Laser-Scan Ltd.

FRT User Guide

Issue 3.8

Science Park, Milton Road, Cambridge, England CB4 4FY tel: (01223) 420414

Document "FRT User Guide"		Category "USER"
Issue 1.0	TA Adams, I Rutherford	06-Jan-1986
.	.	.
.	.	.
Issue 3.0	Ron Russell	5-Mar-1992
Issue 3.1	Clarke Brunt	10-Apr-1992
Issue 3.2	Clarke Brunt	12-Feb-1993
Issue 3.3	Clarke Brunt	21-Jul-1993
Issue 3.4	Clarke Brunt	14-Sep-1993
Issue 3.5	Clarke Brunt	30-Aug-1994
Issue 3.6	Clarke Brunt	4-Jan-1995
Issue 3.7	Clarke Brunt	5-Apr-1995
Issue 3.8	Clarke Brunt	1-Dec-1997

CONTENTS

1 PREFACE 4
1.1 Objective of this Guide 4
2 INTRODUCTION 5
3 PARAMETER FILES ASSOCIATED WITH FEATURE REPRESENTATION 5
3.1 Feature representation file types. 5
3.2 Maximum numbers in parameter files. 7
3.3 THE FEATURE REPRESENTATION TABLE (FRT) 9
3.3.1 General 9
3.3.2 Comment Lines 11
3.3.3 The FRT Section of the Feature Representation Table 11
3.3.3.1 FRT Parameter 1 - the Record Identifier 12
3.3.3.2 FRT Parameter 2 - the Feature Code Identifier . 12
3.3.3.3 FRT Parameter 3 - the Graphical Type (GT) Entry 12
3.3.3.4 FRT Parameter 4 - Colour Entry 14
3.3.3.5 FRT Parameter 5 - Width Entry 15
3.3.3.6 FRT Parameter 6 - Size 15
3.3.3.7 FRT Parameter 7 - Secondary Code (SC) 16
3.3.3.8 FRT Parameter 8 - Flags 18
3.3.3.9 FRT Parameter 9 - H/W 19
3.3.3.10 FRT Parameter 10 - Description 19
3.3.4 The Priority Table of the Feature Representation Table 20
3.3.4.1 PRIORITY Parameter 1 - the Record Identifier . . 20
3.3.4.2 PRIORITY Parameter 2 - the Default Identifier or the Feature Code 20
3.3.4.3 PRIORITY Parameters 3 and 4 - a priority and representation pair 21
3.3.4.4 PRIORITY Parameters 5 to 18 - optional priority and representation pairs 21
3.3.5 The Symbol Code Table (SCT) of the Feature Representation Table 22
3.3.6 The Group Table of the Feature Representation Table 23
3.3.6.1 Group Parameter 1 - the Record Identifier . . . 23
3.3.6.2 Group Parameter 2 - the group name 24
3.3.6.3 Group parameter 3 - the Group Feature Code Range 24
3.3.6.4 Group Parameter 4 - the Optional Description . . 24
3.3.7 The Pattern Table of the Feature Representation Table 24
3.3.8 The Fill Area Table of the Feature Representation Table 28
3.3.8.1 Fill Parameter 1 - the Record Identifier 29
3.3.8.2 Fill Parameter 2 - the Fill Index 29
3.3.8.3 Fill Parameter 3 - the Hatch Style 29
3.3.8.4 Fill Parameter 4 - the Line Pattern Number . . . 30
3.3.8.5 Fill Parameter 5 - the Optional Description . . 30
3.3.9 The Attribute Code Definition (ACD) part of the Feature Representation Table 30
3.3.9.1 ACD Parameter 1 - the Record Identifier 33
3.3.9.2 ACD Parameter 2 - the data type entry 34
3.3.9.3 ACD Parameter 3 - the code entry 34
3.3.9.4 ACD Parameter 4 - the name entry 35

3.3.9.5	ACD Parameter 5 - the minimum value	35
3.3.9.6	ACD Parameter 6 - the maximum value	35
3.3.9.7	ACD Parameter 7 - the minimum step	35
3.3.9.8	ACD default values	36
3.3.9.9	The LSL default attribute code definition	36
3.4	The Symbol Representation IFF file (SRI)	37
3.4.1	Overview	37
3.4.2	Creating a template SRI file	37
3.4.3	Digitising the SRI File	37
3.4.3.1	Feature Codes in the SRI file	38
3.4.3.2	Feature Serial Numbers (FSN)	38
3.4.4	Guidelines for SRI Files	39
3.4.5	Post Processing SRI Files	39
3.5	The Text Representation IFF File (TRI)	40
3.5.1	Overview	40
3.5.2	Digitising Text Fonts	40
3.5.3	Digitising Special Characters	42
3.5.4	Character substitution	43
3.6	The PostScript Font List File (PSFONTLIST)	44
3.6.1	Overview	44
3.6.2	Comment lines	44
3.6.3	FONT directive	44
3.6.4	FIND directive	44
3.6.5	SCALE directive	45
3.6.6	ENCODING directive	45
3.6.7	NEWNAME directive	45
3.6.8	KERNING directive	46
3.6.9	AFM directive	46
3.6.10	DIRECTION directive	46
3.6.11	ARABIC directive	47
3.6.12	HEADER directive	48
3.6.13	Composite Characters	48

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

PREFACE

Objective of this Guide

This introductory guide is intended to teach new operators how to generate and modify the graphic presentation of map information on Laser-Scan's digitising, display and plotting systems. This graphics generation is achieved by the creation and use of special parameter files called Feature Representation Tables (FRT files) and the nature of these and their associated files is described in this manual.

It is assumed that the reader has a basic understanding of the VAX/VMS operating system and an awareness of the contents and structure of the Laser-Scan IFF (Internal Feature Format) digital map data format, (see the IFF User Guide).

The reader should also be familiar with the use of the Laser-Scan IMP (IFF Map Processing) modules IPATCH, ISORT and ISTART, (for details see the IMP Reference Manual).

For experienced users the Feature Representation software libraries are described in the the FRTLIB Programmer Reference Manual.

INTRODUCTION

All Laser-Scan cartographic systems store and manipulate digital maps in a simple data format known as IFF (**I**nternal **F**eature **F**ormat). The collection of elements comprising the map information is held in the digital file as a series of discrete **features**. These map features can be lines, in-filled areas, points or texts; this is referred to as the feature's GT or "Graphical Type". Each different type of feature (e.g. railways or telegraph poles) is assigned a different numeric code referred to as the FC or "Feature Code" Laser-Scan plotting systems use the feature code to determine how a feature should be portrayed on the graphics display or on the film and paper plots.

It is necessary to pre-design a scheme of feature codes before any production digitising and plotting work can begin on Laser-Scan systems. The feature coding scheme should be capable of sub-dividing the map data into a sufficient number of categories in order to provide appropriate control of the plotting symbology at the output stages. For example, if it is necessary to display two different road types with different colours, line-widths or line styles then it will be necessary to distinguish them within an IFF file with different feature codes. Once designed, production digitising can adhere to the feature coding scheme and all Laser-Scan digitising products have mechanisms to feature code data in schemes designed by the user. Readers are referred to the relevant user guide to achieve this.

Associated with any feature coding scheme is a complementary **feature representation scheme**. This is the means by which graphics output systems can relate appropriate symbol definitions to the numeric feature codes held within the digital IFF files. Each feature representation scheme should match exactly with the chosen feature coding scheme; without this compatibility, system errors or warnings will result when attempts are made to plot a digital map on a plotter or display it on the LITES2 graphic screens. Any enhancements or modifications which are made to a feature coding scheme should equally be made to the feature representation scheme. The following sections describe the nature of the parameter files associated with feature representation schemes including details on how they can be created and amended.

PARAMETER FILES ASSOCIATED WITH FEATURE REPRESENTATION

Feature representation file types.

There are three parameter files associated with any feature representation scheme on Laser-Scan systems. These files are known as:

- The FRT (**F**eature **R**epresentation **T**able) is the major file in any representation scheme. It is this file which associates each feature code in the feature coding scheme with its corresponding graphical type (point, line, area, etc) and gives details of line styles, widths and colours. This file is also used to specify user definable attribute code definitions (ACDs).

- The SRI (**S**ymbol **R**epresentation **I**FF) file is a complementary file to the FRT which describes the nature and appearance of point symbols used either to portray discrete point features or to compose particularly intricate line styles. Since point symbols are inherently graphic in nature, they are held in the feature representation scheme within an IFF file. Their format is similar to that used to store the map data themselves. The SRI file, therefore, can be looked upon as a library of graphic symbols to be utilised when displaying digital maps.

- The TRI (**T**ext **R**epresentation **I**FF) file is a complementary file to the FRT which describes the nature and appearance of the type fonts to be used when displaying text and map names. The system is capable of handling different type fonts. Fonts can either be supplied by Laser-Scan or they can be created and maintained by the customer. Since text fonts are inherently graphic in nature, they are held in the feature representation scheme within an IFF file. The TRI file, therefore, can be looked upon as a library of characters held as type fonts to be utilised when displaying digital maps.

  The development of the PostScript plotting language, with its own built in fonts, has meant that for some applications, the importance of the TRI file has diminished. When PostScript fonts are used, the TRI file is sometimes used just for character width information, and sometimes not at all, though a TRI file must still always be specified. See the section below about the PostScript Font List File (PSFONTLIST) for details.

All three files are necessary to define a feature representation scheme. Files can be used for more than one scheme if required: in fact it is commonly the case that the text font description file (the TRI) is used in many representation schemes at a user site.

Detailed descriptions of each of the component representation files are given in subsequent sections of this document.

Directory location of feature representation files

All three feature representation files - the FRT, SRI and TRI files - reside in the VAX/VMS directory with logical name:

LSL$FRT

Thus, users can summarise all the available feature representation files by setting default to the directory defined by the logical name LSL$FRT and issuing the VMS command 'DIRECTORY'.

$ **SET DEFAULT LSL$FRT<CR>**
$ **DIRECTORY<CR>**

It is wise to use the same name within the file specification for the complementary three files in the representation scheme, though this is not mandatory. The name of each representation file within a scheme is distinguished by its three-letter file extension. Thus, a typical representation scheme for the display of a digitising exercise for soil mapping might use the following three files - all held in directory LSL$FRT

SOILS.FRT
SOILS.SRI
UNIVERS.TRI

The file UNIVERS.TRI might be a text font library set up by a user to produce map text in the Univers type face.

Both LITES2 (the **L**aser-Scan **I**N**T**eractive **E**diting **S**ystem) and FPP (the Laser-Scan **F**ast **P**lotter **P**rogram) require the relevant file specifications to be supplied during initialisation stage for the relevant FRT, SRI, and TRI files being used. It is not generally necessary to supply all three file specifications if the filenames are the same for each file; in such cases, it is sufficient merely to supply the filename of the FRT.

The definition of a working feature representation scheme is based on all three files - the FRT, SRI and TRI. The files interrelate with one another and there are references and pointers between them. To explain the whole system therefore requires an understanding of the contents of all three files. Novice readers are urged to read all sections before attempting to create and manipulate FRT files. The content and format of the individual files is presented below.

Maximum numbers in parameter files.

