Laser-Scan Ltd.

LITES2

Reference Manual

Issue 4.2

Document "LITES2 Reference"   Category "REFERENCE"

Document Issue 1.0	Paul Hardy	7-Mar-1985
.		
.		
.		
Document Issue 3.0	Ron Russell	9-Oct-1989
Document Issue 3.1	Ron Russell	5-Jan-1990
Document Issue 3.2	Ron Russell	19-Jul-1990
Document Issue 3.4	Clarke Brunt	22-Mar-1991
Document Issue 3.5	Clarke Brunt	30-Sep-1991
Document Issue 4.0	Paul Hardy	22-May-1992
Document Issue 4.2	Clarke Brunt	21-Jul-1993
Document Issue 4.3	Clarke Brunt	18-Jan-1994

CONTENTS

1 Introduction

This manual is the main reference document for the Laser-Scan Cartographic Editor LITES2. It is intended for any person who uses LITES2 and it describes the editor environment and use, and each editor command in detail.

LITES2 is an interactive graphical editing program which has been designed to be particularly suitable for work with cartographic type data, but can also be used on other types of feature-orientated data. It has facilities for reading, drawing, amending, creating, and deleting features.

It is assumed that the user is already familiar with general use of the VAX/VMS operating system (see the DEC VAX/VMS manuals), and with the principles of digital cartography. This manual is supplementary to the "LITES2 User's Guide", which should be read before any use is made of LITES2, and before referring to this document.

This manual describes the full LITES2 command language, including some commands which are included for compatibility with future enhancements. The presence or absence of any facility from this manual does not imply any future commitment by LSL. See the current version of the LITES2 SPS (Software Product Specification) for a description of currently supported facilities.

LITES2 is a reimplementation of the previous LSL cartographic editor LITES1, (variously referred to as LITES, MADES, IGES, or SOLADI). The reimplementation has achieved the following major advances over LITES1.

* More performance and greater throughput as LITES2 takes advantage of 32 bit computer architectures and modern software techniques. This also gives extended program lifetime and easier maintenance.

* LITES2 is easier to use and without the petty restrictions of LITES1 which arose from its PDP11 and HRD-1 ancestry. This gives more flexibility and adaptability to changing workloads.

* Command names may be of arbitrary length instead of LITES1's three characters, and may be abbreviated to minimum non-ambiguous abbreviation.

* Macro commands are supported; users may define a name for a sequence of commands. The sequence can then be invoked by typing the macro name, or by assigning it to a menu square or cursor button, which are user-programmable.

* All commands may be journalled to file to provide audit trails and error analysis and recovery.

* Command input from file is supported. This allows recovery from system failure using the command journal file, and also use of generated "guidance files" for training, demonstrations, and semi-automated editing using information from validation programs.

* The sorted, sectored workspace file concept central to LITES1 has been removed entirely. It has been replaced by operation directly on an IFF file copy, giving greater speed, versatility and ease of future enhancement. This also gives greatly increased compatibility with the

LSL IFF utilities. The speed of finding features within the IFF file is maintained by keeping sectoring information in arrays in memory.

* LITES2 uses the Feature Representation Table (FRT) mechanisms instead of the Legenda used by LITES1 to determine the graphical representation of features. This means that symbols and text characters can include arbitrary complexity of lines, curves and circle arcs, and that patterned lines, fill areas, and user-defined character fonts are supported. Overall, a screen picture much truer to finished map sheet can be achieved.

* LITES2 calls graphics subroutines at the level of the new International Standards Organisation's Graphical Kernel System (GKS). This defines an abstract graphics device and greatly aids portability. Use of GKS also allows easy integration of the new generation of GKS-based graphics devices as they become available, with consequential increases in performance.

2 **Environment**

2.1 **Computer**

LITES2 runs on Digital Equipment Corporation (DEC) VAX series computers, running under the VMS operating system. Sufficient disc space must be available to hold input, workspace, and output versions of the IFF file containing the map being edited, as well as about 5M bytes for program images and static data files. An appropriate graphics workstation must also be available (see below) if graphic interaction is required.

For further details of the VAX computer series, see the DEC documents 'VAX Technical Summary', and the 'VAX Handbooks' (which describe the Architecture, Hardware, and Software of the machine).

For further details of VAX/VMS, see 'VAX Software handbook', published by DEC. Other manuals of direct relevance to the user of this manual are 'Introduction to VAX/VMS', and 'VAX/VMS Command Language Reference Manual'.

Refer to the LITES2 SPS (Software Product Specification) for currently supported VMS versions and hardware and software prerequisites.

2.2 **Workstation**

LITES2 can be run without graphic interaction on any DEC-compatible alphanumeric terminal eg VT320 VDU.

For graphic interaction it requires an LSL-supported GKS workstation configuration. This can include either one or two graphics screens (referred to as primary and secondary displays). Refer to the LITES2 SPS (Software Product Specification) for currently supported workstation types.

The refresh capability of the graphical display serves to provide a positional marker or cursor on the screen, to highlight features selected on the screen and to provide temporary display within graphical constructions.

3 Files used by LITES2

3.1 Input data

LITES2 is used to display and edit map data. At initialisation it reads one or more IFF files of map data and other files containing information about the graphic representations of features, layout of menus, tailoring options etc. At the end of a session, new edited versions of the IFF files are produced.

Each IFF file contains a single MAP. Data within a map may be grouped into LAYERS (sometimes known as OVERLAYS). The basic elements of map data are called FEATURES. The editor deals with twelve different categories or graphical types of feature. See the FRTLIB Reference Manual for details of treatment of graphical types. The list is:

1	line string
2	clockwise arc
3	anticlockwise arc
4	three point arc
5	full circle
6	curve
7	unoriented symbol
8	oriented symbol
9	scaled symbol
10	text string
11	symbol string
12	fill area

3.2 Logical names

For LITES2 to run successfully, certain logical names must be set up to point to directories and files used by the program. These will normally be set up automatically, but there may be a need to alter them for special purposes.

Logical name LSL$LITES2LOCK must be set to the full file specification of a file containing the LITES2 licence.

Logical name LSL$LITES2WORK must be set to a directory in which LITES2 is to put workspace and dump files. If a LITES2 session is terminated abnormally (e.g. by computer failure, or CTRL/Y) then the workspace file will be left in this directory, so the directory should be checked periodically, and old versions of files deleted.

Logical name LSL$LITES2CMD must be set to a directory in which LITES2 expects to find command files (default extension .LCM). The edgematching problem file is written to this directory.

Logical name LSL$LITES2JNL must be set to a directory into which LITES2 writes a journal file of commands given during a session (extension .LJN). It is advisable either to regularly purge this directory, or to set a version limit on it (using the DCL command SET DIRECTORY/VERSION_LIMIT=n), so that journal files do not build up without limit.

Logical name LSL$LITES2SETUP must be set to a directory in which LITES2 preserves table setup parameters for use in future sessions (extension .SET). It is often convenient to use the same directory as LSL$LITES2JNL for this.

If the logical name LSL$LITES2TERMINAL is set to a string, then this string is used as the terminal name when naming device dependent files. If LSL$LITES2TERMINAL is not assigned then the logical translation of SYS$COMMAND is used as the terminal name.

If the logical name LSL$LITES2ROUTINES is set to the full file specification of a shareable image containing the LITES2 user routines, then the LITES2 command USER will execute the code in this image.
Similarly the logical names LSL$LITES2ROUTINES_n (where n is an integer in the ranges 1 - 5 or 101 - 105) supply the images to be executed with the corresponding ROUTINE command.
See the chapter on user routines for details of how to create your own user routines. If these files are not present, then the corresponding LITES2 commands will not be available.

Logical name LSL$IF must be set to a directory in which LITES2 expects to find IFF files (input and output map data).

Logical name LSL$FRT must be set to a directory in which LITES2 expects to find FRT, SRI, and TRI files (feature representation files).

Logical name LSL$DTI must be set to a directory in which LITES2 expects to find DTI image files.

Logical name LSL$LSI must be set to a directory in which LITES2 expects to find LSI image files.

Logical name LSL$LSR must be set to a directory in which LITES2 expects to find LSR image files.

Logical name LSL$HELP must point to the directory containing the LITES2 help library, LITES2.HLB. This may be used outside LITES2 by typing e.g. $ HELP/LIBRARY=LSL$HELP:LITES2.

If the logical name LSL$LITES2INI is set to a file specification, then the contents of this file will be read as a series of LITES2 commands, when the program is first invoked (see section 4.3 below).

By default the maximum number of user macros that may be declared is 600. This value may be altered by defining the logical name LSL$LITES2_MACROMAX to be the appropriate number.

By default the maximum number of menu squares and puck buttons that may be declared is 500. This value may be altered by defining the logical name LSL$LITES2_MENUSQUAREMAX to be the appropriate number.

By default the maximum number of user variables that may be declared is 1000. This value may be altered by defining the logical name LSL$LITES2_VARIABLEMAX to be the appropriate number.

As in other Laser-Scan programs which are able to fill areas, the maximum number of points allowed in a solid filled area may be controlled by defining the logical name LSL$FILL_POINTSMAX to be the required number. The default value is 8192 points, with the minimum allowed being 100. Exceeding the limit for number of points will result in fill areas being drawn incorrectly or being drawn as an outline. When drawing raster images, LITES2 uses the same logical name to control the maximum number of pixels that can be drawn across an image, so the message "Buffer too small to draw DTI/LSR - zoom in or increase LSL$FILL_POINTSMAX" can be avoided in a future run of LITES2 by increasing the value of this logical name. Similarly the maximum number of times which a scan line may cross the boundary of an area may be controlled by defining the logical name LSL$FILL_CUTSMAX to be the required number. The default value is 100 intersections, with the minimum allowed being 10. Exceeding the limit for intersections will result in messages 'FILL_SIDE - Too many intersections found - ignored'. Memory has to be allocated in proportion to these numbers, so unnecessarily large values should be avoided.

If the logical name LSL$IFF_OUTPUT_REVISION is set to the value 1, then the IFF files that LITES2 outputs will contain CB entries and edits made to point attributes will be retained. If, however, LSL$IFF_OUTPUT_REVISION is set to the value 0, or is not set up at all, the IFF files that LITES2 outputs will contain ST (and ZS entries). The commands which add other attributes will still appear to work, but the attributes will not appear in the file.

The logical names LSL$TEXT_ROUTINE and/or LSL$SYMBOL_ROUTINE may be set to point to shared images that can be used to draw texts and symbols differently from the standard FRT routines used by LITES2. More details of this facility is available in the chapter on alternative text and symbol drawing routines.

The logical name LSL$LITES2_GET_SHEET_ROUTINES may be set to point to a shared image which can be used to supply a user defined map indexing system. This image is then called when the system variable $MAP_SHEET is accessed. This substituted routine may be passed either the absolute position of the cursor or its geographical position. Example source files that contain instructions to build this image are supplied in LSL$PUBLIC_ROOT:[LITES2.ROUTINES.EXAMPLES] and are called GET_SHEET_GEOG_EXAMPLE.FOR and GET_SHEET_GRID_EXAMPLE.FOR.

If this logical name is not set up, the variable $MAP_SHEET will use the sheet naming convention specified for the Ordnance Survey of Great Britain.

If the logical name LSL$OS_MH_TABLE points to a file name, this file will be opened and read as a translation table to locate the update flags in OS map headers type 3 and 4 for use with the OPERATION OS_MH_FLAGS command. If this logical name is not set up prior to invoking LITES2, it will not be possible to read IFF files with these type of map headers if an OPERATION OS_MH_FLAGS command has been given, or it will not be possible to give an OPERATION OS_MH_FLAGS command after reading IFF files that have these type of map headers. See the CONVERT package documentation for details of this table and the logical name.

The logical name LSL$LITES2_TERMINATOR_MASK may be used to control which characters will be taken as line terminators when typed at the terminal. It should be defined as a comma-separated list of ASCII codes and ranges of codes (use quotes if the list contains commas, otherwise a search-list will result). For example

```
$ DEFINE LSL$LITES2_TERMINATOR_MASK "13,26,128-159"
```

would cause carriage-return (13), Ctrl-Z (26), and also the upper control characters (128-159) to terminate lines.

If the logical name is not defined, then it is left to the VMS terminal driver to decide which characters are terminators. This is affected by the command
$ SET TERM/LINE_EDITING

The purpose of this is to allow characters which would by default have terminated the line to be input as characters, for example into text strings or AC entries. Note that even if they are set not to terminate the line, most control characters will not echo as anything sensible at the terminal.

Certain functions in LITES2 are supplied as shared images, which are only mapped when they are first accessed. This keeps the LITES2 image smaller, and thus more efficient, for those users who do not require the extra functionality that these shared images supply. These shared images are pointed to by the following logical names:

1. LSL$LITES2_GEOG_ROUTINES This logical name should point to LSL$EXE:LITES2GEOGSHR.EXE, an image supplied by Laser-Scan that implements grid -> geographical conversions as used by the GEOGRAPHICAL and SHOW GEOGRAPHICAL commands and the $LONGITUDE and $LATITUDE system variables.

2. LSL$LITES2_GEOM_ROUTINES This logical name should point to LSL$EXE:LITES2GEOMSHR.EXE, an image supplied by Laser-Scan that allows the manipulation of the dynamic data structures called `geometries'. The GEOMETRY command requires a suitable licence.
   This shared image itself makes use of the image LSL$LIBRARY:LSLGOTHICSHR which is pointed at by the logical name LSL$LSLGOTHICSHR.

3. LSL$LITES2_VIEW_ROUTINES This logical name should point to LSL$EXE:LITES2VIEWSHR.EXE, an image supplied by Laser-Scan when the licence for the VIEW command is supplied. This image implements the perspective viewing capability of LITES2, as supplied through the VIEW command.

4. LSL$LITES2_KERN_ROUTINES This logical name should point to an image, supplied by Laser-Scan, that is used when a photogrammetric restitution instrument is used as a 3-dimensional input device to LITES2. The name of the image will vary depending on the instrument in use, and whether it is equipped with an image superimposition system for graphical display from LITES2.

5. LSL$LSLGOTHICSHR If LSR image files are to be displayed or edited, then this logical name should point to a shareable image (normally LSL$LIBRARY:LSLGOTHICSHR, in the LSLSYSTEM package). This will normally be done as part of the system initialisation.

3.3 **Files created by LITES2**

3.3.1 **Workspace files**

Maps read in by the IFF command are copied to a workspace IFF file LSL$LITES2WORK:filename.WRK. This file is operated on during the editing session, and is normally deleted at the end, unless the DUMP command is used, when it is renamed as LSL$LITES2WORK:filename.DMP and may be used for subsequent input to LITES2. The dump file is a valid IFF file, and may be used as such by other programs, but as a result of editing it may contain dead space, and the layers are likely to be fragmented. For this reason, it is advisable to finally use the EXIT or WRITE command to cause the workspace file to be tidied into a new version of the IFF file.

3.3.2 **Journal file**

During a session, LITES2 writes all commands issued to a file LSL$LITES2JNL:terminal.LJN (where "terminal" is either the SYS$COMMAND device, e.g. TTA0, or the name defined by the logical name LSL$LITES2TERMINAL). Optionally, any macros or command file directives may be journalled, but preceded by the comment delimiter (!), so that the file may be used as command input to LITES2 to recover a session lost for any reason. (See Recovery.)

3.3.3 **Table setup file**

LITES2 saves the information concerning the positions of maps, menus, and tracking areas in a file LSL$LITES2SETUP:terminal.SET. This enables the old positions to be re-used in another LITES2 session, without re-digitising. (See Table Setup.)

3.3.4 **Edgematching problem file**

The edgematching routine writes a LITES2 command file LSL$LITES2CMD:terminalPROB.LCM, usually overwriting any previous version. If however ENABLE APPEND has been given, the LITES2 commands are added to any existing version of the file.
This file is automatically invoked by the REVIEW command, to guide the operator through any problems encountered while edgematching.

3.3.5 **SAVE MACRO files**

The SAVE MACRO command causes a file LSL$LITES2CMD:macroname.LCM to be written for subsequent input to LITES2.

3.3.6 **Sector list file**

The SAVE SECTORS command causes a file LITES2.PRT to be written to the current directory. This is intended as a program development aid.

4 Using LITES2

4.1 Running the program

In order to run LITES2, you must be logged in to VMS at a terminal. See your System Manager if you do not have the Username and Password required to do this.

At some sites, the system manager will have included LITES2 as part of a captive command procedure so that the user will not have to give the actual command to start up LITES2. However, if you are presented with the "$" (dollar) prompt, then you can start up the program by typing

LITES2

The program will announce itself as

LITES2 ([version]) V[n] of [hh:mm:ss dd-mmm-yy]",

where [version] is the name of the hardware specific version of LITES2 in user, [n] is the version number, and [hh:mm:ss dd-mmm-yy] is the time and date of linking of this release. This is followed by a message indicating the licensed users of this version.

There will then be a pause while first stage initialisation is carried out, and normally an initialisation command file containing options and menu definitions will be read. You will then be presented with a prompt from LITES2, which is an asterisk by default, but may have been changed in the initialisation command file eg

*

and commands to the program may be given (eg HELP).

4.2 Command States

There is underlying structure to editing operations, which often need to be performed in a certain sequence or in the correct context. This structure is reflected in the fact that LITES2 is always in one or other of a small number of command states. Within each state, a given range of commands is valid and certain commands or operator actions are used to move between states. The main command states and their transitions are shown diagramatically in the figure below.

On starting up, the editor is in INITIAL state in which options and attribute files can be specified, and which continues until the commands have been given which specify the map file(s) to be edited. The map is then read-in to the workspace file, and optionally drawn. The editor then enters READY state. It is possible to return to INITIAL state by using EXIT, DUMP, or QUIT commands when DISABLE EXIT is in operation.

The majority of commands are available in READY state, which is the principal operating level of the editor. LINE state is entered when a linear feature is identified using the FIND command. TEXT state and SYMBOL state are the equivalents for text features and symbols.

EDIT state and MODIFY state are both entered as the result of editing operations which require an END command to terminate. ON state is entered by commands which require the cursor to be constrained on a linear feature. CONSTRUCT state is entered during construction of linear features.

WINDOW state is entered by a WINDOW command to allow specification of an area of the map to be enlarged onto the graphics screen, or by a REGION WINDOW command for constructing a rectangular region. MACRO state is entered while defining macro commands. AC state is entered to allow editing of ancillary coding (AC/TC/CH entries). SETUP state is entered while digitising map corners on a raster image on the screen. PAINT state is entered while editing a raster image using the IMAGE PAINT command.

The current program state is displayed by the command SHOW STATE. It is also displayed in the status line (ENABLE STATUS), and is used as a prompt if PROMPT STATE is in action. System variable $STATE is set to the current state.
The possible program states are:

```
INITIAL   - Awaiting specification of input maps (IFF files)
READY     - Ready for an editing command, no operations in progress
LINE      - Line type feature found
CIRCLE    - Circle arc feature found
TEXT      - Text feature found
SYMBOL    - Symbol feature found
EDIT      - Edit operation on line type feature in progress
MODIFY    - Modification of text or symbol feature in progress
ON        - Constrained on line feature during part edit operation
WINDOW    - Defining a window for drawing
CONSTRUCT - Constructing a line type feature, or a BRIDGE
AC        - Editing ancillary codes of a feature or attribute set
DEBUG     - Not used
RECOVER   - Deleted (or 'limbo') feature found
SETUP     - Setting up corners of IFF files on a raster image
PAINT     - Editing a raster image using the IMAGE PAINT command
MACRO     - Defining a macro
```

```
                           INIT state
                               |
                          (IFF/READ commands)
                               |
                               V
                           READY state
                               |
        +------------------+---+------------+----------+--------+
        |                  |                |          |        |
        |                  |                |          |        |
(FIND text/symbol)    (FIND line)    (construct line)  |  (WINDOW command)
        |                  |                |          |        |
        V                  V                V          |        V
   TEXT or SYMBOL     LINE state    CONSTRUCT state    |   WINDOW state
      states         /        |                        |
        |           /         |                        |
        |          /          |                        |
        |         /           |                        |
        |        /            |                        |
  (editing command)     (constraining)         (MACRO command)
        |                     |                        |
        |                     |                        |
        V                     V                        V
 EDIT or MODIFY state     ON state                MACRO state
```

4.3 **LITES2 initialisation**

When started up, LITES2 implicitly obeys the command @LSL$LITES2INI, thus, if logical name LSL$LITES2INI is set up to point to a file (with defaults LSL$LITES2CMD:---.LCM), then this will be executed. This mechanism is normally used to tailor LITES2 for a particular application.

Next, LITES2 obtains the device name from the translation of LSL$LITES2TERMINAL if it exists, or else from SYS$COMMAND. This will normally be the user's login terminal e.g. TTA0. This is used to gain access to several files specific to this terminal. The command @terminal is implicitly obeyed, so that if, for instance, the user is on TTA0, the file LSL$LITES2CMD:TTA0.LCM will be obeyed if it exists. This mechanism is normally used to configure LITES2 for the particular hardware available on each terminal line, or to set up menus and pucks.

Finally LITES2 obeys any commands given on the DCL command line, and is then ready for interactive input. If LITES2 is to be run non-interactively, then the commands for the complete session should be contained in the initialisation files or command line. (These may of course invoke other command files.)

Certain LITES2 commands ("privileged commands") are only valid if encountered in initialisation files, so that a degree of control may be exercised over the use of these by ordinary users.

While reading initialisation files, the "Now in state..." messages are suppressed independently of the setting of ENABLE NOW, so that the defining of menus or pucks or other macros does not cause an excess of messages to be output.

4.4 **LITES2 recovery**

If a LITES2 session is lost for any reason (e.g. computer failure, operator error), then there are two possible ways to recover it. The preferred route is to utilise the journal file, but it may also be possible to use the workspace file if this still exists.

4.4.1 **Recovery from the journal file**

Find out the name of the terminal on which the session was run (e.g. TTA0), possibly using the DCL command SHOW TERMINAL. If LSL$LITES2TERMINAL was used, then the translation of this will be needed instead. Look for the journal file, LSL$LITES2JNL:terminal.LJN. If any sessions have been run at this terminal since, the required journal file may not be the most recent version, so check the file's creation date.

Check that the file is complete. If the file is LOCKED, then use the DCL command UNLOCK on it. If part (or all) of the file appears to be missing, and the number of blocks used is less than the number of blocks allocated (as shown by the DCL command DIRECTORY), then use the DCL command SET FILE/END_OF_FILE, and then edit to remove any garbage at the end. If necessary, edit the file. If for instance the session was lost due to the operator accidentally typing QUIT, then the QUIT command must be removed, otherwise it will be executed again.

It is advisable to RENAME or COPY the journal file to a different filename, to avoid confusion when LITES2 creates another of the same name. It is convenient to rename the file into LSL$LITES2CMD: with extension .LCM, so that the directory and extension need not be specified when it is used e.g.
$RENAME LSL$LITES2JNL:terminal.LJN LSL$LITES2CMD:CRASH.LCM
where CRASH is a name of your choice.

Define logical name LSL$LITES2REC as the journal file, e.g.
$DEFINE LSL$LITES2REC CRASH (if file is LSL$LITES2CMD:CRASH.LCM)
and run LITES2.

The command in the recovery file will be read instead of any initialisation files. It will be necessary to set up maps on the table if this was done in the original session. After reading the recovery file, provided that it does not contain an EXIT, DUMP, or QUIT command, LITES2 will prompt for interactive input, and editing may be continued.

Finally DEASSIGN logical name LSL$LITES2REC, so that LITES2 does not use the recovery file for the next session. Delete the recovery file when it is finished with.

While reading a recovery file, LITES2 always continues on error, whether or not ENABLE CONTINUE has been specified.

4.4.2 **Recovery from the workspace file**

It may be possible to use the workspace file to recover a LITES2 session. The workspace file is a valid IFF file, and may be input to LITES2 as an IFF file in the usual way. It will usually be necessary to mend the workspace file before use using the IMP utility IMEND (which replaces the old MIFF and IFT), as it will not have been closed correctly. It is probable that one block (and possibly more) of the workspace file will not have been written to disk when LITES2 terminates abnormally. If the missing block is at the end of the file, then the result will just be the loss of some data, but if a block in the middle of the file is missing, then it may be impossible to access the data after it.

If using the workspace file for input, then great care should be taken to ensure that data is not lost.

4.5 **Interactive devices**

Each version of LITES2 supports a number of interactive devices, such as digitising tables and bitpads, in addition to a keyboard. For details of the particular hardware supported by each version, see the appropriate Workstation Guide. Nevertheless some general principles apply:

If a digitising table is used, then it will be possible to set up source documents (maps), menus, and a tracking area on it. The digitising puck will normally have more than one button on it, and in this case each button may be programmed to execute a LITES2 command. The lowest numbered button is usually reserved for tracking the screen cursor around the map, or tracking area, and may not be programmed to do anything else. If any other button is pressed while within a map or tracking area, the cursor will be positioned to the appropriate place before the button's own command is obeyed, so it is not compulsory to track the cursor to the correct place first using the lowest numbered button. Some LITES2 commands cause the screen cursor to move to a computed position, for instance CLOSE and LOOP (which create closed loops), and FIND (which moves onto a nearby feature). If the next command after one of these is given from a puck button, then no cursor movement will occur even if within a map or tracking area. This allows, for instance, an END command to be given without spoiling the computed cursor position, or having to move the puck off the map area. Any button (but sometimes not the lowest numbered button) may be used to access a menu box, but a facility exists (PRIORITY PUCK) to cause certain buttons to perform their puck functions even when pressed over a menu.

The same comments as for tables apply to bitpads (tablets), but there are some differences. Source documents may not be set up on bitpads (they are usually too small and inaccurate). Bitpads are always connected locally to the workstation, and can perform some sort of cursor tracking on the screen without the intervention of the computer - this often leads to LITES2 having two screen cursors - its own position and the bitpad position. In these cases, the lowest numbered button on the bitpad puck causes the LITES2 cursor to move to the position of the bitpad cursor. Because of this local tracking, there is not usually any reason to set up a tracking area on a bitpad. The other buttons may be used to access bitpad menu boxes, or be programmed in their own right.

If a screen menu is provided, then it is usually accessed by moving the bitpad cursor to the required box and pressing the lowest numbered button.

4.6 **LITES2 Table Setup**

If a digitiser is in use, then before reading in maps, or when the SETUP AGAIN command is used, LITES2 will prompt the user to digitise the positions of maps, menus, and tracking areas. At this point, the user may digitise the points as requested, but also has the chance to abort the setup, or to use the setup from a previous LITES2 session if possible.

The setup of a map, menu, or tracking area may be aborted by pressing CTRL/C at the keyboard (Control and C keys pressed simultaneously), or by pressing the highest numbered button on any puck (provided that the puck has been defined using the PUCK command).

The previous setup values may be used by pressing the penultimate button on any puck (again providing that the puck has been defined).

4.7 **Use of CTRL/C**

LITES2 has its own CTRL/C handler, so that CTRL/C (pressing Control and C simultaneously at the keyboard) may be used for several purposes. Note that which function occurs will depend on what the program is doing at the time. The functions of CTRL/C are as follows:

* Abort execution of any command files and macros returning to interactive input. Also terminate RESPOND input. This will occur only when any currently executing command completes.

* Terminate reading in of a map and proceed with the next map, if any. Only the data read in so far is available for editing and output. This feature may be useful for demonstration purposes or in cases when the wrong map has been specified. A subsequent QUIT with DISABLE EXIT in force will return to INITIAL state and allow the correct map to be given. If reading in of an INSITU map is prematurely terminated, then unless an EXIT/DUMP/QUIT command is given immediately, the remaining data will be lost irrevocably.

* Abort re-drawing of the map. There may be a delay between pressing CTRL/C and drawing actually stopping. This is useful if the wrong area is being drawn. Editing may continue even though only part of the picture is drawn on the screen.

* Abort re-drawing an image, and proceed with the next one, if any.

* Abort the drawing of labels.

* Abort edgematching as soon as possible.

* Pause MEND AUTOMATIC as soon as possible.

* Abort the construction of a region using REGION IMAGE.

* Abort image speckle removal using IMAGE SPECKLE CLEAR/FILL.

* Abort image editing operations using IMAGE CLEAR/FILL/COPY/MOVE.

* Terminate the output produced by the SHOW MACRO, SHOW MENU, SHOW PUCK, and SHOW REGION commands.

* Terminate a delay initiated by the WAIT command.

* Abort a user routine that is creating multiple features. Note that CTRL/C will only abort the user routine between calls of the user supplied routines; it will not abort while the user supplied routines are being executed.

If enabled, then CTRL/Y will return to a DCL prompt as usual. If done by accident, then CONTINUE (or C) will continue with LITES2.

4.8 **Messages and Errors**

LITES2 has 4 types of messages which can be output to the alphanumeric terminal, indicating conditions of varying severity.

Some messages are purely informational and are sometimes unsolicited, and sometimes the result of an operator command requesting the information. These are known as 'INFORM' type messages, and may be suppressed if required using the DISABLE INFORM command.

The sort of operator errors which are expected to occur occasionally, such as failing to find a feature, or an invalid command, result in the production of a 'MOAN' type message, which is preceded by '???', and is usually accompanied by an audible warning.

'NASTY' type messages are preceded by '!!!' and should not normally occur. They usually indicate that an internal consistency check has failed, which may lead to catastrophic failure of the program.

The final type of message is 'LEARN' type, which is preceded by '...'. These are only output if ENABLE LEARN is in operation, and are intended to provide extra information while learning to use the program.

The occurrence of a 'MOAN' or 'NASTY' error will result in any macros and command files being abandoned, and LITES2 returning to interactive input, unless ENABLE CONTINUE is used.

5 Editing Operations

5.1 Feature Selection

A fundamental concept of the editor is the concept of the FOUND-FEATURE, and of the FEATURE-IN-HAND. The found-feature is a feature selected by the operator to be the subject of editor operations. Selection is normally by positioning the cursor close to the desired feature and issuing the FINd command, but there are alternative provisions for finding a feature by global searches. Refer to the FIND, SEARCH, and LOCATE commands below for more information.

When an editing command is given, any found-feature becomes the FEATURE-IN-HAND in order that a second feature can be found if needed for the editing operation. For example, during a complex editing operation such as JOIn, there will be both a feature-in-hand which is the line to join from, and also a found-feature, which is the line to join to.

The current found-feature, and feature-in-hand, if any, are highlighted in refresh mode on the display screen. For linear, arc, curve and symbol string features, the string of line segments linking the data points is highlighted (or a portion of it if the whole cannot be refreshed). For symbol features the whole symbol is highlighted. For text features each character is highlighted, or on some workstations a highlighted box is drawn around the text,

5.2 Attributes of Points

When carrying out geometrical edits on features which have attributes associated with their defining points, it is not always possible, or meaningful, to retain these attributes.

In particular, when linear features are FILTERed, then the whole new feature will be created with no point attributes. (Note that squaring and offsetting does retain point attributes)

When new points are created within features (e.g. with the various PART operations or with the CLIP, INSERT or BRIDGE commands) then the new point created does **not** inherit any of the attributes of the points adjacent to it. When JOINing two features together, two points are condensed into one; in this case the resulting point inherits the attributes of the point that was found when the JOIN command was given.

NOTE

> With all these editing operations, attributes can be automatically edited or added by the use of the OPERATION command.

6 LITES2 command language

This section is not concerned with the functions of LITES2 editing commands, which are described individually below, but rather with the use of the command language and programming facilities to set up menus and pucks, and to combine the primitive operations into more sophisticated procedures. Where a particular command is mentioned, its exact syntax and function should be looked up in the individual descriptions.

6.1 Command lines

LITES2 command lines are lines of text, up to 255 characters long, which may be obtained from the keyboard, or other interactive controls, or from a command file on disc. (The interactive interface ensures that button presses and menu probes have the identical effect to typing commands at the keyboard.)

A command line consists of zero or more **commands**, separated by the character "#", followed by an optional comment preceded by the character "!". The characters "#" and "!" may **never** be used for any other purpose.
For example: FIND # DELETE ! deletes the nearest feature

6.2 Commands

LITES2 commands consist of an optional **label** (beginning with "." and ending with ":"), optionally followed by a **primary command**. In some cases, this may form the entire command (eg FIND). Some primary commands must be followed by one of several **secondary commands** (eg SEARCH ALL). In some cases a default secondary command is assumed if this is omitted (eg SEARCH is equivalent to SEARCH NEXT). Both primary and secondary commands may be shortened to the minimum non-ambiguous abbreviation.

Commands may be followed by compulsory or optional **arguments**

```
eg TOLERANCE FIND 3.0   (argument is compulsory)
   POSITION 400.0 450.0 (arguments are optional)
   POSITION             (position to centre of screen is assumed)
```

The precise effect of omitting an optional argument is described with each individual command.

6.3 Arguments

Command arguments are in general integer numbers, real numbers, or text strings. An integer number may not have a decimal point. Where a real number is required, the decimal point may be omitted when an integral value is desired, and scientific or exponent notation (eg 1.3E-3 = 0.0013) is permitted. Real numbers may also be entered in a 'rational' format (eg 2/3 for two thirds, or 0.6666667).

In the case of text arguments, exactly what is required is described with the individual commands, but it should be noted that in the case of commands taking an arbitrary text string (eg for insertion in the map data, or as a message), any leading spaces or tab characters are ignored. The text may be enclosed in double quotation marks, within which a repeated quotation mark may be used to include a quotation mark in the text. Quotation marks **must** be used if leading spaces or tabs are to form part of the text.

A few commands that deal with ancillary codes (ACs) or point attributes take a special form of argument, consisting of a code, (usually referred to as a "type" when referring to ACs), possibly followed by a value.

The code is either an integer which identifies the attribute, or a name that corresponds to this integer. Laser-Scan has defined some codes and names that are available to all users, but it is possible for the user to define his own in addition. This is done in the ACD (attribute code definition) part of the FRT. Details of the Laser-Scan standard codes and of how to define further codes can be found in the "FRT User Guide".

The format of the value part of the argument depends on the data type associated with the attribute code. Each of the Laser-Scan standard attribute codes has a data type associated with it; user defined attribute codes have data types associated with them when they are defined in the FRT.

