.
WORKSTATION 6000 is relevant only to the 8 plane system, and allows the user to define 128 rather than 64 colours.
WORKSTATION 7000 is relevant to both 4 and 8 plane systems. The number of colours used is then allowed to reach the maximum for the particular number of planes (16 and 256 respectively), the defaults being 16 and 250. The choice of 250 colours on an 8 plane system allows FPP to run concurrently with any terminal emulation windows on the screen without any contention over the colours. If logical name LSL$UIS_MAX_COLOUR is defined to be an integer, then this number of colours will be used instead. This could be used for example to use the full 256 colours, even though this will then result in contention with any terminal windows.

14.2 **Colour table**

The colours used for the picture are defined using a text file on logical name

LSL$VAX_COLOUR (if this is set up). See Appendix A for the format of this file.

14.3 **Control options**

Two logical names are available to control the way in which drawing on the UIS workstation takes place.

If logical name LSL$UIS_HW_FILL is defined (as anything), then the hardware area filling facilities are used for solid fill areas (by default, close horizontal hatch lines generated by the software are used). Hardware filling may well be faster than the software fill. Hardware fill may be of particular benefit if a UIS metafile is being created (see below), since it will take advantage of the filling facilities of the final output device. Filled areas are limited to 8192 points by default. The limit may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for software fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'. correctly. Hardware fill may well be limited to a smaller number.

Logical name LSL$UIS_RESOLUTION may be set to a line width in mm (the default is 0.3, or approximately one pixel). This width will be used for lines with zero width in the FRT, and will be the thickness and separation of hatch lines in solid filled areas if hardware fill is not used. The main purpose of this is that lines with a width less than the screen resolution may be specified if output is to a UIS metafile for final output on a different device.

14.4 **UIS plot files**

In order to produce a UIS plot file, you have to set the logical name LSL$UIS_RETAIN_FILENAME - this is the name of the file which will contain the UIS commands (i.e. the metafile).

It is important that this logical-name is cleared when UIS output-to-file is complete because display-lists have to be enabled which can have unpleasant side-effects on other programs using UIS.

(NB. A useful VMS utility which operates on UIS files is RENDER - it can translate the UIS commands into several formats eg. SIXEL, PostScript etc - see "MicroVMS WORKSTATION SOFTWARE" manual.)

CHAPTER 15

Versatec electrostatic plotters

Versatec electrostatic plotters may be used by Laser-Scan programs FPPVRSC2V1 and FPPVRSVGS (package PLOTTING), ROVERVRSC2V1 (package TVES), and LITES2MOTIF (package LITES2).

15.1 Versatec library

There are two versions of FPP producing output suitable for Versatec plotters, which use different libraries supplied by Versatec. In order to use them, one must have obtained the appropriate library and associated data files from Versatec, and the plotter itself must be have appropriate hardware and firmware to plot the data formats produced by these libraries.

FPPVRSC2V1 uses a library usually called "Versatec Random Enhanced" (the object library itself is COL2V1.OLB). This library produces .RAN files, which need to be further processed using the Versatec supplied utility CRFOUT before being sent to the plotter. Further details are given below.

FPPVRSVGS uses a library called "Versatec Graphics Software (VGS)" (the object library itself is VGSC.OLB or VGSH.OLB). This library can produce a variety of output formats, depending on the settings in a parameter file. With this library, no separate utilities are needed to process the plot data. Further details are given below.

15.2 Colour table

The Versatec colour electrostatic plotters differ from normal pen plotters in that the colours produced by a particular colour index (from an FRT file, for instance) may be defined by the user. In this respect, the plotter resembles a colour display terminal. The Laser-Scan programs initially define the colours by reading a disc file on logical name LSL$VERSATEC_COLOUR. The format of this file is the same as that used for the VAXstation and Tektronix displays, and the same file may be used, although it will probably be found that changes are necessary to get the desired colours on the Versatec. Also white should not normally be used on the Versatec, as it will be invisible against the white paper background. FPP and LITES2 will use the colours defined by the file on logical name LSL$VERSATEC_COLOUR for the plot, while ROVER always reads in another table subsequently (LSL$LOOKUP:ROVER5.DAT by default, or others by use of the PLANES command, or explicitly by use of the LUT command).

An example colour table file is given in Appendix A.

15.3 **Area filling**

The maximum number of points allowed in a filled area is 8192 by default, but may be altered by defining logical name LSL$FILL_POINTSMAX to be the desired number. Similarly, the maximum number of times which a hatch line (for hatched fill) may cut the polygon is 100 by default, but may be altered by defining logical name LSL$FILL_CUTSMAX to be the desired number - exceeding this limit results in the message 'FILL_SIDE - Too many intersections found - ignored'.