The definitions contained in the FRT, SRI, and TRI parameter files are stored in computer memory when the files are read. The amount of memory reserved for this limits the number of definitions which may be contained in each file. Defaults are provided which allow enough space for average files, but if this is exceeded, then a mechanism is provided to increase the space. For each limit, there is a logical name which may be defined to be the maximum number of the particular entity allowed. For example, to increase to number of vectors allowed in a TRI file to 20000, one might do:

$ **DEFINE LSL$FRT_TRIMAX 20000<CR>**

The setting of limits way in excess of the size of your files should be avoided, as computer memory will be wasted. If large parameter files are used regularly, then you should arrange for these logical names to be defined system wide.

The logical names available, together with their default values and maximum allowed values, are summarised in the table below:

Logical name	**Default**	**Maximum**	**Controls**
LSL$FRT_ACDMAX	50	32767	ACD entries allowed in FRT
LSL$FRT_FILMAX	100	32767	FILL entries allowed in FRT
LSL$FRT_FONTMAX	5	127	Fonts allowed in TRI
LSL$FRT_FRTMAX	1000	32767	FRT entries allowed in FRT
LSL$FRT_GRPMAX	30	32767	GROUP entries allowed in FRT
LSL$FRT_PATMAX	100	32767	PATTERN entries allowed in FRT
LSL$FRT_PRIOMAX	20	32767	PRIORITY entries allowed in FRT
LSL$FRT_SCTMAX	200	32767	SCT entries allowed in FRT
LSL$FRT_SRIMAX	7000	1000000	Vectors allowed in SRI
LSL$FRT_TRIMAX	10000	1000000	Vectors allowed in TRI
LSL$FILL_POINTSMAX	8192	none	Points in patterned area
LSL$FILL_CUTSMAX	100	none	Intersections in patterned area

THE FEATURE REPRESENTATION TABLE (FRT)

General

The FRT (**F**eature **R**epresentation **T**able) provides the display and plotting software with the association between the numeric feature codes in the feature coding scheme and the required display symbology in the feature representation scheme.

FRT files are standard VAX/VMS ASCII character text files which can be printed on the line printer and edited in a normal text editing session as required. The file is composed of a series of records which each comprise a set of numeric parameters separated by at least one space or a tab character.

This 'free format' enables easy creation and updates and it is a good rule to tabulate the parameters into a sequence of columns to facilitate user understanding of the file. There are also facilities for including comment lines within the table to aid interpretation by the reader; the software ignores these comments at run time. Examples can be found in Appendix A.

The FRT provides each feature code with a definition of its graphical type (line, in-filled area, point or text) and how it is to be drawn, (e.g. colour, line width, size of symbol etc). Sub-sections of the FRT control the specification of line styles to be used for symbol definitions (in the SRI and TRI), general map lines (in the PATTERN section of the FRT), or patterned fill areas (in the FILL section of the FRT). The PRIORITY sub-section allows the definition of how feature codes are to be drawn in terms of other feature codes.

A further section of the FRT file called the ACD (for **A**ttribute **C**ode **D**efinition) provides the opportunity to specify the data type, name, maximum value and minimum value of any attribute codes required, over and above those default codes defined by LSL.

Overview of the component tables of an FRT file

Each FRT file is sub-divided into six sections:

- FRT, the **F**eature **R**epresentation **T**able,
- PRIORITY, the definitions of multi-pass representations
- SCT, the **S**ymbol **C**omponents **T**able,
- GROUP, feature code group definitions,
- PATTERN, line style pattern definitions,

- FILL, fill area pattern definitions, and,
- ACD, the **A**ttribute **C**ode **D**efinitions

Only two of these sections are mandatory (FRT, SCT) and the remaining four are optional depending on the requirements of the representation scheme. Conventionally, all seven sections (if present) are held sequentially in the file in the order: FRT, PRIORITY, SCT, GROUP, PATTERN, FILL, ACD.

Each of the seven sections perform the following tasks:

- The FRT section relates each feature code in the coding scheme to its associated graphical type (GT) - ie point, line, area, etc - and the required graphical representation for that feature, viz its colour, linewidth, size (if relevant), and details of the hardware to be used for drawing and symbol or line pattern style (if relevant).
- The PRIORITY section relates (up to 8) additional feature codes, and the priority in which to draw them, with a master feature code - the feature code that is associated with the feature in a file.
- The SCT section relates graphical types, and the required graphical representation, to the lines which are used in the composition of point symbols (as held in the SRI file) and text characters as held in the TRI file).
- The GROUP section (which is optional) provides a useful means of grouping similar features (with different feature codes) under one global group name. From this series of group definitions it is then possible to plot information by groups or select them for manipulation in LITES2.
- The PATTERN section (which is optional) provides the specifications of patterned line styles and provides the pointers to the relevant symbols in the SRI file if these are required in the creation of a particularly intricate line style.
- The FILL section is included when a representation scheme makes it necessary to define fill area patterns. These are defined in terms of hatched patterned lines. This section specifies the hatch direction and patterned line style to be used.
- The ACD section is included when there is a requirement for a site to define additional attribute codes, over and above those that are supplied by default by LSL.

Comment Lines

Though not essential, it is advantageous and useful to include comment lines of information in between each section (and also within sections if it is helpful). Furthermore, it is suggested that comment lines are included at the head of the file to give details of its purpose, author, date of creation, etc, and also at the head of each section to provide explanatory text headings to each column in the table to follow. Comment lines are represented by any alphanumeric characters which follow an exclamation mark (!). These exclamation marks can be placed in any character position within a record. The effect, though, will be to cause the software to ignore characters which lie to the right of the exclamation mark in the record. In general cases, it is most appropriate to use column 1 for the exclamation mark and hence convert the entire record into a comment line. The FRT file presented in Appendix A gives examples of the use of comment lines. The user is strongly recommended to use comments copiously. Comment lines are particularly useful for recording the nature of any changes made to the file. It should be noted that all comment lines should begin with a non-numeric character, particularly when comments are made at the end of a line in the FRT to give an explanation of the line style being represented.

The FRT Section of the Feature Representation Table

The FRT relates the numeric FC (**F**eature **C**ode) of an IFF feature to the graphical representation to be used for that feature when plotting. This information is supplied as ASCII character information in columns. For each feature code (FC), the graphics information is held as one record (line of information). Each record is typed in free format - ie, at least one space or tab character between columns, and within the FRT section the records must be sequenced as ascending numeric FCs. Appendix A details a typical Feature Representation Table from which can be seen an example FRT section within the full table.

In total there are ten parameters which comprise the FRT section of the Feature Representation Table, though one of these parameters is the textual comment (or description) which is optional to aid its understanding. The nature and importance of the majority of these parameters is dependent on the value given to the parameter known as the graphical type (GT) which is parameter 3 in the record. This is described in detail subsequently but for example, the size parameter for lines and curves indicates the amount that they are to be offset, for symbols and characters it is their height and for hatched fill areas it is the separation of the hatching.

Each of the parameters will now be explained in the same sequence as they appear from left to right within each FRT record:

	FC	GT	COLOUR	WIDTH	SIZE	SC	FLAG	H/W	DESCRIPTION
FRT	30	1	2	0	0	0	2	1024	Minor Building
FRT	32	1	3	50	0	5			Parish Boundary
FRT	35	7	1	0	20	13	1		Trig Point

FRT Parameter 1 - the Record Identifier

In order to confirm that the current record belongs to the FRT section, it is necessary to include the three character identifier

FRT

This parameter is mandatory and must be the first parameter within the FRT record.

FRT Parameter 2 - the Feature Code Identifier

The second parameter within the FRT record is the FC identifier. It is mandatory and takes an integer number between 0 and 32767. This entry refers to the feature code used within the IFF map data files to distinguish features within a pre-designed feature coding scheme. It is essential that there is a valid FC identifier within a record of the FRT for every feature code referenced by data within an IFF file. Missing FRT entries will result in the error message:

FRTLIB - Feature code 'integer' not found in FRT

at the time when LITES2 or FPP attempts to display a map file.

Although it is necessary to have a sufficient number of FRT entries to relate to every FC in the chosen data, it is possible to use an FRT schema which is far more complex than is actually necessary to suit an individual IFF file.

FC identifiers cannot be duplicated within an FRT and the records **must** be sorted numerically in ascending FC order.

FRT Parameter 3 - the Graphical Type (GT) Entry

The third parameter within the FRT record is known as the Graphical Type or GT entry. It is mandatory and currently takes an integer number between 0 and 12. This entry instructs the Laser-Scan software systems of the graphical type to be associated with the feature code specified in the preceding FC identifier entry. The graphical types currently available on Laser-Scan systems are:

1 = LINEAR	Join digitised points by straight lines.
2 = CLOCKWISE ARC	A clockwise arc based on three digitised points - edge, centre, edge.
3 = ANTICLOCKWISE ARC	As 2 above in anticlockwise direction.
4 = CIRCUMCIRCLE ARC	A circular arc based on three digitised points, all positioned on the arc's circumference.
5 = FULL CIRCUMCIRCLE	A circle based on three points all positioned on the circle's circumference.
6 = INTERPOLATED CURVE	Interpolate a curve through a given sequence of points to generate a smooth line feature. FPP and LITES2 have options for both the Akima and McConalogue algorithms.
7 = UNORIENTED SYMBOL	Insert at the single digitised point a pre-defined point symbol (held either in the SRI file or the plotting hardware's symbol library). The symbol is drawn at the orientation at which it is stored in the symbol library and at the size specified in the FRT.
8 = ORIENTED SYMBOL	Insert at the digitised point a predefined point symbol (held either in the SRI file or in the plotter hardware's library). The symbol is drawn at the size specified in the FRT, but at the orientation defined either by a second point or an explicit angle in the IFF file.
9 = SCALED SYMBOL	Insert at the digitised point a predefined point symbol (held either in the SRI file or in the plotter hardware's library). The symbol is drawn at the orientation and size defined by a second locating point in the IFF file. The size parameter within the FRT is only used as a default in instances where a scaled symbol has not been given its second reference point.
10 = TEXT	All text and map names are held as a digitised point, an angle and an ASCII string. The height of the text can either be taken from the FRT or be stored in the IFF file.
11 = SYMBOL STRING	This is the graphical type associated with special line features in which a pre-defined point symbol (held either in the SRI file or the plotting hardware's symbol library) is positioned at each digitised point location.

12 = FILL AREA — This is the graphical type associated with features to be shown as an in-filled area. These data are generally held as closed polygons (ie. the boundary of the area); in instances where the first and last points are not coincident, the systems will automatically include an additional last point whose position is coincident with the first. Subsequent parameters associated with GT12 provide details of the mechanism to fill the areas when displayed.

The relevance and importance of the subsequent parameters which comprise the remainder of the FRT record is dependent on the value taken by the GT parameter. The table which follows associates the relevance of each subsequent parameter based on the value of GT.