The possible data types are:-

* Integer
  This is a integer or "whole" number.
  An example of valid integer value is

```
24
```

* Real
  This is a real or "floating point" number.
  Examples of valid real values are

```
3.414
366
0.123E3
```

* Character
  This is a string of 4 characters. It can optionally be surrounded by quotation marks when entered as an argument. The string will be padded with spaces on the right if shorter than 4 characters.
  Examples of valid character values are

```
"abcd"
 abcd
"   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
```

* 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
2:00:00.00
::                        just now
```

6.4 **Command files**

LITES2 commands may be stored in a disk file. Each line of the file is a single command line as described above. The command file is invoked by the command @filename, optionally followed by parameters. The commands in the file are then obeyed until the end of file is reached, when control returns to the command following the @filename directive. Command files may invoke other command files up to a certain level of nesting.

The most common use of command files is for LITES2 initialisation, but their use is not restricted to this. A possibly lengthy and rarely used sequence of editing commands may be better stored as a command file than as a macro (see below). A macro could still be set up to invoke the file rather than having the user type @filename.

6.5 **Macros**

Macros provide a mechanism for the user to extend the LITES2 command language by adding new 'commands' which perform sequences of existing commands. The definition of a macro is begun using the MACRO command, after which all commands are stored as part of the macro (rather than being obeyed immediately) until the ENDMACRO command is encountered. Any calls to other macros, or @filename directives are merely checked for syntax during macro definition. Once defined, a macro is invoked merely by giving its name, optionally followed by parameters. It may also be used in JUMP commands (see below).

It is possible to define a macro with the same name as one of the LITES2 primitive commands (or whose abbreviation is the same as that of a primitive command). In these circumstances, the macro will take precedence. If the primitive command is intended, the command name should be preceded by the character "%" which prevents the command being looked up as a macro.

It is good practice to use the "%" escape character with primitive commands in macros and command files, so that these will continue to work even if macros with ambiguous names are defined later. (Command decoding will also be slightly faster.)

6.6 **Parameters**

Parameters may be passed to command files and macros using the commands @file, JUMP, JTRUE, JFALSE, and the normal macro call invoked by giving the name of the macro. All these commands may be followed by a line of parameters, each delimited by one or more spaces or tabs, or enclosed in double quotes. Within the command file or macro, the values of the parameters are available in the character system variables $P1, $P2 etc. The number of parameters is available in $PCOUNT, and the whole line of parameters in $PLINE.

The parameters are only available within the command file or macro actually called. If this calls another command file or macro, then a new set of parameters will be available within the inner procedure. When the flow of control returns to the outer procedure, then the first set of parameters will be available again.

For example:
* add_text Red House
calls a macro passing two parameters. $P1 = "Red", $P2 = "House", $PCOUNT = 2, and $PLINE = "Red House" (the parameters do not actually contain the quotes).

* add_text "Red House"
This time there is only one parameter, $P1 = "Red House". $PLINE does contain the quotes.

6.7 **Pucks and Menus**

Programmable menus and pucks are implemented using macros. Once defined, using a PUCK or MENU command, the macros menuname1 up to menunameN (where menuname is the name of the menu or puck, and N is the number of boxes or buttons) are available to be defined using MACRO commands.

Spaces between the menu name and the box number are optional. The macros for unused boxes or buttons should be left undefined - they will then do nothing. Menu or puck macro commands are normally generated automatically by the interactive interface when the boxes are probed or the buttons pressed, but there is nothing to prevent them being used at the keyboard, or in command files or other macros.

6.7.1 **Keyboard function keys**

The function keys on DEC VDU terminals may be used as a programmable puck in LITES2. Use the PUCK command to define a puck on device 0, and then program the puck macros to execute the desired LITES2 commands. Buttons 1-49 are used, though not all of these have a corresponding keyboard key. This facility depends on a key transmitting the appropriate escape sequence to the computer and cannot utilise any key definitions using the DCL DEFINE/KEY command. Some terminals may allow keys to be programmed to send a string of characters to the computer just as if they had been typed - such a facility may be used instead of the mechanism described here.

F1 through to F5 cannot be used to give LITES2 commands. Which of the other keys are available depends on the terminal settings using the DCL commands SET TERM/[NO]LINE_EDITING, and SET TERM /APPLICATION_KEYPAD or /NUMERIC_KEYPAD. If the terminal is set to perform line editing, then F6 through F14 and the arrow keys will perform their line editing function and cannot be used to give LITES2 commands. If line editing is disabled, then these keys may be programmed. The keypad numeric keys, and also '.' '-' ',' and Enter may only be programmed if APPLICATION_KEYPAD is set, otherwise they just transmit their own character. The remaining keys (there are 16 of them) may always be programmed. The DCL SET TERM commands may be SPAWNed from LITES2, and some settings (particularly APPLICATION or NUMERIC KEYPAD) may sometimes be set locally on the terminal.
The layout of a VT220 type keyboard is shown below, with the puck button number shown below each key

```
F1 F2 F3 F4 F5  F6 F7 F8 F9 F10  F11 F12 F13 F14  Help Do  F17 F18 F19 F20
 x  x  x  x  x   6  7  8  9  10   11  12  13  14   15  16   17  18  19  20

                                            Fin Ins Rem    PF1 PF2 PF3 PF4
                                             40  41  42     21  22  23  24

                                            Sel Prv Nxt     7   8   9   -
                                             43  44  45    37  38  39  25

                                                Up          4   5   6   ,
                                                46         34  35  36  26

                                          Left Down Right   1   2   3
                                           48   47   49    31  32  33

                                                            0       .  Ent
                                                           30      28   27
```

6.8 **Labels**

Any command may be preceded by a label. This begins with ".", consists of letters, numbers, and underline characters, and is terminated by ":". The case of the letters in a label is not significant. The label may be used as a target for JUMP, JTRUE, and JFALSE commands (qv), and must be in the same command line, or macro as the command which refers to it. These is no need for labels in different macros to be different, but if duplicate labels occur in the same macro, then the first one will always be found.

6.9 **Flow of control**

The command file directive (@filename) and the macro call (macroname), which have already been introduced, pass execution to the appropriate command file or macro. Calls may be nested up to a certain limit.

Unless affected by the mechanisms described below, commands are obeyed in order one by one until the end of a command line is reached. A new line is then obtained from the current source (command file or interactively). A macro is treated internally as a single line with multiple commands separated by "#". When the current source is exhausted, execution continues at the command following that which invoked the command file or macro.

One way in which the flow of control can be altered is if a command causes a MOAN or NASTY type error. Unless ENABLE CONTINUE has been used, this causes all currently active command files and macros to be abandoned, and control is returned to interactive. This mechanism (with DISABLE CONTINUE in force) can be used to terminate a repetitive sequence (eg when SEARCH NEXT gives an error because there are no more features), but this method is not recommended since it will fail if there is no interactive input available (as for instance in a batch job).

The next method of altering the flow of control is the JUMP command. This transfers control to a named macro or label. In the case of a JUMP to a macro, any commands following the JUMP at the current level (macro or command file) are abandoned and execution begins at the start of the named macro. This means that is is only sensible for JUMP to occur as the last command in a command file or macro, unless the following command has a label which is the target of a JUMP elsewhere. A label must occur in the same macro or line as the JUMP command which refers to it.

JUMP does **not** use up a level of macro/command file nesting, and the target of the JUMP **can** be the current macro. This means that simple repetitive loops can be constructed using macros which repeat until a error occurs. NB DISABLE CONTINUE **must** be in force, otherwise CTRL/C must be used to break out of the loop.

More versatile than JUMP are JTRUE, JFALSE, THEN, ELSE, and ABORT. The behaviour of these depends on whether the **condition flag** (see below) is TRUE or FALSE.

JTRUE and JFALSE are the same as JUMP, except that the jump is only made if the conditional flag is TRUE or FALSE respectively. Unlike JUMP, these therefore can usefully be followed by other commands in a command file or macro.

THEN and ELSE are followed by a command (possibly itself a macro or @filename) which is obeyed only if the condition flag is TRUE or FALSE respectively. These **do** use up a level of command nesting, and also differ from JUMP in that, whether or not the command is obeyed, execution continues at the next command.

The ABORT commands may be used just to terminate execution of procedures. ABORT INPUT terminates all active command files and macros and returns to interactive input, while ABORT ALWAYS (the ALWAYS secondary command is optional) just skips the rest of the current line or macro. ABORT TRUE and ABORT FALSE are the same as ABORT ALWAYS, but depend on the setting of the condition flag. ABORT FILE skips the rest of the current command file, even if called from within a macro invoked by the command file.

The condition flag used by JTRUE, JFALSE, THEN, ELSE, and ABORT is set using the commands TEST, OR, and AND. These compare the value of a LITES2 **variable** (see below), with an **expression** (see below).

The RESPOND command will pause execution of a macro or command file and prompt for the user to enter commands interactively. When finished, the CONTINUE command will return control the the command file or macro, alternatively CANCEL RESPOND or CTRL/C may be used to abandon all command files and macros and return to ordinary interactive input.

6.10 **LITES2 Variables**

LITES2 variables are named entities which may be used to hold a character or numerical value. Variables may be one of four types:

CHARACTER: Contains a string of characters
INTEGER: Contains an integer value in the range -2147483648 to 2147483647
REAL: Contains a real value value in the range +/- 0.29E-38 to 1.7E38 with a precision of 7 to 8 decimal digits. Real variables are displayed with 8 significant figures (with trailing zeros suppressed) but the eighth figure may not be completely accurate.
DOUBLE: Contains a real value value in the range +/- 0.29E-38 to 1.7E38 with a precision of 15 decimal digits.

Variables may be used in several different ways:

* Their value may be substituted into commands

* Their value may be tested in TEST, AND, and OR commands

* They may be used to perform arithmetic in LET commands

* They may be used in INQUIRE to obtain a value interactively

Some variables always exist within LITES2. These have names which begin with the character "$", and are known as **system variables**. They contain values such as the current cursor position, or the current point on the found feature, and may not be set by the user. Some system variables have a compulsory argument (or subscript), e.g. $CUTREGION 2, and a few have an optional argument, e.g. $CURSINWIN 0.8. A complete list of available system variables is given below.

User variables must be declared using the DECLARE command before use. They can then be given values using the LET and INQUIRE commands. It is possible to declare array variables by following the name by the number of elements required.

By default the maximum number of user variables that may be declared is 1000. This maximum may be altered by setting the logical name LSL$LITES2_VARIABLEMAX to the required value before LITES2 is started.

The current value of variables can be displayed using the SHOW VARIABLES command, and can be tested using the TEST, AND, and OR commands.

The value of a variable may be substituted into a command by enclosing its name in single quotation marks (provided that ENABLE SUBSTITUTION is set). The trailing quotation mark may usually be omitted, as the variable name is taken to end at the first character which is not alphabetic or an underline. The trailing quote therefore must be present if two variables are to be substituted without any intervening spaces. If present, the trailing quote must immediately follow the variable name. There is no restriction on which parts of a command may use a variable. The substituted value may be a single argument, or (usually for a character variable) several arguments or even the whole command.

Eg. The command: SEARCH FSN 'NUMBER' will search for serial number 3 if NUMBER is an INTEGER variable with value 3. The effect would be identical if NUMBER were a CHARACTER variable containing the string "3", but note that only INTEGER, REAL, and DOUBLE variables can be used for arithmetic and to perform numerical comparisons.

In the case of array variables, another integer variable may be used as the subscript, thus if FSNS was an integer array containing a list of feature serial numbers, and I was an integer variable containing the required subscript, then the command: SEARCH FSN 'FSNS'I'' could be used. Both the trailing quotes are optional. A particular subscript may of course be used explicitly, as in SEARCH FSN 'FSNS4'.

The values of variables are never substituted while a macro is being defined. The presence of ' in any line in a macro will prevent the line being checked for syntax as the macro is defined, since the value of the variable is unknown at this stage.

6.10.1 **Expressions**

Expressions are used in the commands LET, TEST, AND, OR, and INQUIRE to set and test the values of variables.

For CHARACTER variables, an expression consists simply of a string of characters. The string may be completely absent (the empty, or null string), and may optionally be enclosed in double quotation marks. Within double quotes, the character " is represented by "". The trailing quotation mark may always be omitted. The case of letters in character variables matters, so that, for instance a variable containing "A" will not be considered equal to the expression "a".

For INTEGER, REAL, or DOUBLE variables, an expression consists of one or more numbers together with the operators + - * / and ^ (exponentiate), and functions (SIN, COS, TAN, ASIN, ACOS, ATAN, ABS, LN, LOG - see below). Unary minus and functions have highest precedence, followed by ^, then * and /, then + and -. Operators of equal precedence are evaluated from left to right. Parentheses (and) may be used to force the order of evaluation. Note that the exponentiation operator may be used to obtain square roots, e.g. 4^0.5 is 2.

When setting an INTEGER variable using LET or INQUIRE, an attempt is made to evaluate the expression using integer arithmetic. If this fails (due to the presence of a decimal point, an E exponent, or a real valued function), then real arithmetic is used, with the final result being truncated to integer.

When setting a REAL or DOUBLE variable, the expression is always evaluated using real arithmetic.

If an INTEGER variable is compared with a real expression in a TEST, AND, or OR command, then the integer is converted to real before the comparison.

For example (assuming I is an INTEGER variable, and R is a REAL):

```
   LET I=(3+8)/3 + 2/3   sets I to 3 using entirely integer arithmetic
   LET I=(3+8)/3.+ 2/3   sets I to 4 since decimal point forces real arithmetic
   LET R=(3+8)/3 + 2/3   sets R to 4.333333 (real arithmetic always used)
```

6.11 **Logical variables**

Variables may be tested as logicals in a TEST, AND, or OR command. If the variable name is specified with no inequality or expression, then the logical result of the test is as follows:

INTEGER variable	True if low bit is set (value is odd)
REAL variable	True if not equal to 0.0
DOUBLE variable	True if not equal to 0.0
CHARACTER variable	True if low bit of first character is set

In particular, this means that for INTEGER variables, 0 is false and -1 is true. These values are used for system variables which take a logical value. For CHARACTER variables, strings beginning with "Y" or "y" are true, while strings beginning with "N" or "n" are false. The null string is false.

6.12 **System variables**

System variables have names beginning with the character $. The available system variables can be displayed using the command "SHOW VARIABLES $". An error results if an attempt is made to use a system variable which is undefined e.g. $FSN when there is no found feature. A list of available system variables follows:

$ABSX

DOUBLE Contains absolute X value of cursor position (IFF units). It is equivalent to $CURSX + $MDOFFSET1

$ABSY

DOUBLE Contains absolute Y value of cursor position (IFF units). It is equivalent to $CURSY + $MDOFFSET2

$ACCVALUE
CHARACTER Contains the value (as 4 characters) of the current AC of the feature.

$ACDATATYPE
INTEGER Contains the data type of the current AC of the feature. The data type is an integer in the range 1 - 5.

1 means interpret the value as an integer
2 means interpret the value as a real number
3 means interpret the value as 4 characters
4 means interpret the value as a date
5 means interpret the value as a time

$ACIVALUE
INTEGER Contains the value (as an integer) of the current AC of the feature.

$ACNAME
CHARACTER Contains the name allocated to the type of the current AC (in either the LSL supplied list of types, or in the current FRT)

$ACPRESENT
INTEGER -1 if the found feature has any ACs, TCs or CHs, else 0

$ACRVALUE
REAL Contains the value (as a real) of the current AC of the feature.

$ACSVALUE
CHARACTER Contains the value (as an character string) of the current AC of the feature. This string is encoded using an appropriate format, depending on the data type.

$ACTEXT
CHARACTER Contains the text of the current AC (or TC or CH) of the feature.

$ACTEXTLEN
INTEGER Contains the length of the text of the current AC (or TC or CH) of the feature.

$ACTOTAL
INTEGER Contains the total number of ACs, TCs and CHs associated with the feature.

$ACTYPE
INTEGER Contains the type of the current AC of the feature.
Note: TCs are considered to be ACs with a type of -1, and CHs are considered to be ACs with a type of -2.

$ANGLE
REAL Contains the angle (anti-clockwise from horizontal) in degrees of the current feature. For linear features this is the angle of the current vector. This variable is not available for circle arcs.

$ANNOTATION_JOURNAL_NAME
CHARACTER Contains the name of the current annotation journal macro.

$ANNOTATION_JOURNAL_STATUS
INTEGER Contains the status of annotation journalling. 0 closed, 1 off, 1 on.

$AREA
DOUBLE Contains the area enclosed by the current feature in square IFF units. For features that are not closed, the first point is considered to be joined to the last, by a straight line. For texts and symbols the area of the bounding box is given.
A positive area indicates that the feature has been digitised in a clockwise direction, negative areas indicate counter clockwise digitising. Degenerate features (with two or less vertices) give an area of 0.0

$ASK_CHAR n Must be followed by an integer in the valid range; this range is determined by which ASK command has been called.
CHARACTER Contains character value(s) from the last successful invocation of an ASK command that returned character values.

$ASK_INT n Must be followed by an integer in the valid range; this range is determined by which ASK command has been called.
INTEGER Contains integer value(s) from the last successful invocation of an ASK command that returned integer values.

$ASK_REAL n Must be followed by an integer in the valid range; this range is determined by which ASK command has been called.
REAL Contains real value(s) from the last successful invocation of an ASK command that returned real values.

$ATTCODE n Must be followed by an integer in the range 1 - $ATTTOTAL.
INTEGER Contains the code (as an integer) of the specified attribute of the point.

$ATTCVALUE n Must be followed by an integer in the range 1 - $ATTTOTAL.
CHARACTER Contains the value (as 4 characters) of the specified attribute of the point.

$ATTDATATYPE n Must be followed by an integer in the range 1 - $ATTTOTAL.
INTEGER Contains the data type of the specified attribute of the point. The data type is an integer in the range 1 - 5.

1 means interpret the value as an integer
2 means interpret the value as a real number
3 means interpret the value as 4 characters
4 means interpret the value as a date
5 means interpret the value as a time

$ATTIVALUE n Must be followed by an integer in the range 1 - $ATTTOTAL.
INTEGER Contains the value (as an integer) of the specified attribute of the point.

$ATTNAME n Must be followed by an integer in the range 1 - $ATTTOTAL.
CHARACTER Contains the name allocated to the type of the specified attribute of the point (in either the LSL supplied list of types, or in the current FRT).

$ATTRVALUE n Must be followed by an integer in the range 1 - $ATTTOTAL.
REAL Contains the value (as a real) of the specified attribute of the point.

$ATTSVALUE n Must be followed by an integer in the range 1 - $ATTTOTAL.
CHARACTER Contains the value (as a character string) of the specified attribute of the point. This string is encoded using an appropriate format, depending on the data type.

$ATTTOTAL
INTEGER Contains the total number of attributes associated with the point.

$BEARING
REAL Contains the bearing (clockwise from grid north) in degrees of the current feature. For linear features this is the bearing of the current vector. This variable is not available for circle arcs.

$BOX n Must be followed by an integer in the range 1 - 4.
REAL Contains the coordinates of the limits of the box surrounding the current feature.

$BOX 1 is the minimum X coordinate (IFF units)
$BOX 2 is the maximum X coordinate (IFF units)
$BOX 3 is the minimum Y coordinate (IFF units)
$BOX 4 is the maximum Y coordinate (IFF units)

$BUTTON
INTEGER Contains the button number of the last puck button used.

$CATEGORY
INTEGER Contains category field for text.

$CLOSED
INTEGER -1 if the found feature is closed, else 0.

$COEFFS n Must be followed by an integer in the range 1 - 6.
REAL Contains the current transformation parameters

$COEFFS 1 is the rotation angle (in degrees)
$COEFFS 2 is the scale factor
$COEFFS 3 is the X coordinate of the rotation point (IFF units)
$COEFFS 4 is the Y coordinate of the rotation point (IFF units)
$COEFFS 5 is the translation in the X direction (IFF units)
$COEFFS 6 is the translation in the Y direction (IFF units)

$COLOUR
INTEGER Contains the colour (from the FRT) of the current feature.

$CONSTRUCTION_FC
INTEGER Contains the feature code to be used when a feature is constructed

$CONSTRUCTION_GT
INTEGER Contains the graphical type of the feature code to be used when a feature is constructed

$CONSTRUCTION_LAYER
INTEGER Contains the number of the layer that constructed features go in

$CONSTRUCTION_MAP
INTEGER Contains the number of the map that constructed features go in

$CPxzz n Must be followed by a map number in the range 1 - 100.
x is either X or Y and zz is one of NW, SW, SE or NE.
REAL Contains coordinates of the control points for each map

$CPXNW n is the X coordinate of the NW control point for map n
$CPYNW n is the Y coordinate of the NW control point for map n
$CPXSW n is the X coordinate of the SW control point for map n
$CPYSW n is the Y coordinate of the SW control point for map n
$CPXSE n is the X coordinate of the SE control point for map n
$CPYSE n is the Y coordinate of the SE control point for map n
$CPXNE n is the X coordinate of the NE control point for map n
$CPYNE n is the Y coordinate of the NE control point for map n

$CURSINGEOMETRY n Must be followed by the number of a defined area type geometry.
INTEGER -1 if the cursor is in the specified geometry, +1 if it is on the boundary, else 0.

$CURSINIMAGE
INTEGER The number of the selected image in which the cursor lies, or 0 if the cursor is not in a selected image.

$CURSINREGION n Must be followed by a valid region number.
INTEGER -1 if the cursor is in the specified region, +1 if it is on the boundary, else 0.

$CURSINWIN [r]
INTEGER -1 if the cursor is in the current window, else 0. May be followed by a number in the range 0.0 -> 1.0, to indicate the part of the window to be tested. For example, a value of 0.8 indicates that a window (centred on the centre of the screen) with sides 0.8 that of the screen will be tested.

$CURSX
REAL Contains X value of cursor position (IFF units)

$CURSY
REAL Contains Y value of cursor position (IFF units)

$CURSZ
REAL Contains Z value of cursor position (IFF units)
Note the cursor may not have a Z value. The variable $CURSZ_EXIST tells if this variable is valid

$CURSZ_EXIST
INTEGER -1 if the variable $CURSZ contains valid data, else 0

$CUTGEOMETRY n Must be followed by a valid area type geometry number.
INTEGER -1 if the found feature cuts the specified area geometry, else 0

$CUTREGION n Must be followed by a valid region number.
INTEGER -1 if the found feature cuts the boundary of given REGION, else 0.

$DATETIME
CHARACTER Contains the current date and time, in VMS format.

$DISPLAY
INTEGER -1 if there is a graphics display in use on the LITES2 workstation, else 0.

$DISPLAYCOLUMNS
INTEGER Contains the number of pixels across the display that has been selected with the DISPLAY NUMBER command. (Only available on certain hardware).

$DISPLAYNUMBER
INTEGER Contains the current display number (ie the DISPLAY selected by the DISPLAY NUMBER command)

$DISPLAYROWS
INTEGER Contains the number of pixels up and down the display that has been selected with the DISPLAY NUMBER command. (Only available on certain hardware).

$DISTANCE
REAL Contains the distance in IFF units from the start of the feature to the current cursor position, measured along the feature.

$ELAPSEDSEC Synonym for $SYSELAPSED

$END
INTEGER -1 if at first or last point of found feature, else 0

$EOF
INTEGER -1 if at end of file has been read with the FILE READ command, else 0

$EXIT_RANGE n Must be followed by an integer in the range 1 - 4.
DOUBLE Contains the range (in absolute coordinates) of the last map that was output.

$EXIT_RANGE 1 is the minimum X coordinate (IFF units)
$EXIT_RANGE 2 is the maximum X coordinate (IFF units)

$EXIT_RANGE 3 is the minimum Y coordinate (IFF units)
$EXIT_RANGE 4 is the maximum Y coordinate (IFF units)

$FC
INTEGER Contains the feature code of the found feature.

$FILELINE
CHARACTER Contains the text string read with the latest FILE READ command.

$FILENAME n Must be followed by a valid file number
CHARACTER Contains the name of the specified file. This will fail if the specified file is not open.

$FILESELECTED
INTEGER Contains the number of the file selected by the most recent FILE command that selects files. If no file is selected, then 0 is returned.

$FILESTATUS n Must be followed by a valid file number
INTEGER Contains the status of the specified file

0 - file closed
1 - file opened for read
2 - file opened for write
3 - file opened for append

$FINDTOL
REAL Contains the current find radius. If the find radius is fixed, then the value represents IFF units, and can be reset at any time by the commands

```
UNITS IFF
TOLERANCE FIND 'xxxx
```

(where xxxx is a user variable that has been assigned the value in $FINDTOL).

If the find radius is currently being zoomed, then the value represents screen mm, and it can be reset at any time by the command

```
TOLERANCE FIND 'xxxx
```

$FIND_COUNT
INTEGER After a FIND command, contains the number of found features which repeated use of FIND (without moving the cursor) will cycle round. 0 if the use of FIND would find new features. See also $FIND_ITEM.

$FIND_ITEM
INTEGER After a FIND command, contains the position of the found feature in the list of nearby features which repeated use of FIND (without moving the cursor) would cycle round. It will be 1 after an initial FIND, and range up to $FIND_COUNT for subsequent FINDs.

$FIRST
INTEGER -1 if at first point of the found feature, else 0.

$FIXEDFIND
INTEGER -1 if the find radius is currently fixed, and is not altered when the picture is zoomed, else 0.

$FLY_TRANSFORMATION
INTEGER -1 if displaying on a different projection from the data, else 0

$FOUND
INTEGER -1 if there is a found feature, else 0.

$FRT
CHARACTER Contains the name of the FRT file currently in use.

$FSN
INTEGER Contains the feature serial number of the found feature. Note that $FSN is not necessarily unique. The only way of uniquely identifying a feature is to save $IFFADDR (and $MAP if there is more than one map).

$GEOG_STRING
CHARACTER Contains the latitude and longitude of the current cursor position, in degrees, minutes and seconds. This variable only exists if all maps have valid version 2 map descriptors.

$GEOMETRY n Must be followed by a valid geometry number.
INTEGER -1 if the specified geometry exists, else 0

$GEOMETRY_PARTS n Must be followed by a valid geometry number.
INTEGER Contains the number of parts that a geometry consists of.

$GEOMETRY_TYPE n Must be followed by a valid geometry number.
INTEGER Contains the type of the specified geometry

0 - point type geometry
1 - line type geometry
2 - area type geometry

$GROUP
CHARACTER Contains a list of the group names, separated by commas, that the found feature is in.

$GROUP_FC n Must be followed by an integer.
CHARACTER Contains a list of the group names, separated by commas, that the specified feature code is in. If the feature code does not exist in the FRT, an error occurs

$GT
INTEGER Contains the graphical type of the found feature.

$GT_FC n Must be followed by an integer.
INTEGER Contains the graphical type of the specified feature code. If the feature code does not exist in the FRT, 0 is returned.

$HADSELECT
INTEGER -1 if there is any selection in force, else 0

$HADSELECT_AC
INTEGER -1 if there is any selection by AC in force, else 0

$HADSELECT_CATEGORY
INTEGER -1 if there is any selection by text category in force, else 0

$HADSELECT_FC
INTEGER -1 if there is any selection by feature code in force, else 0

$HADSELECT_FLAG
INTEGER -1 if there is any selection by edited, unedited or deleted in force, else 0

$HADSELECT_FSN
INTEGER -1 if there is any selection by feature serial number in force, else 0

$HADSELECT_GEOMETRY
INTEGER -1 if there is any selection by geometry in force, else 0

$HADSELECT_LAYER
INTEGER -1 if there is any selection by layer in force, else 0

$HADSELECT_MAP
INTEGER -1 if there is any selection by map in force, else 0

$HADSELECT_PRIORITY
INTEGER -1 if there is any selection by feature code priority in force, else 0

$HADSELECT_REGION
INTEGER -1 if there is any selection by region in force, else 0

$HADSELECT_STYLE
INTEGER -1 if there is any selection by text style in force, else 0

$HEIGHT
REAL Contains the height of found text feature (mm).

$HWTYPE
CHARACTER Contains the type of hardware this version of LITES2 runs on.

$IFFADDR
INTEGER Contains the address in the IFF file of the found feature. This provides a unique reference (within a map) of the feature. Note that $FSN is not necessarily unique.

$IFF_REVISION n Must be followed by a valid map number.
INTEGER Contains the output revision level (0 or 1) of the specified input file.

$IMAGEASPECT
REAL Contains the 'aspect' at the cursor position, derived from an image file in the same way as $IMAGEVALUE. The aspect is the direction of the normal to the surface, or the direction of maximum 'downhill' gradient. It is measured in degrees clockwise from North.

$IMAGECOLUMNS n Must be followed by an image number in the range 1 - 8.
INTEGER Contains the number of columns in the specified image.

$IMAGEGRADIENT
REAL Contains the gradient, or slope, at the cursor position, derived from an image file in the same way as $IMAGEVALUE.

$IMAGENAME n Must be followed by an image number in the range 1 - 8
CHARACTER Contains the name of the specified image

$IMAGEORIGINX n Must be followed by an image number in the range 1 - 8
DOUBLE Contains the X value of the origin for the specified image

$IMAGEORIGINY n Must be followed by an image number in the range 1 - 8
DOUBLE Contains the Y value of the origin for the specified image

$IMAGEPIXSX n Must be followed by an image number in the range 1 - 8
REAL Contains the pixel size in X for the specified image

$IMAGEPIXSY n Must be followed by an image number in the range 1 - 8
REAL Contains the pixel size in Y for the specified image

$IMAGEROWS n Must be followed by an image number in the range 1 - 8.
INTEGER Contains the number of rows in the specified image.

$IMAGEVALUE
INTEGER or REAL Contains the value extracted from the image file pixel in which the cursor currently lies. An error message is given if the cursor does not lie within one of the images specified in the most recent IMAGE SELECT command. In the event of several images overlapping, the highest numbered image will be used. The image from which the value is taken need not be currently visible on the screen.

$IMAGE_EXIST n Must be followed by a valid image number.
INTEGER -1 if the specified image exists, else 0.

$IMAGE_RANGE_zzzz n Must be followed by a map number in the range 1 - 100.
zzzz is one of XMIN, XMAX, YMIN, YMAX, ZMIN or ZMAX
DOUBLE Contains the range of each image in absolute units.

$IMAGE_RANGE_XMIN n is the minimum X coordinate in image n
$IMAGE_RANGE_XMAX n is the maximum X coordinate in image n
$IMAGE_RANGE_YMIN n is the minimum Y coordinate in image n
$IMAGE_RANGE_YMAX n is the maximum Y coordinate in image n
$IMAGE_RANGE_ZMIN n is the minimum Z coordinate in image n
$IMAGE_RANGE_ZMAX n is the maximum Z coordinate in image n

$IMAGE_SETUP
INTEGER -1 if the vector data has been setup to align with the image(s), else 0

$INGEOMETRY n Must be followed by a valid area type geometry number.
INTEGER -1 if the found feature is entirely inside specified area geometry, else 0

$INREGION n Must be followed by a valid region number.
INTEGER -1 if the found feature is entirely inside given REGION, else 0.

$INVISIBLE
INTEGER -1 if the current vector of the found feature is INVISIBLE, else 0

$LAST
INTEGER -1 if at last point of the found feature, else 0.

$LATITUDE
DOUBLE Contains the latitude of the current cursor position in decimal degrees. May only be used if all maps have valid version 2 map descriptors.

$LAYER
INTEGER Contains the layer number of the found feature.

$LAYER_EXIST n Must be followed by a valid layer number.
INTEGER -1 if the specified layer exists, else 0.

$LENGTH
REAL Contains the total length of the found feature in IFF units.

$LIMITS n Must be followed by an integer in the range 1 - 4.
REAL Contains the coordinates of the limits of the LITES2 working area. This is the total range of the maps that were originally read in, plus 5% all round. Note that the range may have been altered by subsequent edits.

$LIMITS 1 is the minimum X coordinate (IFF units)
$LIMITS 2 is the maximum X coordinate (IFF units)
$LIMITS 3 is the minimum Y coordinate (IFF units)
$LIMITS 4 is the maximum Y coordinate (IFF units)

$LINE
REAL Contains the length of the current vector of the found feature, in IFF units. For non-linear features (texts, symbols and circle arcs) this variable is identical to $LENGTH.

$LOCATION
INTEGER Contains text location field for text.

$LONGITUDE
DOUBLE Contains the longitude of the current cursor position in decimal degrees. May only be used if all maps have valid version 2 map descriptors.

$MAP
INTEGER Contains the map number of the found feature.

$MAPNAME n Must be followed by a map number in the range 1 - 100.
CHARACTER Contains the name of the specified source map.

$MAPSTATUS n Must be followed by a valid map number
INTEGER Contains the status of the specified map (IFF file)

0 - not opened yet
1 - opened with READ command (map is read only)
2 - opened with IFF command
3 - opened with INSITU command

$MAPTOTAL
INTEGER Contains the total number of maps read in, excluding any since removed using the QUIT n command.

$MAP_NUMBER
INTEGER Contains the next map number containing the string specified by an ASK MAP_NUMBER command

This variable is a synonym for the variable $ASK_INT 1

$MAP_SHEET [n] May be followed by an integer representing the scale of the map sheet whose name is required (eg 1250 or 10560). If this value is not supplied the sheet scale currently being used by LITES2 is used.

CHARACTER Contains the name of the map sheet that covers the current cursor position. By default, the sheet naming convention is that used by the Ordnance Survey of Great Britain. This provides map names for the scales 1/1250, 1/2500, 1/10000, 1/10560, 1/25000. In addition a name will be generated for scales greater than 1/250000.

This default algorithm can be substituted by a user supplied one which can be passed either the absolute position of the cursor or its geographical position. This substitution is achieved by supplying a shared image pointed at by the logical name LSL$LITES2_GET_SHEET_ROUTINES. Example source files that contain instructions to do this are supplied in LSL$PUBLIC_ROOT:[LITES2.ROUTINES.EXAMPLES] and are called GET_SHEET_GEOG_EXAMPLE.FOR and GET_SHEET_GRID_EXAMPLE.FOR.

$MAXFSN n Must be followed by a map number in the range 1 - 100.
INTEGER Contains the maximum FSN number for the specified map (excluding features specified by any FIDUCIAL command).

$MDOFFSET n Must be followed by an integer in the range 1 - 2.
DOUBLE Contains the coordinate offset for the LITES2 coordinate system ($MDOFFSET 1 is x value, $MDOFFSET 2 is y value). This value is to be added to any IFF coordinate, to get the real projection coordinate of the point.

$MDSCALE
REAL Contains the scale of the LITES2 working area. It is the scale of the first map that is read into LITES2. The value comes either from the MD entry in the IFF file, or if that value is less than or equal to 0.0 or DESCRIPTOR has been disabled, then from the MH entry.

$MHARR n Must be followed by an integer in the range 1 - $MHLEN
INTEGER Contains the contents of the MH entry, of the first map that was read into LITES2, as a series of 32 bit integers. See IFFLIB documentation for the format of the MH entry.

$MHLEN
INTEGER Contains the number of 32 bit integers in $MHARR.

$MMFACTOR
REAL This is the factor to be used to convert from IFF units to sheet mm.

$MOANED
INTEGER -1 if the last command caused an error, else 0.

$MODTCC
INTEGER Contains the text component code of the current subtext of the text feature being modified.

$MODTEXT
CHARACTER Contains the text string for the current subtext of the feature being modified.
For composite texts, this is the current text component (i.e. the component that the cursor is on)

$MODTEXTLEN
INTEGER Contains the length of the text string of the current subtext of the feature being modified.
For composite texts, this is the current text component (i.e. the component that the cursor is on)

$OPTBIG
INTEGER -1 if BIG is currently enabled, else 0.

$OPTBLIN
INTEGER -1 if BLINK is currently enabled, else 0.

$OPTCOMP
INTEGER -1 if COMPOSITE text is currently enabled, else 0.

$OPTCONT
INTEGER -1 if CONTINUE is currently enabled, else 0.

$OPTENDS
INTEGER -1 if ENDS is currently enabled, else 0.

$OPTHEIG
INTEGER -1 if HEIGHT is currently enabled, else 0.

$OPTINFO
INTEGER -1 if INFORM is currently enabled, else 0.

$OPTLEAR
INTEGER -1 if LEARNER is currently enabled, else 0.

$OPTPATT
INTEGER -1 if PATTERN is currently enabled, else 0.

$OPTPSIZE
INTEGER -1 if PSIZE is currently enabled, else 0.

$OPTSUBS
INTEGER -1 if SUBSTITUTE is currently enabled, else 0.

$OPTVERI
INTEGER -1 if VERIFY is currently enabled, else 0.

$OUTGEOMETRY n Must be followed by a valid area type geometry number.
INTEGER -1 if the found feature is entirely outside specified area geometry, else 0

$OUTREGION n Must be followed by a valid region number.
INTEGER -1 if the found feature is entirely outside given REGION else 0.

$OVERLAY
INTEGER Contains the overlay number that the current feature will be drawn in. It is the lowest numbered overlay that the overlay selections satisfy for this feature; if no overlay satisfies the overlay selections, overlay number 0 is returned

$OVERLAYNUMBER
INTEGER Contains the current overlay number (ie the OVERLAY selected by the OVERLAY NUMBER command)

$P n
CHARACTER Contains the value of the n'th parameter supplied to the current command file or macro, or a null string if none was supplied.

$PATTERN
INTEGER Contains the pattern index for linear features.

$PC
INTEGER Contains the process code of the found feature.

$PCOUNT
INTEGER Contains the number of parameters supplied to the current command file or macro.

$PI
DOUBLE Contains the value of PI, the ratio of the circumference of a circle to its diameter.

$PID
CHARACTER Contains the process identification string for the current process

$PLINE
CHARACTER Contains the entire line of parameters supplied to the current command file or macro, or a null string if none was supplied.

$POINT

INTEGER -1 if at a point of the found feature, else 0.

$POINTNO

INTEGER Contains the point number of the cursor on the found feature.

$PRIVPOINT

INTEGER -1 if at a point has an attribute with a value specified by a previous PRIVILEGE POINT command, else 0.

$PSIZE

INTEGER Contains the point size of found text feature.

$RANDOM [n] May be followed by a integer to be used as a seed value.

REAL Contains a random number in the range 0.0 - 1.0.
NOTE: subsequent references to $RANDOM will produce different values. By using the optional integer a repeatable series of random numbers can be initiated.

$RANGE_PROBLEM

INTEGER Reports problems in calculating the sectored area after reading an IFF file and transforming it to another projection.

This variable is valid after going from INITIAL to READY state, or when an IFF file has been read in READY state. It remains set until another file is read. It is -1 if a point has been detected outside the sectored area after transformation and 0 if no such point was detected.

$RANGE_zzzz n Must be followed by a map number in the range 1 - 100.

zzzz is one of XMIN, XMAX, YMIN or YMAX
REAL Contains the (original) range for each map.

$RANGE_XMIN n is the minimum X coordinate in map n
$RANGE_XMAX n is the maximum X coordinate in map n
$RANGE_YMIN n is the minimum Y coordinate in map n
$RANGE_YMAX n is the maximum Y coordinate in map n

NOTE

The coordinates are in terms of the current LITES2 space, and reflect the range of the IFF file that was originally read in. They do not take account of any subsequent edits.

$REFRESH

INTEGER Contains the number of points that are refreshed when an object is found.

$REGION n Must be followed by a valid region number.

INTEGER -1 if the specified region exists, else 0.

$REGIONAREA n Must be followed by a valid region number.

DOUBLE Contains the area enclosed by the specified region in square IFF units.

A positive area indicates that the region runs in a clockwise direction, negative areas indicate a counter clockwise direction.

$RESPOND
INTEGER -1 if in second level interactive input (ie the RESPOND command has been given - awaiting a CONTINUE command), else 0.

$SCRFACTOR
REAL This is the factor to be used to convert from IFF units to screen mm.

$SECONDARY
INTEGER Contains the secondary code (from the FRT) for the current feature. See FRTLIB documentation, for what this represents for different types of features.

$SIZE
REAL Contains the size of the currently found symbol.

$SIZE_FC n Must be followed by an integer.
REAL Contains the size entry of the specified feature code in the current FRT file.

$SRI
CHARACTER Contains the name of the SRI file currently in use.

$STATE
CHARACTER Contains the current program state. (See HELP STATE)

$STYLE
INTEGER Contains the style index for the current text feature.

$SYSBUFIO
INTEGER Contains the number of buffered input/output operations during this run of LITES2. This includes operations to terminals on serial lines.

$SYSCPU
REAL Contains the CPU time elapsed in seconds during this run of LITES2.

$SYSDIRIO
INTEGER Contains the number of direct input/output operations during this run of LITES2. This includes operations to disc files (in particular IFF files).

$SYSELAPSED
REAL Contains the time elapsed in seconds during this run of LITES2.

$SYSFAULTS
INTEGER Contains the number of page faults incurred during this run of LITES2. Useful as a performance tool to check whether LITES2 would benefit from increased memory or working set size.

$TABLE
INTEGER -1 if there is a digitising table in use on the LITES2 workstation, else 0

$TABLEXY n Must be followed by an integer in the range 1 - 2.
REAL Contains the coordinates of the table cursor after the last successful invocation of the command ASK TABLE. Until the command ASK TABLE has been executed successfully, contains the lower left coordinate of the available LITES2 working area.

$TABLEXY 1 is the X coordinate (IFF units)
$TABLEXY 2 is the Y coordinate (IFF units)

This variable is a synonym for the variable $ASK_REAL

$TCC
INTEGER Contains the text component code of the current subtext of found text feature.

$TEXT
CHARACTER Contains the text string for the current text feature. For composite texts, this is the text component that the text was found by.

$TEXTLEN
INTEGER Contains the number of characters in the text string for the current text feature. For composite texts, this is the text component that the text was found by.

$TEXTTOTAL
INTEGER Contains the number of subtexts in a composite text feature.

$TOPFC
INTEGER Contains the highest feature code in the FRT table currently in use.

$TOPGEOMETRY
INTEGER Contains the number of the highest geometry that has been defined.

$TOPMAP
INTEGER Contains the highest map number currently in use.

$TRI
CHARACTER Contains the name of the TRI file currently in use.

$UIC
CHARACTER Contains the user identification code of the user.

$UNIT_DESC
CHARACTER Contains the descriptive string specified in the last UNITS FACTOR command.

$UNIT_FACTOR
REAL Contains the number specified in the last UNITS FACTOR command.

$UNIT_TYPE
INTEGER Contains the type of UNITS command currently in force. 0 - none or NORMAL, 1 - IFF, 2 - MMS, 3 - FACTOR. Contains the number specified in the last UNITS FACTOR command.

$USER
CHARACTER Contains the user name of the user

$VERSION
CHARACTER Contains the LITES2 version number.

$WARP_COEFFS_IMAGE Must be followed by an integer in the range 1 - 8 identifying which coefficient is required.
DOUBLE Contains the coefficients for transforming an image coordinate to a map coordinate.

$WARP_COEFFS_MAP Must be followed by an integer in the range 1 - 8 identifying which coefficient is required.
DOUBLE Contains the coefficients for transforming a map coordinate to an image coordinate.

$WARP_DIRTY
INTEGER -1 if the warp points have been altered since WARP FIT was performed, otherwise 0.

$WARP_FITTED
INTEGER -1 if WARP FIT has been performed, and WARP ON will activate the fit, otherwise 0.

$WARP_IMAGE_N
INTEGER The number of image control points for warping.

$WARP_IMAGE_X Must be followed by an integer identifying the control point.
REAL The X coordinate of an image control point.

$WARP_IMAGE_Y Must be followed by an integer identifying the control point.
REAL The Y coordinate of an image control point.

$WARP_MAP_N
INTEGER The number of map control points for warping.

$WARP_MAP_X Must be followed by an integer identifying the control point.
REAL The X coordinate of a map control point.

$WARP_MAP_Y Must be followed by an integer identifying the control point.
REAL The Y coordinate of a map control point.

$WARP_MODE
INTEGER The current warp mode. 0 if warping is off, 1 if warping the image, 2 if warping the map.

$WARP_RESIDUAL_X Must be followed by an integer identifying the control point.
REAL The X residual at a warp control point (transformed map coordinate minus image coordinate).

$WARP_RESIDUAL_Y Must be followed by an integer identifying the control point.
REAL The Y residual at a warp control point (transformed map coordinate minus image coordinate).

$WARP_RMS_X
REAL The root mean square X residual at the warp control points (calculated at the time the warp was fitted).

$WARP_RMS_Y
REAL The root mean square Y residual at the warp control points (calculated at the time the warp was fitted).

$WARP_TRANSFORM
CHARACTER The name of the current warp transform (LINEAR, HELMERT, AFFINE, EXTENDED, or PROJECTIVE).

$WIDTH
REAL Contains the line width of the current linear feature.

$WINDOW n Must be followed by an integer in the range 1 - 4.
REAL Contains the coordinates of the limits of the current window.