15.4 **Versatec model and other parameters**

FPPVRSC2V1 by default assumes a model 3436 plotter is being used, whilst for FPPVRSVGS, a model must always be specified in the parameter file. Amongst other things, the default paper size and numbers of nibs per inch are based on the model. Although FPP does not mind plotting outside the paper size (as printed out after the IFF command is given), the library will clip the data, so it is important to use the correct model number. If using FPPVRSC2V1, and you have a model other than a 3436, then set logical name LSL$VERSATEC_MODEL to be its number e.g.

```
$ Define LSL$VERSATEC_MODEL 3444
```

Other parameters may be set similarly:

The logical name LSL$VERSATEC_MEMORY may be set to an integer to control the amount of memory (in bytes) assumed to be present in the plotter. The default is 5500000. If using FPPVRSVGS, the logical name is ignored - if required them memory size should be set in the parameter file.

If logical name LSL$VERSATEC_MONO is defined (at all) then a plot suitable for a monochrome plotter will be produced. All line work is done in black. In the case the default model is an 8242. If using FPPVRSVGS, the logical name is ignored - all plots are done in colour mode, but if a monochrome model is selected in the parameter file, then a monochrome plot is produced.

If logical name LSL$VERSATEC_MENU is defined (at all), then a menu of options is displayed when the plot file is initialised. It may be possible to change other parameters using this menu - though it is not guaranteed that the program will plot correctly if this is done. If using FPPVRSVGS, the logical name is ignored.

If logical name LSL$VERSATEC_BACKGROUND is defined (at all) then plots of raster data will plot pixels which have colour index 0 in colour 0 as defined in the colour table. If the logical name is not defined, then these pixels will not be plotted at all, and so will remain paper coloured. If using FPPVRSC2V1, the logical name is ignored - pixels with 0 value are never plotted.

15.5 **Paper size and pixel size**

FPP always has the concept of a sheet, in which it positions the plot according to the commands given by the user. On the Versatec, the sheet size is based on the Versatec model in use (as described above). The maximum Y dimension is the width of the plotter, while the maximum X dimension is usually the same, giving a square area. It is normally advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

ROVER always fills the whole sheet, which is assumed to consist of 1448 pixels in the x direction, and an appropriate number in the y direction to give square pixels. Since this may produce a plot which is too large, a means is provided to change the sheet size, and the number of pixels.

If you wish to set the paper size to be something other than the default, perhaps longer along the roll of paper, or to acheive different positioning using the POSITION command, then logical name LSL$VERSATEC_SIZE may be set to to point to a file containing a single line with 4 real numbers defining the x size (metres), y size (metres), x number of pixels, and y number of pixels, for instance to specify an A0 sheet, the file would have the single line:
1.189 0.841 1448 1024
The pixel numbers are not relevant to FPP, but control the number of pixels assumed to cover the area for ROVER and LITES2. LITES2 sets the number of pixels and their size itself, but will never draw more than the specified number of pixels across a plot. The number may be altered within LITES2 using the PLOT PIXELS command.

The maximum number of pixels which may be specified in either direction is 16384. Programs may impose other limits of their own e.g. 10000 for ROVER. The limit for LITES2 depends on the format of the raster data. For DTI files, the limit is 32768 for byte data, 16384 for word data, and 8192 for long data. There is no limit for LSI or LSR format files. All of these limits (except ROVER's 10000) are proportional to the maximum number of points in filled areas and hence may be increased using logical name LSL$FILL_POINTSMAX. The values given here are for the default of 8192.

If using FPPVRSVGS, then the paper size specified in the file will be overridden by any size specified in the parameter file. (Note that if PAPER is specified in the parameter file, the size must be in cm.)

15.6 **Versatec specific plotting**

The VRSC2V1 Versatec driver can only plot in transparent mode, which means that where parts of the plot overlap, the inks will be combined, generally giving darker colours. The VRSVGS driver also has the ability to plot in opaque mode, where anything drawn replaces what is there already (underlying colours do not show through), but only if RASTER or VCGL output formats are used. The following is only relevant in this case.

Several mechanisms are provided to select the drawing mode, to satisfy the needs of the programs FPP, ROVER, and LITES2.

- By default FPPVRSVGS plots in transparent mode. The command ESCAPE 1 will cause opaque mode to be used for subsequent plotting, while ESCAPE 2 will revert to transparent mode. As an alternative, giving the

command WORKSTATION 1000 before beginning a plot will select opaque mode initially.

- By default, ROVERVRSVGS draws the vector overlay using opaque mode, so that the colours replace any underlying colour. In order to allow the overlay to be transparent, define the logical name LSL$VERSATEC_TRANSPARENT (as anything).

- By default, LITES2 will draw any data in overlays whose base plane is greater than zero in opaque mode. If it is required to draw everything in transparent mode, then either define the logical name LSL$VERSATEC_TRANSPARENT (as anything), or give the command PLOT TYPE 1000 (before PLOT OPEN). Any number other than 1000 will revert to the original behaviour.