The relationships between FRT parameters and Graphical Type are summarised in the table below:

Feature Type	**GT**	**Colour**	**Width**	**Size**	**SC**	**Flag**	**H/W**
linear	1	Y	Y	Y	Y	O	O
clockwise arc	2	Y	Y	N	Y	O	O
anticlockwise arc	3	Y	Y	N	Y	O	O
circumcircle arc	4	Y	Y	N	Y	O	O
full circumcircle	5	Y	Y	N	Y	O	O
interpolated curve	6	Y	Y	N	Y	O	O
unoriented symbol	7	Y	Y	Y	Y	O	O
oriented symbol	8	Y	Y	Y	Y	O	O
scaled symbol	9	Y	Y	Y	Y	O	O
text	10	Y	Y	Y	Y	O	O
symbol string	11	Y	Y	Y	Y	O	O
fill area	12	Y	Y	Y	Y	O	O

where Y = Yes, relevant
N = No, not relevant
O = Optional

FRT Parameter 4 - Colour Entry

The integer values assigned to the colour entry are limited only by the number of colours available on the plotting device.

For example, if a 3 colour plotter has pen 1 assigned as red and 1 is entered as the colour for a specific feature code then all line work digitised in that feature code will be plotted in red. What happens when an attempt is made to display a feature in a pen number that does not exist on the plotting device is dependent on that device, but Laser-Scan programs always draw it visibly.

Colour 0 indicates the background colour, and will generally be converted to colour 1. However if, when drawing symbols and texts from the SRI or TRI, the colour entry is 0, then the colour of the individual components of the symbol or character is taken from the relevant SCT (see below).

FRT Parameter 5 - Width Entry

The real value assigned to this entry is in mm on the map sheet, although it is only applicable to plotting devices that are capable of generating different line widths.

If zero is assigned then the minimum line width will be allocated. If any attempt is made to display a feature with a line width that does not exist on the current plotting device, it will be defaulted to the minimum width.

FRT Parameter 6 - Size

The real value assigned to the size parameter is in mm on the map sheet.

As indicated earlier the meaning of this entry is dependent on the graphical type

- for line and curve features (GT1 and GT6) it is the distance (in mm) the feature is to be offset from its defining points. A positive value indicates that the feature is to be drawn offset to the right and a negative value that it is to be offset to the left.

- for symbols and texts (GT7 - GT11) it is the size of that the symbol or text is to be drawn at. The method of achieving the correct size is discussed below in the section which deals with digitising of symbols into the Symbol Representation file (SRI).

- for fill areas (GT12) it is the spacing to be used when using a hatched or patterned fill area style (see below). If a value of 0.0 is given, then the default value for this spacing is used (usually 5.0mm).

- for lines, curves, and areas (GT1, GT6, and GT12) used as part of the prioritised representation of a symbol or text, it is the offset (outwards) from the symbol or text feature. A positive value is interpreted as mm, while a negative value (converted to positive) is interpreted as a fraction of the size of the symbol or text.

FRT Parameter 7 - Secondary Code (SC)

The secondary code is responsible for looking up additional characteristics, other than those indicated by the GT, about the specific feature code. For example the style of symbol or the pattern in a linear feature. However, the target area for the lookup of these characteristics is determined by the value of the Graphical Type (GT) entry.

The following table shows the relationship between SC (**S**econdary **C**ode) and GT (**G**raphical **T**ype)

GT	**Target area for look up of corresponding value in SC**
1	PATTERN definitions (in FRT)
2	"
3	"
4	"
5	"
6	"
7	FSN (**F**eature **S**erial **N**umber) in the SRI
8	"
9	"
10	Text representation file layer number (TRI)
11	FSN (**F**eature **S**erial **N**umber) in the SRI
12	Has specific values (detailed below)

An SC of 0 is used if there is no ancillary information.

The contents and format of both the SRI and TRI files are discussed later in the sections dealing with those file types. Briefly here, it is important to understand that both files are graphic in nature; the library of symbols or the font of text characters are held in the IFF format rather like the map data themselves. In TRI files, it is possible to hold more than one font at a time and these are subdivided into different IFF layers. In SRI files each symbol is referenced with a unique FSN (**F**eature **S**erial **N**umber), located in the NF entry of each IFF feature which forms a part of that symbol. These two items are important when assigning the integer parameters to the secondary codes in the FRT.

For IFF features which are to be represented by symbols (GT = 7, 8, 9) the SC parameter relates to the feature serial number of the symbol in the SRI file.

For IFF text features (GT = 10) the SC parameter relates to the font number in the TRI file. (Different fonts are stored in different layers in the TRI file). To draw a text font italicised, the negative value of the font should be put in the SC entry.

By default italic text will be drawn at a slope of 26.565 degrees. The slope can be set by the user through the logical name LSL$ITALIC_ANGLE. This logical name should point to a real value in

the range 0.0 to 45.0 which represents the required slope angle in degrees. e.g.

$ Define LSL$ITALIC_ANGLE 15

to draw italic text with an angle of 15 degrees.

NOTE

It is intended that this logical name should be set system wide for a site. Changes in the value it points to between runs of LITES2 and FPP may lead to inconsistent results

In the case of special line styles (GT = 1, 2, 3, 4, 5, 6) the SC parameter relates to the pattern number in the PATTERN section of the FRT file.

Once the user has indicated the relevant pattern or feature serial number in the SC entry then these can be retrieved from the pattern table, SRI file or TRI file.

The SC for GT 12 has a different function as it defines the style of the fill area that is to be used.

For fill areas, the SC codes relate to the following:

SC	**Fill area definition**
< -2	FILL area pattern definitions (in FRT)
-1	Solid
0	Hollow
1	Hatch with horizontal lines
2	Hatch with vertical lines
3	Hatch with 45 degree lines
4	Hatch with -45 degree lines
5	Cross hatch with horizontal and vertical lines
6	Cross hatch with 45 and -45 degree lines.
101-106	as 1-6 above but include the perimeter line
10000-13600	Number is 10000 + angle in tenths of degrees
20000-23600	as 10000-13600 but include perimeter line

The spacing between lines in hatched styles is determined by the size entry. If this is 0.0, then the default value of 5.0mm is used.

Styles 10000-13600 and 20000-23600 allow hatching at an arbitrary angle, but not cross-hatching. Use PRIORITY and multiple representations if cross-hatching is required.

The colour of the area-fill is determined by the colour definition for that particular feature code.

For solid, hollow or hatched styles, the fill area attributes are used for the perimeter line. For patterned styles, inclusion of the perimeter line is defined in the fill area table (see below).

FRT Parameter 8 - Flags

This is an optional column which contains a bit significant integer. This means that the following values should be added together as required to obtain a value which will have the desired effect. The meaning of the bits is defined in the following table:

Bit	Value	Meaning
0	1	use hardware circle arc, curve, symbol, or text. This is ignored if the graphical type is not a circle (2-5), curve (6), symbol (7-9,11), or text (10) or if the plotter or display does not support the facility. In the case of symbols, the number of the symbol to be taken from the hardware symbol library is in from the H/W entry.
1	2	use GKS line type 2 (dashed)
2	4	use GKS line type 3 (dotted)
3	8	use GKS line type 4 (dashed - dotted)
10	1024	for graphical type 12 (fill area); when using patterned fill areas clip any symbols at edge of area for graphical types 1 and 6 (lines and curves); when using a line pattern, then continue the phase of the pattern through any invisible segments, rather than re-starting
11	2048	when this feature is rendered into a perspective view it is to "stand up" in the view. This flag may only be set for symbols, texts, symbol strings, line features with no line pattern, and fill areas that do not contain pattern fill
12	4096	when this feature is to "stand up" in a perspective view, if this bit is clear, then the bottom of the feature follows the ground; if the bit is set, the bottom of the feature is straight between data points
13	8192	when this feature is to "stand up" in a perspective view, if this bit is clear, then the top of the feature follows the ground; if the bit is set, the bottom of the feature is straight between data points
14	16384	when this bit is set, features with this feature code will not be zoomed in LITES2, but will always be displayed at the same size on the screen - at the FRT size taken to be screen mm. This applies to symbol heights, text heights, line thicknesses and line spacing in fill areas.
15	32768	when this bit is set, lines, curves, and areas used as

			part of the prioritised representation of a symbol or text	
			will follow a 'complex' (more closely fitting) boundary of	
			the symbol of text. When unset, a 'simple' boundary (just	
			a box) will be used.	

Notes

- If none of bits 1, 2 or 3 are set, or if more than one is set, GKS line type 1 (solid) is used.
- Not all hardware has the capability of drawing any or all of these line types.
- Setting bit 10 for areas may cause FPP and LITES2 to take considerable time to draw patterned fill areas. This may be acceptable for final plots, but it may be desirable to DISABLE PATTERN when using these programs before the final plot stage is reached. Alternatively a similar FRT entry without bit 10 set may be used during the development stage.
- bits 12 and 13 are only relevant to lines and fill areas

FRT Parameter 9 - H/W

This is an optional column which contains an integer indicating either the hardware tool to be used when drawing, or the symbol from the hardware symbol library to be used (if bottom bit of flag word is set). For Ferranti master plotters (one of the devices presently supported that have this facility) this integer would be built up as follows:

0 - 63 slug number

add 100 if tangential control is to be used
add 1000 if magnification is to be used

If the optional entry in the FRT is omitted the default of 0 is used.

FRT Parameter 10 - Description

This is an optional column where a descriptive text of the feature can be included to assist the operator or other users in interpreting the FRT file.

The Priority Table of the Feature Representation Table

This is an optional section of the feature representation table which provides the specifications of how to draw features when using multiple passes.

Each record is typed in free format and comprises of up to 19 parameters or columns of information. There is one record or line associated with each priority definition:

```
!
!     FC      Prio    Repr    Prio    Repr    Prio    Repr
!
! a 'b' road that goes under a motorway
PRIO  6100    4       6100
!
! a 'b' road that goes over a motorway
PRIO  6105    9       6100
!
! a motorway drawn with a thick black line, a thinner blue (infill)
! and a thin central line
PRIO  6210    6       6910    7       6911    8       6912
!
! other features are all drawn first
PRIO  DEFAULT 2
```

PRIORITY Parameter 1 - the Record Identifier

The identifier

'PRIORITY'

is mandatory and must be the first parameter in the record. It may be abbreviated to 'PRIO'.

PRIORITY Parameter 2 - the Default Identifier or the Feature Code

The identifier

'DEFAULT'

indicates that the next parameter is the priority at which to draw features not referenced by a PRIORITY record.

Otherwise this parameter is an integer representing the feature code whose representation or drawing priority is being redefined. In this case the feature code must already have been defined in a FRT record.

Priority records must be in ascending order of this parameter in the FRT file.

PRIORITY Parameters 3 and 4 - a priority and representation pair

If parameter 2 was 'DEFAULT' then parameter 3 is the default priority at which to draw feature codes that are not referenced in the PRIORITY table. The rest of the line is then treated as a comment.

Otherwise this pair of integers represent a priority and a feature code that are to be used for one pass of drawing features with the feature code of parameter 2. The representation feature code must previously have been defined in the FRT table.

Generally, the graphical type of the representation feature code must be of the same type as the master feature code, but the following combinations are allowed:

- Lines can be represented by symbol strings
- Symbol strings can be represented by lines
- Area (boundaries) can be represented by symbol strings
- Area (boundaries) can be represented by lines
- Texts and symbols can be represented by lines, curves, or areas. In this case, a boundary around the text or symbol is drawn.

PRIORITY Parameters 5 to 18 - optional priority and representation pairs

Further pairs of integers representing other priorities and feature codes to be used in multi-pass drawing of features with the master feature code.