```
$WINDOW 1 is the minimum X coordinate  (IFF units)
$WINDOW 2 is the maximum X coordinate  (IFF units)
$WINDOW 3 is the minimum Y coordinate  (IFF units)
$WINDOW 4 is the maximum Y coordinate  (IFF units)
```

$ZOOM
REAL Gives the number of times that the current picture on the screen is magnified from the full map on the screen.

6.13 **Functions**

Within expressions the following functions may be used. Their argument may optionally be enclosed in parentheses.

SIN COS TAN ASIN ACOS ATAN ABS LN LOG

Trigonometric functions deal with angles in degrees, and return a real value.

ABS returns a real or an integer value, depending on the context.

LN and LOG return the natural (base e) and common (base 10) logarithm respectively as a real value.

For example, the command LET R=SIN30 will set variable R to 0.5, while the command LET R=COS('ANGLE'+30) will set R to the cosine of ANGLE+30, assuming that ANGLE is a REAL variable, and that SUBSTITUTION is enabled.

7 **Commands**

Each command of LITES2 is described below, grouped approximately by function. The classifications used are; initialisation, option, command handling, general, identification, construction, constraint, positioning, editing, joining, text and symbol, attribute, ancillary coding, interrogation, windowing, exiting, and miscellaneous. Use the index to locate the description of a particular command.

The primary commands of LITES2 in alphabetic order are as follows.

ABANDON	ABORT	ABSOLUTE	ADD	AFTER	ALIGN
ALTER	ANCILLARY	AND	ANNOTATION	ARC	ASK
BASE	BEND	BRIDGE	CANCEL	CHANGE	CIRCLE
CLIP	CLOSE	COLLAPSE	CONTINUE	COPY	CREATE
CURVE	DEBUG	DECLARE	DELETE	DEPOSIT	DESCRIBE
DESELECT	DISABLE	DISPLAY	DISTANCE	DRAW	DUMP
EDGEMATCH	EDIT	ENABLE	END	ENDMACRO	EXAMINE
EXIT	EXTEND	FEATURE	FIDUCIAL	FILE	FILTER
FIND	FIRST	FOLLOW	FORCE	FRACTION	FREE
FRT	GEOGRAPHICAL		GEOMETRY	GET	HELP
IFF	IMAGE	INCLUDE	INQUIRE	INSERT	INSITU
INTERPOLATE	INVISIBLE	JFALSE	JOIN	JTRUE	JUMP
LABEL	LARGER	LAST	LATLONG	LET	LOCATE
LOOP	MACRO	MAPS	MARGIN	MATCH	MEND
MENU	MERGE	MESSAGE	MIDDLE	MODIFY	MOVE
NEXT	NULL	OFFSET	ON	OPERATION	OR
ORIENT	OSSETUP	OVERLAY	PARAGRAPH	PICTURE	PING
PLOT	POLARC	POLYGON	POINT	POSITION	PREVIOUS
PRIORITY	PRIVILEGE	PROJECTION	PROMPT	PROPAGATE	PTOLERANCE
PUCK	PUT	QUIT	RANGE	RASPBERRY	READONLY
RECOVER	RECTANGLE	REFRESH	REGION	REMOVE	RENAME
REPEAT	REPLACE	RESPOND	REVERSE	ROTATE	ROUTINE
SAVE	SCALE	SCROLL	SEARCH	SECTOR	SELECT
SET	SETUP	SHEET	SHOW	SMALLER	SORT
SPAWN	SPLIT	SQUARE	SRI	START	STRETCH
SUBSTITUTE	SUPPRESS	TAKE	TEST	TEXT	THIS
TIE	TIME	TOGGLE	TOLERANCE	TRACK	TRAIL
TRANSFORM	TRI	TURN	UNITS	UNSET	USER
VERIFY	VIEW	WAIT	WARP	WHOLE	WINDOW
WORKSTATION	WRITE	ZOOM			

7.1 **Initialisation Commands**

7.1.1 **FRT**

Specifies FRT file name and causes it to be read in. An FRT file contains a Feature Representation Table which tells the program how to draw features according to their Feature Codes (FCs). Default filename is LSL$FRT:---.FRT;0

If used in READY state to read a new FRT, then great care should be taken that the feature codes in the file are compatible with any maps which are in use at the time. This means that all the feature codes are present, and any changes of graphical type are to a similar type. It is reasonable to change between line, curve, symbol string, and area, and also from one type of symbol to another. Note that LITES2's spatial index for a feature is not recalculated, so for instance if a symbol or text is changed to one of a different size, then attempts to find it by a point on its bounding box may fail.

If the new FRT file has missing codes, or codes which have graphical types inconsistent with the old ones (e.g. text instead of line), then LITES2 may fail and enter its "collapse" routine when these feature codes are used.

```
Format:  FRT  filename

eg       FRT DR1:[LSL.FRT]TESTCARD
or       FRT OS
```

Valid in states INITIAL READY

7.1.2 **SRI**

Specifies SRI file name. An SRI (Symbol Representation IFF) file holds the shapes of symbols. Default filename is the FRT filename (qv) with .SRI substituted for .FRT

If used in READY state to read a new SRI, then great care should be taken that the symbols in the file are compatible with those in any maps which are in use at the time.

```
Format:  SRI  filename

eg       SRI DR1:[LSL.FRT]TESTCARD
or       SRI OS
```

Valid in states INITIAL READY

7.1.3 **TRI**

Specifies TRI file name. A TRI (Text Representation IFF) file holds the shapes of characters. Default filename is the FRT filename (qv) with .TRI substituted for .FRT

If used in READY state to read a new TRI, then great care should be taken that the characters in the file are compatible with the text features in any maps which are in use at the time.

Format: TRI filename

eg   TRI DR1:[LSL.FRT]TESTCARD
or   TRI OS

Valid in states INITIAL READY

7.1.4 **MAPS**

Specify number of IFF files to be read in, or the number of IFF files that can be open at once.

Format: MAPS [subcommand] integer

Valid in state INITIAL

* **MAPS [IN]**
  Allows a number of IFF files to be specified in INITIAL state. Default is 1 so command is only needed if more than one map is to be read using IFF, READONLY, or INSITU commands.
  Note that only maps specified in INITIAL state can be set up on a digitising table.
  The command MAPS 0 may be used, provided that at least one image file has been specified using IMAGE commands, to cause LITES2 to move into READY state without any IFF files. The image files can then be displayed as required.

  Format: MAPS [IN] integer

  eg   MAPS 4
  or   MAPS IN 4

* **MAPS OPEN**
  Specifies number of IFF files that LITES2 can keep open at once when multiple maps are being used.
  LITES2 opens and closes IFF files as they are accessed, and if dealing with a large number of files (eg when using LITES2 in a "continuous mapping" mode) it may be advantageous to allow more than a maximum of 3 (the default) to be open at once.
  Users should note that there is a limit to the number of files (of all kinds) that they can have open at once. As the MAPS OPEN command only takes effect when LITES2 accesses files (after they have been read in) unrecoverable errors can occur if this number of open files is exceeded at this stage.

  Note that it is NOT necessary to allow LITES2 to open all the IFF files it is accessing at once. Unless the MAPS OPEN command shows an obvious performance improvement, it is recommended that it should not be used.

Format: MAPS OPEN integer

7.1.5 **IFF**

Specifies name of IFF file to be edited. Use READONLY command (qv) if file is not to be amended. The IFF file will be allocated the lowest available map number. Default filename is LSL$IF:---.IFF;0

Format: IFF filename

```
eg       IFF DU3:[LSL.IFF]TESTCARD
or       IFF J5012
```

Valid in states INITIAL READY

7.1.6 **INSITU**

Specifies IFF file to be edited in situ. Not recommended for normal use as original data can be lost if the session is aborted for any reason. EXIT, DUMP, and QUIT commands all have the same effect: the edits are still preserved. See IFF command for syntax.

Format: INSITU filename

Valid in states INITIAL READY

7.1.7 **READONLY**

Specifies IFF file to be read in and inspected without amendment. A subset of the file can still be written out using SELECT and WRITE commands, and features in the map can be copied to another (not read-only) file.
See IFF command for syntax, and for amending files.

Format: READONLY filename

Valid in states INITIAL READY

7.1.8 **MENU**

Specifies the dimensions and name of a menu. Menu names must consist of up to 16 alphabetic characters (including underline).
This information is used to define the shape and size of the menu, and also the reserved macro names which are used to program the menu.
The operator will be prompted to digitise the corners of the menu when all the IFF files to be read have been specified, or if in READY state, then when a SETUP AGAIN command is given.

Format: MENU x y name

eg   MENU 9 14 CMDMEN

Valid in states INITIAL READY

7.1.9 **PUCK**

Specifies device number, number of buttons or boxes and name of a digitising puck, screen menu, function buttons or mouse (or tracker ball). Puck names must consist of up to 16 alphabetic characters (including underline).
This information is used to define the reserved macro names which are used to program the puck buttons using the MACRO command (qv).
A puck can only be defined once in any run of LITES2

Device numbers are 0 for keyboard function buttons, 1 for screen, 2 for bitpad, 3 for the digitising table, 4 for tracker ball or mouse, 5 for separate function buttons, and 6 for a stereo digitising instrument. Other numbers may be available in some LITES2 implementations - see the appropriate Workstation Guide for details. Device numbers with no corresponding physical device may be used, but in this case the puck macros may only be used as normal commands - they will never be generated automatically by the interactive controls.

The devices are initialised using the appropriate ENABLE command (qv), except for the keyboard function buttons which are always enabled.

Format: PUCK devno buttons name

eg   PUCK 3 16 ALTEKPUCK

Valid in states INITIAL READY

7.1.10 **TRACK**

Specifies the device number for a tracking area to be set up on a digitising surface.
See the PUCK command for information on device numbers.

Format: TRACK integer

eg   TRACK 2

Valid in states INITIAL READY

7.1.11 **DESCRIBE**

Describe certain entities to allow them to be displayed.

Format: DESCRIBE subcommand

Valid in states INITIAL READY

* **DESCRIBE MACRO**
  Describe the annotation for a macro. This description is used to annotate screen menu boxes when they are available, and any macro so described will have the annotation output by the SHOW MACRO and SHOW MENU commands (qv).

  Format: DESCRIBE MACRO macroname description

  eg DESCRIBE MACRO SCREEN7 "Zoom 3"

* **DESCRIBE SCREENMENU**
  Describe the layout for a screen menu (when available). This command only has any effect if the hardware allows screen menus. The screen menu must have previously defined as a PUCK on line 1 (the screen).

  The arguments give:

  1. the number of columns in the menu
  2. the number of rows in the menu
  3. the location of the menu with respect to the locating point (in the range 0 - 8, as for text justification)
  4. the x size of the menu (in mm on the screen or as a fraction of the screen if in the range 0.0 to 1.0)
  5. the y size of the menu (in mm on the screen or as a fraction of the screen if in the range 0.0 to 1.0)
  6. the x position of the locating point of the menu (as a fraction between 0.0 and 1.0 of the X extent of the screen)
  7. the y position of the locating point of the menu (as a fraction between 0.0 and 1.0 of the Y extent of the screen)
  8. the name of the menu
  9. the label to be written into the top of the menu when it is displayed

  The menu is displayed on the screen by the ENABLE SCREENMENU command (qv).

  ```
  Format:  DESCRIBE SCREENMENU integer integer integer real real real
                               real menuname label
  ```

  ```
  eg   DESCRIBE SCREENMENU 2 16 6 50.0 100.0 1.0 0.0 screen Our menu
  or   DESCRIBE SCREENMENU 2 16 6 0.15  0.3  1.0 0.0 screen Our menu
  ```

  This command describes a screen menu that is 2 columns by 16 rows, 50.0 mm wide and 100mm long (or in the second case 0.15 times the screen width by 0.3 times the screen height), located at the bottom right of the screen, that had previously been defined by a PUCK command to be called SCREEN. It will have the header "Our menu".

7.1.12 **SCALE**

Specifies drawing scale to be used for texts, symbols etc. As these are specified in the FRT in sheet mm, a correspondence between IFF file units and sheet mm must be specified somehow.

The default action is to take the SHEET scale from the IFF Map Descriptor (MD) entry. If this is unset, or DISABLE DESCRIPTOR is used, then a scale from the IFF Map Header (MH) entry is used. If this is also unset, or if DISABLE EXTERNAL is used, then SCALE FACTOR 1 is assumed.

If the IFF units are related to the map sheet (eg inches, table units etc) then SCALE FACTOR should be used to relate them to sheet mm.

If the IFF units are related to the ground (eg map projection coordinates) SCALE IFF should be used to relate these units to ground mm (the default is SCALE IFF 1000 - ie IFF units are metres) and SCALE SHEET relates ground mm to map sheet mm.

If the nature of the IFF units is unknown, SCALE AUTO will produce a scale factor such that 1mm on the screen represents 1mm on the map

If the SCALE SHEET command has been given, but it is subsequently required to revert to the default action of taking the scale from the map header or map descriptor, this may be achieved by giving a SCALE FACTOR or SCALE AUTO command, followed by an appropriate SCALE IFF command.

Format: SCALE subcommand

Valid in state INITIAL

* **SCALE AUTO**
  Automatic scaling with whole sheet equal to whole screen.

  Format: SCALE AUTO

* **SCALE FACTOR**
  Number of sheet mm represented by one IFF file unit.

  Format: SCALE FACTOR real

* **SCALE IFF**
  Number of ground mm represented by one IFF file unit.

  Format: SCALE IFF real

* **SCALE SHEET**
  Number of ground units represented by one sheet unit. (eg 50000 for a 1:50000 sheet)

Format: SCALE SHEET real

7.1.13 **FIDUCIAL**

Defines which features are fiducial features (grid, tick marks etc) and which should not be taken into account when calculating FSNs for new features.

Format: FIDUCIAL subcommand

Valid in state INITIAL

* **FIDUCIAL LAYERS**
  All features in these layers are ignored when calculating FSNs for new features. The default is layer 0.

  To add to the current list of fiducial layers, specify the layers in the range argument.

  If no argument is given, all the existing layers are removed from the list (including layer 0, the default).

  Format: FIDUCIAL LAYERS [range]

  eg FIDUCIAL LAYERS 11-12,32

7.1.14 **SETUP**

Specifies type of digitising table setup.

Maps can be set up on the table using several procedures to determine the position of the control points of the map on the table. The control points of the map are taken from the right hand side of the CP entry in the IFF file.

Having determined the position of the control points, several different transformations can be used to relate any point digitised on the table to the map coordinate system.

By default, all the maps specified while in INITIAL state are set up on the table when the workstation is initialised (on going from INITIAL state to READY state) or after a SETUP AGAIN command. The commands SETUP CANCEL and SETUP MAP allow a selection of the maps that have been read in to be set up. SETUP INITIAL returns to the default action outlined above.

There is a limit of 9 maps that can be set up at any one time.

Format: SETUP subcommand

Valid in states INITIAL READY

* **SETUP AGAIN**
  Set up maps and menus on table again.

  Format: SETUP AGAIN

* **SETUP CANCEL**
  Cancels all the maps currently on the list to be set up, either when the workstation is initialised or after a SETUP AGAIN command. After this command has been given, only maps specified with a SETUP MAP command are put on this list.

  Format: SETUP CANCEL

* **SETUP EDGE**
  This setup uses points digitised along the neat edges of the map sheet to determine where the control points lie.
  For this setup the right hand side of the CP entry in the IFF file must represent a true rectangle.

  The operator is requested to digitise points along each map edge in turn. The number of points to be digitised on each edge can be set by the PTOLERANCE EDGESETUP command.

  The digitised points are used to define 4 straight lines, using a least squares regression, and if the Root Mean Square (RMS) of the residuals for any line is greater than a limit (that can be set by the PTOLERANCE EDGESETUP command) then the operator is required to digitise points along that edge again.

  The 4 lines are then used to calculate the corners of the map. Having done this it is possible to check if the digitised points are well enough distributed along the edges. This is done using the concept of an "ideal gap" for each edge. This is the gap that would be left between points if they were evenly distributed along the edge. Points must be greater than a minimum factor of this gap apart, and less than a maximum factor of this gap apart. These factors can be set by the PTOLERANCE EDGESETUP command. If the points on any edge do not conform to this spacing, the operator is required to redigitise points along that edge again.

  Several sets of observations can be made, the number being set by the PTOLERANCE EDGESETUP command. In this case the maximum range of the positions of the derived corner points at any corner must not exceed a specified tolerance, and the sum of the ranges must also not exceed a specified tolerance. These tolerances are given in the PTOLERANCE EDGESETUP command. If either of these criteria are not met, then the operator is requested to reobserve sets, until there are the required number of sets that fulfil the criteria. Where there are more than one acceptable group of sets, the group that gives the minimum sum of the ranges is accepted.

  The mean of the accepted sets of map corners is used as the values for the transformation.

  Format: SETUP EDGE

* **SETUP FOUR**
  4-point setup (default).
  Corners are requested in the order NW, SW, SE, NE.

  Format: SETUP FOUR

* **SETUP INITIAL**
  Cancels the effect of SETUP CANCEL, so that maps specified while in INITIAL state will be added to the list of maps to be set up on the table at the appropriate time.

  Format: SETUP INITIAL

* **SETUP MAP**
  Add the specified map to the list of maps that will be set up on the table at the appropriate time.

  Format: SETUP MAP n

* **SETUP NONE**
  No setup i.e. no document on table.

  Format: SETUP NONE

* **SETUP OSMULTI**
  OS multiple point or piecemeal setup

  For this setup the right hand side of the CP entry in the IFF file must represent a true rectangle.

  The map is divided into boxes, and the corner of each box is digitised on the table. Each point is digitised several times, and a mean taken. Observations that fall outside a predefined tolerance must be repeated. The number of boxes, number of repetitions and acceptance criteria are set by the OSSETUP command.

  Grid intersections are requested from the SW corner, going E.

  A transformation is set up for the whole map, and also one for each individual box. When a point is digitised on the table then the whole map transformation is used to determine which box the point falls in, and it is the transformation for this box that is used to determine the coordinates of the digitised point in the map coordinate system

  Format: SETUP OSMULTI

* **SETUP TABLE_COUNT**
  Used to specify the size of the counts returned by the digitising table in mm. The default value is 0.02 mm, the setting Laser-Scan recommend for ALTEK digitising tables.

  Format: SETUP TABLE_COUNT real

* **SETUP TRANSFORM**
  Determines the type of transformation used to relate table coordinates (x,y) to map coordinates (X,Y).

Where the transformation is overdetermined, a least squares solution is employed to determine the best fit and there will be residuals left at the control points. A warning is given if the maximum residual is greater than RESID_WARN times the range of the map, and the transformation will not be accepted if the maximum residual is greater than RESID_LIMIT of the range of the map. RESID_WARN and RESID_LIMIT are set by the PTOLERANCE RESIDUAL command (qv).

These residuals can be a result of observational errors (which the least squares solution should deal with sensibly). They can also occur when the control points on the source document do not define the same shape as the control points in the IFF file, perhaps because of paper distortion. (Note that in this case none of the transformations will solve the problem with any degree of certainty).

```
Format:  SETUP TRANSFORM type

         where type = AFFINE (default)
                    = EXTENDED
                    = ORTHOGONAL
                    = PROJECTIVE
```

- Affine
  This transformation is of the form
  X = a0 + a1*x + a2*y
  Y = b0 + b1*x + b2*y

  It corrects for scaling, rotation, shearing and translation. Shapes may be altered, but in a predictable way (eg a square may become a parallelogram). It only requires three points to define it, so there will be residuals left at the control points.

  This is the default transformation.

- Extended
  This transformation is of the form
  X = a0 + a1*x + a2*y + a3*x*y
  Y = b0 + b1*x + b2*y + b3*x*y

  It will force the 4 control points to fit (so there is no checking carried out), but it is not obvious how other points in the map sheet will be distorted. This transformation is used by some other Laser-Scan programs, for example LASERAID.

- Orthogonal - sometimes known as the HELMERT transformation
  This is of the form
  X = a0 + a1*x - a2*y
  Y = a3 + a2*x + a1*y

  This transformation corrects for scaling, rotation and translation. Shapes are maintained. It only requires two points to define it, so there will be residuals left at the control points.

o Projective

This is of the form

X = (a0*x + a1*y + a2) / (a6*x + a7*y +1)
Y = (a3*x + a4*y + a5) / (a6*x + a7*y +1)

This transformation uses a projective algorithm to relate table coordinates to map coordinates. It forces the 4 control points to fit (so there is no checking carried out). This transformation is often used with the EDGE setup procedure.

* **SETUP TWO**

2-point setup (SW and NE corners). This is a fast setup. It has the effect of forcing the use of the orthogonal transformation (with no checking)

Format: SETUP TWO

7.1.15 **OSSETUP**

This command has been superceded by the PTOLERANCE OSSETUP command. When used in conjunction with the SETUP TABLE_COUNTS command, then this command must be given after the SETUP TABLE_COUNTS command.

Overrides the default settings for doing a multiple point setup. It defines the number of boxes the map is to be split into along with the number of observations required at each point, the number of observations at each point that must fall within the tolerance of the mean, and that tolerance.

Format: OSSETUP fulx fuly repeat numok tolerance