  As an alternative to the above mechanisms, the PLOT ESCAPE command may be used, in the same way as for FPPVRSVGS. The command PLOT ESCAPE 1 will switch to opaque plotting, while PLOT ESCAPE 2 will revert to transparent plotting.

A Versatec colour electrostatic plotter usually has four coloured toners, yellow, magenta, cyan, and black. Any dot on the paper is either toned, or left blank - the toners cannot be applied at less than full intensity. This means that colours which are not fully saturated can only be acheived using a screen pattern with some dots on and some dots off. The particular screen pattern used is chosen by the Versatec library. These patterns are used for filled areas and thick lines. For the thinnest lines, the screen pattern sometimes produces a poor effect, because the line may lie entirely in a gap in the pattern. For this reason, thin lines are drawn in the nearest pure colour (yellow, magenta, cyan, red, green, blue, black, white) to the desired colour.

FPPVRSC2V1 draws lines in 8 different widths ranging from 1 to 16 nibs (i.e. up to 1mm on a 400dpi plotter), while FPPVRSVGS uses 31 different widths ranging from 1 to 31 nibs (i.e. up to 2mm on a 400dpi plotter).

15.7 **Producing plots**

FPPVRSC2V1 produces Versatec Random Format output. By default this will go to a file RANDOM.RAN in the current directory, but logical name RANDOM may be set to point to another file name. The directory containing the Versatec colour and pattern files (these have names like CE3400.DAT and CE3400.PAT) must be pointed to by logical name LSL$LOOKUP.

The method used to plot the Random Format file depends on the Versatec installation. The colour and pattern files mentioned above, and also any device drivers and programs to send the Random Format file to the plotter, must be obtained from Versatec. These are not supplied by Laser-Scan. The procedure is usually something like the following:

To produce an OFFLINE plot the following should be typed

```
$ Define RANDOM 'filename'
Then run FPP to produce Random Format output in 'filename'

$ Define VEROUT 'filename'
$ Define VREMOTE DUMMY
$ Define VRECL 2000
$ Run CRFOUT (supplied by Versatec, reads RANDOM, writes VEROUT)
```

The resulting VEROUT file can then be copied to a foreign mounted magnetic tape for use on an offline plotter.

To produce an ONLINE plot the following should be typed

```
$ Define RANDOM 'filename'
Then run FPP to produce Random Format output in 'filename'

$ Define VEROUT 'filename'
$ Run CRFOUT (supplied by Versatec, reads RANDOM, writes VEROUT)
```

NB: The logical names VREMOTE and VRECL should be deassigned when producing an online plot for the versatec.

The resulting VEROUT file can then be copied or printed to an online Versatec plotter.

FPPVRSVGS can produce any of three output formats, RASTER, VRF, or VCGL. All the files required at run time are pointed to by logical names, which are usually assigned by executing a command file VGSLOGNAM.COM or VGSSYSLOGNAM.COM which is generated during installation of the VGS library software. The parameter file, pointed at by logical name PRMFIL, must at least specify which output driver, and which plotter model to use. Many other items may optionally be specified. See the Versatec VGS User's Guide for details. If including a PAPER directive in the parameter file, then ensure that it is specified in centimetres. If possible, ensure that the Versatec test programs can be used to produce plots before trying the Laser-Scan software. The output plot file uses logical name VEROUT, which may be assigned to the actual plotter, or a disk file depending on the plotter configuration. Note that incorrect settings in the parameter file or the configuration file may well prevent FPPVRSVGS from working. FPPVRSVGS expects to define pens and colours itself according to the colour table on logical name LSL$VERSATEC_COLOUR, so do not attempt to define these in the parameter file.

CHAPTER 16

WILD TA2 and TA10 flatbed plotters

Image FPPWILD.EXE produces output suitable for WILD TA2 and TA10 flatbed plotters.

16.1 Mode of operation

The program produces output on logical name LSL$WILD. This logical name is usually assigned directly to the plotter serial line, but may under some circumstances be assigned to a file for transmission to the plotter later. For example

```
$ Define LSL$WILD TXA1:
or
$ Define LSL$WILD WILD.DAT
```

If the logical name LSL$WILD is not set up, then the file LSL$WILD.DAT is produced.

16.2 Plotting area

When operating on line, FPPWILD will only plot in the area above, and to the right of the currently set origin. It assumes that the size of paper available extends from the origin to the upper right corner of the currently set window. This is the area used by the POSITION command, so, for example, the command POSITION 0 will plot as close as possible to the currently set origin, while the command POSITION 8 will plot in the top right corner of the currently set window.