The representation feature codes must already have been defined in the FRT records, and their graphical types must conform to the rules given above.

The priorities must be in increasing order across the record, although two or more priorities may be equal. In this case it should not matter which representation is drawn first.

When using lines, curves, or areas to represent texts or symbols (often to 'blank' an area behind the text or symbol), then the SIZE and FLAG fields in the FRT record for the line, curve, or area are used to control the offset of the boundary, and whether the shape of the boundary is 'complex' or 'simple'. The line, curve, or area may be patterned, filled, or whatever; the only limitation being that hatched areas always use a default hatch spacing, since the SIZE field is already used to specify the offset.

The Symbol Code Table (SCT) of the Feature Representation Table

This section of the FRT is mandatory and relates directly to the descriptive feature codes used with lines which compose the symbols and text characters in the SRI files and TRI files.

The SCT can be considered as the specialised FRT for the symbols in the SRI files and text in the TRI. In the majority of cases, and for elementary use, the SCT only serves to define the graphical type of the feature codes contained in the SRI and TRI. The graphical type values are exactly the same as those described above. The layout of the SCT is identical to that of the main FRT, as it contains the same parameters:

!	FC	GT	COL	WIDTH	SIZE	SC	FLAG	H/W	DESCRIPTION
!									
SCT	1	1	0	0	0	0			line
SCT	2	6	3	0	0	0			curve
SCT	3	5	1	0	0	0			circle

The first parameter which was 'FRT' in records of the FRT section is replaced by 'SCT' to identify the record as part of the Symbol Component Table.

More advanced use of the SCTs can include the use of multiple colours and line widths in the creation of symbols. This is achieved by composing in the SRI and TRI files symbols or text characters which have several line features each possessing different descriptive feature codes. In this way, the definition of colours or linewidth or graphical type for the different feature codes relates to the descriptions given in the relevant columns of the SCT.

When drawing texts and symbols (including those in patterned lines), the characteristics of the line used to draw a particular component is determined by the following rules:

1. The colour is taken from the entry in the SCT, unless it is 0, when the colour from the FRT entry is used. If this is 0, colour 1 is used.

2. If the device is capable of drawing variable weight lines, the thickness is taken from the width entry in the SCT, unless it is 0, when the width in the FRT entry is used. If this is 0, then the line is drawn with the minimum line thickness.

3. For fill areas, the hatch spacing is taken from the size entry in the SCT; if this is 0, then the default spacing is used.

4. The hardware line type (if applicable on the hardware) is taken from the flags entry in the SCT, if that entry exists. If it does not exist, then it is taken from the flag entry in the FRT for the symbol or text. If this does not exist then a default of 0 is used.

5. The hardware tool to use is determined in a similar manner to the hardware line type.

6. Flag bit 0 (value 1) in the SCT can be used to indicate a hardware circle or curve, as for the FRT flags field.

Instructions for the creation of symbols is given in the section dealing with SRI files below.

Examples of the FRT and SCT components of the FRT file can be found in Appendix A.

The Group Table of the Feature Representation Table

This is an optional section of the feature representation table which is intended to speed up the selection or deselection of groups of feature codes when operating LITES2 or FPP. By grouping together several different feature types by their descriptive feature codes and referring to them under a 'global' group name it is possible to identify the group quickly and easily. The group information is supplied as ASCII character information in four tabular columns. For each group entry, there is one record within the group table:

```
! Group Definitions
!               NAME              FC          DESCRIPTION
GROUP           Isolators         1-10        ! Locked isolators
GROUP           Pylons            120-124     ! 300Kv Pylons
GROUP           Sub__stations      21,31,41    ! Local
sub-stations
!
```

Each record is typed in free-format - ie, at least one space or tab character between columns. Appendix A to this document details a typical feature representation table from which can be seen an example group section within the full table.

Each of the four parameters will now be explained in the same sequence as they appear from left to right within each group record.

Group Parameter 1 - the Record Identifier

In order to confirm that the current record belongs to the group section, it is necessary to include the identifier:

'GROUP'

This parameter is mandatory and must be the first parameter within the GROUP record.

Group Parameter 2 - the group name

This parameter takes on a user defined keyword which identifies the group of feature codes being aggregated. The total number of characters comprising all group names must accumulate to no more than 240 characters (eg 30 group names of eight characters would be acceptable). Typical names that might be chosen include:

WATER
HYPSOGRAPHY
REVISED
BUILDINGS

Group parameter 3 - the Group Feature Code Range

This parameter defines the feature codes or range of feature codes which should be aggregated to create the group. Each feature code to be included is separated by a comma and ranges of feature codes can be defined by the start of range and end of range separated by a dash '-'. Thus for example,

GROUP WATER 1, 3, 5, 7-9

will class feature codes 1, 3, 5, 7, 8, 9 in the class WATER which may be selected or deselected using the class name, in LITES2 or FPP.

Group Parameter 4 - the Optional Description

A description of the group may be entered at the end of the line.

The Pattern Table of the Feature Representation Table

This is an optional section of the feature representation table which provides the specifications of patterned lines associated with linear features which are to be displayed in symbol form.

The pattern table is required if a secondary code was given for any feature codes with graphical type 1-6 in the main body of the FRT.

Each record is typed in free format and comprises 12 parameters or columns of information. There is one record or line associated with each patterned line.

The first parameter (or column) is the pattern record identifier and it is mandatory. It takes the following identifier in all cases:

'PATTERN'

The second parameter takes an integer number which is a pattern index; this relates to the identifier for the pattern used as the SC (secondary code) parameter for linear features in the FRT. Patterns must be ordered and sorted sequentially in ascending numeric order.

The remaining columns can be split into 5 general categories. These categories are:

Unit codes This relates to the number of the symbol to be used in the relevant symbol library.

Repeat entries Define the number of times that a component is to be repeated within the overall pattern cycle.

Size entries Define the length of the component in order to create a pecked line feature.

Width Defines the width (size) of symbols being plotted along a line feature.

Flag Gives additional control of the pattern definition

Offset Allows the pattern cycle to start part way through

Any line pattern is governed by its 'overall size'. This is the length, in mm, in which a complete cycle of the pattern is generated and which LITES2 and FPP can repeat to create the appearance of continuous, patterned line features.

The pattern facility is capable of producing either pecked lines or more complex linework whose symbols are contained within the symbol library. The pattern technique works on the philosophy of the user defining the unit lengths of lines or sizes of symbols to be displayed or plotted, with a repeated cycle provided by the 'overall size' parameter. Any gaps which remain are distributed equally between all the components of the pattern. Major and minor size parameters are used to input the length of the dashes in a line segment. The size of the gaps is determined by subtracting the total line length from the overall size entry and dividing it by the total number of features in the segment.

The following section gives a complete description of the columns contained in the pattern table, of which there is an example in Appendix B. Since there are optionally up to two pattern components within a single cycle of the pattern, the two are distinguished by the descriptor major or minor component. The columns are defined as follows:

	Patt Index	**Maj UC**	**Min UC**	**Maj Rep**	**Min Rep**	**Overall Size**	**Maj Size**	**Min Size**	**Major Width**	**Minor Width**	**Flags**	**Offset**
PATTERN	10	0	15	0	1	8.0	3.0	1.0	0.0	2.0	136	6.0

PATTERN Statement is the record identifier.

Pattern Index This figure corresponds with the number specified by the SC entry in the FRT.

Major Unit Code This number represents the symbol number of a symbol in the relevant symbol library to be used as the major component. 'Zero' produces a line instead of a symbol. The symbol is drawn where the middle of the corresponding line would have been.

Minor Unit Code This number represents the symbol number of a symbol in the relevant symbol library to be used as the minor component. 'Zero' produces a line instead of a symbol. The symbol is drawn where the middle of the corresponding line would have been.

Major Repeat This indicates the number of times that component is to be repeated in the overall size. Therefore a repeat equal to 1 will result in two occurrences of the major component.

Minor Repeat This is a similar command to parameter five above for the minor component.

Overall Size This figure is the maximum size in which the full pattern must be contained. It is imperative that the total of major and minor sizes is not greater than the overall size figure. Units are in millimetres and the parameter given is a real (or decimal) number.

Major Size The unit length of the major component is entered in this section. Units are in millimetres. A size of zero is valid - the line will not be drawn but may still be substituted by a symbol.

Minor Size The unit length of the minor component is entered under this entry. As for the major size, zero is valid, but if major and minor size are both zero, then at least one must be substituted by a symbol. The gaps are calculated from the amount left over after the major and minor components are taken out of the overall size.

Major Width This defines the size of the symbol, substituted in the major UC, in millimetres. Note that the length of the unit (Major Size) does not affect the symbol size - only its position.

Minor Width Defines the corresponding size of the symbol to be substituted in the minor component in millimetres.

Flags word This is bit significant (16 bits). This means that the following values should be added together as required to obtain a value which will have the desired effect. The flag value may be specified in

hexadecimal if required by prefixing the number with ^X (or octal with ^O, or binary with ^B). This is likely to be more meaningful than decimal especially if the large values are used (e.g. 32768 is ^X8000). If omitted the value is taken as zero. Note also the use of flag bit 10 (value 1024) in the FRT section, to indicate that a pattern should maintain its phase across invisible segments of line. By default, the pattern is restarted after an invisible segment.

Bit	Value	Effect of flag
0	1	phase restore at each vertex
1	2	scale pattern to fit line length
2	4	invert major symbol in alternately
3	8	invert minor symbol in alternately
4	16	do not rotate major symbol (i.e. always draw it horizontal)
5	32	do not rotate minor symbol (i.e. always draw it horizontal)
6	64	use hardware to draw major symbol
7	128	use hardware to draw minor symbol
8	256	stretch the major symbol if the pattern is scaled (bit 1)
9	512	stretch the minor symbol if the pattern is scaled (bit 1)
10	1024	fit whole number of major components when scaling (bit 1)
11	2048	fit whole number of minor components when scaling (bit 1)
12	4096	draw major dash as well as symbol (if no symbol, omit major dash)
13	8192	draw minor dash as well as symbol (if no symbol, omit dash)
15	32768	plot pattern using hardware if possible

Other bits are not used at present.

NOTE

Bit 1 (2) controls whether any scaling of the pattern to fit the line being drawn takes place. Some other bits are only meaningful when used in combination with this bit.

Bit 0 (1) used alone means restart the pattern at each vertex of the line, but when used together with bit 1 (2) means fit a whole number of patterns between vertices.

If either (or both) of bits 10 (1024) and 11 (2048) are set in addition to bit 1 (and possibly bit 0), then rather than fitting a whole number of patterns, merely a whole number of major or minor components respectively (including the following gap)

must fit. If just one of bits 10 (1024) or 11 (2048) is set, then the other component is allowed to span vertices.

Bits 8 (256) and 9 (512) allow the major or minor symbol to be stretched in the direction along the line when the pattern is scaled. This is likely to be most useful when adjacent symbols are supposed to touch each other, and this is to be maintained while scaling.

Bits 12 (4096) and 13 (8192) are used to allow the plotting of both a dash **and** a symbol for a given component, or alternatively to suppress the component completely (while still using up the space allocated to it). This latter effect is achieved when the corresponding UC is set to zero.