```
where fulx      is the number of boxes in the X direction (integer)
      fuly      is the number of boxes in the Y direction (integer)
      repeat    number of observations of each point      (integer)
      numok     number that must be within tolerance      (integer)
      tolerance tolerance in table units                  (real)
```

The defaults are: 5 5 4 2 19.05

Valid in state INITIAL (privileged command;
only valid in initialisation file)

7.1.16 **AFTER**

Defines a command (possibly a macro or an @file command) which will be obeyed immediately after some definable event.

Format: AFTER subcommand

Valid in states INITIAL READY

* **AFTER ERROR**
  Defines a command which will be obeyed after an error has caused LITES2 to return to interactive input, while executing a macro or obeying a command read from a command file.

  If the argument is omitted, then any existing command will be removed, and no extra actions will be taken after an error occurs.

  The current command that is set up can be shown with the SHOW AFTER command.

  Format: AFTER ERROR [command]

  eg AFTER ERROR CLEAR_UP

* **AFTER INPUT**
  Defines a command which will be obeyed after the last map is read in, just after LITES2 enters READY state from INITIAL state. If there are any errors during input (such as "feature code not found"), then the command will only be obeyed if ENABLE CONTINUE is in force. The is useful for commands such as WORKSTATION COLOUR which are not valid in INITIAL state.

  If the argument is omitted, then any existing command will be removed, and no extra actions will be taken after reading the maps.

  The current command that is set up can be shown with the SHOW AFTER command.

  Format: AFTER INPUT [command]

  eg AFTER INPUT @SET_OPERATION.LCM

7.2 **Option Commands**

7.2.1 **ENABLE**

Activates specified optional facilities. Use DISABLE (qv) to deactivate facilities.

Format: ENABLE subcommand

Valid in all states (except where specified)

* **ENABLE AND**
  Use a logical AND of whatever regions have been selected.
  (ie to be considered, a feature must be in all the selected regions)
  The default setting is DISABLE AND, when a logical OR of the selected regions is carried out.
  (ie to be considered, a feature must be in at least one of the selected regions)

  NOTE

  If any one region has more than one area selected (eg INREGION and CUTREGION) and ENABLE AND has been given, no features are selected because the three areas of a region (INREGION, CUTREGION and OUTREGION) define mutually exclusive features.

  Format: ENABLE AND

* **ENABLE APPEND**
  When writing a edge matching problem file, append to existing file (if one exists). If disabled (default) an existing file is overwritten by the new information.

  Format: ENABLE APPEND

* **ENABLE BALL**
  Tracker ball (or mouse) to be used to control the cursor.

  Format: ENABLE BALL

* **ENABLE BELL**
  Audible warning when "moan" messages are output.

  Format: ENABLE BELL

* **ENABLE BIG**
  Use a big cursor (default is small cursor).

  Format: ENABLE BIG

* **ENABLE BITPAD**
  Bitpad to be used.

  Format: ENABLE BITPAD

Valid in state INITIAL

* **ENABLE BLANK**
Blank out the data behind texts with the specified colour. This command is only effective on hardware with a raster display and affects both text features and text in annotation.

The size of the area blanked out is controlled by the TOLERANCE EXPAND command.

The default colour is 0 (background). If another colour is specified then it becomes the default. The current blanking colour is shown with the SHOW ANNOTATION command. This default can be overridden in any particular overlay by the OVERLAY BLANK command.

Format: ENABLE BLANK [colour]

* **ENABLE BLINK**
Blink the cursor (default is not to blink).

Format: ENABLE BLINK

* **ENABLE BOX**
Draw text features as their bounding box, rather than as characters. This may speed up the display of map data.

Format: ENABLE BOX

* **ENABLE BRIEF**
Only commands actually executed are written to the journal file (default).
If disabled, then other commands which change the flow of control, such as JUMP, macro names, @filename, are also journalled, but are preceded by the comment character '!'.

Format: ENABLE BRIEF

* **ENABLE BUTTON**
Use function buttons.
Function buttons are not supported on all hardware. See hardware dependent reference manual.
The buttons are defined using the PUCK command on line 5.

Format: ENABLE BUTTON

* **ENABLE CHECKS**
When defining macros and user variables, check that the name does not already exist.

If CHECKS are disabled, the user is allowed to multiply define macros and variables, but every time the user defined macro table and variable table is accessed then a warning message is output (even if the multiply defined entity is not being accessed). If a multiply defined entity is accessed its value is undefined.

The purpose of this command is to speed up reading large, proven initialisation files in production environments by disabling checks while they are being read. To get the maximum benefit the following guidelines should be followed:

1. All primitive LITES2 commands should be preceded by a "%"

2. Initialisation of variables (with LET commands) should be grouped together at the end of the initialisation after all the variables have been declared.

3. Checks should be ENABLEd before returning to interactive mode. There is no advantage in having checks disabled while not declaring macros and variables.

Format: ENABLE CHECKS

* **ENABLE CLEAR**
Clear the screen on initial draw (default).

Format: ENABLE CLEAR

* **ENABLE COMPOSITE**
An appropriate licence is required to use this command.
Allow use of composite text. Composite text is text which may contain more than one text string; each text string has its own locating point, orientation, text status etc, which can be edited individually, but the whole feature can also be edited as an entity.

Format: ENABLE COMPOSITE

Valid in state INITIAL

* **ENABLE CONTINUE**
When an error occurs, continue execution with the next command.
If disabled (default) then if an error occurs during expansion of a macro command or command file, the macro or file is abandoned and control returns to interactive.

Format: ENABLE CONTINUE

* **ENABLE DATE**
Update ACs holding "date of last edit" information if the feature has been edited. If no AC of the required type is present then one is inserted.
The optional argument specifies the AC type which has to be updated. (current default 110). This option is only valid if FLAGS is enabled.

Format: ENABLE DATE [integer]

* **ENABLE DESCRIPTOR**
The origin (and scale, when EXTERNAL is enabled) of maps are to be taken from the map descriptor (MD) in the IFF file, rather than from the map header (MH) (default).
If there is no type 2 map descriptor, or if the entries in it are zero

then the values from the map header are used if they seem to be set.

Format: ENABLE DESCRIPTOR

* **ENABLE DIAGNOSTICS**
Program development diagnostics to be output.

Format: ENABLE DIAGNOSTICS

* **ENABLE DSR**
See the description of the command ENABLE SD (stereo digitiser).

Format: ENABLE DSR

* **ENABLE ECHO**
Type out all commands as they are executed. Useful to trace the progress of command files or macros.

Format: ENABLE ECHO

* **ENABLE ENCLOSING**
Draw centre of fill areas, even when their borders do not impinge on the screen.

Default action is to only draw areas whose boundaries impinge on the screen.

Format: ENABLE ENCLOSING

Valid in state INITIAL

* **ENABLE ENDS**
Find only on ends of line features.

Format: ENABLE ENDS

* **ENABLE EXIT**
Exit from program after an EXIT, DUMP, or QUIT command (default).
If EXIT is disabled, then LITES2 will return to INITIAL state after these commands, in preparation for reading in new map(s). Note that QUIT in INITIAL state will always terminate the program.

Format: ENABLE EXIT

* **ENABLE EXTERNAL**
Use map scale in IFF header (default).
See SCALE command for further information.

Format: ENABLE EXTERNAL

Valid in state INITIAL

* **ENABLE FILL**
Draw area features (graphical type 12) filled according to the FRT (default). If disabled, then all area features are drawn hollow (just outlined).

Format: ENABLE FILL

* **ENABLE FIXED**
Implements an enhanced squaring algorithm for SQUARE PART and SQUARE WHOLE. When this option is enabled (default) the following features are included in the squaring algorithm:

  o Points specified with the PRIVILEGE POINT command are held fixed

  o When base squaring, after all the bases have been used as data, the remaining unsquared lines are part squared.

  o Redundant points are removed from parallel lines

* **ENABLE FLAGS**
Flag edited and constructed features during editing.
Preserve edit and deleted flags on read in.
Preserve edited and deleted flags on output if SELECT OUTPUT deselected
Activate selection by flags. (SELECT EDITED,DELETED,UNEDITED)
For detailed description of the effect of this option consult the chapter on "Flagging of Edited Features".

Format: ENABLE FLAGS

* **ENABLE GRAPHICS**
Use graphic and other interactive devices.
If graphics are disabled, then only the alphanumeric terminal is used. In order to use interactive devices with no display, then ENABLE GRAPHICS but DISABLE PRIMARY/SECONDARY displays.

Format: ENABLE GRAPHICS

Valid in state INITIAL

* **ENABLE HEIGHT**
Use text height from TH entry in IFF file, not from FRT.

Format: ENABLE HEIGHT

Valid in state INITIAL

* **ENABLE HWTEXT**
Use hardware text facilities on display. This option is only effective on certain types of display.

The SIG6000 display will be loaded with the character shapes from the TRI file, so that they are drawn much faster subsequently.

The MOTIF version will use either Display PostScript, or X-Windows itself draw draw text if the FRT includes a hardware bit in the flags entry for a text feature code. (See Workstation Guide.)

Attempts to use hardware text on a device which does not support it may result in text not appearing at all.

Format: ENABLE HWTEXT

* **ENABLE IFFLOCK**
Lock IFF files to ensure that the same file is not edited by more than one LITES2 user simultaneously (default).
Checks that the IFF file specified by an IFF or INSITU (not READONLY) command is not already being edited by another LITES2 user. An error message gives some details of the user accessing the file. No checking is performed if the option is disabled.
The checking works across all the nodes in a VAXcluster for users in the same group, but not for files accessed using DECnet (with node:: at the start of the file name).

Format: ENABLE IFFLOCK

* **ENABLE IFFMAP**
Access workspace IFF files by mapping them into computer memory.

The setting of the option at the time when each IFF file is specified (using IFF, INSITU, or READONLY commands) is taken to apply to that particular file. This option can give faster access, but needs a certain amount of care in use. It cannot be used to access workspace files over DECNET, and is not advisable when more maps than the number specified by MAPS OPEN (3 by default) are in use simultaneously. Extending the file as a result of editing may lead to fragmentation of the program address space, as might a series of returns to INITIAL state followed by reading in of more maps. The ideal use is for a READONLY background map. If this option is used inappropriately, a "Virtual address space full" error may eventually occur, which means that the number of virtual pages (as displayed by $SHOW PROCESS/CONTINUOUS) has exceeded the SYSGEN parameter VIRTUALPAGECNT.

Format: ENABLE IFFMAP

* **ENABLE INFORM**
Allow output of INFORM and LEARNER messages.
Note: when disabled, commands like SHOW and EXAMINE will produce no output.

Format: ENABLE INFORM

* **ENABLE INTERPOLATION**
Draw time interpolation of curves (default).
Disable to speed up drawing of curved features (eg contour maps).

Format: ENABLE INTERPOLATION

* **ENABLE KRISS**
See the description of the command ENABLE SI (superimposition).

Format: ENABLE KRISS [subcommand]

* **ENABLE LEARNER**
Extra messages to be output to aid a learner.

Format: ENABLE LEARNER

* **ENABLE MESSAGE**
Allow output from MESSAGE commands even when INFORM is disabled. This allows LITES2 informational messages to be suppressed while still displaying messages from macros and command files. Disabled by default.

Format: ENABLE MESSAGE

* **ENABLE MONITOR**
Use table monitor to obtain data from digitising table

Format: ENABLE MONITOR

Valid in state INITIAL

* **ENABLE NARROW**
If disabled (default), then the window defined using the WINDOW command is adjusted to fill the screen.
If disabled, then only features within the defined window are drawn, possibly leaving a blank area of screen in which it is nevertheless possible to find features. This facility may be used to speed up redrawing if the exact area required is known.

Format: ENABLE NARROW

* **ENABLE NOW**
"Now in XXX state" message is output at every change of state

Format: ENABLE NOW

* **ENABLE PATTERN**
Drawing of line patterns, including pattern filled areas.
If disabled then all lines are drawn solid.

Format: ENABLE PATTERN

* **ENABLE POSITIONING**
Text positioning by 9 possible locating points (default).
If disabled then all texts are positioned by bottom left corner.

Format: ENABLE POSITIONING

* **ENABLE PRIMARY**
Primary display to be used (default).

Format: ENABLE PRIMARY

Valid in state INITIAL

* **ENABLE PSIZE**
Height of text is in points, not mm (default).

Format: ENABLE PSIZE

Valid in state INITIAL

* **ENABLE QUIET**
Suppress acknowledgement of button presses with a bell (default).

Format: ENABLE QUIET

* **ENABLE SAME_REVISION**
Create output files with the same output revision level as the corresponding input files.

When this option is disabled (the default state) output files are created with the output revision level set by the logical name LSL$IFF_OUTPUT_REVISION. When files are merged on output, the type of file output always depends on the setting of this logical name.

Format: ENABLE SAME_REVISION

* **ENABLE SCREENMENU**
Use screen menu (when using suitable hardware). The screen menu can be ENABLED/DISABLED as required. If the definition of the menu is changed by DESCRIBE SCREENMENU, then ENABLE SCREENMENU will draw the new menu.
The screen menu must previously have been defined using the PUCK command on line 1 (the screen) (qv) and must also have been described using the DESCRIBE SCREENMENU command (qv).
The boxes on the screen menu can be defined using a series of DESCRIBE MACRO commands (qv).

Format: ENABLE SCREENMENU

* **ENABLE SCRUB**
Deleted (but recoverable) features are to be 'scrubbed out' if the display allows this (default). If disabled, deleted features remain displayed until a re-draw is performed. At present, this option applies only to the TEK4014 display.

Format: ENABLE SCRUB

* **ENABLE SD**
Allow 3D input from a stereo digitising instrument. The stereo digitising instrument is to be used to control the cursor. The instrument is initialised and driven to the LITES2 cursor position. If the ENABLE SD command is given while in INITIAL state, then the instrument will be initialised when the map(s) are read in. While active, the floating mark will follow the LITES2 cursor position, and pressing any button on the instrument will move the LITES2 cursor to the position of the floating mark. SD may be enabled or disabled at any time.

This option is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware.

Use of a stereo digitiser depends on a shared image pointed at by the logical name LSL$LITES2_KERN_ROUTINES. This image is supplied by Laser-Scan. The name of this image depends on what type of instrument is to be used, and whether it is used with an image superimposition system.

Format: ENABLE SD

* **ENABLE SECONDARY**
  Secondary display to be used.

  Format: ENABLE SECONDARY

  Valid in state INITIAL

* **ENABLE SI**
  A stereo superimposition device is to be used to display data in a stereo digitising instrument. This command is only valid in conjunction with the ENABLE SD command. The superimposition device is initialised and any subsequent LITES2 graphics will be drawn on it. If the ENABLE SI command is given while in INITIAL state, then the device will be initialised when the map(s) are read in. If the device is already initialised, then any ENABLE/DISABLE SI commands will just make the superimposition image visible/invisible.

  The subcommands (if given) control further SI options.

  This option is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware.

  Use of a stereo digitiser and superimposition device depends on a shared image pointed at by the logical name LSL$LITES2_KERN_ROUTINES. This image is supplied by Laser-Scan. The name of this image depends on what type of instrument is to be used, and whether it is used with an image superimposition system.

  Format: ENABLE SI [subcommand]

  o **ENABLE SI DIALOG**
    Specifies an area on the superimposition device to be used as a one line dialogue area. Messages may be written to this area using the ENABLE SI MESSAGE command. The arguments specify the position of the bottom left of the area, and also its height and length, all in pixels in the range 0-1023. The maximum allowed height is 100. The default dialogue area is at position 0,61 with height 20 and length 1023.

    Format: ENABLE SI DIALOG xpos ypos height length

  o **ENABLE SI MESSAGE**
    Displays the given text in the superimposition dialogue area. If no text is given, the area is cleared. If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

    Format: ENABLE SI MESSAGE [text]

- o **ENABLE SI REGISTRATION**
  Performs a manual registration of the superimposition image with the photographs. The command is only used with certain devices. On a KERN DSR instrument, the user is prompted to move the photographs until registration is achieved, and then press the right DSR button to accept, or the left button to leave the registration unchanged. Once set, the registration will be maintained for the duration of the LITES2 session. If the argument n is given as zero, then the effect of any manual registration is removed.

  Format: ENABLE SI REGISTRATION [n]

- o **ENABLE SI SIDE**
  Specifies which eye is to see the superimposition screen menu, status area, and dialogue area. The argument should be 1 for left (default), 2 for right, 3 for both, or 0 to make the menu invisible.

  Format: ENABLE SI SIDE side

- o **ENABLE SI STATUS**
  Specifies an area on the superimposition device to be used as a status area. When enabled, the area will continuously display the current state, plus the currently set map, layer, and feature code. The optional arguments specify the position of the bottom left of the area, and also its height, all in pixels in the range 0-1023. The maximum allowed height is 100. If the arguments are omitted, the command just enables/disables the status area without changing its position. The default status area is at position 0,30 with height 20.

  Format: ENABLE SI STATUS [xpos ypos height]

* **ENABLE SEGMENTS**
  Use segments on displays for which this is possible (default).
  The use of segments enables fast redraw of the picture, but may be disabled if the map is too large for the display's segment store. If ENABLE SEGMENTS is given while in READY state, then the next re-draw will clear the segment store and re-load it, possibly with only part of the data if selections have been made or the display is zoomed in. Any features not loaded will not become visible until another ENABLE SEGMENTS is given followed by a re-draw.

  Format: ENABLE SEGMENTS

* **ENABLE SORT**
  Re-draw features sorted in IFF or FSN order. Sorting is not usually relevant on a display which supports ENABLE SEGMENTS. In this case it should remain disabled.
  Use the SORT command (qv) to specify type of sorting.

  Format: ENABLE SORT

* **ENABLE STATUS**
  Inverse video status lines to be written on VDU screen.
  This option requires a DEC VT100 VDU or equivalent, and reduces the available scroll area. The position of the status line may also be specified by including an optional argument. The default position is the bottom two lines of the screen.
  DISABLE STATUS will increase the scroll area which has been set.

  Format: ENABLE STATUS [integer]

* **ENABLE SUBSTITUTION**
  Substitute the values of any variables enclosed in single quotation marks into command lines. If disabled (default), then single quote is treated as a normal character and may be inserted into texts, for instance. Note that variables are never substituted while defining a macro.

  Format: ENABLE SUBSTITUTION

* **ENABLE TABLE**
  Digitising table to be used (default).

  Format: ENABLE TABLE

  Valid in state INITIAL

* **ENABLE THICK**
  Draw using line thickness if hardware permits.
  If disabled then all segments are drawn minimum thickness.

  Format: ENABLE THICK

* **ENABLE TRACEBACK**
  Traceback of routine calls is output for system errors (diagnostic).
  If disabled (default) then just error message is output.

  Format: ENABLE TRACEBACK

* **ENABLE VECTOR**
  Draw vectors as well as raster.
  This command only has any effect when using raster images. By DISABLing VECTOR the raster images can be redrawn without the vector data being displayed.

  Format: ENABLE VECTOR

* **ENABLE VERIFY**
  Allows verification of found features (default).
  The style of verification is set by the VERIFY command (qv).

  Format: ENABLE VERIFY

* **ENABLE Z**
  Allows LITES2 to be used as a 3 dimensional editor.
  An appropriate licence is required to use this command.

See the chapter about using LITES2 as a 3 dimensional editor for more details on the effect of this command.

Format: ENABLE Z

Valid in states INITIAL READY

7.2.2 **DISABLE**

Deactivates specified optional facilities.
See ENABLE command.

Format: DISABLE subcommand

Takes same subcommands as ENABLE (qv)

7.2.3 **TOGGLE**

If the selected facility is currently activated, then TOGGLE deactivates it, and if it is currently deactivated, TOGGLE activates it. This is most useful when programmed as a menu box or puck button.

Format: TOGGLE subcommand

Takes same subcommands as ENABLE (qv)

7.2.4 **INTERPOLATE**

Sets drawing & construction interpolation method.

Format: INTERPOLATE subcommand

Valid in states INITIAL READY

* **INTERPOLATE AKIMA**
  Akima curve interpolation (default).
  Akima is a bicubic spline method which preserves linearity if possible.

  Format: INTERPOLATION AKIMA

* **INTERPOLATE MCCONALOGUE**
  McConalogue curve interpolation.
  McConalogue is a circular arc pair method.

  Format: INTERPOLATION MCCONALOGUE

7.2.5 **SORT**

Sets type of sorting used for re-drawing (if ENABLE SORT is set).

Format: SORT subcommand

Valid in all states

* **SORT FSN**
  Re-draw sorted in order of increasing Feature Serial Number. This may be used to ensure that features such as filled areas are drawn first.

  Format: SORT FSN

* **SORT GT**
  Re-draw sorted by graphical type of features.
  Features are drawn in the following order:

  1. Fill areas (graphical type 12)

  2. Line work (graphical types 1 - 6)

  3. Symbols (graphical types 7 - 9 and 11)

  4. Texts (graphical type 10)

  Within each of these groups, features are drawn in order of their feature code.

  Format: SORT GT

* **SORT IFF**
  Re-draw sorted in IFF file order. This will often give a considerable speed increase for the re-draw (default).

  Format: SORT IFF

* **SORT PRIORITY**
  Re-draw features according to the priority defined for feature codes in the PRIORITY entries in the FRT table that is being used. If a feature code's priority is not defined in a PRIORITY record, then it will be drawn at the default priority.

  For features with the same priority, features are drawn in order of their feature code.

  The use of this sorting allows features to be drawn in a predictable order, so that for example, one type of road will always lie over another one. It also allows features to be drawn several times with different representations, allowing complex line types to be constructed, for example cased roads drawn by drawing a thick line in the casing colour, followed by a thinner line in the fill colour.

  Note that the components of a single feature may be drawn at different times, so that road junctions, for example, will be correctly cased.

When drawing in this mode, colour 0 in the FRT table will be drawn in background colour, allowing parts of features to blank out what is below them.

When drawing texts in this mode, ENABLE BLANK will cause all but the text representation with the highest priority to be invisible.

Format: SORT PRIORITY

7.2.6 **REFRESH**

Changes the characteristics of the refresh (or highlighted) picture.

Format: REFRESH subcommand

Valid in states INITIAL READY LINE CIRCLE EDIT ON CONSTRUCT

* **REFRESH BITS**
  Sets the bit planes used for refresh (if the display allows it). This command may be used when the allocation of colours to overlays makes the cursor and other highlighting difficult or impossible to see. Refresh works by inverting the bits in the display, then inverting them back again to turn it off. By default all bit planes are inverted. The integer given in this command is a bit set mask specifying the particular planes to be inverted. If the argument is omitted, then the default is used. In the MOTIF version of LITES2, the default may be set by defining logical name LSL$DECW_REFRESH_BITS.

  For example: If you had an 8 plane overlay using colours 0 to 255, then by default the cursor would display in colour 255 when over an area of the screen with colour 0 (background). If you used the command REFRESH BITS 1, then the cursor would display in colour 1 instead.

  Format: REFRESH BITS [integer]

* **REFRESH CURSOR**
  Sets the shape of the cursor (if the display allows it). This command allows the cursor to be set to the same shape as the 'brush' used by the IMAGE PAINT and IMAGE ERASE commands.

  Format: REFRESH CURSOR subcommand

  o **REFRESH CURSOR CIRCLE**
    Use a circle or ellipse as the cursor. The first argument gives the diameter of the circle. If it is positive, then it is measured in IFF units (or mm if a UNITS command has been given), and the cursor changes size if the picture is zoomed. If it is negative, then it is measured in screen mm, and the cursor will be of constant size. If the second argument is given, it specifies a height, allowing an ellipse to be used.

    Format: REFRESH CURSOR CIRCLE width [height]

- o **REFRESH CURSOR CROSS**
  Use a simple cross as the cursor. The first argument gives the size. If it is positive, then it is measured in IFF units (or mm if a UNITS command has been given), and the cursor changes size if the picture is zoomed. If it is negative, then it is measured in screen mm, and the cursor will be of constant size. If the second argument is given, then a second cross with this size if drawn on top of the first, cancelling out where the two overlap. This is used to achieve a cross with the centre missing.

  Format: REFRESH CURSOR CROSS size [size2]

- o **REFRESH CURSOR DEFAULT**
  Restores the cursor to its default appearance.

  Format: REFRESH CURSOR DEFAULT

- o **REFRESH CURSOR RECTANGLE**
  Use a rectangle or square as the cursor. The first argument gives the width. If it is positive, then it is measured in IFF units (or mm if a UNITS command has been given), and the cursor changes size if the picture is zoomed. If it is negative, then it is measured in screen mm, and the cursor will be of constant size. The second argument specifies a height, allowing a rectangle to be used. If omitted, the result is a square.

  Format: REFRESH CURSOR RECTANGLE width [height]

* **REFRESH LINE**
  Sets maximum refresh line length (not yet implemented).

  Format: REFRESH LINE integer

* **REFRESH POINTS**
  Sets maximum number of refreshed points (range 2 to 1000, default 50).

  When refreshing composite text each component takes 5 vertices. At least one component is refreshed.

  Format: REFRESH POINTS integer

7.2.7 **SCROLL**

Sets the size and position of the scroll area on the terminal.
SCROLL 0 0 will use the maximum amount of screen area depending on the position of the status line.

Format: SCROLL numlines startline

where numlines is the number of lines in the scroll area (integer)
      startline is the first line of the scrolling area (integer)

Valid in all states

7.2.8 TOLERANCE

Specifies various tolerances.
Distances are specified in sheet mm, unless otherwise stated. This behaviour may be overridden if the TOLERANCE command is preceded by a UNITS command.

Format: TOLERANCE subcommand [argument]

Valid in states INITIAL READY

* **TOLERANCE BUNCH**
  Sets the tolerances to be used in filtering, using the BUNCH algorithm.

  The three coefficients (a,b,c) control the spacing of filtered points. They represent:

  a the minimum separation between successive master points

  b the lateral threshold distance from the trend line

  c the maximum separation between successive master points. A maximum separation of 0.0 is equivalent to one of infinity.

  Note: c must be greater than or equal to a, which must be greater than or equal to b. If this condition is not met, then the default values are reset.

  Trailing arguments may be omitted (the setting is unchanged).

  Format: TOLERANCE BUNCH a b c

* **TOLERANCE CIRDRAW**
  Sets the point density for drawn circles (graphical types 2-5). The three coefficients (a,b,c) control the spacing of interpolated points. The approximate separation of points (d) is given by:

  d = a + 2*SQRT(2br) + cr (r is radius)

  which means (if other coefficients were zero) that

  a gives a constant separation of a mm

  b gives a constant 'arc to chord' distance of b mm

  c gives a constant angular deviation of c radians (2*PI/c points in a circle)

  Trailing arguments may be omitted (the setting is unchanged).

  The default setting is a=0, b=0.05, c=0, which gives an 'arc to chord' distance of 0.05mm, or approximately 30 points in a circle of radius 10mm, with the number proportional to the square root of the radius.

Format: TOLERANCE CIRDRAW a b c

* **TOLERANCE CIRGEN**
Sets the point density for generated circles (CIRCLE and ARC commands). The three coefficients (a,b,c) control the spacing of interpolated points as for the TOLERANCE CIRDRAW command.

The default setting is a=0, b=0.05, c=0, which gives an 'arc to chord' distance of 0.05mm, or approximately 30 points in a circle of radius 10mm, with the number proportional to the square root of the radius.

Format: TOLERANCE CIRGEN a b c

* **TOLERANCE CURDRAW**
Sets the point density for drawn curves (graphical type 6). The three coefficients (a,b,c) control the spacing of interpolated points as for the TOLERANCE CIRDRAW command, with r being an approximation to the radius of curvature for each span of the curve.

The default setting is a=0.25, b=c=0, which gives 4 points per mm.

Format: TOLERANCE CURDRAW a b c

* **TOLERANCE CURGEN**
Sets the point density for generated curves (CURVE command). The three coefficients (a,b,c) control the spacing of interpolated points as for the TOLERANCE CIRDRAW command, with r being an approximation to the radius of curvature for each span of the curve.

The default setting is a=0.25, b=c=0, which gives 4 points per mm.

Format: TOLERANCE CURGEN a b c

* **TOLERANCE DEGREES**
Angle squaring tolerance in degrees (for SQUARE ANGLES).

Format: TOLERANCE DEGREES real

* **TOLERANCE EDGE**
Edge matching tolerance in mm on the nominal sheet.

Format: TOLERANCE EDGE real

* **TOLERANCE EXPAND**
Sets the proportion of the character height that regions defined by texts will be expanded beyond the limits of the text itself. If the expansion is greater than the average character width, oddities in the region boundary may occur.
By default the value of 0.1 is used.

Format: TOLERANCE EXPAND real

* **TOLERANCE FAR_MOVE**
Sets the distance that is to be used as the criterion for the OPERATION FAR_MOVE_POINT command (qv).

Format: TOLERANCE FAR_MOVE real

* **TOLERANCE FIND**
Sets the find radius to be used in the FIND command.
The value is in screen mm. If there is no display, then the screen size is assumed to be 360 mm. It is not possible to set a find radius of more than half the screen width.
When the find radius is specified in screen mm, then it gets larger and smaller as the picture is zoomed in and out. To specify a fixed find radius in IFF units, sheet mm, or other UNITS, then precede this command with a UNITS command. The find radius will then be fixed until it is again specified in screen mm.

Format: TOLERANCE FIND real

* **TOLERANCE FOLLOW**
Sets the tolerances to be used in following, using the BUNCH algorithm.

The four coefficients (a,b,c,d) control the spacing of points added while following.
They represent:

a the time interval at which to inquire the position from the device being followed

b the minimum separation between successive master points

c the lateral threshold distance from the trend line

d the maximum separation between successive master points.
A maximum separation of 0.0 is equivalent to one of infinity.

These arguments are specified in sheet millimetres. To set them in other units, precede this command with a UNITS command.

Note: d must be greater than or equal to b, which must be greater than or equal to c. If this condition is not met, then the default values are reset.

Trailing arguments may be omitted (the setting is unchanged).

Format: TOLERANCE FOLLOW a b c d

* **TOLERANCE JUSTIFY**
The width of text characters specified in the TRI file usually allows some blank space after each character. The space after the final character in a string must be allowed for when centre or right justification is used.

The TOLERANCE JUSTIFY command sets the amount to be subtracted from a text string to represent this space. The amount is specified as a fraction of the text height.

By default the value 0.333333 is used.

Format: TOLERANCE JUSTIFY real

* **TOLERANCE OFFSET**
  Sets the proportion of the height that texts/symbols will be offset if no argument is given to the OFFSET command. If the argument is -ve, then the text/symbol will be offset above the original.

  Format: TOLERANCE OFFSET real

* **TOLERANCE PROPAGATE**
  Distance along the features to propagate a mismatch during edgematching or when the PROPAGATE command has been given during a TIE or JOIN operation. The distance is specified in mm on the nominal sheet.
  Note: to avoid propagation when edgematching, this tolerance should be set to 0.0

  Format: TOLERANCE PROPAGATE real

* **TOLERANCE RADIANS**
  Angle squaring tolerance in radians (for SQUARE ANGLES).

  Format: TOLERANCE RADIANS real

* **TOLERANCE SQDEF**
  OS squaring tolerance - default setting used for all SQxx parameters

  Format: TOLERANCE SQDEF

* **TOLERANCE SBMT**
  OS squaring tolerance - as SQMT but for based squaring

  Format: TOLERANCE SBMT real

* **TOLERANCE SBLT**
  OS squaring tolerance - as SQMT but for based squaring

  Format: TOLERANCE SBLT real

* **TOLERANCE SQBT**
  OS squaring tolerance - length of base must be longer than this distance (mm)

  Format: TOLERANCE SQBT real

* **TOLERANCE SQCT**
  OS squaring tolerance - SQCT is used by the OS squaring algorithm to test if a feature forms a closed loop. If the distance between the first and last point is less than SQCT sheet mm, then the feature is considered to be closed.

  Format: TOLERANCE SQCT real

* **TOLERANCE SQLT**
  OS squaring tolerance - minimum length of line (mm)

  Format: TOLERANCE SQLT real

* **TOLERANCE SQMT**
  OS squaring tolerance - maximum lateral distance a point will be moved, for a line to be included in this squaring pass (mm)

  NOTE

  Points may finally be moved by more than this amount, especially when points have been removed from the feature.

  Format: TOLERANCE SQMT real

* **TOLERANCE SQPL**
  OS squaring tolerance - maximum angle (in degrees) that two lines may differ by (after adjustment) and still be considered parallel

  Format: TOLERANCE SQPL real

* **TOLERANCE SQWT**
  OS squaring tolerance - warning issued when point moved more than this distance (mm)

  Format: TOLERANCE SQWT real

7.2.9 **PTOLERANCE**

Specifies the various tolerances, that are privileged (ie can only be given in initialisation files).

Format: PTOLERANCE subcommand [argument]

Valid in state INITIAL (privileged command;
only valid in initialisation file)

* **PTOLERANCE EDGESETUP**
  Overrides the default settings for doing an EDGE setup.
  It defines the number of pointings to be made to each edge of the map, the number of complete sets of observations to be made, and the limits and tolerances for acceptance of the results. See SETUP EDGE for details of how these tolerances are used.

  Format:
  PTOLERANCE EDGESETUP numpts numsets sidetol tolmax tolsum mingap maxgap

  where

```
numpts  - number of pointings to each side (maximum 5)
numsets - number of sets that must be within tolerance (maximum 5)
sidetol - maximum RMS of points from lines they define (mm)
tolmax  - range between points at any one corner (mm)
tolsum  - sum of the ranges between points at the corners (mm)
mingap  - factor of ideal gap that points must be apart
maxgap  - factor of ideal gap that points must be closer than
```

The defaults are: 5 2 0.075 0.2 0.425 0.2 2.0

* **PTOLERANCE OSSETUP**
  Overrides the default settings for doing a multiple point setup. It defines the number of boxes the map is to be split into along with the number of observations required at each point, the number of observations at each point that must fall within the tolerance of the mean, and that tolerance.

  Format: PTOLERANCE OSSETUP fulx fuly repeat numok tolerance

  ```
  where fulx         is the number of boxes in the X direction (integer)
        fuly         is the number of boxes in the Y direction (integer)
        repeat       number of observations of each point      (integer)
        numok        number that must be within tolerance      (integer)
        tolerance    tolerance in table mm                     (real)
  ```

  The defaults are: 5 5 4 2 0.381

* **PTOLERANCE RESIDUAL**
  Specifies the tolerances for residuals (in x and y) after a least squares solution to a set up. These numbers are in terms of the range in x and y of the distances between the control points.

  The first argument is the limit at which a setup will be accepted, the second is the level at which a warning will be output

  Format: PTOLERANCE RESIDUAL limit warn

  The defaults are: 0.0025 0.00025

7.2.10 **VERIFY**

Sets the style of verification of found features. Whether features are verified or not is controlled by the VERIFY option, set by ENABLE VERIFY, DISABLE VERIFY or TOGGLE VERIFY (qv)

Format: VERIFY subcommand

Valid in all states

* **VERIFY AC**
  Verify any AC, TC, or CH entries for each found feature.

  Format: VERIFY AC

* **VERIFY FEATURE**
  Verify FSN, Map, Layer, Feature code, and Point number for each found feature.
  If the process code (PC) is not 0, then it is also shown.

  Format: VERIFY FEATURE
  or   VERIFY

* **VERIFY GROUP**
  Shows the group(s) that the feature code of the found feature is in.
  This is only effective if VERIFY FEATURE is also set.

  Format: VERIFY GROUP

* **VERIFY OFF**
  Disable all verification of found features.

  Format: VERIFY OFF

* **VERIFY TEXT**
  Verify the text of each found text feature.

  Format: VERIFY TEXT

7.3 **Command Handling Commands**

7.3.1 **@FILE**

Take commands from the specified command file until end-of-file, or error. The default filename is LSL$LITES2CMD:---.LCM. The filename may be followed by a series of parameters, each delimited by one or more spaces or tabs, or enclosed in double quotes. Within the command file, the values of these parameters are available as system variables $P1, $P2 etc. The number of parameters is available in $PCOUNT, and the whole line of parameters in $PLINE.

Format: @filename [p1 p2...]

Valid in all states

7.3.2 **RESPOND**

Suspends a command file or macro and obtains input from the interactive controls (terminal, table, or bitpad) until a CONTINUE command (resume command file or macro) or CANCEL RESPOND command (abandon command file or macro and return to interactive) is given.

Format: RESPOND

Valid in all states

7.3.3 **CONTINUE**

Continue execution of a command file or macro which has been suspended with RESPOND (qv) after interactive input is complete.

Format: CONTINUE

Valid in all states

7.3.4 **MACRO**

Enter MACRO state to define a macro. All commands will be stored rather than obeyed until ENDMACRO (qv) Macro names must consist of up to 16 alphabetic characters (including underline), followed by a box or button number in the case of a PUCK or MENU macro.

Format: MACRO name

eg MACRO FRED

Valid in states INITIAL READY

N.B. This command is not valid in MACRO state

7.3.5 **ENDMACRO**

End the definition of a macro started with a MACRO command (qv). This is the only command actually obeyed, rather than stored, while in MACRO state.

Format: ENDMACRO

Valid in state MACRO

7.3.6 **JUMP**

Transfers control to the specified macro (which may be the one currently executing), or to a label.

In the case of a jump to a macro, any remaining commands at the present level (ie in the current command file) are lost, as are any remaining commands on the current line, or in the current macro. The macro name may be followed by a series of parameters, each delimited by one or more spaces or tabs, or enclosed in double quotes. Within the macro, the values of these parameters are available as system variables $P1, $P2 etc. The number of parameters is available in $PCOUNT, and the whole line of parameters in $PLINE.

In the case of a jump to a label (which must include its leading "."), the label must be found in the current line or macro. The case of letters in the label is not significant.

NOTE

> The command sequence THEN JUMP macro or ELSE JUMP macro has the same effect as THEN macro or ELSE macro. The commands in the macro will be executed, and then the remaining commands in the current line or macro will be executed. The commands THEN JUMP .label or ELSE JUMP .label will be unable to find the label. To achieve the desired effect use the JTRUE or JFALSE commands.

```
Format:  JUMP macro [p1 p2...]
or       JUMP .label

eg       JUMP FRED
```

Valid in all states

7.3.7 **JTRUE**

As JUMP (qv), but only transfers control if the condition flag is set to TRUE. The condition flag can be set by the TEST, OR and AND commands and also by the user routines.

```
Format:  JTRUE macro [p1 p2...]

eg       JTRUE FRED
```

Valid in all states

7.3.8 **JFALSE**

As JUMP (qv), but only transfers control if the condition flag is set to FALSE. The condition flag can be set by the TEST, OR and AND commands and also by the user routines.

Format: JFALSE macro [p1 p2...]

eg JFALSE FRED

Valid in all states

7.3.9 **ELSE**

Obeys the command-line only if the condition flag is FALSE.
The condition flag is set by the TEST, OR and AND commands (qv) and also by the user routines.

Several ELSE and THEN commands may be given in any order without re-setting the condition flag, as long as the condition flag is not altered by the macro that is called.

NOTE

> Due to the nature of the command separator #, it is not possible to put more than one command in this command line (the # would be taken as the separator between the ELSE command and the next command), however the command line can consist of a macro or a @filename directive.
>
> Note also that this command should not be followed by a JUMP, JTRUE, JFALSE, ABORT ALWAYS, ABORT TRUE, or ABORT FALSE command.

Format: ELSE command-line

```
eg       TEST $FOUND         ! (is there a found feature)
         ELSE MESSAGE "No found feature"
```

Valid in all states

7.3.10 **THEN**

Obeys the command-line only if the condition flag is TRUE.
The condition flag is set by the TEST, OR and AND commands (qv) and also by the user routines.

Several ELSE and THEN commands may be given in any order without re-setting the condition flag, as long as the condition flag is not altered by the macro that is called.

NOTE

Due to the nature of the command separator #, it is not possible to put more than one command in this command line (the # would be taken as the separator between the THEN command and the next command), however the command line can consist of a macro or a @filename directive.

Note also that this command should not be followed by a JUMP, JTRUE, JFALSE, ABORT ALWAYS, ABORT TRUE, or ABORT FALSE command.

Format: THEN command-line

```
eg       TEST $LAYER>3 # AND $FC=4 ! layer above 3, and feature code 4
         THEN DELETE
```

Valid in all states

7.3.11 **ABORT**

Causes the current input stream to be aborted. There are several levels of severity of ABORT

Format: ABORT subcommand

Valid in all states

* **ABORT ALWAYS**
  Abort the current input line or macro (default). This has the effect of ignoring the rest of the commands on the line or in the macro. It is the equivalent of JUMP to an empty macro.

  NOTE

  Note that this command should not be used as the argument to a THEN or ELSE command.

  Format: ABORT ALWAYS
  or ABORT

* **ABORT FALSE**
  Abort the current input line or macro, if the condition flag is FALSE. This has the effect of ignoring the rest of the commands on the line or in the macro. It is the equivalent of JFALSE to an empty macro.

NOTE

This command should not be used as the argument to a THEN or ELSE command.

Format: ABORT FALSE

* **ABORT FILE**
Abort the current input line or macro and all the rest of the lines in the current command file.

Format: ABORT FILE

* **ABORT INPUT**
Aborts all current command input and returns to first level interactive input. This means that all commands on command lines or macros are ignored as are all unread lines in any command files. These command files are closed. The effect of any RESPOND command is cancelled.

This command is the equivalent to entering CTRL/C while reading commands. After this command has aborted all current input, any command line specified by the AFTER ABORT command is executed.

Format: ABORT INPUT

* **ABORT RESPOND**
Cancels a previous RESPOND command.

Returns input from second level interactive input (ie after a RESPOND command has been given, and awaiting a CONTINUE command), to first level interactive input.

Format: ABORT RESPOND

* **ABORT TRUE**
Abort the current input line or macro, if the condition flag is TRUE. This has the effect of ignoring the rest of the commands on the line or in the macro. It is the equivalent of JTRUE to an empty macro.

NOTE

This command should not be used as the argument to a THEN or ELSE command.

Format: ABORT TRUE

7.3.12 **TEST**

Sets the condition flag depending on the result of the test.
The condition flag is used by the JTRUE, JFALSE, THEN, and ELSE commands.
The variable may be any system or user-declared variable.

```
The inequality may be any of:     =     >    >=     <    <=    <>
               with synonyms:   .EQL. .GTR. .GEQ. .LSS. .LEQ. .NEQ.
                         and:                       .LT.
```

Inequality names may be abbreviated.
If the inequality is absent, but expression is present, then = is assumed.
If both inequality and expression are absent, then the variable is tested as a logical.
For CHARACTER variables, the result of a comparison is determined according to the ASCII collating sequence, assuming that the shorter string is padded with spaces to the length of the longer.

Format: TEST variable [[inequality] expression]

```
eg       TEST NAME=Fred         True if CHARACTER variable NAME is "Fred"
         TEST $FOUND            True if there is a found feature
         TEST R>3.14            True if REAL variable R is greater than 3.14
```

Valid in all states

7.3.13 **OR**

ORs the existing value of the condition flag with the result of the test.
If the condition flag is already TRUE, the test is not performed.
Syntax is exactly as for TEST.

Format: OR variable [[inequality] expression]

Valid in all states

7.3.14 **AND**

ANDs the existing value of the condition flag with the result of the test.
If the condition flag is already FALSE, the test is not performed, so a sequence such as TEST $FOUND # AND $FC=3 will not test $FC (which would cause an error) if $FOUND was FALSE.
Syntax is exactly as for TEST.

Format: AND variable [[inequality] expression]

Valid in all states

7.3.15 **CANCEL**

Allows macros, regions, variables, RESPOND input and lists of operations to be cancelled.

Format: CANCEL subcommand

Valid in states INITIAL READY

* CANCEL ADD_FEATURE

  Cancels the list of ACs set up by the OPERATION ADD_FEATURE command.

  Format: CANCEL ADD_FEATURE

* CANCEL ANGLESQ_POINT

  Cancels the list of point attributes set up by the OPERATION ANGLESQ_POINT command.

  Format: CANCEL ANGLESQ_POINT

* CANCEL BREAK_POINT

  Cancels the list of point attributes set up by the OPERATION BREAK_POINT command.

  Format: CANCEL BREAK_POINT

* CANCEL CIRCLE_POINT

  Cancels the list of point attributes set up by the OPERATION CIRCLE_POINT command.

  Format: CANCEL CIRCLE_POINT

* CANCEL CODE_CH_FEATURE

  Cancels the list of ACs set up by the OPERATION CODE_CH_FEATURE command.

  Format: CANCEL CODE_CH_FEATURE

* CANCEL CURVE_POINT

  Cancels the list of point attributes set up by the OPERATION CURVE_POINT command.

  Format: CANCEL CURVE_POINT

* CANCEL DIGITISE_POINT

  Cancels the list of point attributes set up by the OPERATION DIGITISE_POINT command.

  Format: CANCEL DIGITISE_POINT

* CANCEL FAR_MOVE_POINT

  Cancels the list of point attributes set up by the OPERATION FAR_MOVE_POINT command.

  Format: CANCEL FAR_MOVE_POINT

* CANCEL FILTER_POINT

  Cancels the list of point attributes set up by the OPERATION FILTER_POINT command.

  Format: CANCEL FILTER_POINT

* CANCEL GEO_CH_FEATURE

  Cancels the list of ACs set up by the OPERATION GEO_CH_FEATURE command.

  Format: CANCEL GEO_CH_FEATURE

* CANCEL JOIN_POINT

  Cancels the list of point attributes set up by the OPERATION JOIN_POINT command.

  Format: CANCEL JOIN_POINT

* **CANCEL MACRO**
  Deletes the specified macro. The storage is freed and becomes available for re-use. A macro must be deleted using this command before it can be redefined.

  Format: CANCEL MACRO name

  eg   CANCEL MACRO FRED
  or   CANCEL MACRO CMDMEN23 (in the case of a menu box)

* CANCEL MOVE_POINT

  Cancels the list of point attributes set up by the OPERATION MOVE_POINT command.

  Format: CANCEL MOVE_POINT

* CANCEL OFFSET_POINT

  Cancels the list of point attributes set up by the OPERATION OFFSET_POINT command.

  Format: CANCEL OFFSET_POINT

* CANCEL OS_MH_FLAGS

  Cancels the group of feature codes and the corresponding flag character associated with the specified flag number by the OPERATION OS_MH_FLAGS command.

  Format: CANCEL OS_MH_FLAGS flagno

* CANCEL OS_MH_TEXTCAT

  Cancels all the text categories that are currently set to modify the flag setting action of texts for the specified flag.

Format: CANCEL OS_MH_TEXTCAT flagno

* CANCEL OTHER_POINT

Cancels the list of point attributes set up by the OPERATION OTHER_POINT command.

Format: CANCEL OTHER_POINT

* CANCEL PARTSQ_POINT

Cancels the list of point attributes set up by the OPERATION PARTSQ_POINT command.

Format: CANCEL PARTSQ_POINT

* **CANCEL REGION**

Removes the specified region from the currently defined list of regions. This frees the storage, and allows the region to be redefined.

Format: CANCEL REGION integer

eg CANCEL REGION 3

* **CANCEL RESPOND**

Returns input from second level interactive input (ie after a RESPOND command has been given, and awaiting a CONTINUE command), to first level interactive input.

This command is a synonym for ABORT RESPOND.

Format: CANCEL RESPOND

* CANCEL SQUARE_POINT

Cancels the list of point attributes set up by the OPERATION SQUARE_POINT command.

Format: CANCEL SQUARE_POINT

* CANCEL TRANSFORM_POINT

Cancels the list of point attributes set up by the OPERATION TRANSFORM_POINT command.

Format: CANCEL TRANSFORM_POINT

* CANCEL USER_FEATURE

Cancels the list of point attributes set up by the OPERATION USER_FEATURE command.

Format: CANCEL USER_FEATURE

* CANCEL USER_POINT
  Cancels the list of point attributes set up by the OPERATION USER_POINT command.

  Format: CANCEL USER_POINT

* **CANCEL VARIABLE**
  Deletes the specified variable. The storage is freed and becomes available for re-use. The variable may be declared again if required, possibly with a different type. In the case of an array variable, the whole array is cancelled - no subscript may be given.

  Format: CANCEL VARIABLE name

  eg CANCEL VARIABLE X

7.3.16 **DECLARE**

Declares a variable, which may be set by the LET or INQUIRE commands, tested by the TEST, AND, and OR commands, or substituted into commands by enclosing its name in single quotation marks. Variable names must consist of up to 16 alphabetic characters (including underline). An array variable may be declared by following the name by an integer in the range 1 to 65535, thus the command DECLARE INTEGER I20 would allow the use of integer variables I1, I2, ... , I20. Caution should be exercised in declaring large arrays, as the computer memory may be insufficient.

Format: DECLARE subcommand

Valid in all states

* **DECLARE CHARACTER**
  Declare a character variable, or array of character variables, and initialise to null (zero characters). The variable may used to contain a string characters.

  Format: DECLARE CHARACTER name[n]

  eg DECLARE CHARACTER NAME
  or DECLARE CHARACTER NAME3

* **DECLARE DOUBLE**
  Declare a double precision real variable or array and initialise to 0.0. The variable may contain real values with absolute value in the range 0.29E-38 to 1.7E38, with a precision of 15 decimal digits.

  Format: DECLARE DOUBLE name[n]

  eg DECLARE DOUBLE D
  or DECLARE DOUBLE D20

* **DECLARE INTEGER**
  Declare an integer variable or array and initialise to 0. The variable may contain integers in the range -2147483648 to 2147483647.

  Format: DECLARE INTEGER name[n]