16.3 Number of pens

The WILD TA plotters may be fitted with a 2 or 4 pen carriage. When running FPPWILD on line to the table, then the software knows what type of carriage is fitted. When running off line to a file, then a 2 pen carriage is assumed, unless the logical name LSL$WILD_PENS is defined as 4.

Features that have a colour index greater than the available number of pens, will be drawn with a pen from the available range. For example if a four pen

carriage is fitted, a feature with colour index 5 would be drawn with pen 1 while a feature with colour index 12 would be drawn with pen 4.

16.4 **Running Off Line**

As a debugging aid FPPWILD can be run with the output going to a file. This is achieved by assigning the logical name LSL$WILD to the name of a file (see above). In this case, when the program initialises the plotting table, it will prompt the user with the input:

```
Enter TA2 version number:
```

Any value can be input as it is ignored. A similar prompt is output when the plot is completed. If attempting to run FPP from a command file, the a response to this question must be included in the FPP commands, after the initial IFF command, and after the final EXIT command.

As noted above, when in this mode of operation a two pen carriage is assumed unless logical name LSL$WILD_PENS is defined as 4, and the coordinates written to the file assume that the whole plotting area of the TA table is available for use.

CHAPTER 17

Xynetics flatbed plotter

Image FPPXYNETICS.EXE produces output suitable for a Xynetics flatbed plotter.

17.1 Mode of operation

The program is not designed to drive the plotter directly, but to produce a magnetic tape of plot instructions. The program writes these instructions to a nine track magnetic tape on logical name LSL$XYNETICS.

This logical name **must** be assigned to a tape drive, which can write tapes at a density of 1600 bpi. For example
$ Define LSL$XYNETICS MSA0:

The tape must previously have been mounted as a foreign device, for example
$ MOUNT/FOREIGN MSA0:

The tape is always rewound on initialisation, i.e. after the first IFF command, and after any subsequent WORKSTATION commands.

If an ICL type header is required at the start of the tape, then the logical name LSL$FPP_HEADER_ICL should be set up. For example
$ Define LSL$FPP_HEADER_ICL true

It is possible to put several XYNETICS plots on a tape, in one run of FPP, by separating the IFF commands with ADVANCE.

17.2 Paper size

FPP always has the concept of a sheet, in which it positions the plot according to the command given by the user. On the Xynetics, the sheet is assumed to be 1.189x0.841 metres (A0 size). The initial pen position (the origin) should be at the lower left of the plotting area. It is advisable to use FPP plot position 0, 1 or 2 (POSITION command) to avoid wastage of paper.

Xynetics plotters can use either rolls of paper or sheets of paper. By default sheet format is assumed, however this can be overridden by giving the command WORKSTATION 100 (the default is WORKSTATION 101).

After a plot, the pen will move back to the origin.

If sheet feed is being used then the plotter will pause, while the sheet is replaced; if the WORKSTATION 100 command has been given, the paper will advance 2 centimetres passed the end of the last plot.

APPENDIX A

Colour Table

The following is an example of a file describing the colours to be used on plotters with user defined colours. It should be set up on the logical name described in the appropriate section (e.g. LSL$BENSON_COLOUR or LSL$CALCOMP_COLOUR).

The character ';' introduces a comment. The colours are specified as proportions of red, green, and blue, in hexadecimal in the range 0-FF. An example file is in LSL$LOOKUP:COLOUR_TABLE.COL.

```
; example colour table
;       Red     Green   Blue    Blink   Comment
16                                      ; number of colours
         0       0       0       0      ; 1 for background
         0       0       0       0      ; 2
        60      A0      FF       0      ; 3 sea
        30      30       0       0      ; 4
        20      50       0       0      ; 5
        20      70       0       0      ; 6
        10      90       0       0      ; 7
        10      A0       0       0      ; 8
         0      C0       0       0      ; 9
         0      D0       0       0      ; 10
        10      EE      10       0      ; 11 green
        41      EE      10       0      ; 12
        61      EE      10       0      ; 13
        81      EE      10       0      ; 14
        91      EE      10       0      ; 15
        A0      EE      10       0      ; 16
```

The example colour table file, LSL$PUBLIC_ROOT:[PLOTTING.LOOKUP]COLOUR_TABLE.COL, may be plotted out using the command file LSL$PUBLIC_ROOT:[PLOTTING.COM]COLOUR_TABLE.COM. These files should be in the LSL$LOOKUP and LSL$COM search lists respectively. The command file accepts 4 parameters - the name of the FPP to use (e.g. FPPCAL5800), the colour table logical name (e.g. LSL$CALCOMP_COLOUR), the ENLARGE factor for the plot (e.g. 1.0), and any other FPP commands needed (normally not required). You should set up any other logical names needed for plotting and then either @ or SUBMIT the command file. Type out the command file for further details.

INDEX