Bit 15 (32768) will only have any effect if the program in use, and the plotter hardware support hardware patterned lines. If not, or if the pattern is not one supported by the hardware, then a software pattern will be used.

Offset This specifies the amount of the pattern in millimetres that should be taken to have occurred before the first visible point. It allows the pattern to start part way through rather than with the first major component, and should be in the range 0.0 (no offset) to (less than) Overall Size. If omitted, an offset of 0.0 is assumed.

Specification of 0 'zero' in any position within a pattern description will result in that description being ignored.

The Fill Area Table of the Feature Representation Table

This is an optional section of the feature representation table which provides the specifications of patterned fill areas. Such patterns can be designed in such a way as to create the impression of symbol fill if desired. Note that fill area attributes are also used to draw the perimeter line. The number of data points allowed in these areas may be controlled by defining logical name LSL$FILL_POINTSMAX to be the required number (default 8192, minimum 100). Similarly the maximum number of times which a scan line may cut the polygon may be controlled using LSL$FILL_CUTSMAX (default 100, minimum 10).

The fill area table is required if a secondary code of less than -1 was given for any feature codes with graphical type 12 in the main body of the FRT.

Each record is typed in free format and comprises 5 parameters or columns of information. There is one record or line associated with each fill area pattern:

```
!
!     INDEX   HATCH   FILLSC  DESCRIPTION
!
FILL   -2       3       2     ! Urban areas
FILL   -3       4       3     ! Deciduous woodland
```

Fill Parameter 1 - the Record Identifier

The identifier

'FILL'

is mandatory and must be the first parameter in the record.

Fill Parameter 2 - the Fill Index

This parameter is the fill index and corresponds to the number used as the SC (secondary code) parameter for fill area patterns in the FRT. Note that it is always a negative number and must be less than -1. Fill area patterns must be ordered and sorted sequentially in ascending numeric order, taking account of the sign.

Fill Parameter 3 - the Hatch Style

This parameter is analogous to the FRT SC entries greater than 0 for graphical type 12. The following values may be used to specify the hatch style:

Value	Hatch style effect
1	Hatch with horizontal patterned lines
2	Hatch with vertical patterned lines
3	Hatch with 45 degree patterned lines
4	Hatch with -45 degree patterned lines
5	Cross hatch with horizontal and vertical lines
6	Cross hatch with 45 and -45 degree lines
101-106	as 1-6 above but include the perimeter line
10000-13600	Number is 10000 + angle in tenths of degrees
20000-23600	as 10000-13600 but include perimeter line

Note that GKS internal style 2 (PATTERN) is not supported by FRTLIB.

The spacing between lines in hatched styles 1-6 and 101-106 is determined by the FRT size entry. If this is 0.0, then the default value of 5.0mm is used.

The colour of the area-fill is determined by the colour definition for that particular feature code.

The current line attributes are used for the perimeter line.

Fill Parameter 4 - the Line Pattern Number

This parameter relates to the line pattern number in the PATTERN section of the FRT file.

Fill Parameter 5 - the Optional Description

A description of the fill area pattern may be entered at the end of the line and will be ignored.

The Attribute Code Definition (ACD) part of the Feature Representation Table

The primary description of a feature within the IFF file is held in its feature code or FC entry (see above). Secondary characteristics of the feature are held as Ancillary Codes (ACs). The points within a feature may also have characteristics associated with them - these are called the attributes of the point. (Logically the X and Y coordinates are attributes of the point, but as they must be present in every point the attributes of a point are often considered to be in addition to its X and Y coordinates).

The form of ACs and point attributes are similar and consist of :-

1. a number in the range 0-32767.

   This is generally called the "type" when referring to ACs and the "code" when referring to point attributes.

   It is an identifier that specifies the characteristic that is being referred to eg

   3 is a floating point height
   5 is a right hand boundary code

2. a value

   the form of the value depends on the data type of the attribute or AC. The possible data types are:-

   a) Integer

This is an integer or "whole" number

b) Real

This is a real or "floating point" number

c) Character

This is a string of 4 characters. In ACD (**A**ttribute **C**ode **D**efinition) entries, the characters *must* be surrounded by single quotation marks e.g.

'abcd'

If less than four characters are specified, then the string is padded with spaces.

When comparisons are made between two values of this type (for example when data is being verified on input to a program) then each of the four characters is compared independently.

d) Date

This is the date part of a VAX/VMS date/time string. It consists of a one or two digit integer representing the day of the month (in the range 1 - 31), followed by '-', followed by three upper case letters representing the month (JAN,FEB...DEC), followed by '-', followed by a four digit integer representing the year (eg 1987). Any part of the date can be missed out, but not the '-'s. If any part is missing, then the field is filled in from the current date. In particular the date '--' is interpreted as 'today'.
The valid range of dates is from 17-NOV-1858 to 31-DEC-9999

Examples of valid dates are

```
 2-JAN-1987
31-DEC-1899
--                          today's date
```

e) Time

This is the time part of a VAX/VMS date/time string. It consists of a one or two digit integer representing the hour of the day (in the range 0 - 23), followed by ':', followed by a two digit integer representing the minute of the hour, followed by ':', followed by a real number with two digits before the decimal point representing the seconds of the minute. Any part of the time can be missed out, but not the ':'s. If any part is missing, then the field is filled in from the current time. In particular the time '::' is interpreted as 'just now'.
The valid range of times is from 00:00:00.00 to 23:59:59.99

Examples of valid times are

```
02:23:45.76
12:00:00.00
::                          just now
```

In addition ACs (but not point attributes) can also have

3. an optional text string

The Attribute Code Definition table allows specific AC and attribute types to be defined as :-

1. being one of the five data types specified above (integer, real, character, date or time)

2. having a name (of up to 20 characters) which can be used to refer to the attribute type, rather than having to remember its numeric type.

3. the allowable range of values for the attribute. This information is used to verify values on input.

4. for real data types, the "granularity" of the values. This is the least significant step in the data. For example it may be 0.1 for ground heights that are specified in metres, or 0.001 for the diameter of a water pipe, again specified in metres. This information is used to format the value when it is being displayed.

The Attribute Code Definitions are divided into 33 tables (tables 0 - 32) each containing up to 1000 code definitions (codes 0 - 999), not all of which need be present. (Note table 32 can only have codes up to 767, not 999). Table 0 is for general use and is mostly predefined by LSL.

Further tables may be defined in the ACD section of the FRT files.

NOTE

> As the codes 100 - 139 have in the past been allocated to specific users, it is possible to define this part of table 0 in the FRT file.
>
> It is recommended that this is only done on these sites that historically used these AC codes.

The number of Attribute Codes which may be defined in an FRT file is 50 by default. If more than this are required, then define logical name LSL$FRT_ACDMAX to be the required number (in the range 0-32767) before running the program which reads the FRT. e.g.

```
$ Define LSL$FRT_ACDMAX 200 !  allow 200 ACDs
```

At sites where a large number of ACDs are used regularly, this logical name may be set system wide to a suitable default. If the logical name is set to an invalid number, then 50 will be assumed.

Each record is typed in free format and comprises up to 7 parameters or columns of information.

```
!
! Attribute codes
! ===============
!
!
ACD TABLE     1               ! survey information
!
!      Code   Name            Minimum         Maximum       Min step
!
ACD D   1     SURVEY_DATE     17-NOV-1858     31-DEC-9999
ACD D   2     CHANGE_DATE     17-NOV-1858     31-DEC-9999
ACD I   3     CHANGE_TYPE     1               6
ACD I   4     CAPTURE_XY      0               100
ACD I   5     CAPTURE_Z       0               100
ACD I   6     SECURITY_MARKER 0               1
!
ACD TABLE     9               ! other examples
!
!      Code   Name            Minimum         Maximum
!
ACD C   1     FOUR_CHARS      '    '          '~~~~'
ACD T   2     TIME_EXAMPLE    00:00:00.0      23:59:59.99
ACD R   3     REAL_EXAMPLE       0.0          1000.0          .01
ACD I   4     INTEGER_EXAMPLE    0
ACD R   5     ANOTHER_EXAMPLE    ?            0
```

Each of the seven parameters will now be explained in the same sequence as they appear from left to right within each group record.

ACD Parameter 1 - the Record Identifier

In order to confirm that the current record belongs to the ACD section, it is necessary to include the identifier:

```
'ACD'
```

This parameter is mandatory and must be the first parameter within the ACD record.

ACD Parameter 2 - the data type entry

This mandatory parameter is a keyword that specifies how the rest of the parameters on the line have to be identified. The available keywords are :-

- o TABLE - the start of a new ACD table
- o I - an entry in the ACD table specified as integer
- o R - an entry in the ACD table specified as real
- o C - an entry in the ACD table specified as character
- o D - an entry in the ACD table specified as date
- o T - an entry in the ACD table specified as time

For the keyword TABLE, only one more (mandatory) parameter is required which is an integer in the range 1 - 32 which specifies the table that following ACDs will be put in, until the next ACD TABLE entry in the FRT. This integer must not be duplicated in ACD TABLE entries and all ACD TABLE entries must occur in ascending order.

NOTE

> Those sites that have been allocated AC codes in the range 100-139 may specify TABLE 0, to allow them to define the ACs within this range.

If any other ACD entry occurs before an ACD TABLE entry is given, then an occurrence of the entry

```
ACD TABLE 1
```

is assumed.

ACD Parameter 3 - the code entry

This mandatory parameter is an integer in the range 0 - 999 (767 in the case of table 32). It is used to derive the type of the AC or the attribute code. This is achieved by multiplying the current table number by 1000 and adding on this entry. Thus in the example shown above, TABLE 1 defines attributes 1001, 1002, ... 1006, while TABLE 9 defines attribute codes 9001, 9002 ... 9005.

Within any one table the code entries must be unique, and in ascending order.

ACD Parameter 4 - the name entry

This mandatory parameter consists of a name by which the attribute can be identified. The name contains only letters and the character '_'. Names can be up to 20 characters long and must be unique over all the ACD entries. Note that this includes the names used in the default definitions in table 0 (see below). Note also that the case of the letters is ignored in making these comparisons, so the name:

DHEIGHT is considered to be the same as the name dheight

None of the following entries are mandatory. If they are missing (or are specified wrongly) then default values are used. If one entry is to be defaulted, but a following one is to be specified, the defaulted one cannot just be missed out (due to the free format of the data) so the character '?' should be inserted.

ACD Parameter 5 - the minimum value

This entry contains the minimum that the value of an AC or point attribute can take. Its form depends on the data type of the attribute being defined (see the specification of the format of various data types above). The purpose of this entry, and of the maximum value, is to allow programs to validate values when they are entered.

ACD Parameter 6 - the maximum value

This entry contains the maximum that the value of an AC or point attribute can take. Its form depends on the data type of the attribute being defined (see the specification of the format of various data types above).

ACD Parameter 7 - the minimum step

This entry contains the minimum increment that is significant in the value of the attribute being defined. It is only meaningful for attributes whose data type is real. It should take the form of a real number. If the value 0.0 is entered, then the minimum possible increment is assumed, within the constraints of real*4 (four byte) floating point arithmetic.