  ```
  eg       DECLARE INTEGER I
  or       DECLARE INTEGER I2048
  ```

* **DECLARE REAL**
  Declare a real variable or array and initialise to 0.0. The variable may contain real values with absolute value in the range 0.29E-38 to 1.7E38, with a precision of 7 to 8 decimal digits.

  Format: DECLARE REAL name[n]

  ```
  eg       DECLARE REAL R
  or       DECLARE REAL R24
  ```

7.3.17 **LET**

Allows the value of a variable to be set. The variable must have been declared using a DECLARE command (qv). System variables may not be set. See the section on the LITES2 command language for further information on expressions.

Format: LET variable [=] expression

```
eg       LET NAME=Building      Sets CHARACTER variable NAME to "Building"
         LET NAME               Sets CHARACTER variable NAME to the null string
         LET I=3                Sets INTEGER variable I to 3
         LET I=(3+4)/2          Sets INTEGER variable I to 3
         LET R=(3+4)/2          Sets REAL variable R to 3.5
         LET RR23=3.14          Sets the 23rd element of REAL array RR
                                to 3.14
```

Valid in all states

7.3.18 **INQUIRE**

Obtains the value for a variable from the interactive controls (keyboard, or pucks, or menus). INQUIRE is most useful in command files or macros. The prompt string is displayed at the terminal. If prompt is omitted, a default of "Enter TYPE VARIABLE: " e.g. "Enter Real R: " is used.
If the prompt is to have leading spaces or tabs, then it must be enclosed in double quotes.
The string entered by the user may be any expression which would have been valid in a LET command for the variable. It is quite possible to respond with a button press or menu probe - the puck or menu macro name will be returned. The cursor may be tracked before the response is entered. If a blank line is entered, a character variable is set to the null string, while for other types the user is reprompted for a valid answer. Pressing CTRL/Z will leave the variable

unchanged.

Format: INQUIRE variable [prompt]

eg   INQUIRE NAME "What is your name? "

Valid in all states

7.3.19 **HELP**

Gives information about LITES2 commands and other subjects. If no subject given then a list of subjects will be printed. If HELP is entered interactively (i.e. not from a command file or macro), the user is prompted for further topics. The possible responses are the same as in the DCL HELP utility. CTRL/Z will exit from help and return to the LITES2 prompt (as will a sufficient number of blank lines).

Format: HELP [text]

eg   HELP
or   HELP FIND

Valid in all states

7.3.20 **MESSAGE**

Outputs the given string to the terminal, or a blank line if none given. Useful in command files to let the operator know what is going on. If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

Format: MESSAGE [text]

Valid in all states

7.3.21 **PING**

Sounds a bell at the terminal to attract the operator's attention.

Format: PING

Valid in all states

7.3.22 **RASPBERRY**

Sounds a distinctive noise ("raspberry") at the terminal. The is usually two bells unless the hardware permits anything else.
Useful to signal errors.

Format: RASPBERRY

Valid in all states

7.3.23 **WAIT**

Pauses execution for the given time (up to 60 seconds).
Useful in command files to allow the operator time to think.
The wait may be terminated prematurely by CTRL/C.

Format: WAIT real

Valid in all states

7.3.24 **PROMPT**

Allows control of the interactive command prompt.

Format: PROMPT subcommand

Valid in all states

* **PROMPT OFF**
  Disable all prompting.

  Format: PROMPT OFF

* **PROMPT ON**
  Enable prompting using last specified prompt type (default). If no prompt has been specified then default is prompt with '*'.

  Format: PROMPT ON
  or   PROMPT

* **PROMPT STATES**
  Enable prompting using current command state as prompt.

  Format: PROMPT STATES

* **PROMPT TEXT**
  Set prompt string to given text.
  If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

  Format: PROMPT TEXT textstring
  eg: PROMPT TEXT Yes, Oh Master ?

7.3.25 **PRIORITY**

Set priority of puck buttons when pressed over a menu area.

Format: PRIORITY subcommand

Valid in all states

* **PRIORITY POSITION**
  Instructs the specified puck buttons to do an implicit POSITION command to move the LITES2 cursor to the device position before executing the command on the button. This POSITION command will not be executed if the previous command specifically positioned the LITES2 cursor (eg a FIND, SEARCH, POINT or NEXT commands).

  The default action is for table pucks to have PRIORITY POSITION commands on all buttons (if they are pressed while in a map or tracking area), as do buttons on a stereo digitising instrument. Bitpads and mice do not have PRIORITY POSITION commands on their buttons by default.

  The command cancels any previous priority position settings for the puck, and if the range is omitted, no buttons will execute an implicit POSITION command.

  Format: PRIORITY POSITION puckname [range]

  eg PRIORITY POSITION ALTEKPUCK 13-16

* **PRIORITY PUCK**
  Set priority for the buttons on a particular puck. The given buttons will obey their puck function even if pressed over a menu area. The command cancels any previous priority puck settings for the puck, and if the range is omitted, no buttons have priority.

  Format: PRIORITY PUCK puckname [range]

  eg PRIORITY PUCK ALTEKPUCK 4,13-16

7.3.26 **OPERATION**

Defines operations that will produce automatic updating of ACs (ancillary codes) of features, attributes of points and parts of the map header when features are edited.

When features are to be updated, the AC or attribute to be updated can be specified by its name or by the corresponding integer. This integer is referred to as the "type" when considering ACs, and the "code" when considering point attributes.

See the section on LITES2 command language for details of the format of the value for this type of command argument.

More than one AC or attribute can be specified to be updated by each operation. The currently set operations can be displayed with the SHOW OPERATION command.

If when defining an operation to update features, the AC or attribute to be updated is specified without a value, then on completion of the operation, that AC or attribute will be deleted from the feature or point rather than the value being updated or inserted.

When elements of the map header are to be updated, the syntax is different - see OPERATION OS_MH_FLAGS and OPERATION OS_MH_TEXTCAT below.

Operations can be cancelled with the CANCEL command (qv).

NOTE

> If the form of the output file is not set to output revision level 1, either by setting the logical name LSL$IFF_OUTPUT_REVISION to 1 or by the use of the ENABLE REVISION_LEVEL command, any point attributes (apart from Z) will be lost on completion of editing. Points with the attribute Z will produce ZS entries in the IFF file, rather than ST entries. See the IFF user guide for more information on IFF files and LSL$IFF_OUTPUT_REVISION

Format: OPERATION subcommand

Valid in states INITIAL (after FRT has been read) READY

* OPERATION ADD_FEATURE

  When a feature is constructed, it will have the ACs and values specified by this OPERATION command, in addition to any specified in the current set of construction attributes. As the ACs in the construction attributes are deleted whenever a new feature code is set, the use of OPERATION ADD_FEATURE allows all features (whatever their feature code) to have the ACs specified by this command.

  Format: OPERATION ADD_FEATURE type [value]

  eg OPERATION ADD_FEATURE Secondary_FC 24

* OPERATION ANGLESQ_POINT

  Points in features generated by the SQUARE ANGLE command will inherit the attributes specified by this OPERATION command.

  Format: OPERATION ANGLESQ_POINT type [value]

  eg OPERATION ANGLESQ_POINT Z -999.9

* OPERATION BREAK_POINT

  When points are inserted in features during a LITES2 edit, eg PART operations, CLIP commands or the BRIDGE command, then the point will take the attributes specified by THIS OPERATION.

Format: OPERATION BREAK_POINT type [value]

eg OPERATION BREAK_POINT Z -999.9

* OPERATION CIRCLE_POINT

Any point that is generated with the CIRCLE, ARC, POLYGON or POLARC commands will inherit the attributes specified by this OPERATION command in addition to those attributes specified in the construction attribute set at the point when END was entered to complete the feature.

Format: OPERATION CIRCLE_POINT type [value]

eg OPERATION CIRCLE_POINT Z -999.9

* OPERATION CODE_CH_FEATURE

Any feature that has an AC of the type specified by this OPERATION command, will have the value of that AC replaced by the specified value whenever the feature undergoes an edit that alters its feature code, feature serial number, AC or the attribute of a point.
Note that this operation will not insert an AC in a feature.

Format: OPERATION CODE_CH_FEATURE type [value]

eg OPERATION CODE_CH_FEATURE Secondary_FC 24

* OPERATION CURVE_POINT

Any point that is generated with the CURVE command will inherit the attributes specified by this OPERATION command in addition to those attributes specified in the construction attribute set.

Format: OPERATION CURVE_POINT type [value]

eg OPERATION CURVE_POINT Z -999.9

* OPERATION DIGITISE_POINT

Any point that is digitised with the START or INSERT commands will inherit the attributes specified by this OPERATION command in addition to those attributes specified in the construction attribute set.

Format: OPERATION DIGITISE_POINT type [value]

eg OPERATION DIGITISE_POINT Z 23.6

* OPERATION FILTER_POINT

Points in features generated by the FILTER (FEATURE) command will inherit the attributes specified by this OPERATION command. Note that the attributes of the original points will not be transferred to the new feature.

Format: OPERATION FILTER_POINT type [value]

eg OPERATION FILTER_POINT Z -999.9

* OPERATION FAR_MOVE_POINT

Mark points that have been moved, during an editing operation, by more than a specified horizontal distance. This distance criterion is set by the TOLERANCE FARMOVE command (qv).

The attributes affected by this operation are set after any others controlled by other (more specific) OPERATION commands. Thus, for example, if the following OPERATIONS are set up:

```
OPERATION MOVE_POINT     text  -9
OPERATION FAR_MOVE_POINT text  -8
```

a point that is edited, and moved by less than TOLERANCE FARMOVE, will have its `text' attribute set to -9, while one that is moved by more than TOLERANCE FARMOVE will have its `text' attribute set to -8.

Format: OPERATION FAR_MOVE_POINT type [value]

eg OPERATION FAR_MOVE_POINT Z

* OPERATION GEO_CH_FEATURE

Any feature that has an AC of the type specified by this OPERATION command, will have the value of that AC replaced by the specified value whenever the feature undergoes an edit that alters its position or geometry.
Note that this operation will not insert an AC in a feature.

Format: OPERATION GEO_CH_FEATURE type [value]

eg OPERATION GEO_CH_FEATURE Secondary_FC 24

* OPERATION JOIN_POINT

When two linear features are joined together, then the new point that replaces the two end points takes the attributes of the point from the first feature found. These attributes are updated by the attributes specified by this OPERATION command.
Note that the attributes of the corresponding point in the second feature are lost.

Format: OPERATION JOIN_POINT type [value]

eg OPERATION JOIN_POINT Z -999.9

* OPERATION MOVE_POINT

Points in features altered by the MOVE, TIE, EDIT, EXTEND or LOOP commands will inherit the attributes specified by this OPERATION command; if the point has this attribute already, then the value will be updated, otherwise the attribute will be added.

Format: OPERATION MOVE_POINT type [value]

eg OPERATION MOVE_POINT Z -999.9

* OPERATION OFFSET_POINT

Points in linear features generated by the OFFSET command will inherit the attributes specified by this OPERATION command. Note that the attributes of the original points will not be transferred to the new feature.
Note that when symbols and texts are offset the OPERATION TRANSFORM_POINT is the operation that is effective.

Format: OPERATION OFFSET_POINT type [value]

eg OPERATION OFFSET_POINT Z -999.9

* OPERATION OS_MH_FLAGS

Set the specified flag in the (OS type) map header with the specified character, when a feature with a feature code in the specified FRT group has been edited, added or deleted.

This operation is rather different from all the others:

1. It is specific to files with an OS type map header
2. Only one FRT group and character can be specified for each flag
3. The updating of the map header only takes place when the file is finally written out (after an EXIT or WRITE command), so if the operation is cancelled before this, the flag will not be changed.

NOTES

1. The present definition of the OS map header limits the number of flags to 8, and insists that the character must be an upper case letter or a digit.

2. If files with new OS map headers (ie those with customer number 3 or 4) are to be updated, then the logical name LSL$OS_MH_TABLE must have been set up to point to a translation table before LITES2 was invoked. See CONVERT package for details of the file that this logical name should point to.

Format: OPERATION OS_MH_FLAGS flagno group char

eg OPERATION OS_MH_FLAGS 8 WATER A

* OPERATION OS_MH_TEXTCAT

Refine the effect of the OPERATION OS_MH_FLAGS command, by allowing specific categories of texts to trigger the flag setting.

By default, if a text feature code is in a group specified by a OPERATION OS_MH_FLAGS command, edits to any feature with that feature code will cause the corresponding flag to be updated. When the command OPERATION OS_MH_TEXTCAT has been given for a specific flag and text feature code, only texts with the specified category will cause the flag to be updated. The command may be repeated for any flag and feature code, to allow several different text categories of text to be set.

Format: OPERATION OS_MH_TEXTCAT flagno fc category

eg   OPERATION OS_MH_TEXTCAT 8 28 2

* OPERATION OTHER_POINT

  If a point has been edited, but the relevant OPERATION xxxx_POINT command has not been given, then this OPERATION command takes effect.

  Format: OPERATION OTHER_POINT type [value]

  eg   OPERATION OTHER_POINT Z -999.9

* OPERATION PARTSQ_POINT

  Points in features generated by the SQUARE PART command will inherit the attributes specified by this OPERATION command.

  Format: OPERATION PARTSQ_POINT type [value]

  eg   OPERATION PARTSQ_POINT Z -999.9

* OPERATION SQUARE_POINT

  Points in features generated by the SQUARE (WHOLE) and the RECTANGLE command will inherit the attributes specified by this OPERATION command.

  Format: OPERATION SQUARE_POINT type [value]

  eg   OPERATION SQUARE_POINT Z -999.9

* OPERATION TRANSFORM_POINT

  Points in features altered by the TRANSFORM command will inherit the attributes specified by this OPERATION command; if the point has this attribute already, then the value will be updated, otherwise the attribute will be added.
  Note that texts and symbols manipulated by the ROTATE, TURN, LARGER, SMALLER, OFFSET, ALIGN and STRETCH commands will also inherit the attributes specified by the OPERATION TRANSFORM_POINT command updated.

  Format: OPERATION TRANSFORM_POINT type [value]

  eg   OPERATION TRANSFORM_POINT Z -999.9

* OPERATION USER_FEATURE
  Any feature generated by the USER (or ROUTINE) command that has an AC of the type specified by this OPERATION command, will have the value of that AC replaced by the specified value.
  This would allow user routines to create features with an AC that contains a date or a time that may otherwise be difficult to construct in a user routine. The user routine would produce a feature with the appropriate AC with a dummy value, whose correct value would be inserted by this OPERATION.

  Note that this operation will not insert an AC in a feature.

  Format: OPERATION USER_FEATURE type [value]

  eg   OPERATION USER_FEATURE Secondary_FC 24

* OPERATION USER_POINT
  Points in features generated by the USER command will inherit the attributes specified by this OPERATION command; if the point has this attribute already, then the value will be updated, otherwise the attribute will be added.

  Format: OPERATION USER_POINT type [value]

  eg   OPERATION USER_POINT Z -999.9

7.3.27 **PRIVILEGE**

Defines commands that cannot subsequently be given, attributes that may not be altered and points that are held fixed while squaring.

Format: PRIVILEGE subcommand

Valid in state INITIAL (privileged command;
                         only valid in initialisation file)

* **PRIVILEGE ATTRIBUTE**
  Defines point attributes and ACs that may not be edited during the current LITES2 session.

  Note that this does not inhibit the editing of the position of a point with the specified attributes, or of a feature with the specified AC.

  It will not be possible to subsequently give an OPERATION command that will alter privileged attributes, but any OPERATION command given before the PRIVILEGE ATTRIBUTE command will be honoured.

  The AC or attribute to be privileged can be specified by its name or by the corresponding integer. This integer is referred to as the "type" when considering ACs, and the "code" when considering point attributes.

  Format: PRIVILEGE ATTRIBUTE type

eg PRIVILEGE ATTRIBUTE Z

* **PRIVILEGE COMMAND**
Defines commands that may not subsequently be given during the current LITES2 session.

If a primary and secondary command are given, then only the specified secondary command will be inhibited; if however only a primary command is given then none of the secondary commands associated with the primary command will be subsequently accepted in the current LITES2 session.

Format: PRIVILEGE COMMAND primary [secondary]

eg PRIVILEGE COMMAND TOLERANCE SQMT

* **PRIVILEGE POINT**
Defines points that are to be held fixed during PART and WHOLE squaring operations. Points cannot be held fixed during ANGLE squaring.

NOTE

This command is only effective if the FIXED option is switched on with the ENABLE FIXED command (on by default). See the chapter "Squaring within LITES2" for details of the squaring algorithm used.

Points with the given attribute and value will be held fixed during PART and WHOLE squaring of features.

The attribute to be tested can be specified by its name or by the corresponding integer.

The format of the value depends on the data type of the specified attribute code, either defined by Laser-Scan or defined by the user in his FRT.

Format: PRIVILEGE POINT attribute value

eg PRIVILEGE POINT CAPTURE_XY 6

7.3.28 **PROJECTION**

Controls the use of projection information in LITES2

LITES2 has the ability to work on several IFF files which are in different projections at the same time. It does this by specifying the projection to display the files in, called LITES2 space, and transforming the coordinates in the individual files to this space before they are used by LITES2. It should be noted that the data files remain in their original projection throughout the

LITES2 session.

This facility depends on all the files being used having a valid projection set up in their map descriptor and these projections all being referred to the same spheroid. See the Documentation for the program ITRANS for more information about map descriptors.

NOTE

> When using this mode, and setting maps up to be digitised on a digitising table, the paper maps should be in the same projection as the LITES2 space.

Format: PROJECTION subcommand

* **PROJECTION IFF**
  To select the projection for the LITES2 coordinate space, a map descriptor is required. This is supplied as the map descriptor in any IFF file that is in the required projection. When an IFF file with a valid map descriptor representing a valid projection is given with this command, all subsequent files that are read in will be displayed in this projection.

  If the command is given with no file name, then LITES2 reverts to the default action of assuming all the files are in the same projection.
  Format: PROJECTION IFF [filename]

  eg PROJECTION IFF testcard

  Valid in state INITIAL

* **PROJECTION OUTPUT**
  When working with "transformations on the fly", when individual maps are output (with either the WRITE or EXIT commands with no filename) the files are written back to their original coordinate system. If more than one map is selected for output and WRITE or EXIT is given with a file name, the maps are combined in LITES2 coordinate space, and the combined map is output in this space. When only one map is selected and it is output to a named file, whether it is output in its original projection or in LITES2 coordinate space is controlled by the PROJECTION OUTPUT command.

  Valid in states INITIAL READY

  o **PROJECTION OUTPUT OFF**
    Select the (default) mode where individual maps being output to a named file are output in the coordinate system of the original map.

    Format: PROJECTION OUTPUT OFF

  o **PROJECTION OUTPUT ON**
    Select the mode where individual maps being output to a named file are transformed into LITES2 coordinate space while being output.

    Format: PROJECTION OUTPUT ON

* **PROJECTION RANGE**

When working with a LITES2 space that is different from that of the IFF files that hold the data, it is necessary for LITES2 to work out the range of the data in each file in LITES2 space. There are three possible ways of calculating this range.

Unfortunately, it is not possible to know if the range has been calculated correctly before the IFF data has been read, but if while inputting the data LITES2 detects a possible problem an informational message is output and the system variable $RANGE_PROBLEM is set to TRUE.

Valid in states INITIAL READY

- **PROJECTION RANGE CORNER**

  Transform the corners of the minimum bounding rectangle defined by the Range entry in the IFF file, and take the minimum and maximum of the x and y of these transformed points. This is fast, but may not give an accurate (or indeed any) result for data that is being transformed into a radically different size or shape.

  Format: PROJECTION RANGE CORNER

- **PROJECTION RANGE DATA**

  Transform all the coordinate data in the file and calculate a new range in LITES2 space. This provides the most accurate solution, with a genuine minimum range, but it may be time consuming if the data files are large.

  Format: PROJECTION RANGE DATA

- **PROJECTION RANGE SIDE**

  Transform a selection of points along the boundaries of the minimum bounding rectangle as defined by the range entry, and use the transformed points to calculate the range of the data in LITES2 space. This method will give a correct result if enough points are selected between the corners of the minimum bounding rectangle. If no number is given, then the current specified number is used.

  This is the default setting, with the number of intermediate points along each edge being set to 20.

  Format: PROJECTION RANGE SIDE [number]

7.4 **General Commands**

7.4.1 **ABANDON**

Abandons current operation.
Drops any found feature and returns to READY state.

Format: ABANDON

Valid in all states

7.4.2 **END**

Ends editing or construction type operation.

Format: END

Valid in states EDIT MODIFY ON WINDOW CONSTRUCT AC RECOVER PAINT

7.4.3 **CREATE**

Creates a new entity.

Format: CREATE subcommand

Valid in states INITIAL READY

* **CREATE ABORT_MAILBOX**
  Creates a mailbox which can be used by other processes to abort LITES2 operations. When a process writes a message to the mailbox, LITES2 behaves as though CTRL/C had been pressed, aborting certain commands, or returning to interactive input. The message written to the mailbox is output by LITES2 after the "Operation aborted" message.

  If the message begins with the digit 0, 1, or 2, then this controls the "severity" of the abort. 0 (or no digit at all) means the same as a keyboard CTRL/C - it will abort commands such as DRAW if any are in progress, but if not will abort the reading of any macros and command files. 1 will abort commands such as DRAW, but will not affect command decoding. 2 will abort certain commands, and also abort command decoding. The digit is removed from the message before it is printed.

  If the name argument is given, then a mailbox with the given name will be created. The name is converted to upper case. It should not contain spaces or other punctuation marks. If the name argument is omitted, the default name LSL$LITES2ABORT will be used.

  By default, the logical name for the mailbox goes into the JOB logical name table, so that it will be visible to any subprocesses. If it was required, for example, that the logical name for the mailbox was placed in the group logical name table, then give the DCL command

```
$ Define/table=LNM$PROCESS_DIRECTORY -
               LNM$TEMPORARY_MAILBOX LNM$GROUP
```

before starting LITES2.

Format: CREATE ABORT_MAILBOX
or CREATE ABORT_MAILBOX name

* **CREATE LAYER**
Create a new layer.

Format: CREATE LAYER integer

Valid in state READY

* **CREATE LOGICAL**
Creates (defines) a logical name. If either the logical name or the equivalence string contains spaces then it should be enclosed in double quotes. Both strings are converted to upper case unless enclosed in double quotes. If the equivalence string is omitted, then the logical name is deassigned.

The logical name is defined in the process table (LNM$PROCESS) at user mode. This means that it will automatically be deassigned when LITES2 exits, and is therefore only useful during the run of LITES2. A particular use is to define logical names for use by hardcopy plotting output routines.

Format: CREATE LOGICAL logical_name equivalence_string
or CREATE LOGICAL logical_name ! (deassign)

eg CREATE LOGICAL LSL$PS TTA2:

* **CREATE MAILBOX**
Creates a mailbox which can be used for communication between LITES2 and other processes. The command has two forms, taking either an integer in the range 1-4, or a name.

The integer argument is the auxiliary input number, thus CREATE MAILBOX 1 creates a mailbox with logical name LSL$LITES2AUX, while CREATE MAILBOX n for n = 2, 3, or 4 creates a mailbox with logical name LSL$LITES2AUX_n. LITES2 will attempt to read interactive commands from these mailboxes. The logical name should not already exist when the CREATE MAILBOX command is given - LITES2 will automatically attempt to read commands from these logical names if they exist.

If the name argument is given instead of the integer, then a mailbox with the given name will be created, but LITES2 will not automatically attempt to read from it. The name is converted to upper case. It should not contain spaces or other punctuation marks. Once created, the mailbox may be treated as a file by any of the LITES2 commands which use text files (e.g. the FILE commands, or @). For example, one could use CREATE MAILBOX OUTPUT, then FILE CREATE 1 OUTPUT:, and FILE WRITE string to send text strings to another process. Note that attempts to read or write the mailboxes will not complete until the cooperating process writes or reads, so care must be taken not to hang LITES2 in a permanent wait state.

By default, the logical name for the mailbox goes into the JOB logical name table, so that it will be visible to any subprocesses. If it was required, for example, that the logical name for the mailbox was placed in the group logical name table, then give the DCL command

```
$ Define/table=LNM$PROCESS_DIRECTORY -
                LNM$TEMPORARY_MAILBOX LNM$GROUP
```

before starting LITES2.

Format: CREATE MAILBOX integer
or   CREATE MAILBOX name

* **CREATE MEMORY**
Reserves the specified number of pages of virtual memory for use by LITES2 dynamic memory allocation. It is intended that this command be used when image files are to be repeatedly opened and closed. If LITES2 has to allocate memory at a higher address than currently open images, then when an image is closed, the memory used by it will not be able to be reused, and the next image opened will use up more memory. Use the CREATE MEMORY command to reserve enough space for the expected use of dynamic memory.

Format: CREATE MEMORY integer

7.4.4 **UNITS**

Overrides the type of units expected for the following command.

Format: UNITS subcommand

Valid in all states except INITIAL

* **UNITS FACTOR**
Specifies a temporary conversion factor for coordinate units. The given number multiplies IFF units to give the required units, thus if IFF units were metres, then the command UNITS FACTOR 0.001 would allow the argument to the next command to be specified in kilometres. A descriptive string may be given after the factor. This appears in the output from some EXAMINE commands. The last specified factor is available in the system variable $UNIT_FACTOR, and the descriptive string in $UNIT_DESC.

Format: UNITS FACTOR real [string]

```
eg      UNITS FACTOR 0.001 kilometres
        OFFSET 0.2    (offsets by 200 IFF units rather then 0.2)
```

* **UNITS IFF**
  Commands that expect their arguments in sheet mm will accept them as IFF units if preceded by this command

  Format: UNITS IFF

  ```
  eg      UNITS IFF
          TOLERANCE SQMT 10    (takes 10 as 10 IFF units, not 10 mm)
  ```

* **UNITS MMS**
  Commands that expect their arguments in IFF units will accept them as sheet mm if preceded by this command

  Format: UNITS MMS

  ```
  eg      UNITS MMS
          OFFSET 1.2    (offsets by 1.2 sheet mm, not 1.2 IFF units)
  ```

* **UNITS NORMAL**
  Cancel the effect of a previous UNITS command. The effect of UNITS is cancelled automatically by the first command to use it, but UNITS NORMAL may be used to cancel the effect before use.

  Format: UNITS NORMAL

7.4.5 **MERGE**

Merges two entities

Format: MERGE subcommand

Valid in state READY

* **MERGE LAYER**
  Merges one layer into another, which must already exist

  Format: MERGE LAYER old new

7.4.6 **RENAME**

Renames an entity.

Format: RENAME subcommand

Valid in state READY

* **RENAME LAYER**
  Renames one layer as another, which must not already exist

  Format: RENAME LAYER old new

7.4.7 **USER**

Specifies that a particular user-supplied operation is to be performed. If there is a found feature it is passed from LITES2 to the user-routine image, which can interrogate aspects of the feature, or modify it and pass it back.

Other information which is passed to the user-routine includes an identifying integer and an optional string (as input with the command), the current cursor position, the current state and the value of the condition flag (used for JUMP etc (qv)).

Data that can be passed from the user-routine to LITES2, as well as a feature, includes the condition flag and a LITES2 command string which will be the next command string to be obeyed.

The shareable image to be used is pointed to by the logical name LSL$LITES2ROUTINES

For information about writing user-routines, see section 8.

Format: USER integer [string]

Valid in states:
INITIAL READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON WINDOW CONSTRUCT AC

7.4.8 **ROUTINE**

An extended version of the USER command that allows up to 10 external images to be accessed. (see USER command).

There are two sets of ROUTINE commands available -

ROUTINE 1   to ROUTINE 5   are for images supplied by the user,

ROUTINE 101 to ROUTINE 105 are reserved for additional images supplied by Laser-Scan.

The shareable image to be used is pointed to by the logical name LSL$LITES2ROUTINES_n, where n is the first argument to the command.

ROUTINE 0 is an alias for the command USER

See the section 8 for more information, and details of how to write user routines.

Format: ROUTINE integer integer [string]

Valid in states:
INITIAL READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON WINDOW CONSTRUCT AC

7.4.9 **BASE**

Specifies that the current segment of the found feature is to be used as a baseline. The currently set bases can be examined with the SHOW BASES command.(qv)

Format: BASE subcommand

Valid in state LINE

* **BASE EDGE**
  The baseline is used when the next EDGEMATCH command is given. Only lines that are longer than the tolerance SQBT will be accepted as bases for edgematching.
  On completion of the command, LITES2 returns to READY state.

  Format: BASE EDGE

* **BASE ORIENT**
  The baseline is used when the next ORIENT command is given. Only lines that are longer than the tolerance SQBT will be accepted as bases for orienting.
  On completion of the command, LITES2 returns to READY state.

  Format: BASE ORIENT

* **BASE SQUARE**
  The current line is added to the list of bases to be used for OS type squaring.
  Only lines that are longer than the tolerance SQBT will be accepted for squaring.
  Up to five of these bases can be defined before any SQUARE command is given and the definitions are lost when either an explicit or an implicit ABANDON command is given. On completion of this command, the program remains in LINE state.

  Format: BASE SQUARE
  or BASE

7.4.10 **ASK**

A command that places information in associated system variables.

Commands that return integers, place the information in the variables $ASK_INT n, those that return real values place them in the variables $ASK_REAL n and those that return character (string) values place them in $ASK_CHAR n.

Commands may return values in one or more of these variables.

For upwards compatibility, the integer variable $MAP_NUMBER is a synonym for $ASK_INT 1 and the real variables $TABLEXY are synonyms for $ASK_REAL.

Format: ASK subcommand

Valid in all states (where appropriate)

* **ASK CHARACTER**
  Takes the argument as an ascii value, and returns the corresponding character in the variable $ASK_CHAR 1. The length of $ASK_CHAR 1 is itself returned in the variable $ASK_INT 1.

  Format: ASK CHARACTER n

  ```
  eg      ASK CHARACTER 65
          SHOW VARIABLE $ASK_CHAR 1
          SHOW VARIABLE $ASK_INT  1
  ```

* **ASK GEOMETRY**
  Returns information about geometries

  Format: ASK GEOMETRY subcommand

  o **ASK GEOMETRY POINT**
    Returns the coordinate of a point that is guaranteed to lie on the specified geometry in the variables $ASK_REAL 1 and $ASK_REAL 2.

    For point type geometries this will be (one of) the point(s)
    For line type geometries this will be a point on a line (not a vertex)
    For area type geometries this will be a point that lies within the bounding line work of the area.

    Format: ASK GEOMETRY POINT integer

* **ASK HLS**
  Returns the Hue Lightness and Saturation of the specified colour in the current display and overlay in the variables $ASK_REAL 1, $ASK_REAL 2 and $ASK_REAL 3. It also returns its attribute in the system variable $ASK_CHAR 1, and the number of characters of this string in the variable $ASK_INT 1.

  This command is only available on versions with suitable hardware facilities.

  Format: ASK HLS n

* **ASK HSV**
  Returns the Hue Saturation and Value of the specified colour in the current display and overlay in the variables $ASK_REAL 1, $ASK_REAL 2 and $ASK_REAL 3. It also returns its attribute in the system variable $ASK_CHAR 1, and the number of characters of this string in the

variable $ASK_INT 1.

This command is only available on versions with suitable hardware facilities.

Format: ASK HSV n

* **ASK IDENTIFIER**
Returns TRUE (-1) in the variable $ASK_INT 1 if the user has the specified identifier in his rights identifier list, and FALSE (0) if not. If the string does not represent a valid rights identifier or if it does not exist as a rights identifier, then the command will moan, and the variable will be unset.

Format: ASK IDENTIFIER string

```
eg       ASK IDENTIFIER INTERACTIVE
         SHOW VARIABLE $ASK_INT  1
```

* **ASK LEGEND_SIZE**
Returns the size (length and height as a proportion of the screen) that the legend would take up when drawn with a DRAW LEGEND command on the current screen with the current annotation settings. The length is returned in $ASK_REAL 1 and the height in $ASK_REAL 2. The number of boxes in the menu is returned in $ASK_INT 1.

Note that the height ($ASK_REAL 2) may be more than 1.0. In this case a subsequent DRAW LEGEND command will fail. (The width may also be greater than 1.0, but in this case the DRAW LEGEND command will not fail, but will simply truncate the legend).

This command is not available in INITIAL state.

Format: ASK LEGEND_SIZE

* **ASK MAP_NUMBER**
If the optional string argument is present then this command fills in the system variable $ASK_INT 1 ($MAP_NUMBER) with the number of the first map that contains the string in its name; if no string is present then the variable will contain the number of the next map that contains the string given in the last ASK MAP_NUMBER command that had an argument.

In either case, if the string is not found in any map name, the variable will contain 0.

Format: ASK MAP_NUMBER [string]

```
eg       ASK MAP_NUMBER .IFF
         SHOW VARIABLE $ASK_INT 1

                    1                        for example
         ASK MAP_NUMBER
         SHOW VARIABLE $ASK_INT 1
```

```
                2                        for example
        ASK MAP_NUMBER
        SHOW VARIABLE $ASK_INT 1

                0                        for example
```

* **ASK POSITION**
  Converts a position between LITES2 coordinate space and the coordinate space of any specified map. LITES2 space is the coordinate system that the command POSITION and SHOW POSITION work with.
  The result of the conversion is put in the system variables $ASK_REAL 1 and $ASK_REAL 2.

  This command is not available in INITIAL state.

  Format: ASK POSITION subcommand

  o **ASK POSITION FROM_MAP**
    Converts between the point `x,y' in the coordinate space of the specified map and the LITES2 coordinate space.

    Format: ASK POSITION FROM_MAP map x y

  o **ASK POSITION TO_MAP**
    Converts between the point `x,y' in LITES2 space and the coordinate system of the specified map.

    Format: ASK POSITION TO_MAP map x y

* **ASK RGB**
  Returns the Red Green and Blue values of the specified colour in the current display and overlay in the variables $ASK_REAL 1, $ASK_REAL 2 and $ASK_REAL 3. It also returns its attribute in the system variable $ASK_CHAR 1, and the number of characters of this string in the variable $ASK_INT 1.

  This command is only available on versions with suitable hardware facilities.

  Format: ASK RGB n

* **ASK STATUS**
  Returns status information about the specified entity.

  Format: ASK STATUS subcommand

  o **ASK STATUS MACRO**
    Returns information about the specified macro, menu or puck.

    In this context a subscripted puck or menu name is treated as a macro. Also note that, as when using macro names in other contexts, the name may be abbreviated until it becomes ambiguous.

If the name does not represent a macro, menu or puck 0 is returned in $ASK_INT 1 and no other $ASK variables are set.

If the name represents a macro, then 1 is returned in $ASK_INT 1 and the number of characters in the macro is returned in $ASK_INT 2 (this can be used to test if a menu square or puck button can be redefined without cancelling first); if the name represents a menu or a puck then 2 is returned in $ASK_INT 1 and the number of boxes in the menu or buttons on the puck is returned in $ASK_INT 2. In both these cases the full (non-abbreviated) name is returned in $ASK_CHAR 1.

Format: ASK STATUS MACRO name

- **ASK STATUS VARIABLE**
  Returns information about the specified variable.

  The type of the variable is returned in $ASK_INT 1 as follows:

  = 0 - variable does not exist
  = 1 - integer
  = 2 - real
  = 3 - character
  = 4 - double

  If the variable exists and if it represents a variable array, then the number of elements in the array is returned in $ASK_INT 2. Simple variables return 0 in $ASK_INT 2.

  Note that subscripts are not allowed in the variable name in this command.

  Format: ASK STATUS VARIABLE name

* **ASK STRING**
  Returns information and carries out lexical operations on the specified string.

  Where a string is returned in the variable $ASK_CHAR n, then the length of this string is generally returned in the variable $ASK_INT n.

  Format: ASK STRING subcommand

  - **ASK STRING ANNOTATION_SIZE**
    Returns the size (length and height in IFF units) that the specified string would take up when drawn with a DRAW TITLE or DRAW TEXT command on the current screen with the current annotation settings. The length is returned in $ASK_REAL 1 and the height in $ASK_REAL 2.

    This command is not available in INITIAL state.

    Format: ASK STRING ANNOTATION_SIZE string

- **ASK STRING ASCII**
  Converts the first character in the specified string to an 8 bit ASCII character and returns in in $ASK_INT 1

  Format: ASK STRING ASCII string

- **ASK STRING COLLAPSE**
  Removes all spaces and tabs from a string and returns the resultant string in the variable $ASK_CHAR 1

  Format: ASK STRING COLLAPSE string

- **ASK STRING COMPRESS**
  Replaces all occurrences of multiple spaces and tabs by single spaces and returns the resultant string in the variable $ASK_CHAR 1

  Format: ASK STRING COMPRESS string

- **ASK STRING DCLSYMBOL**
  Returns the value of the specified DCL symbol in the variable $ASK_CHAR 1. If the symbol does not exist, then a null string is returned.

  Format: ASK STRING DCLSYMBOL string

- **ASK STRING ELEMENT**
  Extracts a specified element from a string in which the elements are separated by a specified delimiter. The result is returned in the variable $ASK_CHAR 1.

  The delimiter must only be 1 character long

  If the delimiter does not exist in the string, or there are less elements in the string than the one specified, then the delimiter is returned in $ASK_CHAR 1.

  Note that the element is the substring after the specified delimiter, thus the command :