ACD default values

The default values used for missing parameters are given in the table below, for the various data types

Data type	Minimum value	Maximum value	Minimum step
I	-2147483647	+2147483647	
R	-1.0E37	+1.0E37	0.0
C	' '	'~~~~'	
D	17-NOV-1858	31-DEC-9999	
T	00:00:00.00	23:59:59.99	

The LSL default attribute code definition

LSL has predefined some attribute codes. These should not be specified in the FRT file - they are always available - but are the equivalent of the following ACD entries

```
!
!
ACD TABLE     0              ! default table
!
!      Code   Name            Minimum         Maximum
!
ACD I   1     Secondary_FC      0           32767
ACD I   2     Contour           -2147483647 2147483647
ACD R   3     Height            -1.0E37     1.0E37
ACD I   4     LH_boundary       0           32767
ACD I   5     RH_boundary       0           32767
ACD I   6     Text              0           32767
ACD I   7     DFAD_FADT         0           0
ACD I   8     DFAD_ACC          0           0
ACD I   9     Parent_FSN        0           65535
ACD I  10     RELHT_START       0           100
ACD I  11     RELHT_END         0           100
ACD R  80     Cliff_left        -1.0E37     +1.0E37
ACD R  81     Cliff_right       -1.0E37     +1.0E37
ACD R  82     Polygon_info      -1.0E37     +1.0E37
ACD R  91     X                 -1.0E37     +1.0E37
ACD R  92     Y                 -1.0E37     +1.0E37
ACD R  93     Z                 -1.0E37     +1.0E37
ACD R  94     ZB                -1.0E37     +1.0E37
ACD R  95     ZC                -1.0E37     +1.0E37
ACD R  96     ZD                -1.0E37     +1.0E37
ACD R  97     Dheight           -1.0E37     +1.0E37
```

The Symbol Representation IFF file (SRI)

Overview

The SRI is an IFF file which provides a graphics library in order to hold all the symbol features for the series of maps that are to be digitised and displayed.

Creating a template SRI file

Before the symbols can be digitised into the IFF file it is desirable that a template file be established in order that the symbols used in production will be of the correct size and scale. The SRI file is said to be **normalised** - viz the control points fall in the range -1 to +1.

This template SRI file can be created using the IMP utility ISTART.

Use ISTART in default template mode and specify the /USER_ORIGIN qualifier. Specify a scale of 1. It is necessary that the control point values specified in response to ISTART prompts are exactly as follows:

Top left or NW	-1.0	1.0
Bottom left or SW	-1.0	-1.0
Bottom right or SE	1.0	-1.0
Top right or NE	1.0	1.0

Specify the origin offset as (0.0,0.0)

This results in the centre point of the file lying at (0.0,0.0).

Digitising the SRI File

The input of the symbols into the SRI file is generally performed using LITES2. In order to read an IFF file in LITES2 an FRT, TRI and SRI must be specified. An FRT must be set up so that it includes all the Graphical Types that may be required to digitise the symbols in the SRI file.

As the single SRI file must contain all the symbols for a specific digitising job, it is important that the user constructs some structure in the file so that it can be easily altered or updated by another user at a later date. It is suggested that symbols are divided up by layer within the SRI using criteria such as grouping symbols for patterned lines and point symbols into separate layers.

Layers can be created and set whilst in LITES2 ready state using:

```
CREATE LAYER  n and
SET LAYER     n
```

NOTE

Symbols digitised into layer 0 will be ignored by Laser-Scan plotting software and most IFF utilities, (see the IFF User Guide). Do not use layer 0 in SRI or TRI files, except possibly to contain a grid or box to aid the positioning of symbols.

Once a layer has been created it is only necessary to supply the SET LAYER n command before commencing digitising.

Feature Codes in the SRI file

The feature code of lines in the SRI is directly related to the FC entry in the SCT section of the FRT. This enables the plotting programs to add colour, curves or width to the symbol being drawn or displayed. Additionally (and more important) symbols can be constructed from lines, curves, arcs, etc if desired. Symbols within symbols are not allowed; neither are area fill patterns.

Before digitising into a certain feature code, the following LITES2 commands must be entered:

```
SET FC n
```

Feature Serial Numbers (FSN)

The FSN (**F**eature **S**erial **N**umber) of a feature in an SRI file is directly related to the secondary code (SC) entry in the main body of the FRT for isolated symbols. When a symbol is used to form part of a patterned line, then the feature serial number of the symbol relates to the major and minor components within the PATTERN section of the FRT. Each time a symbol is plotted, the program associates all lines in the SRI with the same relevant FSN in order to create it.

Since some symbols have complex linework, many lines may have to be digitised in order to create the full symbol feature. The default LITES2 convention is to increase the FSN by one for each line digitised. Therefore, in order to create numerous lines of the same FSN using LITES2 the

```
SET FSN n
```

command must be used before commencing digitising each line, where n corresponds to the SC entry in the FRT.

Guidelines for SRI Files

The following should be noted before attempting to create an SRI file:

- As the SRI file becomes increasingly complicated, it should be noted that the symbols can be selected by layer, feature code or individually by Feature Serial Number.

- A given symbol representation may consist of one or several features having the same Feature Serial Number (FSN) which is the symbol index number.

- Invisible line segments may be used in features - the effect is usually identical to beginning a new feature, but less space in the file is required. In a symbol component which is a filled area, invisible segments will not cause a new area to be started, but the points afterwards will continue to be part of the original area. This is useful if a hatching type including the perimeter line is used, since these segments will not be drawn.

- The features in the SRI must be in ascending FSN order. A file may be put into ascending FSN order (allowing for duplicate FSNs) using the IMP utility ISORT. (The /DUPLICATES option should be used, this is the default - see the IMP Reference Manual).

- Features making up a symbol may be digitised at enlarged scale, and may be normalised by a special program, SRINORM. For detailed information about SRINORM see the SRINORM User Guide. Even if an enlarged digitising scale is used, features should be digitised with the coordinate origin (0,0) at the required symbol locating point.

- In order that the height value of a symbol, when plotted using either LITES2 or FPP, is equivalent to the value specified in the FRT "SIZE" column, a set procedure must be followed when digitising symbols into the SRI file.

Post Processing SRI Files

Once the symbols have been digitised and the operator has finished in LITES2, it is necessary to sort the lines within the SRI file so that all FSNs are contiguous and held in ascending order. This is achieved with the IMP utility ISORT:

```
$ ISORT file-spec file-spec
```

If the symbols were digitised at an enlarged scale SRINORM should be used to rescale the feature coordinates to fit the unit square:

```
$ SRINORM file-spec file-spec
```

The Text Representation IFF File (TRI)

Overview

The text representation file is the equivalent of the symbol representation file, except that it is designed specifically to hold all text fonts. As with the SRI, a normalised file is required, (this is explained for the SRI above). The input mechanism for text fonts within the TRI files differs slightly; these differences are discussed below.

Programs able to display hardware text (using the hardware flag bit in the FRT file) may not need character shapes in the TRI file, but the character width information is still used to allow justification of text string. Plotting using the PostScript language (or Display PostScript on an X-Windows display) can sometimes determine character widths from other sources, so an empty TRI file may suffice. See the section below on the PostScript Font List File (PSFONTLIST) for details of specifying PostScript fonts.

Digitising Text Fonts

All linework for a particular font must be digitised into a separate layer. The layer number used should relate to the SC entry within the FRT for text features described by graphical type 10. Currently, the layer numbers are limited in the range 1-127. If the SC calls up a layer in the TRI that does not exist then the following error message is returned:

```
FRTLIB Font n is not defined.
```

The feature serial number that is assigned to linework works on a similar philosophy to SRIs in that for a specific text character, all component features of a single character must have the same feature serial number.

TRI files differ from SRI files in that the feature serial numbers of the text character line components must be equal to the ASCII value for that character. For example, FSN 65 will relate to all the line work which composes an uppercase 'A' character. The range of ASCII values is from 0 to 255, which enables the creation of special characters, explained below. The ASCII character set is presented in Appendix C.

All FSNs have to be assigned a type 3 AC code using the LITES2 command:

```
ADD HEIGHT n
```

where n is equivalent to the spacing width of the character. The AC must appear in only one of the features with a given FSN. The spacing

width is the space between the defining point of this character and that of the next character in a text string. The spacing width is not necessarily related to the actual size of the character, for example if a character has a spacing width of 0.0, then it will be overprinted by the next character. This mechanism may be used to add diacritical marks (such as accents) to characters.

If a uniformly spaced typestyle is required, similar in appearance to a typewriter, then n can be set to a constant value.

The feature codes used in the TRI files will be equivalent to those indicated in the SCT section of the main FRT file. It is important that only feature codes with a linear graphical type (GT) are used GT = 1-6 and graphical type GT 12 for fill areas.

As indicated with the guidelines for digitising symbols into an SRI file, the user should choose a point on the text character to be the locating point. This is traditionally specified as the bottom left hand corner of the text character.

See diagram Appendix B.

The SRINORM command qualifier /CHARACTER=n is required, where n represents the ASCII value of the character that is to be used as the template size which all the other characters will be scaled to.

For example, the following command could be given:

```
$ SRINORM FONT.TRI FONT.TRI/CHARACTER=65
```

This would have the result that all the characters digitised into the TRI file would be scaled according to the size of the character digitised with an FSN equivalent of 65, namely uppercase "A", as in the example illustrated above.

For further information about the SRINORM utility see the SRINORM User Guide.

Digitising Special Characters

Two characters in each font are treated specially. These are ASCII values 0 (null) and 32 (space). Character 0 should not be included in a font - even if it is, any occurrence of the null character in a text string will be ignored. Character 32 is always plotted as a space, irrespective of whatever linework is digitised for it.

Example: Space character FSN 32 (ie ASCII 32).

The space character can be digitised with a single line, although the important factor is defined by the AC 3 entry which will dictate the size of the gap left between the previous and ensuing character. As the FSN is equal to 32, the software will automatically ignore any feature and only read the AC entry.

Example: special characters such as circumflex and acute accents

ASCII values in the range 128-255 are available for user defined text symbols. The text symbol can be digitised by setting the relevant FSN. However, in order to display the special text symbol the user must subtract 128 from the FSN number that was used to digitise the symbol. The resulting value can be looked up in the ASCII table. For example FSN 229-128 = ASCII value 101 = e. The user can then call up the special character by adding the prefix $ to that character = $e. For example:

  ^
Evesville Nord - special character block

                                                                ^
is typed in as "Ev$esville Nord" if FSN 229 was digitised as e.

An alternative approach would be to define the diacritical mark alone, for instance using character 222 = $^. If this character was given a spacing width of zero, then it would overprint the next character, and the above example would then be typed as "Ev$^esville Nord".

In order to obtain the $ sign in text, it is necessary to type $$.

There is no reason why characters in the range 128-255 cannot be used directly if a means can be found of inputting them (such as using the "Compose Character" key on a VT220). If this mechanism is used, then care should be taken if listing the IFF file in text format, as not all printers may be able to accept these characters.

The special handling of the $ character in text strings can be switched off by defining logical name LSL$DOLLAR_ESCAPE as 0. If this is done, then $ will be treated as an ordinary character. If the logical name is not defined, or is defined as 1, then $ will cause 128 to be added to the character value of the next character, as described above.

Example: Alternative space character (different width).

Some applications may require a space character with width different to the normal one. After choosing a suitable character number for the

alternative space, probably in the range 128-255, the problem is to

digitise some linework which will not be visible. The simplest thing to do is to use a 2-point invisible line, which can be created in LITES2 by the command sequence START, INVISIBLE, END, followed by finding the last (3rd) point of the line, and giving the command REMOVE.

Character substitution

There is a facility to substitute characters in all text strings passed to FRTLIB for drawing. Logical name LSL$SUBSTITUTE_CHARACTERS may be defined to be a string of characters. The first is substituted by the second, the third by the fourth, etc. For example, the string "?#|!" would replace '?' by '#', and '|' by '!'. The logical name is examined at the time the FRT file is read, so could for example be reset in LITES2 by re-reading the FRT. To turn off substitution, either deassign the logical name, or define it to be a single space.

Note that substitution of characters which already have a `special' meaning, such as `$' `{' `}' and `space', is likely to yield unpredictable results, as other programs expect these characters to behave in their usual way.

The PostScript Font List File (PSFONTLIST)

Overview

Programs LITES2MOTIF (LITES2 package) and FPPMOTIF (PLOTTING package), and also PostScript plotting from within LITES2MOTIF, are able to display text using PostScript built in fonts (the hardware flag bit in the FRT file must be specified).

The PostScript Font List File is the means by which a font number in the FRT file relates to a particular PostScript font. To use it, logical name LSL$PS_FONTLIST (or LSL$DPS_FONTLIST for LITES2MOTIF using Display PostScript) must point to the file. Missing parts of the file specification are filled in from the default 'LSL$FRT:.PSFONTLIST'. The PSFONTLIST file is an ordinary text file, containing directives for specifying details of each font.

An example PSFONTLIST file can be found in Appendix D, while two examples are supplied in the directory LSL$PUBLIC_ROOT:[MAPPING.EXAMPLES.FRT].

This section describes the directives which can be used in a PSFONTLIST file.

Comment lines

All blank lines in a PSFONTLIST file are ignored, as are any lines whose first significant character is an exclamation mark (!). Such lines can be used to supply explanatory comments.

FONT directive

This directive is the only compulsory one, and must appear for each font being defined. Any other directives for this font should follow, in any order. It specifies the FRT font number, and the name of the font. In the absence of other directives, the name should be the PostScript name for the font (in the correct case).

```
          FONT number name
    e.g. FONT 1 Times-Roman
```

FIND directive

Specifies a sequence of PostScript to be used to find the required font, overriding the default of applying the 'findfont' operator to the name given in the FONT directive. This directive will not normally be required. The sequence of commands must leave a PostScript font directory on the stack. As in this example, it may include commands to scale or otherwise manipulate the font, but note that the SCALE directive should normally be used for scaling - widths

and kerning information from an AFM file will be incorrect if scaling is performed by the FIND directive.

```
     FIND string
e.g. FIND /Souvenir-Light findfont 1.25 scalefont
```

SCALE directive

Specifies that the size of the font be multiplied by the factor given. This is often required because the design size of a PostScript font does not necessarily correspond to the height of the capital letters - it is more often the design inter-line spacing, so text is plotted smaller than expected.

```
     SCALE factor
e.g. SCALE 1.5
```

ENCODING directive

Specifies that the font be re-encoded. This means changing the mapping between the character codes in text strings, and the set of characters which are plotted. It is usually used to map accented characters into the range of ASCII codes between 128 and 255. The string following the directive is, formally speaking, a sequence of PostScript which when executed will leave an encoding vector on the stack. Less formally, this means that the name of a pre-defined encoding (e.g. ISOLatin1Encoding) will suffice. The ENCODING directive may be repeated for a particular font - the strings will all be appended, which would allow a complete 256 entry encoding vector to be specified explicitly. When a font is re-encoded, it needs to be given a new name. By default, this name is generated by appending '-ReENC' to the font's name, but the NEWNAME directive may be used to supply the new name.

```
     ENCODING string
e.g. ENCODING ISOLatin1Encoding
```

NEWNAME directive

Used to supply a new name for a font which is re-encoded, overriding the default of appending '-ReENC' to the font's name. This directive will not normally be required, unless perhaps using the same font several times with different encodings.

```
     NEWNAME name
e.g. NEWNAME Helvetica-ISOLatin1
```

KERNING directive

Specifies that kerning should be applied to this font. Kerning is the adjusting of inter-character spacing to produce a more pleasing effect. At present, only pair kerning is available (KERNING PAIR), which adjusts the spacing between particular pairs of characters. The details of what kerning to apply are contained in a separate 'font metric' file, which must be specified using the AFM directive.

KERNING subcommand
e.g. KERNING PAIR

AFM directive

Specifies the name of an '**A**dobe **F**ont **M**etric file'. Missing parts of the file specification are filled in from the default 'SYS$PS_FONT_METRICS:.AFM'. These files are usually supplied with any PostScript fonts on the system. Three sets of information are read from the file. Firstly the character width and encoding information. This is used to compute the widths of character strings for justification. If no AFM file is specified, the PostScript plotting will use widths from the TRI file (which may be incorrect), while Display PostScript will request widths from the interpreter (which may slow things down). Secondly, the pair kerning information is read. This is used if KERNING PAIR is specified. Thirdly, the composites information is read. This is used, if logical name LSL$COMPOSITE_CHARACTERS is defined as 1, to draw any named composite characters (e.g. {Zcaron}) encountered in text strings. Font metric files should be appropriate for the font encoding to be used, otherwise width and kerning information may be applied to the wrong characters.

AFM filename
e.g. AFM AVANTGARDE_BOOK

DIRECTION directive

Specifies that the writing direction for this font is not the default of left-to-right. The only alternative available is right-to-left (DIRECTION RIGHT_TO_LEFT). This directive is needed because PostScript fonts for right-to-left alphabets seem usually to be supplied with positive character widths that would result in them still being displayed from left-to-right. The directive causes all the character widths in the AFM file (which must also be specified) to be negated, and the characters to be displayed from right-to-left. In the case of a font which naturally displayed from right-to-left, the directive would not be needed.

DIRECTION subcommand
e.g. DIRECTION RIGHT_TO_LEFT

ARABIC directive

Specifies automatic glyph substitution in fonts for languages such as Arabic, in which a different letter shape must be used, depending on whether the character is Isolated, or occurs at the Start (Initial), Middle (Medial), or End (Final) of a word. Fonts for these languages have up to four glyphs for each letter. The ARABIC directive (repeated for each letter to form a table) specifies the input code in the character string, followed by the four codes for the different glyphs.

The following rules are applied in choosing the glyph for a letter

- Any codes encountered which are not present in the table will be treated as glyph numbers directly, thus providing access to characters such as spacing and punctuation.
- Characters in this table will be assumed to join on to their neighbours if the neighbours are also in the table.
- Characters whose Isolate and Initial forms are the same are assumed not to join to their successor, which therefore takes its Isolate or Initial form.
- Characters whose Isolate and Final forms are the same are assumed not to join to their predecessor, which therefore takes its Isolate or Final form.
- Non-spacing diacritical marks (this means that the glyph in the Isolate column has a zero width in the AFM file) are ignored when working out the connectivity of adjacent characters.
- If the glyph in the Initial, Medial, or Final column is zero, then the Isolate glyph is used instead.

The above rules give rise to some special cases for Arabic:

- The `Hamza' character does not join on to any others, so all 4 glyph variants should be the same.
- The spacing character `Tatweel' only has one glyph variant, but of course must join to adjacent characters, which is achieved by putting the glyph number in the Isolate column, and zero in the others.
- For non-spacing diacritical marks, such as `kasra', the Isolate column must contain the glyph number, while the other columns can contain zero, or the same or a different glyph.

```
      ARABIC input isolate final medial initial
 e.g. ARABIC 193 213 213 213 213 !  Hamza
      ARABIC 228 190 189 188 187 !  Lam
      ARABIC 224 64 0 0 0 !  Tatweel
      ARABIC 240 234 0 0 0 !  kasra
```

HEADER directive

This may appear anywhere in the PSFONTLIST file and does not apply to a particular font. It will normally appear at the top of the file. The supplied string is just passed through to the PostScript interpreter (and must therefore be valid PostScript). A typical use would be to define a new, named, encoding for use by the fonts following later. The HEADER directive may be repeated as many times as required. (The following example rather pointlessly defines MyEncoding as a synonym for StandardEncoding.)

```
          HEADER string
     e.g. HEADER /MyEncoding StandardEncoding def
```

Composite Characters

This facility is described here because it is only available when PostScript text, and a PSFONTLIST file and an AFM file are used. The AFM file for a PostScript font may describe several 'composite' characters. These are characters made up by superimposing two or more existing characters in the font, such as accented letters. Some composite characters may be encoded, so that they may be drawn merely by including the correct ASCII code in the text string, possibly using the '$' escape mechanism described above, but others are not, so they may only be referenced by their name (e.g. Zcaron) in the AFM file. The next section describes the mechanism for including such characters in text strings.

The use of composite characters must be enabled by defining logical name LSL$COMPOSITE_CHARACTERS to be 1, e.g.

```
$ Define LSL$COMPOSITE_CHARACTERS 1 !  enable composite characters
```

The mechanism is disabled if the logical name is not defined, or is defined as 0.

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.

APPENDIX A

SAMPLE FEATURE REPRESENTATION TABLE (FRT)

```
!  Feature Representation Table for contract
!
!  Author:
!  Date:
!  Copyright:  Laser-Scan Ltd
!
!  Feature Representation Table FRT
!
!       FC   GT   COL   WIDTH   SIZE   SC   FLAG   H/W  DESCRIPTION
!
FRT     1     7    0     0       0      1    0      0   control points
FRT     2     1    0     0       0      0    0      0   contours
FRT     3     1    1    1.0      0      0               index contours
FRT     4     1    0    0.5      0      1               pecked lines
FRT     5    12    0     0       0     -1               lake
FRT     6     1    2    0.3      0      2    4      0   symbol line
FRT     7     8    0     0      2.0     2    1     23   wind indicator
FRT     8     7    3     0      1.5     3               spot height
FRT     9    10    0    1.0     3.0     1               title text
FRT    10    10    0    0.5     1.0    -1               minor text (italic)
FRT    11    12    0     0      3.0    -2               symbol fill
!
! Symbol component table SCT
!
!       FC   GT   COL   WIDTH   SIZE   SC   FLAG   H/W  DESCRIPTION
!
SCT     1     1     0     0     0     0                  line
SCT     2     1     1    1.0    0     0                  line
SCT     3     6     0     0     0     0                  curve
SCT     4     5     0     0     0     0                  circle
!
! Group Definitions
!                NAME                FC                DESCRIPTION
!
GROUP            Contours            2-4               ! All contours
GROUP            Symbols             1,7,8             ! All symbols
!
!
! Pattern Definition Table
!
!       Pattern  Major  Minor  Major  Minor  Overall Major  Minor  Major  Minor
```

```
!       Index    UC     UC     Repeat Repeat Size    Size   Size   Width  Width

PATTERN  1       0      0       0      0      5.0     3.0    0.0    0.0    0.0
PATTERN  2       0      0       0      0      5.0     2.0    1.0    0.0    0.0
PATTERN  3       0      0       0      2     13.0     3.0    2.0    0.0    0.0
PATTERN  4       4      0       0      0      3.0     0.0    0.0    1.0    0.0
PATTERN  5       4      5       0      1      7.0     3.0    1.0    1.0    0.5
PATTERN  6       0      1       0      0      2.0     0.5    0.0    0.0    0.5
PATTERN  7       0      5       0      0      6.0     3.0    0.0    1.0    1.0
!
! Fill area table
!
!     Index   Hatch    SC
!
FILL   -2       3       2
!
! Attribute codes
! ===============
!
!
ACD TABLE       1               ! survey information
!
!      Code     Name            Minimum         Maximum       Min step
!
ACD D   1       SURVEY_DATE     17-NOV-1858     31-DEC-9999
ACD D   2       CHANGE_DATE     17-NOV-1858     31-DEC-9999
ACD I   3       CHANGE_TYPE     1               6
ACD I   4       CAPTURE_XY      0               100
ACD I   5       CAPTURE_Z       0               100
ACD I   6       SECURITY_MARKER 0               1
!
ACD TABLE       9               ! other examples
!
!      Code     Name            Minimum         Maximum
!
ACD C   1       FOUR_CHARS      '    '          '~~~~'
ACD T   2       TIME_EXAMPLE    00:00:00.0      23:59:59.99
ACD R   3       REAL_EXAMPLE       0.0          1000.0          .01
ACD I   4       INTEGER_EXAMPLE    0
ACD R   5       ANOTHER_EXAMPLE    ?            0
```