  ```
  ASK STRING ELEMENT 2 /   MON/TUE/WED/THUR/FRI/SAT/SUN
  ```

  will return the string "WED" while the command :

  ```
  ASK STRING ELEMENT 2 /  /MON/TUE/WED/THUR/FRI/SAT/SUN
  ```

  will return the string "TUE"

  Format: ASK STRING ELEMENT n delimiter string

- **ASK STRING EXTRACT**
  Extracts a substring, denoted by its start and end position, from a string and returns it in the variable $ASK_CHAR 1.

  If the end position is less than the start position, the null string is returned.

Format: ASK STRING EXTRACT start_pos end_pos string

eg ASK STRING EXTRACT 2 4 Freddy

will return the string "red"

- **ASK STRING FILE_FIND**
  Returns a character string containing the expanded file specification for the file-spec argument in the variable $ASK_CHAR 1. If the FIND_FILE function does not find the file in the directory, a null ("") string is returned.

  If the device or directory names are omitted from the input file-spec, defaults are supplied from the current default disk and directory. Defaults for a file name or type are not supplied and if the version number is omitted, the specification for the file with the highest version number is returned.

  Wildcards can be used in the file-spec argument. In this case repeated calls with the same input file-spec will return all the file specifications that match the input. When all the files have been found, a null ("") string is returned.

  Format: ASK STRING FILE_FIND filespec

  eg ASK STRING FILE_FIND lsl$com:lites2ini.com

  may return the string "LSL$SITE_ROOT:[LSL.COM]LITES2INI.COM;23"

  while repeated calls of the command

  ASK STRING FILE_FIND lsl$com:lites2ini.com;*

  may return the strings "LSL$SITE_ROOT:[LSL.COM]LITES2INI.COM;23"
  "LSL$SITE_ROOT:[LSL.COM]LITES2INI.COM;22"
  ""

- **ASK STRING INDEX**
  Returns the starting position of string2 in string1 in the variable $ASK_INT 1.

  If string1 does not contain string2, 0 is returned; if string2 occurs more than once in string1, the position of the first occurrence is returned.

  Format: ASK STRING INDEX string1 string2

- **ASK STRING ISALPHA**
  Returns TRUE (-1) in the variable $ASK_INT 1 if all the characters in the string are alphabetic (ie if they all lie between 'A' and 'Z' or between 'a' and 'z'). Otherwise FALSE (0) is returned.

  Format: ASK STRING ISALPHA string

- **ASK STRING ISDATETIME**
  Returns TRUE (-1), in the variable $ASK_INT 1, if the string represents a valid DEC VMS date/time string, otherwise FALSE (0) is returned. If the string is a valid date, then the date part is returned in $ASK_CHAR 1 (and its length in $ASK_INT 2) and the time in $ASK_CHAR 2 (with its length in $ASK_INT 3)

  Format: ASK STRING ISDATETIME string

- **ASK STRING ISDIGIT**
  Returns TRUE (-1) in the variable $ASK_INT 1 if all the characters in the string are numeric (ie if they all lie between '0' and '9'). Otherwise FALSE (0) is returned.

  Format: ASK STRING ISDIGIT string

- **ASK STRING ISINTEGER**
  Returns TRUE (-1) in the variable $ASK_INT 1 if the characters in the string form a valid integer. Otherwise FALSE (0) is returned.

  Format: ASK STRING ISINTEGER string

- **ASK STRING ISREAL**
  Returns TRUE (-1) in the variable $ASK_INT 1 if the characters in the string form a valid real number. Otherwise FALSE (0) is returned.

  Format: ASK STRING ISREAL string

- **ASK STRING LEFT**
  Extracts a substring, denoted by its end position, from a string and returns the result in the variable $ASK_CHAR 1.

  If the end position is less than the start position, the null string is returned.

  Format: ASK STRING LEFT end_position string

  eg   ASK STRING LEFT 4 Freddy

  will return the string "Fred"

- **ASK STRING LENGTH**
  Returns the length of the string in the variable $ASK_INT 1.

  Format: ASK STRING LENGTH string

- **ASK STRING LOWERCASE**
  Converts any upper case alphabetic characters in the string to lower case, and returns the string in $ASK_CHAR 1.

  Format: ASK STRING LOWERCASE string

- **ASK STRING NO_DOLLAR**
  Converts any "$ escape" sequences as used in Laser-Scan's FRTLIB text drawing routines (see the MAPPING package for details) in the string to the corresponding 8 bit ASCII character, and returns the result in the variable $ASK_CHAR 1.

  Note that for this to work satisfactorily, the TRI must contain the appropriate characters from the DEC multinational character set.

  Format: ASK STRING NO_DOLLAR string

- **ASK STRING PAD**
  Pad the given string to the specified length by adding the appropriate number of spaces, and return the new string in the variable $ASK_CHAR 1.

  Note that to use this variable in subsequent LITES2 commands, it must usually be surrounded by quotation marks ("), as trailing spaces are generally stripped off LITES2 commands.

  Format: ASK STRING PAD length string

- **ASK STRING PARSE**
  Parses the given file name and returns the specified field in the variable $ASK_CHAR 1.

  No punctuation is included in the fields (except for '.' in directories). If the specified field does not exist, a null string is returned.

  Valid fields are :

  ```
  NODE
  DEVICE
  DIRECTORY
  NAME
  TYPE or EXTENSION
  VERSION
  ```

  These may all be truncated until they become ambiguous.

  Format: ASK STRING PARSE filename field

- **ASK STRING RIGHT**
  Extracts a substring, denoted by its starting position, from a string and returns the result in the variable $ASK_CHAR 1.

  If the start position is greater than the length of the string, the null string is returned.

  Format: ASK STRING RIGHT position string

  eg ASK STRING RIGHT 4 Freddy

  will return the string "ddy"

o **ASK STRING TEXT_SIZE**
Returns the size (length and height in IFF units) that the specified string would be if constructed with a TEXT command. The size depends on the values in the attribute set (as set with the SET command). The length is returned in $ASK_REAL 1 and the height in $ASK_REAL 2.

This command is not available in INITIAL state.

Format: ASK STRING TEXT_SIZE string

o **ASK STRING TRIM**
Trims any trailing spaces and tabs off the end of the string, and returns it in the variable $ASK_CHAR 1.

Note that to enter the string in the command line in the first place, it must be surrounded by quotation marks (").

Format: ASK STRING TRIM string

o **ASK STRING TRNALL**
Does a recursive translation on the string as a logical name (in the logical name table LNM$FILE_DEV) and returns the result in the variable $ASK_CHAR 1.

If the logical name does not translate, or is not resolved by 10 translations, then the null string is returned.

Format: ASK STRING TRNALL string

o **ASK STRING TRNLNM**
Does a single translation on the string as a logical name (in the logical name table LNM$FILE_DEV) and returns the result in the variable $ASK_CHAR 1.

If the logical name does not translate then the null string is returned.

Format: ASK STRING TRNLNM string

o **ASK STRING UPCASE**
Converts any lower case alphabetic characters in the string to upper case, and returns the string in $ASK_CHAR 1.

Format: ASK STRING UPCASE string

* **ASK TABLE**
Returns the position of the table cursor in the system variables $ASK_REAL 1 and $ASK_REAL 2. It does this without the operator pressing any button on the puck. The cursor must be lying within the area of a map or tracking area that has been set up on the table, for the command to succeed.

NOTE

This command is not available when the table is accessed through a LSL MUART (ie ENABLE MONITOR must have been given). The table must also be of a type that responds to requests from the host computer. In particular, if using an ALTEK 90 controller, the "ENABLE HOST" switch must be set to the on position.

When using non-ALTEK tables, it is possible to specify the string that is used to inquire the position from the table, in the shared image that is used to decode the table string. See the documentation on the TABLE MONITOR for details of this decoding routine, or the example supplied in LSL$PUBLIC_ROOT:[TABLE.EXE]EXAMPLE.FOR. This routine can be compiled and linked using the command file DECODE.COM in the same directory.

```
Format   ASK TABLE

eg       ASK TABLE
         SHOW VARIABLE $ASK_REAL 1
         SHOW VARIABLE $ASK_REAL 2
```

7.4.11 **FILE**

Specifies an action on a user specified text file. Files may be opened for both reading and writing. Each text file opened has an associated file-number which is used to reference the file for selection and closing.
Up to three text files may be open at one time.

Format: FILE subcommand

Valid in all states

* **FILE APPEND**
  Specifies an existing text file, to which data is to be appended, with the given file-number and file name. The file is automatically selected. The default file specification is the current directory, with file extension ".DAT".

  ```
  Format:  FILE APPEND file-number filename

  eg       FILE APPEND 2 FRED
  or       FILE APPEND 2 DUA1:[FRED.DEMO]TEST.DAT
  ```

* **FILE CLOSE**
  The file with the given file number is closed.

  ```
  Format:  FILE CLOSE file-number

  eg       FILE CLOSE 3
  ```

* **FILE CREATE**
Specifies a text file to be created with the given file-number and file name. The file is automatically selected. By default files are created in the current directory with extension ".DAT".

```
Format:  FILE CREATE file-number filename

eg       FILE CREATE 1 FRED
or       FILE CREATE 1 DUA1:[FRED.DEMO]TEST.DAT
```

* **FILE OPEN**
Opens an existing file for reading, with the given file-number and file name. The file is automatically selected. The default file specification is the current directory, with file extension ".DAT".

```
Format:  FILE OPEN file-number filename

eg       FILE OPEN 1 FRED
or       FILE OPEN 1 DUA1:[FRED.DEMO]TEST.DAT
```

* **FILE READ**
Reads the next record from the currently selected file into the system variable $FILELINE. If the end of the text file is reached then the system variable $EOF is set to -1. Otherwise it is 0.

```
Format:  FILE READ
```

* **FILE SELECT**
Specifies that the file with the given file-number is selected for reading or writing.

```
Format:  FILE SELECT file-number

eg       FILE SELECT 1
```

* **FILE WRITE**
Writes the given text string to the currently selected file. If no text string is supplied then a blank line is written. If the text is to have leading spaces or tabs, then it must be enclosed in double quotes. This command may not be used on files opened for read access with FILE OPEN.

```
Format:  FILE WRITE [text]
```

7.5 **Identification Commands**

7.5.1 **FIND**

Searches for nearest feature to cursor position.
Finds up to 4 features nearest to cursor and highlights the best fit which becomes the FOUND FEATURE and the subject for editing operations. If FIND is given again without moving the cursor, then the program will cycle round the 4 best hits.

The behaviour of FIND can by modified by various other commands, including ENABLE ENDS and TOLERANCE FIND (qv). If there is a cursor constraint in operation, either explicitly using ON or FORCE (qv), or implicitly by COPY and MODIFY commands, then FIND will locate the nearest intersections of features with the constraint.

Format: FIND

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON WINDOW CONSTRUCT

7.5.2 **RECOVER**

Searches for nearest feature in limbo.
Works as FIND (qv) but only finds deleted features. The END command must be used to accept the feature. If RECOVER is given again without moving the cursor, the program will cycle round up to 4 nearest candidates.

Format: RECOVER

Valid in states READY LINE CIRCLE TEXT SYMBOL RECOVER

7.5.3 **SEARCH**

Searches whole IFF file using spiral search for feature matching given specification. The SEARCH command should normally only be used when the intention is to use SEARCH NEXT to find several features in turn. If only one feature is required, then the LOCATE (qv) command is more efficient.

The order in which features are found by SEARCH is not very well defined except for SEARCH NEAREST.

Format: SEARCH subcommand

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY WINDOW CONSTRUCT RECOVER

* **SEARCH ALL**
  Search for any feature matching current SELECT constraints.

  Format: SEARCH ALL

* **SEARCH DELETED**
  Search for feature in limbo. Same as SEARCH ALL but finds only deleted features.

  Format: SEARCH DELETED

* **SEARCH FSN**
  Search for feature with given FSN, optionally in a particular map.

  Format: SEARCH FSN fsn [map]

* **SEARCH LAST**
  Search for the last feature that was found (using FIND or SEARCH), constructed, or edited, and make it the found feature again. A SEARCH NEXT sequence is unaffected by this command.

  Format: SEARCH LAST

* **SEARCH NEAREST**
  Search for the nearest feature matching current SELECT constraints. This command is similar to SEARCH ALL (qv), but the features are found in strict order of increasing distance. The cursor will snap onto a point of a feature if one is within the current find tolerance. SEARCH NEAREST is likely to take longer than SEARCH ALL, especially at large distances from the cursor.

  Format: SEARCH NEAREST

* **SEARCH NEXT**
  Continue current search for next candidate (default).

  Format: SEARCH NEXT
  or SEARCH

* **SEARCH TEXT**
  Search for text feature containing given sub-string.
  If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

  Format: SEARCH TEXT text

7.5.4 **LOCATE**

Searches whole IFF file using spiral search for feature matching given specification. LOCATE is similar to SEARCH (qv) but finds only the first matching feature. LOCATE may be used in the middle of a SEARCH NEXT sequence without destroying the search context.

Format: LOCATE subcommand

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY WINDOW CONSTRUCT RECOVER

* **LOCATE ALL**
Locate any feature matching current SELECT constraints.

Format: LOCATE ALL
or   LOCATE

* **LOCATE DELETED**
Locate feature in limbo. Same as LOCATE ALL but finds only deleted features.

Format: LOCATE DELETED

* **LOCATE FSN**
Locate feature with given FSN, optionally in a particular map.

Format: LOCATE FSN fsn [map]

* **LOCATE TEXT**
Locate text feature containing given sub-string.
If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

Format: LOCATE TEXT text

7.6 **Construction Commands**

7.6.1 **START**

Starts line or symbol feature construction using current attribute set as selected by GET (qv) and modified by SET (qv). If a construction is already started then just adds the given point.
The construction is finished by giving the END command (qv) at the final position. In the case of symbol string features (graphical type 11) features, START then END without moving the cursor will construct a single point feature.
To create a text feature use the TEXT command (qv).
In SETUP state, the START command is used to digitise a corner point.

Format: START

Valid in states READY LINE CIRCLE TEXT SYMBOL WINDOW CONSTRUCT SETUP

7.6.2 **CURVE**

Adds data point to construction but sets curve interpolation on either side of given point.

Format: CURVE

Valid in state CONSTRUCT

7.6.3 **INVISIBLE**

Adds data point to construction with an invisible segment up to the given point.

Format: INVISIBLE

Valid in state CONSTRUCT

7.6.4 **ARC**

The next constructed feature is to be a generated arc.

Format: ARC subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL

* **ARC CENTRED**
  Construct generated arc.
  Defined by the centre point and two points on the circumference

  Format: ARC CENTRED

* **ARC CIRCUM**
  Construct generated arc.
  Defined by three points on the circumference

  Format: ARC CIRCUM

7.6.5 CIRCLE

The next constructed feature is to be a generated circle.

Format: CIRCLE subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL

* **CIRCLE CENTRED**
  Construct generated circle.
  Defined by the centre point and a point on the circumference

  Format: CIRCLE CENTRED

* **CIRCLE CIRCUM**
  Construct generated circle.
  Defined by three points on the circumference

  Format: CIRCLE CIRCUM

7.6.6 POLARC

The next constructed feature is to be a generated polygon arc. The integer gives the number of sides.

Format: POLARC subcommand integer

eg   POLARC CENTRED 5

Valid in states READY LINE CIRCLE TEXT SYMBOL

* **POLARC CENTRED**
  Defined by the centre point and two points on polygon arc.

  Format: POLARC CENTRED integer

* **POLARC CIRCUM**
  Defined by three points on polygon arc

  Format: POLARC CIRCUM integer

7.6.7 **POLYGON**

The next constructed feature is to be a generated polygon. The integer gives the number of sides.

Format: POLYGON subcommand integer

eg POLYGON CENTRED 5

Valid in states READY LINE CIRCLE TEXT SYMBOL

* **POLYGON CENTRED**
  Defined by the centre point and a point on the polygon.

  Format: POLYGON CENTRED integer

* **POLYGON CIRCUM**
  Defined by three points on polygon.

  Format: POLYGON CIRCUM integer

7.6.8 **RECTANGLE**

The next constructed feature is to be a generated rectangle.

Format: RECTANGLE subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL

* **RECTANGLE DIAGONAL**
  Construct a generated rectangle by digitising one corner, a point on an adjacent side and the corner diagonally opposite the first one.

  Format: RECTANGLE DIAGONAL

* **RECTANGLE SIDE**
  Construct a generated rectangle by digitising two adjacent corners and a point on the opposite side.

  Format: RECTANGLE SIDE

7.6.9 **CLOSE**

Closes a feature currently being constructed to form a loop. The cursor is moved to the correct position ready for the END command. If given after the BRIDGE command, and the bridge starts at one end of a feature, then the CLOSE command will locate the other end of the original feature, and END will generate a closed feature from the original plus the bridge. Use END without further cursor movement to create a closed feature.

Format: CLOSE subcommand

Valid in state CONSTRUCT

* **CLOSE NORMAL**
  Positions cursor on first point of construction (default).
  An END command will then result in a closed feature.

  Format: CLOSE NORMAL
  or CLOSE

* **CLOSE SQUARE**
  Extends the current segment or construction to a point such that the line back to the start point forms a right angle. A point is added here, and the cursor is left on the first point of feature. Useful, in conjunction with FORCE CORNER (qv), for adding the final points of a squared construction.

  Format: CLOSE SQUARE

7.6.10 **INCLUDE**

Allows all or some of an existing feature to be included in the current construction. The original feature remains unaltered.

Format: INCLUDE subcommand

Valid in states LINE CONSTRUCT

* **INCLUDE PART**
  Includes the part of the found feature from the current cursor position to the point when the next END command is given.
  After the INCLUDE PART command has been given the cursor is constrained on the found feature until an END (or an ABANDON) command.

  Format: INCLUDE PART

* **INCLUDE WHOLE**
  Includes the whole of the found feature in the current construction (default).
  The cursor must be on one end of the found feature, and after this command it will be located at the other end.

  Format: INCLUDE WHOLE
  or INCLUDE

7.6.11 **FOLLOW**

Add points to a construction by continuously polling the position from the specified device.

Note that not all versions of LITES2 support the devices that can be specified by this command.

Filtering of the points is achieved using a modified form of the BUNCH algorithm that is used by the FILTER command, the main difference being that while the BUNCH filter algorithm computes new master points, the algorithm used by FOLLOW outputs points that have been digitised.

The rate at which the devices are polled and the tolerances for this filtering are set by the TOLERANCE FOLLOW command.

The filtering algorithm uses tolerances related to chords on the incoming point strings. The first point received from the input stream is accepted as a master point. As subsequent points are received, they are added to the backlog buffer, unless one of the following conditions are met.

1. if the point is less than the minimum distance from the last master point or the last point in the buffer, it is ignored.

2. if the maximum distance tolerance is not set to 0.0 (infinity) and the distance from the last master point to the current point is greater than this tolerance, then the last point in the backlog buffer is output as a master point, the buffer is emptied and the current point is added as the first point in the buffer.

3. if the perpendicular distance from any of the points in the backlog buffer to the line joining the last master point to the current point exceeds the lateral tolerance, then the last point in the backlog buffer is output as a master point, the buffer is emptied and the current point is added as the first point in the buffer.

4. if the backlog buffer is full, then it is compressed to leave the points farthest to the right and to the left of the line between the last master point and the current point, before the current point is added to the backlog buffer.

See the FILTER command for more information on the effect of altering the tolerances.

LITES2 will complete any processing that it is engaged in when it is time to poll the device for a coordinate, before reading the position from the device and doing the above filtering. This may mean that on busy systems, FOLLOWing may be somewhat uneven. This problem may be alleviated by reducing the amount of refresh drawing that LITES2 has to do by giving the REFRESH VERTICES command.

FOLLOWing mode is left by giving the END or START command. FOLLOWing will also be abandoned if an error occurs while retrieving a coordinate. If many of these errors occur then the following time interval is probably set too small.

Format: FOLLOW subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL WINDOW CONSTRUCT

* **FOLLOW DSR**
  See the description of the command FOLLOW SD (stereo digitiser).

Format: FOLLOW DSR

* **FOLLOW SCREEN**
  Gets coordinates from the position of the workstation cursor on the screen. Note that this is not the LITES2 cursor, but the cursor that is controlled by the movement of the mouse or bitpad.

  Format: FOLLOW SCREEN

* **FOLLOW SD**
  Gets coordinates from the stereo digitising instrument, if it has been initialised.

  Format: FOLLOW SD

* **FOLLOW TABLE**
  Gets coordinates from the digitising table (default).

  NOTE: This command is not available when the table is accessed through a LSL MUART (ie ENABLE MONITOR must have been given). The table must also be of a type that responds to requests from the host computer. In particular, if using an ALTEK 90 controller, the "ENABLE HOST" switch must be set to the on position.

  When using non-ALTEK tables, it is possible to specify the string that is used to inquire the position from the table, in the shared image that is used to decode the table string. See the documentation on the TABLE MONITOR for details of this decoding routine, or the example supplied in LSL$PUBLIC_ROOT:[TABLE.EXE]EXAMPLE.FOR. This routine can be compiled and linked using the command file DECODE.COM in the same directory.

  Format: FOLLOW TABLE

7.7 **Constraint Commands**

The following commands are available to constrain the movement of the cursor along lines or features. While the cursor is constrained, the FIND command will locate intersections of the constraint with linear features.

7.7.1 **FORCE**

Force constraint on cursor movement, or move cursor on to a surface or a line. The FREE command may be used to restore free cursor movement. Format: FORCE subcommand Valid in states LINE TEXT SYMBOL MODIFY CIRCLE EDIT CONSTRUCT

* **FORCE ANGLE**
  Next line segment to be at given angle (measured anti-clockwise from horizontal).
  If given after FORCE LINE or FORCE DISTANCE, then the cursor position is fixed at the given angle and distance.
  Note that movement is possible in either direction along constraining line.

  Format: FORCE ANGLE real

* **FORCE BEARING**
  Next line segment to be at given bearing (measured clockwise from grid north).
  If given after FORCE LINE or FORCE DISTANCE, then the cursor position is fixed at the given angle and distance.
  Note that movement is possible in either direction along constraining line.

  Format: FORCE BEARING real

* **FORCE CORNER**
  A corner of the given angle (default 90 degrees) at the current point.
  If given after FORCE LINE or FORCE DISTANCE, then the cursor position is fixed at the given angle and distance.
  Note that movement is possible in either direction along constraining line.

  Format: FORCE CORNER [real]

* **FORCE DISTANCE**
  Force the total length of a construction to be a given distance.
  The distance is specified in IFF units (unless a UNITS command has been given), and must be greater than the current length of the construction.
  The cursor is constrained onto a circle around the previous point. If given when a linear constraint is already in force, for example FORCE ANGLE, then the cursor position is fixed at the given angle and distance.

  Format: FORCE DISTANCE real

* **FORCE EDGE**
  Moves cursor onto currently defined base for edgematching.

  Format: FORCE EDGE

* **FORCE FLAT**
  Moves cursor onto horizontal plane through the point that the cursor was moved to when the particular editing command was given. For example, if the editing command EXTEND is given, at any time before a subsequent ABANDON or END command, FORCE FLAT will make the Z value of the cursor the same as the height of the point being extended.

  This command is only valid if Z has been ENABLED.

  Format: FORCE FLAT

  Valid in state EDIT

* **FORCE LINE**
  Force the length of the current segment of a construction.
  The length is specified in IFF units (unless a UNITS command has been given).
  The cursor is constrained onto a circle around the previous point. If given when a linear constraint is already in force, for example FORCE ANGLE, then the cursor position is fixed at the given angle and distance.

  Format: FORCE LINE real

* **FORCE ORTHOGONAL**
  Horizontal or vertical line only (not yet implemented).

  Format: FORCE ORTHOGONAL

* **FORCE PARALLEL**
  Parallel to the current segment of the found feature.
  The cursor is constrained to the right of the found feature if the command argument is positive, and to the left if the argument is negative. If no argument is given, the cursor position is used.

  Format: FORCE PARALLEL [real]

* **FORCE PERPENDICULAR**
  Perpendicular to the current segment of the found feature.
  The cursor position is used to determine the placement of the perpendicular. The initial position of the cursor is to the right of the found feature if the command argument is positive, and to the left if the argument is negative.
  Note that movement is possible in either direction along constraining line.

  Format: FORCE PERPENDICULAR [real]

* **FORCE RADIAL**
  On a radius of the found feature.

Format: FORCE RADIAL

* **FORCE SLOPE**
  Moves cursor onto the plane through the 2 points used to determine the cursor position and any constraint when the editing command took effect. For example, if the editing command EXTEND is given, at any time before a subsequent ABANDON or END command, FORCE SLOPE will move the the cursor so that it lies on the plane through the last two points of the feature.

  This command is only valid if Z has been ENABLED.

  Format: FORCE SLOPE

  Valid in state EDIT

* **FORCE TANGENTIAL**
  On a tangent to the found feature.

  Format: FORCE TANGENTIAL

7.7.2 **FREE**

Frees cursor from constrained movement.

Format: FREE

Valid in states READY LINE CIRCLE EDIT MODIFY CONSTRUCT

7.7.3 **ON**

Constrains cursor to move only on found feature. FIND will then locate intersections of the found feature with other linear features. The feature intersected with is merely used to locate the cursor, and does not become the found feature.

Format: ON

Valid in states LINE CIRCLE EDIT MODIFY CONSTRUCT

7.8 **Positioning Commands**

7.8.1 **POSITION**

Positions cursor to given IFF coordinates.

If all the arguments are omitted the cursor is positioned to the centre of the screen.

```
Format:  POSITION  [ x y [z] ]

eg       POSITION 100 121.6
or       POSITION
```

Valid in all states except INITIAL

7.8.2 **ABSOLUTE**

Positions cursor to given coordinates. The coordinates are specified as full projection coordinates i.e. coordinates that include any origin offset specified in the IFF files.

If all the arguments are omitted, the cursor is positioned to the centre of the screen.

```
Format:  ABSOLUTE  [ x y [z]]

eg       ABSOLUTE 327100.34 6734121.62
or       ABSOLUTE
```

Valid in all states except INITIAL

7.8.3 **GEOGRAPHICAL**

Positions cursor to given latitude and longitude.

This command is only valid if at least one map read in has valid type 2 map descriptor, specifying a valid projection.

The latitude and longitude are specified in degrees, minutes and seconds.

If the arguments are omitted, the cursor is positioned to the centre of the screen.

Geographical conversions make use of a shared image pointed at by the logical name LSL$LITES2_GEOG_ROUTINES. This image is supplied by Laser-Scan. It is called LSL$EXE:LITES2GEOGSHR.EXE.

```
Format:  GEOGRAPHICAL [ lat long ]

eg       GEOGRAPHICAL 56 30 00.000   -3 20 00.000
         GEOGRAPHICAL 56 30 00.000N   3 20 00.000W
```

```
GEOGRAPHICAL 56 30N         3 20W
GEOGRAPHICAL 56.5N          3.333333333W
GEOGRAPHICAL  3 20 00.000W   56 30 00.000N
```

these examples are all valid angles that will position the cursor to the same point.

Valid in all states except INITIAL

7.8.4 **LATLONG**

Positions cursor to given latitude and longitude, where the latitude and longitude are specified as double precision numbers.

This command is only valid if at least one map read in has valid type 2 map descriptor, specifying a valid projection.

If the arguments are omitted, the cursor is positioned to the centre of the screen.

Geographical conversions make use of a shared image pointed at by the logical name LSL$LITES2_GEOG_ROUTINES. This image is supplied by Laser-Scan. It is called LSL$EXE:LITES2GEOGSHR.EXE.

Format: LATLONG [lat long]

eg   LATLONG 56.0 -3.0

Valid in all states except INITIAL

7.8.5 **SHEET**

Positions cursor to given coordinates (given in sheet mm).

If the arguments are omitted, the cursor is positioned to the centre of the screen.

Format: SHEET [x y]

eg   SHEET 352.6 222
or   SHEET

Valid in all states except INITIAL

7.8.6 **FIRST**

Positions to first point, AC, TC, CH or text component.
Only operates on composite texts if there is no other linear found object.
If in MODIFY state, goes into MODIFY (part) state.

Format: FIRST

Valid in states LINE CIRCLE EDIT TEXT MODIFY ON CONSTRUCT AC

7.8.7 **LAST**

Positions to last point, AC, TC, CH or text component.
Only operates on composite texts if there is no other linear found object.
If in MODIFY state, goes into MODIFY (part) state.

Format: LAST

Valid in states LINE CIRCLE EDIT TEXT MODIFY ON CONSTRUCT AC

7.8.8 **MIDDLE**

Positions cursor midway between 2 points

Format: MIDDLE

Valid in states LINE CIRCLE EDIT MODIFY ON CONSTRUCT

7.8.9 **NEXT**

Positions to next point, AC, TC, CH or text component.
Only operates on composite texts if there is no other linear found object.
If in MODIFY state, goes into MODIFY (part) state.

Format: NEXT

Valid in states LINE CIRCLE EDIT TEXT MODIFY ON CONSTRUCT AC

7.8.10 **PREVIOUS**

Positions to previous point, AC, TC, CH or text component.
Only operates on composite texts if there is no other linear found object.
If in MODIFY state, goes into MODIFY (part) state.

Format: PREVIOUS

Valid in states LINE CIRCLE EDIT TEXT MODIFY ON CONSTRUCT AC

7.8.11 **THIS**

Positions to current AC, TC, CH or text component.
If in TEXT or MODIFY state, goes into MODIFY (part) state.

Format: THIS

Valid in states TEXT MODIFY AC

7.8.12 **POINT**

Positions cursor on the given point of the found feature.

Format: POINT integer

Valid in states LINE CIRCLE EDIT TEXT MODIFY ON CONSTRUCT

7.8.13 **FRACTION**

Positions cursor fractionally between 2 points

Format: FRACTION real

eg   FRACTION 0.4

Valid in states LINE CIRCLE EDIT MODIFY ON CONSTRUCT

7.8.14 **DISTANCE**

Positions the cursor a given distance along a feature. The distance is specified in IFF units (unless a UNITS command has been given). The distance may be measured from either end of the feature, or from the current cursor position.

Format: DISTANCE [subcommand] real

Valid in states LINE CIRCLE EDIT MODIFY ON CONSTRUCT

* **DISTANCE [ABSOLUTE]**
  The cursor is positioned to the given distance from the start of the feature (or backwards from the end if the distance is negative). An attempt to move beyond the end of the feature will result in the cursor being positioned at the end.

  Format: DISTANCE [ABSOLUTE] real

  eg   DISTANCE 10.0
  or   DISTANCE ABSOLUTE 10.0

* **DISTANCE RELATIVE**
  The cursor is positioned to the given distance along the feature from the current cursor position. The direction of movement is towards the end of the feature if the distance is positive.
  An attempt to move beyond the end of the feature will result in the

cursor being positioned at the end.

```
Format:  DISTANCE RELATIVE  real

eg       DISTANCE RELATIVE 10.0
```

7.9 **Editing Commands**

7.9.1 **COPY**

Creates a copy of a feature and allows changes to be made to it. Several changes can be made using CHANGE, DELETE, MOVE, OFFSET, and REVERSE. An END command completes the operation.
Compare with MODIFY command.

Format: COPY subcommand

Valid in states LINE TEXT SYMBOL

* **COPY PART**
  Copies part of line features. The old feature is unaffected. See MODIFY PART for details of specifying part features.

  Format: COPY PART

* **COPY WHOLE**
  Copies the whole feature (default).

  Format: COPY WHOLE
  or COPY

7.9.2 **MODIFY**

Allows multiple changes to be made to a feature. Several changes can be made using CHANGE, DELETE, MOVE, OFFSET, and REVERSE. An END command completes the operation. If given without COPY or MODIFY, then these commands in general behave as though MODIFY WHOLE had been used, but for line features the operation is completed immediately.
Compare with COPY command.

Format: MODIFY subcommand

Valid in states LINE TEXT SYMBOL CIRCLE

* **MODIFY PART**
  Modifies part of line features. The initial cursor position is taken as one end of the part to be modified. The program enters ON state and the cursor is constrained to move on the line. Commands may now be given to alter the part feature. After moving the cursor to the end of the part, the END command completes the operation. If no commands to alter the part are given, the effect is merely to split the feature into (up to) three parts.

  Format: MODIFY PART

* **MODIFY WHOLE**
  Modifies the whole feature (default).

  Format: MODIFY WHOLE

or MODIFY

7.9.3 **DEPOSIT**

Deposits the text or symbol that is currently being modified. The feature may still be further modified and DEPOSITed again or the original operation can be ended with the appropriate number of END commands. Alternatively, ABANDON may be used to terminate the original operation.
A new feature serial number is automatically generated for features that are deposited.

Format: DEPOSIT

Valid in state MODIFY

7.9.4 **BRIDGE**

Replaces part of a feature, or adds additional points to the end of a feature. The part of the feature removed may be RECOVERed and possibly JOINed up again, if BRIDGE is used in error.

The sequence of commands involved will normally be
FIND
PREVIOUS/NEXT/FRACTION etc. as needed to get to first point of change
BRIDGE
START at all the new points of the changed part.
FIND again to locate last point of change
END

If the BRIDGE command is given while on the end of a feature, then it is not necessary to FIND the feature again before giving the END command. The additional points are added to the end of the feature.

If the END command if given while on the end of a feature, then this point is **not** included in the bridge. This allows the end section of a feature to be replaced completely.

A common mistake is to bridge all the way from one end of a feature to the other, hoping to make a closed loop. Instead, the original feature is deleted and replaced by the bridge. To get the desired effect, use the CLOSE command (qv) - this will automatically locate the opposite end of the feature from where the bridge was started, and END will then produce a closed loop.

Format: BRIDGE

Valid in state LINE

7.9.5 **TAKE**

Transfers attributes from the current attribute set to the found feature.

Format: TAKE subcommand

Valid in states LINE CIRCLE TEXT SYMBOL

* **TAKE AC**
  Transfers the ancillary codes (ACs) from the current attribute set to the found feature (in addition to any it may have already).

  Format: TAKE AC

* **TAKE ATTRIBUTES**
  This command is only valid if the cursor is on a point.
  Transfers the point attributes from the current attribute set to the current point of the found feature (in addition to any it may have already).

  Format: TAKE ATTRIBUTES

7.9.6 **CHANGE**

Changes characteristics of found feature. For linear features, the change will be completed immediately, unless a COPY or MODIFY command has been given first. For text or symbol features, MODIFY state will be entered, allowing other changes to be made until END is given.

When dealing with composite texts and in MODIFY (part) state, then only the characteristics of the current text component will be altered; when in MODIFY (or TEXT) state, all the component texts will take the specified attribute.

Format: CHANGE subcommand

Valid in states LINE CIRCLE TEXT SYMBOL MODIFY ON

* **CHANGE CATEGORY**
  Changes category field for text.

  Format: CHANGE CATEGORY integer

* **CHANGE CC**
  Changes component code of current text component.

  Format: CHANGE CC integer

* **CHANGE CROSSREF**
  Changes text cross reference (not yet implemented).

  Format: CHANGE CROSSREF integer

* **CHANGE DIRECTION**
  Changes "reversed feature flag" of found feature (not yet implemented). See REVERSE to change the direction of digitising of a feature.

  Format: CHANGE DIRECTION

* **CHANGE EDITED**
  Changes the edit flag of the found feature to the specified setting. This is normally set automatically as a result of editing, and is used for example in SELECT EDITED.

  Format: CHANGE EDITED integer
  eg.

  ```
  CHANGE EDIT 1    sets the flag
  CHANGE EDIT 0    clears the flag
  ```

* **CHANGE FC**
  Changes feature code of found feature.

  Format: CHANGE FC integer

* **CHANGE FSN**
  Changes feature serial number of found feature. If no argument is given a new FSN is generated automatically.

  Format: CHANGE FSN [integer]

* **CHANGE HEIGHT**
  Changes height of found text feature (mm).

  Format: CHANGE HEIGHT real

* **CHANGE LAYER**
  Changes layer of found feature.

  Format: CHANGE LAYER integer

* **CHANGE LOCATION**
  Changes text location field for text.

  Format: CHANGE LOCATION integer

* **CHANGE MAP**
  Changes map of found feature.

  Format: CHANGE MAP integer

* **CHANGE PC**
  Changes process code of found feature

  Format: CHANGE PC integer

* **CHANGE PSIZE**
  Changes text height in points.

  Format: CHANGE PSIZE integer

* **CHANGE STYLE**
  Changes text style for text.

  Format: CHANGE STYLE integer

7.9.7 **DELETE**

Deletes found feature, AC, TC or CH

Format: DELETE subcommand

Valid in states LINE CIRCLE TEXT SYMBOL AC

* **DELETE PART**
  Deletes part of a line feature. Move to end of part then END.
  This command is equivalent to:
  MODIFY PART
  DELETE
  See MODIFY PART command for details.

  Format: DELETE PART

* **DELETE WHOLE**
  Deletes a whole feature, or an AC, TC, CH (default). For features, the feature is marked as deleted and remains in limbo. It may be found only using the RECOVER command (qv), or SEARCH DELETED (qv).
  The feature will finally be deleted and removed from the IFF file when the EXIT command is given.

  Format: DELETE WHOLE
  or   DELETE

7.9.8 **EDIT**

Amends a single data point or line segment.

Format: EDIT subcommand

* **EDIT ATTRIBUTE**
  Edits the specified attribute of the current point.
  If no value is given, then the specified attribute is deleted from the point, otherwise the specified attribute and value is added to the point or, if the point already has the attribute then its value is updated.

  The attribute code can either be an integer, or the corresponding name defined by Laser-Scan or one defined by the user in his FRT.

  See the section on LITES2 command language for details of the format of the value for this type of command argument.

NOTE

If the logical name LSL$IFF_OUTPUT_REVISION is not set to 1, any point attributes (apart from Z) will be lost on completion of editing. Points with the attribute Z will produce ZS entries in the IFF file, rather than ST entries.
See the IFF user guide for more information on IFF files and LSL$IFF_OUTPUT_REVISION

Format: EDIT ATTRIBUTE attribute [value]

Valid in states LINE SYMBOL TEXT

* **EDIT CP**
Edits the specified control point of the specified map. If no map is specified, the control point in map 1 is edited.

The x and y values are interpreted as IFF units (unless a UNITS command has been given).

The new value will be used when the SETUP AGAIN command is next given.

```
Format:  EDIT CP [map] corner x y

         where corner = NW
                      = SW
                      = SE
                      = NE,

         map is an integer and
         x and y are real numbers

eg       EDIT CP 2 NW 0.0 1000.0
```

Valid in state READY

- NE
  Edit the North East (upper right) control point. This is the last point in the CP entry in the IFF file.

- NW
  Edit the North West (upper left) control point. This is the first point in the CP entry in the IFF file.

- SE
  Edit the South East (lower right) control point. This is the third point in the CP entry in the IFF file.

- SW
  Edit the South West (lower left) control point. This is the second point in the CP entry in the IFF file.

* **EDIT POINT**
  Edits the position of a data point (default). The current point is attached to the cursor, and can be moved around until the END command is given.

  Format: EDIT POINT
  or EDIT

  Valid in state LINE

* **EDIT VISIBILITY**
  Edits the visibility of a line segment.
  The cursor must be positioned between points. The line segment is set visible (1) or invisible (0).

  Format: EDIT VISIBILITY integer

  Valid in state LINE

7.9.9 **EXTEND**

Extends (or shortens) the end segment of a line feature.
The cursor is constrained to move in either direction along the final segment of the feature until END is given.
Often used in combination with FIND to extend one line until it exactly touches another. If FREE is given, the effect becomes identical to EDIT POINT.

Format: EXTEND

Valid in state LINE

7.9.10 **INSERT**

When in LINE state adds a data point.
A new point is created, attached to the cursor, and can be moved around until the END command is given.

If in CONSTRUCT state, then an intersected point can be inserted in the construction. The effect is shown in the following diagram:

```
       1              2              I
----X-------------X- - - - - - - X       The INSERT command is given at
                                 !       point 2. INVISIBLE, START or
                                 !       END commands are given at point
                                 ! 3     3 and 4, with the result that
                                 X       points 2 and 3 are replaced by
                                 |       point I.
                                 |
                                 |       If an INVISIBLE command is
                                 | 4     given at point 3, the vector
                                 X       1 - I is invisible; if an
                                 |       INVISIBLE command is given
                                 |       at point 4, the vector I - 4
                                 |       is invisible.
```

Format: INSERT

Valid in states LINE CONSTRUCT

7.9.11 **LOOP**

Makes the found feature a loop (closed feature). Both end points are attached to the cursor, and may be moved around until END is given.

Format: LOOP subcommand

Valid in states LINE CIRCLE

* **LOOP EXTEND**
  Extends first and last spans of feature to new meeting point.

  Format: LOOP EXTEND

* **LOOP NORMAL**
  Moves first and last points to their mean position (default).

  Format: LOOP NORMAL
  or LOOP

7.9.12 **OFFSET**

Generates a new feature parallel to the found feature. The offset distance is specified in IFF units (unless a UNITS command has been given). If the old feature is to be replaced by the offset feature, then give the MODIFY command first.

For linear features the new feature is offset to the right if the command argument is positive, and to the left if the argument is negative. For whole feature operations the argument need not be given, and the cursor position is used to calculate the offset distance. To offset part of a feature, give the OFFSET command after COPY PART or MODIFY PART (qv) in which case OFFSET is not allowed after REVERSE, MOVE, FILTER or TRANSFORM.

For texts and symbols the new feature is offset below if the command argument is positive and above if it is negative. If no argument is given, then the feature is offset by a fixed proportion of the height of the text or symbol. This value can be set by the TOLERANCE OFFSET command. If it is negative, then the OFFSET command without an argument will offset the feature above the original.
Any rotating, moving, stretching or aligning of texts or symbols is lost after giving this command.

Format: OFFSET [real]

Valid in states LINE CIRCLE ON

7.9.13 **MOVE**

Moves a feature to a new position.
This command attaches the feature to the cursor and allows it to be moved around until END is given. For line features, the effect is as if MODIFY WHOLE had been given first. To retain the old feature, use COPY WHOLE. To move part of a feature, use COPY PART or MODIFY PART, in which case the part will only be attached to the cursor after END is given to delimit the part. In this case, MOVE is not allowed after OFFSET, FILTER or TRANSFORM.
For text or symbol features, MODIFY state is entered and the feature is attached to the cursor until END is given. A further END is required to complete the edit, and leave MODIFY state.

Format: MOVE

Valid in states LINE TEXT SYMBOL MODIFY ON

7.9.14 **REMOVE**

Deletes a data point.
The current point is removed from the feature.

Format: REMOVE

Valid in states LINE CONSTRUCT

7.9.15 **REVERSE**

Digitising direction of the found feature is reversed. See MOVE for details of operations on part features. This command is not allowed after OFFSET, FILTER or TRANSFORM.

Format: REVERSE

Valid in states LINE ON

7.9.16 **ORIENT**

Moves the found (linear) feature onto the previously defined orienting base. The selected edge of the found feature is moved onto the orienting base, and the rest of the feature is moved correspondingly. The selected edge is moved so that its centre is projected perpendicularly onto the base.
If the feature has been found by a point, and not between points, then the edge out of that point that makes the smaller angle with the base is used.
The oriented feature (or part of it when dealing with long features) is displayed in refresh and EDIT state is entered. To accept the oriented feature enter END; otherwise ABANDON will cancel the operation.
After orienting, the original feature is deleted but can be recovered.

Format: ORIENT

Valid in state LINE

7.9.17 **TRANSFORM**

Sets up and uses an orthogonal transformation. This is a transformation of the form:

X = ax - by + c1
Y = bx + ay + c2

This applies a rotation, a scale change and a translation to points in the current coordinate system (x,y) to give the points in the target coordinate system (X,Y).

The transformation can either be set up by giving the coordinates of two points in both systems, using the TRANSFORM FROM and TRANSFORM TO commands, or by specifying the rotation, scale factor, point of rotation and translation explicitly, using the TRANSFORM COEFFICIENTS command.

Features and regions can be transformed using the TRANSFORM FEATURE and TRANSFORM REGION commands.

NOTE

It is possible to transform features so that they lie outside the limits of the map. These features cannot be displayed or found, but they may be located using the LOCATE or SEARCH commands (qv).

Format: TRANSFORM subcommand

* **TRANSFORM AC**
  This is an alias for TAKE AC (qv), and is included here for historical reasons. The command will be withdrawn in a future release of LITES2.

* **TRANSFORM COEFFICIENTS**
  Define the transformation, by giving the angle, scale factor, rotation point and required x and y translations.
  The angle is given in degrees.
  The scale factor must be in the range .001 to 1000
  The rotation point is the point about which the rotation takes place.

  The transformation is carried out in the order:

      rotation
      scale change
      translation

  The coordinates of the rotation point and the X and Y translations are in IFF units (unless a UNITS command has been given).

  Format: TRANSFORM COEFFICIENTS angle scale x y dx dy

  Valid in states READY LINE CIRCLE ON TEXT SYMBOL MODIFY

* **TRANSFORM FEATURE**
  Transform the current feature according to the current transformation parameters. (default)

  It is possible to transform part features if the MODIFY PART or COPY PART commands (qv) have been given. In this case TRANSFORM is not allowed after REVERSE, MOVE, FILTER or OFFSET.

  The height or point size of texts will be altered if HEIGHT is enabled, but this can only be done to the nearest .01 mm (if POINT is disabled) or to the nearest valid point size. In these cases subsequent editing of the feature may be required.

  The size and orientation of scaled symbols will be altered, but only the orientation of oriented symbols will be altered, their size being retained. Unoriented symbols will have their position altered, but will be drawn horizontal and will be the size specified in the FRT.

  Format: TRANSFORM FEATURE
  or TRANSFORM

  Valid in states LINE CIRCLE ON TEXT SYMBOL MODIFY

* **TRANSFORM FROM**
  Define the current coordinate system, by giving the coordinates of two points. The two points must define a vector. If the command TRANSFORM TO has been given, then the transformation between the two coordinate systems will be set up.

  The coordinates are in IFF units (unless a UNITS command has been given).

Format: TRANSFORM FROM xa ya xb yb

Valid in states READY LINE CIRCLE ON TEXT SYMBOL MODIFY

* **TRANSFORM REGION**
  Transform the specified region according to the current transformation parameters.

  Format: TRANSFORM REGION reg

  Valid in states READY LINE CIRCLE ON TEXT SYMBOL MODIFY

* **TRANSFORM TO**
  Define the target coordinate space, by giving the coordinates of two points. The two points must define a vector. If the command TRANSFORM FROM has been given, then the transformation between the two coordinate systems will be set up.

  The coordinates are in IFF units (unless a UNITS command has been given).

  Format: TRANSFORM TO xa ya xb yb

  Valid in states READY LINE CIRCLE ON TEXT SYMBOL MODIFY

7.9.18 **FILTER**

Filters the points in an entity according to the BUNCH filtering algorithm.

The BUNCH filter uses tolerances related to chords on the incoming point strings. The filter performs a least squares fit through all the existing data points, and when a point lies more than the lateral threshold distance from the trend line it is considered to be a provisional master point. A new fit is then conducted forwards from the last master point, until again the threshold deviation is exceeded. The last provisional point is taken to be the next master point and the intervening points are rejected. The process is repeated until the end of the line is reached. If the lateral threshold distance is large, it will rarely be exceeded and many points will be thrown away.

The number of points which are kept as master points or are thrown away may be additionally controlled by the minimum and maximum separation.

The minimum separation is the shortest distance allowed between successive master points along the line. If this is set to a large value, more points will be thrown away giving increasingly angular linework.

The maximum separation is the distance travelled along the line before forcing out a master point. A large value will result in very sparse points along straight and nearly straight lines. A maximum separation of 0.0 is equivalent to one of infinity, and means that no points will be forced out on distance criteria.

The maximum separation must be greater than or equal to the minimum separation which must be greater than or equal to the lateral threshold distance.

As the BUNCH filter uses a least squares algorithm, all new points will be placed to the outside of the original curve of the line being filtered.

Format: FILTER subcommand

Valid in states LINE ON

* **FILTER FEATURE**
  Filters the found feature.

  It is possible to FILTER part features if the MODIFY PART or COPY PART commands (qv) have been given. In this case FILTER is not allowed after REVERSE, MOVE, TRANSFORM or OFFSET.
  Format: FILTER FEATURE
  or FILTER

7.9.19 **FEATURE**

Construct a feature from some entity

Format: FEATURE subcommand

Valid in states READY

* **FEATURE GEOMETRY**
  Construct a feature from the specified geometry. If the geometry has several parts, each part will produce a separate feature.

  Area type geometries will produce features with invisible lines connecting inner rings. These are in a form that can be converted back into area geometries with the GEOMETRY FEATURE command.

  The feature is constructed with the attributes in the current attribute set (and so this must be compatible with the type of the geometry).

  If Z is enabled, features will be constructed with all the points at the height of the cursor.

  Format: FEATURE GEOMETRY n

* **FEATURE REGION**
  Construct a feature from the specified region.

  The feature is constructed with the attributes in the current attribute set (and so the current feature code must be linear).

  If Z is enabled, features will be constructed with all the points at the height of the cursor.

Format: FEATURE REGION n

7.9.20 **SPLIT**

Splits a linear feature in two, or splits a text into two or more text components.

Format: SPLIT subcommand

* **SPLIT AFTER**
  This command is only valid if COMPOSITE has been enabled.

  Replaces the text component containing the specified string "text" with two components. The split is made after "text" and the current component becomes the one that does not contain this string.

  If "text" is at the end of a text component, then a warning is given; the following text component becomes the current one. This is a way of moving directly to a particular text component in a composite text feature without NEXTing or PREVIOUSing over the intervening components.

  If in MODIFY or TEXT state, then the whole text is scanned for the first occurrence of "text"; if in MODIFY (part) state then only the current text component is examined

  If the text is to have leading or trailing spaces or tabs, then it must be enclosed in double quotes.

  Format: SPLIT AFTER text

  Valid in states TEXT MODIFY

* **SPLIT APART**
  This command is only valid if COMPOSITE has been enabled.

  Splits the current composite text into two separate features, before the current text component.

  Texts can be joined together again using the JOIN command.

  This command returns LITES2 to READY state.

  Format: SPLIT APART

  Valid in states TEXT MODIFY

* **SPLIT AROUND**
  This command is only valid if COMPOSITE has been enabled.

  Replaces the text component containing the specified string "text" with three components. The splits are made before and after "text" and the current component becomes the one containing this string.

If "text" is already a complete text component, then a warning is given and this text component becomes the current one. This is a way of moving directly to a particular text component in a composite text feature without NEXTing or PREVIOUSing over the intervening components.

If in MODIFY or TEXT state, then the whole text is scanned for the first occurrence of "text"; if in MODIFY (part) state then only the current text component is examined.

If the text is to have leading or trailing spaces or tabs, then it must be enclosed in double quotes.

This command leaves LITES2 in MODIFY (part) state.

Format: SPLIT AROUND text

Valid in states TEXT MODIFY

* **SPLIT BEFORE**
This command is only valid if COMPOSITE has been enabled.

Replaces the text component containing the specified string "text" with two components. The split is made before "text" and the current component becomes the one that contains this string.

If "text" is at the beginning of a text component, then a warning is given; this text component becomes the current one. This is a way of moving directly to a particular text component in a composite text feature without NEXTing or PREVIOUSing over the intervening components.

If in MODIFY or TEXT state, then the whole text is scanned for the first occurrence of "text"; if in MODIFY (part) state then only the current text component is examined.

If the text is to have leading or trailing spaces or tabs, then it must be enclosed in double quotes.

Format: SPLIT BEFORE text

Valid in states TEXT MODIFY

* **SPLIT LINE**
The feature is split at the current cursor position (default).

Format: SPLIT LINE
or SPLIT

Valid in states LINE CIRCLE

7.9.21 **CLIP**

Divides the current feature, where it cuts the specified region boundary, to create one or more new features.

The original feature is deleted.

An error occurs if the feature does not cross the region boundary.

Format: CLIP subcommand

Valid in states LINE CIRCLE

* **CLIP CUTREGION**
  Divides the current feature into features that lie completely inside and completely outside the specified region.

  Format: CLIP CUTREGION

  eg CLIP CUTREGION 4

* **CLIP INREGION**
  Divides the current feature into features that lie completely inside the specified region.

  The parts of the original feature that lie outside the region can be recovered using the RECOVER or SEARCH DELETED commands.

  Format: CLIP INREGION

  eg CLIP INREGION 4

* **CLIP OUTREGION**
  Divides the current feature into features that lie completely outside the specified region. The parts of the original feature that lie inside the region can be recovered using the RECOVER or SEARCH DELETED commands.

  Format: CLIP OUTREGION

  eg CLIP OUTREGION 4

7.9.22 **SQUARE**

Squares the found (linear) feature, using either simple or OS algorithms. The squared feature (or part of it when dealing with long features) is displayed in refresh and EDIT state is entered. To accept the squared feature enter END; otherwise ABANDON will cancel the operation.

After squaring, the original feature is deleted but can be recovered; the squared feature has the appropriate process code set.

When the FIXED option is enabled (default) then an enhanced squaring algorithm for SQUARE PART and SQUARE WHOLE is used. The following features are then included in the squaring algorithm:

- o Points specified with the PRIVILEGE POINT command are held fixed

- o When base squaring, after all the bases have been used as data, the remaining unsquared lines are part squared.

- o Redundant points are removed from parallel lines

See also TOLERANCE command, and the chapter on squaring at the end of this manual.

Format: SQUARE subcommand

Valid in state LINE

* **SQUARE ANGLES**
  Forces all angles within angle tolerance to be right angles.

  Format: SQUARE ANGLES

* **SQUARE PART**
  Applies OS squaring algorithm using distance and length tolerances. Only sides within tolerance are squared.

  Format: SQUARE PART

* **SQUARE WHOLE**
  Applies OS squaring algorithm to all sides (default).

  Format: SQUARE WHOLE
  or SQUARE

7.10 **Joining Commands**

7.10.1 **JOIN**

- o For linear features: Merges two features into one.

  The MATCH command (qv) can be used to specify which attributes of features to be joined must match.

  Normal sequence is:

  FIND first feature at an end
  JOIN
  FIND second feature (can only find ends)
  adjust position of join if required
  END

- o For text features: If there is a found object that is a text, then the found feature is logically joined to the end of the feature being modified. The second feature retains its old position. A PARAGRAPH command will position the second feature at the end of the first.

Format: JOIN

Valid in states LINE MODIFY

7.10.2 **MATCH**

Match conditions for JOIN, TIE, MEND and EDGEMATCH.

Format: MATCH subcommand

Valid in states INITIAL READY LINE EDIT ON

* **MATCH AC**
  Ancillary codes must match. The set of AC types to match may be given as names, numbers, or ranges of numbers. The range of ACs given in this command replace any given in a previous MATCH AC command. If the command is used without giving any AC types, perhaps after a MATCH NONE command, then it just turns on AC matching with the same set of ACs as previously, or if this is the first use of the command, then ACs of types 2 3 4 & 5 (CONTOUR, HEIGHT, LH_BOUNDARY, and RH_BOUNDARY) must match.

  Format: MATCH AC [range]

  ```
  eg       MATCH AC 20-30,40-50,55,78
  or       MATCH AC LH_BOUNDARY,RH_BOUNDARY
  or       MATCH AC HEIGHT,23
  ```

* **MATCH FC**
  FC must match (default).

  Format: MATCH FC

* **MATCH FSN**
  FSN must match.

  Format: MATCH FSN

* **MATCH LAYER**
  LAYER must match (default).

  Format: MATCH LAYER

* **MATCH MAP**
  MAP must match (default).

  Format: MATCH MAP

* **MATCH NONE**
  No attributes need match, for alignment only.

  Format: MATCH NONE

* **MATCH PC**
  Process Code must match.

  Format: MATCH PC

7.10.3 **EDGEMATCH**

An appropriate licence is required to use this command.
Carry out automatic edgematching of features along a baseline defined using the BASE EDGE command (qv).
For more information about edgematching, see separate section of manual below.
Edgematching can be aborted with CTRL/C, but any features that have been altered at this point cannot be recovered. The REVIEW command (qv) may be used after EDGEMATCH to allow areas where problems occurred to be examined individually.

Format: EDGEMATCH subcommand

Valid in state READY

* **EDGEMATCH JOIN**
  EDGEMATCH and JOIN the features together

  Format: EDGEMATCH JOIN

* **EDGEMATCH TIE**
  EDGEMATCH and TIE the features together

Format: EDGEMATCH TIE

* **EDGEMATCH EXTEND**
  EXTEND or truncate features to the baseline

  Format: EDGEMATCH EXTEND

7.10.4 **REVIEW**

Runs a command file generated by EDGEMATCH (qv) which allows areas where problems occurred to be examined and amended individually.

Format: REVIEW

Valid in state READY

7.10.5 **MEND**

Merges several features into one. Acts as JOIN (qv), but jumps to other end of second feature to try to find a plausible connection. The MATCH command (qv) can be used to specify which attributes of features to be joined must match.

Normal sequence is:
FIND first feature at the end near join to second feature
MEND [MANUAL or AUTOMATIC (default)]
If operation pauses then:
FIND next feature to MEND to, adjust position of join if required, then
END

ABANDON will terminate the operation, including the feature in hand in the mended feature, but not any found feature.

Format: MEND subcommand

Valid in state LINE

* **MEND AUTOMATIC**
  Performs whole operation automatically (default). The operation will pause only if there is more than one possible candidate to MEND to. CTRL/C may be used to pause the operation.

  Format: MEND AUTOMATIC
  or MEND

* **MEND MANUAL**
  Allows operator intervention at each break.

  Format: MEND MANUAL

7.10.6 **PROPAGATE**

Smooths feature(s) on either side of point. Used during JOIN or TIE to smooth out the effects of the edit. The PROPAGATE must be given after JOIN or TIE each time an operation is performed.

Format: PROPAGATE

Valid in state EDIT (while TIEING and JOINING)

7.10.7 **TIE**

Does a simultaneous edit of the end points of two features to produce a graphical match. Acts as JOIN (qv) but leaves the two features independent.

Format: TIE

Valid in state LINE

7.11 **Text and Symbol Commands**

7.11.1 **TEXT**

Creates a text feature. MODIFY state is entered, allowing the text to be amended before being entered into the file by END or DEPOSIT. Uses the current text attribute set as defined by SET commands (qv). If the text is to have leading spaces or tabs, then it must be enclosed in double quotes.

NOTE: An implied TEXT or REPLACE command is inserted before any command that is terminated by a Control/Z (that is the CTRL and Z keys pressed together). REPLACE is used if it is valid, and otherwise TEXT is used.

If a character within the text is preceded by $, then 128 is added to the ASCII value of the character. This allows access to ASCII characters 128 to 255, provided that these are defined in the TRI file. To obtain the $ character itself, use $$. If the keyboard allows ASCII characters in the range 128-255 to be entered directly, for instance by using the Compose Character key, then this mechanism may also be used. See the FRTLIB User Guide, in the Mapping Package documentation, for details of defining a TRI file.

```
Format:  TEXT  text
or       text^Z

eg       TEXT High St.
or       High St.^Z
```

Valid in states READY LINE CIRCLE TEXT SYMBOL MODIFY

7.11.2 **LARGER**

Makes text or scaled symbol larger. For texts with height in pointsize units it uses the next available pointsize. Otherwise the text or symbol is increased in size by 10%.

Format: LARGER

Valid in states TEXT SYMBOL MODIFY

7.11.3 **SMALLER**

Makes text or scaled symbol smaller. For texts with height in pointsize units it uses the next available pointsize. Otherwise the text or symbol is decreased in size by 10%.

Format: SMALLER

Valid in states TEXT SYMBOL MODIFY

7.11.4 **MARGIN**

Shifts text to specified OS marginal position.

Format: MARGIN integer

Valid in states TEXT MODIFY

7.11.5 **ROTATE**

Rotates text or symbol in terms of angles measured anticlockwise from the horizontal.

Format: ROTATE subcommand

Valid in states TEXT SYMBOL MODIFY

* **ROTATE BY**
  Rotate the subject anticlockwise by the given angle in degrees.

  Format: ROTATE BY real

* **ROTATE CURSOR**
  Rotate the subject using the cursor (default). When the feature is suitably orientated, disconnect the cursor with an END.

  Format: ROTATE CURSOR
  or   ROTATE

* **ROTATE TO**
  Rotate subject to lie at the given angle in degrees. The angle is measured anticlockwise from the right (East).

  Format: ROTATE TO real

7.11.6 **TURN**

Rotates text or symbol in terms of bearings measured clockwise from grid north.

Format: TURN subcommand

Valid in states TEXT SYMBOL MODIFY

* **TURN BY**
  Rotate the subject clockwise by the given angle in degrees.

  Format: TURN BY real

* **TURN CURSOR**
  Rotate the subject using the cursor (default). When the feature is suitably orientated, disconnect the cursor with an END.

  Format: TURN CURSOR
  or   TURN

* **TURN TO**
  Rotate subject to lie at the given bearing in degrees. The bearing is measured clockwise from grid north.

  Format: TURN TO real

7.11.7 **ALIGN**

Define size and rotation of a scaled symbol with cursor. When symbol is suitably aligned, disconnect cursor with an END.

Format: ALIGN

Valid in states SYMBOL MODIFY

7.11.8 **STRETCH**

Define size of a scaled symbol with cursor. When symbol is of a suitable size, disconnect cursor with an END.

Format: STRETCH

Valid in states SYMBOL MODIFY

7.11.9 **BEND**

Bends a text around a linear feature.

Each character of the text string becomes an individual text component, with its locating point positioned on the linear found feature and its orientation parallel to the found feature at this point. Any characters with zero width are taken to represent diacritical marks (such as accents) and are put together with the following character in a single component.

When BENDing around circles or circle arcs, then the direction (i.e. clockwise or anticlockwise) that the text takes is determined by the original orientation of the text, and the tangent of the arc at the first point used to define the BEND.

This command is only valid if the COMPOSITE option has been enabled and the existing text consists of a single text component. (If text is already composed of multiple components, then use the COLLAPSE command to achieve a single text component)

Format: BEND subcommand

Valid in state MODIFY

* **BEND NORMAL**
  BEND the subject around the found object at the cursor position (default).
  The original justification of the text is maintained.

  Format: BEND NORMAL
  or BEND

* **BEND STRETCH**
  BEND the subject between two points of a linear found object.

  The initial cursor position is taken as the first point, and the cursor is restrained to lie on the found object. The cursor can then be moved to the second point and the END command given to complete the operation.

  Format: BEND STRETCH

7.11.10 **COLLAPSE**

This command is only valid if COMPOSITE has been ENABLED.

Converts two or more component texts into a single text component. If any of the component texts within the feature contain more than one letter, then they are assumed to be words, and a space is added between each collapsed component. If all the components consist of one letter, then no extra spaces are added (this would be the case if the composite text was produced by the BEND command).

When in TEXT or MODIFY state, all the text components are collapsed into one subtext. The resulting text takes its attributes from the first text component.

When in MODIFY (part) state the current text component has the following text component added to it. This combined text component takes its attributes from the current text component.

Format: COLLAPSE

Valid in states TEXT MODIFY

7.11.11 **PARAGRAPH**

This command is only valid if COMPOSITE has been ENABLED.

Treats a text feature as a paragraph of text, and allows word processing operations on it.

The text is reformatted as a paragraph. This paragraph is positioned with the locating point of its first component text positioned so that the first line of the formatted paragraph is as close as possible to the first line of the original feature.
The first component of the original text feature also provides the orientation and the line spacing of the paragraph. The line spacing uses the TOLERANCE OFFSET, to provide an offset for each succeeding line of text (see OFFSET command for details).

This command leaves LITES2 in MODIFY state, with the first text component as the current component.

NOTE

In all PARAGRAPH commands, any per point attributes are lost.

Format: PARAGRAPH subcommand

Valid in states TEXT MODIFY

* **PARAGRAPH FILL**
  Fill a paragraph of text so that no line is longer than the specified length. The text components are split at spaces or tabs, and spaces may be inserted if necessary when two text components are collapsed. Text components are never collapsed if they have different component codes or different heights.
  Leading and trailing spaces and tabs are stripped off text components.

  The length is specified in sheet mm, unless a UNITS command has previously been given. If no length is specified, then the length of the first component is used.

  Format: PARAGRAPH FILL [length]

* **PARAGRAPH JUSTIFY**
  Justifies a paragraph of text so that each line is the same length. It does this by putting each word in a text component of its own, and positioning them so that extra space in the line is evenly distributed between the individual components. If a line is already longer than the specified length, then it is allowed to overhang the margins of the text, so it is usual to precede this command with a PARAGRAPH FILL command.

  The last line of the paragraph is never filled.

  Leading and trailing spaces and tabs are stripped off text components.

  The length is specified in sheet mm, unless a UNITS command has previously been given. If no length is specified, then the length of the first component is used.

  Format: PARAGRAPH JUSTIFY [length]

* **PARAGRAPH NORMAL**
Tidies a paragraph of text.

Text features that have been paragraphed can be upset by several LITES2 commands. For example:

- o changing the size of a subtext means that the subtext takes up more or less space in the line, with subsequent overwriting or unwanted space

- o changing the locating code of the paragraph as a whole, where there is more than one subtext per line, leads to overwriting of these subtexts (although other lines are justified as required)

This command corrects such paragraphs, by repositioning the text components on their own lines to form a correct paragraph. The number of spaces to place between each text component may be specified (default 1). It may be useful to specify zero spaces if there are text components which form only part of a word, and hence do not require spaces around them.

Leading and trailing spaces and tabs are stripped off text components, unless the optional number of spaces is given as 0.

Format: PARAGRAPH NORMAL [spaces]

7.11.12 **WHOLE**

Changes from MODIFY (part) state to MODIFY state; i.e. change from editing individual text components to editing the whole text feature.

Format: WHOLE

Valid in state MODIFY

7.11.13 **REPLACE**

Replaces existing text or AC text with the given string. For an AC, the string may be absent, in which case the AC text is removed. If the string is to have leading spaces or tabs, then it must be enclosed in double quotes.

If editing a composite text feature as a whole, all the individual text components are replaced by one single component.

NOTE: An implied TEXT or REPLACE command is inserted before any command that is terminated by a control Z. REPLACE is used if it is valid, and otherwise TEXT is used.

Format: REPLACE text
or text^Z

Valid in states TEXT MODIFY AC

7.11.14 **SUBSTITUTE**

Substitutes new substring for existing one in text feature or ancillary code. If either the old or new text contains spaces then it should be enclosed in double quotes.

Note: if in MODIFY (part) state, then only the current text component is searched for the existing text string. If dealing with whole texts, then the first occurrence will be changed.

Format: SUBSTITUTE oldtext newtext

```
eg       SUBSTITUTE FRED JIM
or       SUBSTITUTE "Ford Prefect" "Zaphod Beeblebrox"
```

Valid in states TEXT MODIFY AC

7.12 **Attribute Commands**

7.12.1 **GET**

Copy the specified attribute set into the current attribute set. These can then be modified using the SET command (qv) and saved using the PUT command.
Attribute sets contain values of feature code (FC), process code (PC), PROCESS (for use at end of construction), and ACs.
There are 16 attribute sets.

Format: GET integer

Valid in states INITIAL READY LINE CIRCLE TEXT SYMBOL EDIT ON WINDOW AC

7.12.2 **PUT**

Copy the current attribute set into the specified attribute set.
Attribute sets contain values of feature code (FC), process code (PC), PROCESS (for use at end of construction), and ancillary codes (ACs).
There are 16 attribute sets.
See also GET and SET.

Format: PUT integer

Valid in states INITIAL READY LINE CIRCLE TEXT SYMBOL EDIT ON WINDOW AC

7.12.3 **SET**

Sets values for various attributes in the current attribute set, which is used for new constructions. Values for FC, PC, PROCESS, point attributes and ACs, can be saved by PUT, and restored by GET, whereas global attributes cannot.

Format: SET subcommand

Valid in states INITIAL READY LINE CIRCLE TEXT SYMBOL MODIFY EDIT ON WINDOW AC

* **SET AC**
  Sets an ancillary code in the current attribute set to be used for new constructions. Any number of ACs may be set. The ACs in the current attribute may be edited using the ANCILLARY command when there is no found feature. The SET FC command clears any ACs in the current attribute set.
  The type of AC can either be an integer, or the corresponding name from the FRT.

  See the section on LITES2 command language for details of the format of the value for this type of command argument.

  Text is optional. It must be enclosed in double quotes if it is to have any leading spaces or tabs.

  Format: SET AC type value [text] (as for ADD AC qv) eg SET AC PIPE 1234 pressure 19.4

* **SET ARC**
Sets the feature code to be used for the features produced when doing PART operations on full circum-circles (graphical type 5). It must represent a circum-circle arc (graphical type 4) in the current FRT.

Format: SET ARC integer

eg SET ARC 100

* **SET ATTRIBUTE**
Sets a point attribute in the current attribute set to be used for new constructions. Any number of point attributes may be set.
The attribute code can either be an integer, or the corresponding name defined by Laser-Scan or one defined by the user in the FRT file.

See the section on LITES2 command language for details of the format of the value for this type of command argument.

Point attributes are removed from the construction attribute set with the UNSET ATTRIBUTE command.

NOTE

If the logical name LSL$IFF_OUTPUT_REVISION is not set to 1, any point attributes (apart from Z) will be lost on completion of editing. Points with the attribute Z will produce ZS entries in the IFF file, rather than ST entries.
See the IFF user guide for more information on IFF files and LSL$IFF_OUTPUT_REVISION

Format: SET ATTRIBUTE code value

eg SET ATTRIBUTE Z 24.3
or SET ATTRIBUTE 93 24.3

* **SET CATEGORY**
Sets the text category for text constructions (global).

Format: SET CLASS integer

* **SET FC**
Sets the feature code for the current attribute set.
Any process and ACs in the current attribute set are cleared by this command.

Format: SET FSN integer

* **SET FSN**
Sets the feature serial number for the next constructed feature (global).
The setting is cleared by ABANDON, or by SET FSN 0.

Format: SET FC integer

* **SET HEIGHT**
Sets height of text (mm) for text constructions (global).

Format: SET HEIGHT real

* **SET INCREMENT**
Sets the increment (in mm) to be used when using the LARGER and SMALLER commands on text features (when the HEIGHT option has been enabled, and the POINT option disabled). If no argument is given to this command, then the height will be increased/decreased by 10%. This is the default setting.

Format: SET INCREMENT [real]

eg SET INCREMENT 2.5

* **SET LAYER**
Sets the layer to be used for new constructions (global).

Format: SET LAYER integer

* **SET LOCATION**
Sets the text location field for text constructions (global).

Format: SET LOCATION integer

* **SET MAP**
Sets the map to be used for new constructions (global)

Format: SET MAP integer

* **SET PSIZE**
Sets the text height in points for text constructions (global).

Format: SET PSIZE integer

* **SET PROCESS**
Sets a process for the current attribute set.
The process is a command (possibly a macro) which will be obeyed automatically on completion of any construction.
The commands invoked may for instance use SEARCH LAST (qv) to perform some editing operation on the new feature.
The process can be cleared by giving a blank string, and is cleared automatically whenever SET FC is used.

NOTE

This process is only carried out after an END command. It will not be executed if a text or symbol is DEPOSITed.

Format: SET PROCESS string

* **SET STYLE**
  Sets the text style for text constructions (global).

  Format: SET STYLE integer

* **SET TEXT**
  Sets the feature code to be used for constructing text features (global).

  Format: SET TEXT integer

7.12.4 **UNSET**

Unsets values for various attributes in the current attribute set.

Format: UNSET subcommand

Valid in states INITIAL READY LINE CIRCLE TEXT SYMBOL MODIFY EDIT ON WINDOW AC

* **UNSET ATTRIBUTE**
  If an argument is given, removes the specified attribute from the list of point attributes in the current construction attribute set.
  If no argument is given then all the attributes in the list are removed.
  The argument (attribute code) can either be an integer, or the corresponding name from the FRT.

  Format: UNSET ATTRIBUTE [code]

  ```
  eg       UNSET ATTRIBUTE Z
  or       UNSET ATTRIBUTE 93
  or       UNSET ATTRIBUTE
  ```

7.12.5 **REPEAT**

Set the values in the current attribute set to the values of the found feature. Enables future constructions to have the same attributes as the found feature. See also GET and SET.

Format: REPEAT subcommand

Valid in states LINE CIRCLE TEXT SYMBOL

* **REPEAT ATTRIBUTE**
  Sets the point attributes in the current attribute set to be the same as those of the current point of the current feature.

  This command is only valid if the cursor is on a point of a feature.

  Format: REPEAT ATTRIBUTE

* **REPEAT FEATURE**
  Sets the feature wide attributes of the current feature in the attribute set (default).

  These are (for all features) the Map, Layer, Feature Code and any ACs and (for text features) the Category, Font, Location and Text Size.

  Format: REPEAT
  or REPEAT FEATURE

7.12.6 **SELECT**

Makes feature selections by various criteria. Selections may be applied to windowing and FIND and SEARCH operations, and can be activated for output. After SELECT ALL, the first SELECT command for a given attribute implies the deselection of all non-selected values for that attribute.

Format: SELECT subcommand

Valid in states INITIAL READY SETUP

* **SELECT AC**
  Allows selection by ancillary code (AC) entries in features. The AC type may be either an integer, the corresponding name defined by Laser-Scan, or a name defined by the user in the FRT file. AC selections may be stored in up to 10 separate groups (see the SELECT ACGROUP command). A feature will be selected if its ACs satisfy the criteria in any one or more of the groups. Within a group, selections for different AC types are logically ANDed together, so a feature must satisfy all the selection criteria to be selected. For example, to select **all** features with either a type 2 or a type 3 AC, one might use the commands:

  ```
  SELECT ACGROUP 1       ! first group of AC selections (default)
  SELECT AC 2            ! want all type 2 ACs
  SELECT ACGROUP 2       ! another group of selections
  SELECT AC 3            ! and all type 3 ACs also
  ```

  If the SELECT ACGROUP commands were not used, then only features with both the AC types would be selected.

  Format: SELECT AC type [subcommand]

  o **SELECT AC type CANCEL**
    Cancels all selections based on the given AC type. The DESELECT command may not be used with CANCEL.

    Format: SELECT AC type CANCEL

  o **SELECT AC type PRESENT**
    Features must have an AC of the specified type. The value and text of the AC are not considered. PRESENT is the default subcommand for the SELECT AC command, and may therefore be omitted.

The command DESELECT AC type PRESENT implies that features must **not** have an AC of the specified type.

Format: SELECT AC type [PRESENT]

- **SELECT AC type TEXT**
  Allows selection according to the contents of the AC text. If just a text string is specified (in double quotes if it contains spaces), then features will be selected if they contain an AC of the specified type whose text contains the given string (or does not contain the string for a DESELECT command). If an inequality and value(s) are given (as for the SELECT AC type VALUE command), then the given text string within the AC text must be followed immediately by a numerical value in the selected range. The datatype of the value may only be integer. A null text string (specified by "") followed by a value or range of values indicates that the numerical value occurs at the start of the AC text.

  Format: SELECT AC type TEXT text [[inequality] val1 [val2]]

  eg SELECT AC DFAD_FADT TEXT fid 25
  Features must contain a DFAD_FADT type AC in which the text contains the string "fid" followed by a value 25.

- **SELECT AC type VALUE**
  Allows selection according to the value contained in AC entries. Values must be specified in the correct format (integer, real, date, time, character) for the data type of the AC.