APPENDIX B

PATTERN LINE EXAMPLES

```
!
! PATTERN DEFINITION TABLE
!
!       Pattern  Major  Minor  Major  Minor  Overall Major  Minor  Major  Minor
!       Index    UC     UC     Repeat Repeat Size    Size   Size   Width  Width

PATTERN   1      0      0       0      0      5.0     3.0    0.0    0.0    0.0
PATTERN   2      0      0       0      0      5.0     2.0    1.0    0.0    0.0
PATTERN   3      0      0       0      2     13.0     3.0    2.0    0.0    0.0
PATTERN   4      4      0       0      0      3.0     0.0    0.0    1.0    0.0
PATTERN   5      4      5       0      1      7.0     3.0    1.0    1.0    0.5
PATTERN   6      0      1       0      0      2.0     0.5    0.0    0.0    0.5
PATTERN   7      0      5       0      0      6.0     3.0    0.0    1.0    1.0
```

The effect of these pattern definitions is illustrated below. Obviously the symbols incorporated in the patterns are dependent on the symbol table used but the examples do give an indication of the effect of the different pattern parameters.

APPENDIX C

ASCII character set

0	(NUL)	33	!	66	B	99	c
1	(SOH)	34	"	67	C	100	d
2	(STX)	35	#	68	D	101	e
3	(CTRL/C)	36	$	69	E	102	f
4	(EOT)	37	%	70	F	103	g
5	(ENQ)	38	&	71	G	104	h
6	(ACK)	39	'	72	H	105	i
7	(BELL)	40	(	73	I	106	j
8	(BS)	41	)	74	J	107	k
9	(HT)	42	*	75	K	108	l
10	(LF)	43	+	76	L	109	m
11	(VT)	44	,	77	M	110	n
12	(FF)	45	-	78	N	111	o
13	(CR)	46	.	79	O	112	p
14	(SO)	47	/	80	P	113	q
15	(CTRL/O)	48	0	81	Q	114	r
16	(DLE)	49	1	82	R	115	s
17	(DC1)	50	2	83	S	116	t
18	(DC2)	51	3	84	T	117	u
19	(DC3)	52	4	85	U	118	v
20	(DC4)	53	5	86	V	119	w
21	(NAK)	54	6	87	W	120	x
22	(SYN)	55	7	88	X	121	y
23	(ETB)	56	8	89	Y	122	z
24	(CAN)	57	9	90	Z	123	{
25	(EM)	58	:	91	[	124	|
26	(CTRL/Z)	59	;	92	\	125	}
27	(ESC)	60	<	93	]	126	~
28	(FS)	61	=	94	^	127	(DELETE)
29	(GS)	62	>	95	_		
30	(RS)	63	?	96	`		
31	(US)	64	@	97	a		
32		65	A	98	b		

Notes

1. The values 0 - 31 represent unprintable and control characters. These may be included in a TRI file (usually as special characters), the problem being how to insert them in any character string to be plotted.

2. The ASCII character whose value is 32 is a space.

APPENDIX D

Example PSFONTLIST file

```
! EXAMPLE.PSFONTLIST           Created                PGH 21-Jul-1993
!
! This file can be read via logical names LSL$PS_FONTLIST and LSL$DPS_FONTLIST.
! It contains a lookup from Laser-Scan FRT font number to PostScript font info.
! See the LITES2 Motif Workstation Guide for more details.
! For standard fonts see the file TEMPLATE.PSFONTLIST.
!
! First use the header mechanism to define a private encoding vector.
! The starting point is StandardEncoding, with changes and additions.
! This example is actually identical to ISOLatin1.
!
HEADER  mark
HEADER  /MyLatinEncoding
HEADER  8#000 1 8#054 {StandardEncoding  exch get} for
HEADER  /minus
HEADER  8#056 1 8#217 {StandardEncoding exch get} for
HEADER  /dotlessi
HEADER  8#301 1 8#317 {StandardEncoding exch get} for
HEADER  /space /oe /OE /lslash /Lslash
HEADER  /Zmacronunder /brokenbar /section /dieresis /copyright
HEADER  /ordfeminine /guillemotleft /logicalnot /hyphen
HEADER  /registered /macron /degree /plusminus
HEADER  /twosuperior /threesuperior /acute /mu
HEADER  /paragraph /periodcentered /cedilla /onesuperior
HEADER  /ordmasculine /guillemotright /onequarter /onehalf
HEADER  /threequarters /questiondown /Agrave /Aacute
HEADER  /Acircumflex /Atilde /Adieresis /Aring /AE
HEADER  /Ccedilla /Egrave /Eacute /Ecircumflex /Edieresis
HEADER  /Igrave /Iacute /Icircumflex /Idieresis /Eth
HEADER  /Ntilde /Ograve /Oacute /Ocircumflex /Otilde
HEADER  /Odieresis /multiply /Oslash /Ugrave /Uacute
HEADER  /Ucircumflex /Udieresis /Yacute /Thorn /germandbls
HEADER  /agrave /aacute /acircumflex /atilde /adieresis
HEADER  /aring /ae /ccedilla /egrave /eacute /ecircumflex
HEADER  /edieresis /igrave /iacute /icircumflex /idieresis
HEADER  /eth /ntilde /ograve /oacute /ocircumflex /otilde
HEADER  /odieresis /divide /oslash /ugrave /uacute
HEADER  /ucircumflex /udieresis /yacute /thorn /ydieresis
HEADER  /MyLatinEncoding
HEADER  where not {256 array astore def} if
HEADER  cleartomark
!
```

```
! Now define some fonts.
!
! first a minimally defined font.
! (no AFM given, so no kerning, and TRI still needed for character widths)
!
FONT    1       Times-Roman
!
! then one with more information given (for kerning)
!
FONT    2       AvantGarde-Book
AFM             AVANTGARDE_BOOK
KERNING         PAIR
!
! then one re-encoded as ISOLatin1
!
FONT    3       Helvetica
ENCODING        ISOLatin1Encoding
NEWNAME         Helvetica-ISOLatin1
AFM             HELVETICA_ISOLATIN1
!
! then one with special findfont
!
FONT    4       My-Souvenir
FIND            /Souvenir-Light findfont 1.25 scalefont
!
! then one with lots of control, and using the private encoding
!
FONT    5       Souvenir-Light
ENCODING        MyLatinEncoding
NEWNAME         My-Font-MyLatin
AFM             SOUVENIR_LIGHT_MYLATIN
KERNING         PAIR
SCALE           1.25
!
! a Hebrew font which uses right-to-left writing direction
!
FONT    6       David-ISOLatinHebrew
SCALE           1.3
AFM             DAVID_ISOLATINHEBREW
DIRECTION       RIGHT_TO_LEFT
!
! an Arabic font which uses right-to-left writing direction, and
! also a table for glyph substitution
! The values in the `Input' column in this case correspond to the
! standard ISO 8859-6, which is equivalent to ASMO 708.
!
FONT    7       NaskhMT
SCALE           1.3
AFM             NASKHMT
DIRECTION       RIGHT_TO_LEFT
!       Input       Isolate Final   Medial  Initial     Name
ARABIC  172         44      44      44      44          ! Arabic comma
ARABIC  187         59      59      59      59          ! Arabic semicolon
ARABIC  191         63      63      63      63          ! Arabic question mark
ARABIC  193         213     213     213     213         ! Hamza
ARABIC  194         69      70      70      69          ! Madda on Alif
ARABIC  195         67      68      68      67          ! Hamza on Alif
```

```
ARABIC  196         218     219     219     218         ! Hamza on Wau
ARABIC  197         71      72      72      71          ! Hamza under Alef
ARABIC  198         217     216     215     214         ! Hamza on Ye
ARABIC  199         65      66      66      65          ! Alif
ARABIC  200         76      75      74      73          ! Be
ARABIC  201         209     210     210     209         ! Te Mabuta
ARABIC  202         80      79      78      77          ! Te
ARABIC  203         84      83      82      81          ! The
ARABIC  204         88      87      86      85          ! Jim
ARABIC  205         96      92      90      89          ! He
ARABIC  206         100     99      98      97          ! Khe
ARABIC  207         101     102     102     101         ! Dal
ARABIC  208         103     104     104     103         ! Dhal
ARABIC  209         105     106     106     105         ! Re
ARABIC  210         107     108     108     107         ! Ze
ARABIC  211         112     111     110     109         ! Sin
ARABIC  212         116     115     114     113         ! Shin
ARABIC  213         120     119     118     117         ! Sad
ARABIC  214         126     124     122     121         ! Zwad
ARABIC  215         162     161     241     127         ! Toe
ARABIC  216         166     165     164     163         ! Zhoe
ARABIC  217         170     169     168     167         ! Ain
ARABIC  218         174     173     172     171         ! Ghain
ARABIC  224         64      0       0       0           ! Tatweel
ARABIC  225         178     177     176     175         ! Fe
ARABIC  226         182     181     180     179         ! Qaf
ARABIC  227         186     185     184     183         ! Kaf
ARABIC  228         190     189     188     187         ! Lam
ARABIC  229         194     193     192     191         ! Mim
ARABIC  230         198     197     196     195         ! Nun
ARABIC  231         203     204     204     203         ! Wau
ARABIC  232         202     201     200     199         ! Ha
ARABIC  233         212     211     211     212         ! Ye (no dots)
ARABIC  234         208     207     206     205         ! Ye (with dots)
ARABIC  235         231     0       0       0           ! fatatan
ARABIC  236         232     0       0       0           ! dammatan
ARABIC  237         235     0       0       0           ! kasratan
ARABIC  238         228     0       0       0           ! fata
ARABIC  239         229     0       0       0           ! damma
ARABIC  240         234     0       0       0           ! kasra
ARABIC  241         233     0       0       0           ! shadda
ARABIC  242         230     0       0       0           ! sukun
!
! End of EXAMPLE.PSFONTLIST
!
```