```
The inequality may be any of:     =     >    >=     <    <=    <>
              with synonyms:   .EQL. .GTR. .GEQ. .LSS. .LEQ. .NEQ.
                        and:                       .LT.
```

  Inequality names may be abbreviated. If the inequality is omitted, then = is assumed. A range of values may be specified by giving two values (omitting the inequality, or specifying =), which will be taken to mean the range between and including the two values. The command may be repeated to specify additional values or ranges of values.

  Format: SELECT AC type VALUE [inequality] val1 [val2]

  eg SELECT AC HEIGHT VALUE >30.0 (height AC, value >30)
  SELECT AC HEIGHT VALUE 10.0 20.0 (height AC, values 10 to 20)

* **SELECT ACGROUP**
  Specifies a group number (1-10) in which to store subsequent SELECT AC commands. The selections in different groups are logically ORed together.

  Format: SELECT ACGROUP n

* **SELECT ALL**
  Return selections to default state.

  If no subcommand is given, then all selections are reset to default state, ie all maps, layers, feature codes, feature serial numbers, ACs, text categories and text styles are selected and any selections based on regions are cancelled. Selection of features flagged as edited, unedited and deleted is also cancelled. I.e. edited and unedited features are selected, while deleted features are not selected.

If a subcommand is given, then only one of classes of selections is reset.

Format: SELECT ALL [subcommand]

- **SELECT ALL AC**
  All ACs are selected.

  Format: SELECT ALL AC

- **SELECT ALL CATEGORY**
  All text categories are selected.

  Format: SELECT ALL CATEGORY

- **SELECT ALL FCS**
  All feature codes are selected.

  Format: SELECT ALL FCS

- **SELECT ALL FLAGS**
  Selection of features flagged as edited, unedited and deleted is cancelled. ie Edited and unedited features are selected, while deleted features are not selected

  Format: SELECT ALL FLAGS

- **SELECT ALL FSNS**
  All feature serial numbers are selected.

  Format: SELECT ALL FSNS

- **SELECT ALL GEOMETRIES**
  Any selections by geometry are cancelled.

  Format: SELECT ALL GEOMETRIES

- **SELECT ALL LAYERS**
  All layers are selected.

  Format: SELECT ALL LAYERS

- **SELECT ALL MAPS**
  All maps are selected.

  Format: SELECT ALL MAPS

- **SELECT ALL PRIORITIES**
  All representation priority levels are selected for display.

Format: SELECT ALL PRIORITIES

- o **SELECT ALL REGIONS**
  Any selections by region are cancelled.

  Format: SELECT ALL REGIONS

- o **SELECT ALL STYLE**
  All text styles are selected.

  Format: SELECT ALL STYLE

* **SELECT CATEGORY**
Select text features with a category in the given range.
Values for text categories are in the range 0 - 63.

NOTE

Only text features are subject to this selection; other features may also be displayed / found / output

Format: SELECT CATEGORY range

eg SELECT CATEGORY 0-12,15,30-35

* **SELECT CUTGEOMETRY**
Select all features that cut the specified geometry.

Note that only one geometry may be used for selection at a time, and that it must be an area geometry.

Format: SELECT CUTGEOMETRY integer

eg SELECT CUTGEOMETRY 4

* **SELECT CUTREGION**
Select all features that cut the boundary of the specified region.

See also ENABLE AND

Format: SELECT CUTREGION integer

eg SELECT CUTREGION 4

* **SELECT DELETED**
Select all features that have been deleted.
Only operational if FLAGS are enabled.

Format: SELECT DELETED

* **SELECT EDITED**
Select all features that have been flagged as edited.
Only operational if FLAGS are enabled.

Format: SELECT EDITED

* **SELECT FCS**
Feature codes in given range.
Range may be numeric or feature code groups.
See the FRTLIB documentation for information about setting up groups.

Format: SELECT FCS range

```
eg       SELECT FC 20-30,40-50,55,78
or       SELECT FC ROADS,RIVERS
or       SELECT FC 3-13,ROADS
```

* **SELECT FSNS**
Select Feature Serial Numbers in given range.

Format: SELECT FSNS range

eg   SELECT FSNS 20-30,40-50,55,78

* **SELECT INGEOMETRY**
Select all features that lie inside the specified geometry.

Note that only one geometry may be used for selection at a time, and that it must be an area geometry.

Format: SELECT INGEOMETRY integer

eg   SELECT INGEOMETRY 4

* **SELECT INREGION**
Select all features that lie completely within the specified region.

See also ENABLE AND

Format: SELECT INREGION integer

eg   SELECT INREGION 4

* **SELECT LAYERS**
Select by layer.

Format: SELECT LAYERS range

eg   SELECT LAYERS 2-4,10-24

* **SELECT MAPS**
Maps in given range.

Format: SELECT MAPS range

eg   SELECT MAPS 2-4

* **SELECT OUTPUT**
  Only selected features are output on EXIT or WRITE (default is DESELECT OUTPUT).

  Format: SELECT OUTPUT

* **SELECT OUTGEOMETRY**
  Select all features that lie outside the specified geometry.

  Note that only one geometry may be used for selection at a time, and that it must be an area geometry.

  Format: SELECT OUTGEOMETRY integer

  eg   SELECT OUTGEOMETRY 4

* **SELECT OUTREGION**
  Select all features that lie completely outside the specified region.

  See also ENABLE AND

  Format: SELECT OUTREGION integer

  eg   SELECT OUTREGION 4

* **SELECT PRIORITIES**
  Select representation priority levels for display. Note that priority selections do not affect feature operations such as FIND, nor file operations such as WRITE.

  Format: SELECT PRIORITIES range
  or   SELECT PRIORITY range

  eg   SELECT PRIORITIES 2-4,10-24

* **SELECT STYLE**
  Select text features with a style in the given range.
  Values for text styles are in the range 0 - 3.

  NOTE

  Only text features are subject to this selection; other features may also be displayed / found / output

  Format: SELECT STYLE range

  eg   SELECT STYLE 0,2

* **SELECT UNEDITED**
  Select all features that have not been flagged as edited.
  Only operational if FLAGS are enabled.

  Format: SELECT UNEDITED

* **SELECT WINDOW**
  Only selected features are drawn on the screen, and may be found using FIND or SEARCH (default).

  Format: SELECT WINDOW

7.12.7 **DESELECT**

Specifies features which are not to participate in future operations. See SELECT.

Format: DESELECT subcommand

Valid in states INITIAL READY SETUP

Takes same subcommands as SELECT (qv), but DESELECT ALL and DESELECT ACGROUP are not valid.

7.12.8 **GEOMETRY**

Geometries are dynamic data structures (ie they are only available while LITES2 is running) that represent 2 dimensional geometric data. They can be manipulated in more complex ways than features, for example they can be combined together and buffer zones can be created around them. Their main advantage is that they represent areas in a more coherent manner than either IFF features or LITES2 regions.

There are three types of geometry - point (dimension 0), line (dimension 1) and area (dimension 2). Each of these types may consist of one or more parts, thus, for example a river network which in an IFF file consists of several line features, would produce one multi part line type geometry. Where a geometry consists of only one part it is called a simple geometry and when it consists of more than one part it is considered to be complex.

A simple area consists of a single outer boundary or ring (digitised in a counter clockwise direction) and possibly one or more inner boundaries or rings (digitised in a clockwise direction). These boundaries must not self-intersect or intersect each other. Inner boundaries must all lie within the outer boundary and must not lie within another inner boundary.

Features can be selected by geometry, where the geometry is of area type. To do this selection, the features are converted to geometries so they must be representable by valid geometries of their default type. See the GEOMETRY FEATURE command for details of these default types.

There are 32 geometries available.

An appropriate licence is required to use this command.

Geometry manipulations make use of a shared image pointed at by the logical name LSL$LITES2_GEOM_ROUTINES. This image is supplied by Laser-Scan. It is called

LSL$EXE:LITES2GEOMSHR.EXE.
This shared image in turn makes use of the image pointed at by the logical name LSL$LSLGOTHICSHR. This is normally LSL$LIBRARY:LSLGOTHICSHR in the LSLSYSTEM package.

Format: GEOMETRY subcommand

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON CONSTRUCT

* **GEOMETRY AND**
  To combine two geometries together to produce a third using a mathematical AND of the input geometries. This command may produce a complex geometry.

  ```
  If geom1 is an area  and geom2 is an  area, geom3 will be an area.
  If geom1 is a  line  and geom2 is an  area, geom3 will be a  line.
  If geom1 is a  point and geom2 is an  area, geom3 will be a point.
  If geom1 is a  line  and geom2 is a   line, geom3 will be a point.
  ```

  All other combinations of geometry types are invalid.

  Format: GEOMETRY AND geom1 geom2 geom3

* **GEOMETRY ANDNOT**
  To combine two geometries together to produce a third using a mathematical AND of the input geometry 1 with the area outside geometry 2. This command may produce a complex geometry.

  ```
  If geom1 is an area  and geom2 is an  area, geom3 will be an area.
  If geom1 is a  line  and geom2 is an  area, geom3 will be a  line.
  If geom1 is a  point and geom2 is an  area, geom3 will be a point.
  ```

  All other combinations of geometry types are invalid.

  Format: GEOMETRY ANDNOT geom1 geom2 geom3

* **GEOMETRY ADD**
  To add a simple geometry to another geometry. If the second geometry does not exist, one of an appropriate type will be created.

  When adding a geometry to an existing geometry, both geometries must be of the same type.

  Format: GEOMETRY ADD geom1 geom2

* **GEOMETRY BUFFER**
  To produce a buffer of a specified radius around a geometry. The resulting geometry will be of area type.

  By default, the radius is specified in IFF units (unless a UNITS command has been given).

The number of points used to generate arcs around external angles in the original geometry is controlled by the CIRGEN tolerance setting.

The algorithm to produce buffer zones is theoretically correct, but because when it produces circle arcs around external angles it generates them as a series of vectors, it occasionally fails to produce a valid geometry. When this happens, LITES2 divides the vector length by 2 and tries to generate the buffer zone again. It does this 4 times before failing. If this happens more than very occasionally, the TOLERANCE CIRGEN setting should be altered to produce shorter vectors for the required offset.

Format: GEOMETRY BUFFER geom1 geom2 radius

* **GEOMETRY CANCEL**
To cancel the specified geometry.

Format: GEOMETRY CANCEL geom1

* **GEOMETRY COPY**
To copy one geometry into another geometry. This leaves the original geometry as it was.

Format: GEOMETRY COPY input_geom output_geom

* **GEOMETRY FEATURE**
To create a geometry from the current found feature. The type of geometry created, by default, depends on the feature's graphical type:

Graphical types 7 - 10 produce point type geometries.
Graphical types 1 - 6 and 11 produce line type geometries.
Graphical type 12 produces area type geometries.

These default types can be overridden by using the dimension argument to the command. This can take a value of 1 or 2.

If, in this case the found object is a text or symbol, then a box (or boxes) will be generated around the feature (expanded using the TOLERANCE EXPAND setting) to produce the line or area type geometry.

If the found feature is a circle or circle arc, then the number of points generated to produce the line or area geometry is controlled by the TOLERANCE CIRGEN setting.

Curves and symbol strings (graphical types 6 and 11) are treated as lines (graphical type 1).

When creating line type geometries, invisible lines in the feature are ignored (ie they are treated as visible lines).

When creating area geometries, the IFF data must conform to certain rules that are not normally enforced by LITES2. Polygons that are produced by other programs (eg IPOLYGON) do conform to these rules.

They are:

1) Invisible moves imply a jump to another ring.
2) Invisible moves to and from a ring must be coincident.
3) The direction of digitising of the rings is irrelevant, but the ring with the largest area is taken to be the outer boundary, and all the other rings must lie inside this boundary.
4) The visible line work must not intersect.

Format: GEOMETRY FEATURE geom1 [dimension]

* **GEOMETRY NOTAND**
To combine two geometries together to produce a third using a mathematical AND of the area outside input geometry 1 with geometry 2. This command may produce a complex geometry.

This command is only valid for area type geometries.

Format: GEOMETRY NOTAND geom1 geom2 geom3

* **GEOMETRY OR**
To combine two geometries together to produce a third using a mathematical OR of input geometry 1 with geometry 2. This means that the resultant geometry will consist of all the area covered by either of the input geometries.

This command may produce a complex geometry.

This command is only valid for area type geometries.

Format: GEOMETRY OR geom1 geom2 geom3

* **GEOMETRY REGION**
To create a (area) geometry from the specified region.

The input region must conform to the rules for data being used for an area type geometry. See GEOMETRY FEATURE for details.

Format: GEOMETRY REGION geom region

* **GEOMETRY RENAME**
To rename a geometry to another geometry. This command has the effect of cancelling the original geometry.

Format: GEOMETRY RENAME input_geom output_geom

* **GEOMETRY XOR**
To combine two geometries together to produce a third using a mathematical XOR of input geometry 1 with geometry 2. This means that the resultant geometry will consist of all the area covered by either

of the input geometries, but excluding the area covered by both.

This command may produce a complex geometry.

This command is only valid for area type geometries.

Format: GEOMETRY XOR geom1 geom2 geom3

7.12.9 **REGION**

An appropriate licence is required to use this command.

Creates or adds points to the specified region. There are 32 regions available.

The command REGION n defines the found linear, text or symbol feature, or the current text component or symbol when in MODIFY state with no found feature, to be the specified region.
For text features, the region consists of the joined up boxes around each character (qv TOLERANCE EXPAND); for symbols it consists of its bounding box.

Regions are always taken to be closed areas. A closing line is assumed. Once defined, a region can be used for feature selection, using SELECT and DESELECT, and also via the $INREGION, $OUTREGION, and $CUTREGION system variables. Features can also be clipped to regions (qv CLIP command).
Regions can be cancelled with the CANCEL REGION command, transformed with the TRANSFORM REGION command, their vertices listed with the SHOW REGION command and they can be displayed on the screen with the DRAW REGION and DRAW AREAREGION commands.

Format: REGION integer [subcommand]

eg REGION 5
or REGION 5 POINT 10.0 20.0

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON CONSTRUCT

* **REGION n BOX**
  Creates a rectangular region around the limits of the found feature (or text or symbol being modified).

  Format: REGION integer BOX

* **REGION n FEATURE**
  The command REGION n FEATURE is a synonym for the REGION n command (see above).

  Format: REGION integer FEATURE

* **REGION n GEOMETRY m**
  Creates a region from the specified geometry.

This command is only valid for area geometries with a single part. If the geometry has any inner rings, then these will be discarded, as regions cannot satisfactorily cope with this concept.

Format: REGION integer GEOMETRY integer

* **REGION n IMAGE**
  Creates a region around pixels of the same colour. The cursor must be pointing within a raster image specified using the IMAGE SELECT command (the highest numbered image will be used if there is more than one). The colour of the pixel containing the cursor is identified, and a region created around all contiguous pixels of the same colour. Note that the region may contain holes of other colours (not containing the cursor). This command may be terminated prematurely by CTRL/C.

  Format: REGION integer IMAGE

* **REGION n POINT**
  The command REGION n POINT x y adds data point x y to region n, creating the region if it does not already exist. The region cannot be used until it has at least 3 points.

  Format: REGION integer POINT x y

* **REGION WINDOW**
  Defines a rectangular region using the cursor. The current cursor position becomes the bottom left of the region, and WINDOW state is entered. The cursor may then be moved to the top right of the region and the END command given. The START command may be used to set the bottom left to a new position. ABANDON may be used to cancel the operation.

  Format: REGION integer WINDOW

* **REGION n ZONE**
  Creates a buffer zone around the found linear feature (or text or symbol being modified). If the feature is closed, then a positive offset will create a region outside the feature while a negative offset will create a region inside the feature. Negative offsets greater than the extent of the closed feature produce invalid regions. If the feature is open, a region will be created around the whole feature, like a sausage.

  Note that for complex features, buffer zones may be reentrant and will cause problems when testing features against them. However the variable $CURSINREGION will give the correct result for these regions.

  When creating buffer zones, circle arcs are generated around the outside of angles in the original feature. The number of points produced in these arcs is controlled by the CIRGEN tolerance setting. The points in the region are then filtered, and the number of points produced in this stage is controlled by the BUNCH tolerance setting.

  By default, the offset distance is specified in IFF units (unless a UNITS command has been given).

Format: REGION integer ZONE offset

7.13 **Ancillary Coding Commands**

7.13.1 **ANCILLARY**

Allows ancillary code manipulation. If in INITIAL or READY state (no found feature), then the ACs in the current attribute set are edited, otherwise the ACs of the found feature are edited.
AC state is entered, and commands PREVIOUS/NEXT/FIRST/LAST (qv) will move between the TCs/CHs/ACs. DELETE will delete the current TC/CH/AC, ADD will add new ones, ALTER will change the current AC, while REPLACE and SUBSTITUTE will allow the text only to be changed.
To display the current list of ACs, use EXAMINE AC (for feature), or SHOW ATTRIBUTE (for current attribute set).
END or ABANDON will revert to READY state. The ACs belonging to a feature will be unchanged of ABANDON is used, but changes to the current attribute set cannot be 'undone'.

Format: ANCILLARY

Valid in states LINE CIRCLE TEXT SYMBOL

7.13.2 **AC**

Alias for ANCILLARY command (qv).

7.13.3 **ADD**

Add a new AC, TC or CH to the found feature, or to the current attribute set.

Format: ADD subcommand

Valid in state AC

* **ADD AC**
  Standard type AC entry. Any type of AC may be added by this command, but some types still have their own command (such as ADD HEIGHT) for historical reasons.
  The type of AC can either be an integer, or the corresponding name from the FRT.

  See the section on LITES2 command language for details of the format of the value for this type of command argument.

  Text is optional. It must be enclosed in double quotes if it is to have any leading spaces or tabs.

  Format: ADD AC type value [text]

  ```
  eg       ADD AC HEIGHT 24.53 pressure 19.4
  or       ADD AC 3      24.53
  ```

* **ADD CH**
  CH entry (used to carry non-IFF information).

  Format: ADD CH text

  eg   ADD CH LI 0 0

* **ADD CONTOUR**
  Contour (integer height) AC entry (type 2).

  Format: ADD CONTOUR integer

  eg   ADD CONTOUR 200

* **ADD CROSSREF**
  Feature cross reference AC entry (not yet implemented).

  Format: ADD CROSSREF integer

* **ADD HEIGHT**
  Height (real) AC entry (type 3).

  Format: ADD HEIGHT real

  eg   ADD HEIGHT 34.6

* **ADD LH**
  Left hand boundary text AC entry (type 4).

  Format: ADD LH integer text

  eg   ADD LH 23 Bedfordshire

* **ADD REALAC**
  Standard real type AC entry. Superceded by ADD AC which can also deal with reals.
  AC type must be 3 or 80 - 99.
  AC value is any real number.
  Text is optional. It must be enclosed in double quotes if it is to have any leading spaces or tabs.

  Format: ADD REALAC type value [text]

  eg   ADD REALAC 99 22.6 Height of pylon

* **ADD RH**
  Right hand boundary text AC entry (type 5).

  Format: ADD RH integer text

  eg   ADD RH 23 Cambridgeshire

* **ADD SECONDARY**
  Secondary feature code AC entry (type 1).

  Format: ADD SECONDARY integer

eg ADD SECONDARY 23

* **ADD TC**
  TC (transmitted comment) entry

  Format: ADD TC text

  eg ADD TC this is a transmitted comment

7.13.4 **ALTER**

Amend the current AC, TC or CH.

Format: ALTER subcommand

Valid in state AC

Subcommands as ADD (qv)

7.13.5 **TRAIL**

Not yet implemented.
Allows alteration of trailing TC/CH entries for the current layer.

Format: TRAIL

Valid in state READY

7.14 **Interrogation Commands**

7.14.1 **EXAMINE**

Displays attributes of found feature.

Format: EXAMINE subcommand

Valid in states:
LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON WINDOW CONSTRUCT AC RECOVER

* **EXAMINE AC**
  Displays AC, TC, CH entries.

  Format: EXAMINE AC

* **EXAMINE ALL**
  Displays all information about found feature.

  Format: EXAMINE ALL

* **EXAMINE ANGLE**
  Displays the angle (anti-clockwise from horizontal) in degrees of the current feature. For linear features this is the angle of the current vector.
  This information is not available for circle arcs.

  Format: EXAMINE ANGLE

* **EXAMINE AREA**
  Displays the area enclosed by a linear feature. If the feature is not closed, an imaginary vector between the first and last points is assumed.
  A positive area indicates that the feature has been digitised in a clockwise direction, negative areas indicate counter clockwise digitising.
  Degenerate features (with two or less points) give an area of 0.0

  Format: EXAMINE AREA

* **EXAMINE ATTRIBUTES**
  The cursor must be on a point for this command.
  Displays the ATTRIBUTES of the current point in the feature.

  Format: EXAMINE ATTRIBUTES

* **EXAMINE BEARING**
  Displays the bearing (clockwise from grid north) in degrees of the current feature. For linear features this is the bearing of the current vector.
  This information is not available for circle arcs.

  Format: EXAMINE BEARING

* **EXAMINE BOX**
  Displays the coordinates of the limits of the box surrounding the current feature.

  Format: EXAMINE BOX

* **EXAMINE CATEGORY**
  Displays category for text.

  Format: EXAMINE CATEGORY

* **EXAMINE CROSSREF**
  Displays any cross reference FSN's (not yet implemented).

  Format: EXAMINE CROSSREF

* **EXAMINE DISTANCE**
  Displays the distance along the found feature of the current cursor position, measured from the start.

  Format: EXAMINE DISTANCE

* **EXAMINE FC**
  Displays feature code of found feature.

  For subtexts of a composite text, the FC displayed is the FC (or text component code - TCC) of the current subtext.

  Format: EXAMINE FC

* **EXAMINE FSN**
  Displays serial number of found feature.

  Format: EXAMINE FSN

* **EXAMINE GT**
  Displays graphical type of found feature.

  Format: EXAMINE GT

* **EXAMINE HEIGHT**
  Displays height of found text feature (mm).

  Format: EXAMINE HEIGHT

* **EXAMINE LAYER**
  Displays layer of found feature.

  Format: EXAMINE LAYER

* **EXAMINE LENGTH**
  Displays the total length of the found feature, if it is of a linear nature.

  Format: EXAMINE LENGTH

* **EXAMINE LINE**
Displays the length and direction of the segment of a linear feature.

Format: EXAMINE LINE

* **EXAMINE LOCATION**
Displays text location field for text.

Format: EXAMINE MAP

* **EXAMINE MAP**
Displays name of IFF file containing found feature.

Format: EXAMINE MAP

* **EXAMINE PATTERN**
Displays line pattern of found feature.

Format: EXAMINE PATTERN

* **EXAMINE PC**
Displays process code of found feature.

Format: EXAMINE PC

* **EXAMINE POINT**
Displays point number(s) for cursor position.

Format: EXAMINE POINT

* **EXAMINE POSITION**
The cursor must be on a point for this command.
Displays the X and Y coordinates of the current point in the feature.

Format: EXAMINE POSITION

* **EXAMINE PSIZE**
Displays text height in points.

Format: EXAMINE PSIZE

* **EXAMINE SECONDARY**
Displays FRT secondary code for found feature. This is the pattern index for lines, the symbol number for symbols, the font for texts, and the fill style for areas.

Format: EXAMINE SECONDARY

* **EXAMINE SIZE**
Displays size of found symbol (mm).

Format: EXAMINE SIZE

* **EXAMINE STYLE**
Displays typeface field for text.

Format: EXAMINE STYLE

* **EXAMINE SUMMARY**
Displays found feature map, layer, FSN, FC, GT (default).

Format: EXAMINE SUMMARY
or EXAMINE

* **EXAMINE WIDTH**
Displays line width of found feature (mm).

Format: EXAMINE WIDTH

7.14.2 **SHOW**

Displays information requested.

Format: SHOW subcommand

Valid in all states

* **SHOW ABSOLUTE**
Gives current cursor position in full projection units.

While SHOW POSITION gives the position in terms of the coordinates in the IFF files being edited, SHOW ABSOLUTE gives the position taking into account any origin offset specified in the IFF files

If Z has been enabled and the cursor has no Z value, a "?" is displayed.

Format: SHOW ABSOLUTE

* **SHOW ACD**
If no argument, lists all the ACDs defined in the current FRT;
with argument, lists details of the specified ACD.

The argument can either be a number (the AC type or attribute code) or the corresponding name.

Format: SHOW ACD [argument]

* **SHOW AFTER**
Lists the commands that have been set up by the AFTER command.

Format: SHOW AFTER

* **SHOW ANNOTATION**
Lists the characteristics that will be used for annotating features when the DRAW LABEL command is given.

Format: SHOW ANNOTATION

* **SHOW ATTRIBUTE**
  Lists the current attribute set, or the requested attribute set. If no argument is given then the current attribute set is listed.

  Format: SHOW ATTRIBUTE [integer]

* **SHOW BASES**
  Lists the BASES that have been set up for squaring, edgematching, or orienting.

  Format: SHOW BASES

* **SHOW COLOURS**
  Show the specified colours in the current overlay and display.
  If no range of colour indices is given all the available colours are listed. The colours may be listed using the RGB (default), HLS, or HSV scheme.

  This command is only available on versions with suitable hardware facilities.

  Format: SHOW COLOURS [range] [scheme]

  ```
  eg      SHOW COLOURS 1,3,5,7-11
  or      SHOW COLOURS 1 HLS
  ```

* **SHOW COMMANDS**
  Lists all the primary commands available in the current state.

  Format: SHOW COMMANDS

* **SHOW CP**
  Lists the control points currently set in the specified map. If no map is specified, the control points of all the maps are displayed.

  The control points may be set using the EDIT CP command.

  Note that the values given are the actual values in the IFF files. The values given by the system variables $CPXxx and $CPYxx are the values in the LITES2 coordinate space, and may contain an appropriate offset if multiple maps have been read in.

  Format: SHOW CP [integer]

* **SHOW DISPLAYS**
  Gives details of displays. If a display number is specified, gives details of that display, otherwise gives details of all displays.

  Format: SHOW DISPLAYS [integer]

* **SHOW FCS**
  If no argument, lists all the FCs defined in the current FRT; with argument, lists details of the specified FC.

  Format: SHOW FCS [integer]

* **SHOW FCPRIORITIES**
  Shows details of priorities used when drawing with ENABLE SORT and SORT PRIORITY.

  If no argument, lists all the priorities defined in the current FRT, along with all the feature codes that have been assigned priorities and the representations to be used.

  With an argument, lists details of the priorities and representations for the specified FC.

  Format: SHOW FCPRIORITIES [integer]

* **SHOW FILL**
  If an argument is given, gives details of the direction and pattern of lines used to fill areas; if no argument is given, gives details of all the patterned areas in the current FRT.

  Note: fill numbers must be less than -1.

  Format: SHOW FILL [integer]

* **SHOW GEOGRAPHICAL**
  Shows the latitude and longitude of the current cursor position.

  This command is only valid if at least one map read in has valid type 2 map descriptor, specifying a valid projection.

  Geographical conversions make use of a shared image pointed at by the logical name LSL$LITES2_GEOG_ROUTINES. This image is supplied by Laser-Scan. It is called LSL$EXE:LITES2GEOGSHR.EXE.

  Format: SHOW GEOGRAPHICAL

* **SHOW GEOMETRIES**
  Shows information about geometry definitions.

  If a geometry is specified, gives information about that geometry
  If no geometry is specified, gives information about all the defined geometries.

  Format: SHOW GEOMETRIES [integer]

* **SHOW GROUPS**
  If no group is specified, a list of the groups available is given. If a group is specified, a list of feature codes in the group is given.

  Format: SHOW GROUPS [text]

* **SHOW IMAGES**
  Gives details of raster images. If an image number is specified, gives details of that image, otherwise gives details of all images.

  Format: SHOW IMAGES [integer]

* **SHOW INTERPOLATION**
  Gives drawing and construction interpolation method.

  Format: SHOW INTERPOLATION

* **SHOW LABELS**
  Lists the attributes that will be used for annotating features when the DRAW LABEL command is given.

  Format: SHOW LABELS

* **SHOW LAYERS**
  Lists all layers in use, with maximum FSNs.

  Format: SHOW LAYERS

* **SHOW LIGHTS**
  Displays the current settings of each light in each view. (See command VIEW LIGHT). If the optional light number is not given, all the light sources are listed for the specified view. If the optional view number is not given all the light sources in all possible views are listed.

  ::: means that this value has not yet been set.

  Format: SHOW LIGHTS

* **SHOW LIMITS**
  Displays the coordinates of the limits of the LITES2 working area. This is the total range of the maps that were originally read in, plus 5% all round. Note that the range may have been altered by subsequent edits

  Format: SHOW LIMITS

* **SHOW MACROS**
  If no macro is specified, a list of the macros defined is given. If a macro is specified, the macro text expansion is given.

  Format: SHOW MACROS [text]

* **SHOW MAPS**
  Lists all maps, and their IFF filenames.

  Format: SHOW MAPS

* **SHOW MATCH**
  Lists the match settings that are used when finding features during TIEing, JOINing, MENDing, and EDGEMATCHing.

  Format: SHOW MATCH

* **SHOW MEMORY**
  Intended as a program development aid.
  Shows statistics concerning use of dynamic memory, lock usage and file usage.
  If an (optional) integer argument in the range 1-3 is given, then an

increasing amount of information will be given about the various dynamic memory zones used by LITES2.

Format: SHOW MEMORY [level]

* **SHOW MENUS**
If no menu is specified, a list of the menus and pucks defined is given. If a menu is specified, the macro expansion for every menu box (or puck button) is given.
If a menu and a box number is given, the macro expansion for that menu box (or puck button) is given.

Format: SHOW MENUS [text]

* **SHOW OPERATIONS**
Lists the attributes (and values) that have been set up by the OPERATION command. If no value is listed, then this attribute (or AC in the case of xxxx_FEATURE operations) then the attribute or AC will be deleted when the relevant edit is completed.
If a number (in brackets) is listed instead of a name, it means that the attribute does not exist in the FRT.

Format: SHOW OPERATION

* **SHOW OPTIONS**
Lists options selected by ENABLE/DISABLE.

If the optional subcommand FIRST is given, the first half of the table is displayed; SECOND displays the second half of the table. If no additional command is given the whole table is listed.

Format: SHOW OPTIONS [subcommand]

* **SHOW OVERLAYS**
Gives details of display overlays. If an overlay number is specified, gives details of that overlay, otherwise gives details of all overlays.

Format: SHOW OVERLAYS [integer]

* **SHOW PATTERNS**
If a pattern is specified, gives details of that pattern; if no pattern is specified, gives details of all the patterned lines in the current FRT.

Format: SHOW PATTERNS [integer]

* **SHOW PLOT**
Shows details of the hardcopy plot settings.

Format: SHOW PLOT

* **SHOW POSITION**
Gives current cursor position (default).

If Z has been enabled and the cursor has no Z value, a "?" is displayed.

Format: SHOW POSITION
or SHOW

* **SHOW PRIVILEGE**
Lists the commands, attributes and points that have been set privileged by the PRIVILEGE command

Format: SHOW PRIVILEGE

* **SHOW PROJECTION**
Show the projection information about the specified map. If no map number is given, then information about the LITES2 coordinate space is given.

Format: SHOW PROJECTION [map]

* **SHOW REGIONS**
If a region is specified, gives the coordinates of the vertices of that region. If no region is specified, gives the coordinates of all the defined regions.

Format: SHOW REGIONS [integer]

* **SHOW SCALES**
Gives information concerning scales.

Format: SHOW SCALES

* **SHOW SCROLL**
Gives scroll area of terminal.

Format: SHOW SCROLL

* **SHOW SECTORS**
Gives number of sectors.

Format: SHOW SECTORS

* **SHOW SELECTIONS**
Gives details of current selections.

Format: SHOW SELECTIONS

* **SHOW SETUP**
Gives details of the setup and transformations that will be used to setup any maps on the table (see SETUP and PTOLERANCE OSSETUP commands), and if there are any maps already set up on the table, then it also gives details of how these are set up.

Format: SHOW SETUP

* **SHOW SORT**
Gives the current method of sorting for re-draws.

Format: SHOW SORT

* **SHOW STATE**
  Gives current state.

  Format: SHOW STATE

* **SHOW TOLERANCE**
  Lists the TOLERANCES that have been set up using the TOLERANCE command.

  If the optional subcommand FIRST is given, the first half of the table is displayed; SECOND displays the second half of the table. If no additional command is given the whole table is listed.

  Format: SHOW TOLERANCE [subcommand]

* **SHOW TRANSFORMATION**
  Shows details of the transformation that has been set up.

  Format: SHOW TRANSFORMATION

* **SHOW VARIABLES**
  If no variable name is given, then all variable names are listed. If a particular variable name is given, then its type and value are given. If just $ is given, then all system variable names are listed.

  Format: SHOW VARIABLES [name]

  ```
  eg      SHOW VARIABLES          lists user variables
          SHOW VARIABLES $        lists system variables
          SHOW VARIABLES NAME     give type and value of variable NAME
  ```

* **SHOW VERSION**
  Gives version and date of linking of program, plus the licensed optional facilities.

  Format: SHOW VERSION

* **SHOW VIEWS**
  Displays the current settings for each view. If the optional view number is not given, then the settings for all views are given.

  ::: means that this value has not yet been set; numbers in brackets are values that are not yet set, but default to the value of some other item.

  Format: SHOW VIEWS

* **SHOW WARP**
  Shows details of transformation (warping) of raster images.

  Format: SHOW WARP

* **SHOW WINDOW**
  Displays the coordinates of the limits of the current window

  Format: SHOW WINDOW

* **SHOW ZOOM**
  Gives the number of times that the current picture on the screen is magnified from the full map on the screen.

  Format: SHOW ZOOM

7.14.3 TIME

Display and manipulation of timing information. For those subcommands which take an argument, it is used to set or clear that timing field.

Format: TIME subcommand

eg TIME ALL
or TIME FIND 0

Valid in all states

* **TIME ALL**
  All timing statistics are displayed or reset.

  Format: TIME ALL

* **TIME DRAW**
  Draw time displayed or reset.

  Format: TIME DRAW [integer]

* **TIME FIND**
  Search time displayed or reset.

  Format: TIME FIND [integer]

* **TIME NOW**
  Current time and date are displayed (default).

  Format: TIME NOW
  or TIME

* **TIME READ**
  Read-in time displayed or reset.

  Format: TIME READ [integer]

* **TIME STATES**
  Time in each state displayed.

  Format: TIME STATES

* **TIME SUMMARY**
  Display elapsed time, CPU time, and IO operations.

  Format: TIME SUMMARY

7.15 **Windowing Commands**

7.15.1 **DRAW**

Draws on the LITES2 screen(s).

DRAW MAP and DRAW SCREEN clear the display and draw the whole or part of the IFF file(s) being edited. The features in the IFF file are drawn with the characteristics specified for their feature code in the FRT file being used. The action of DRAW may be modified by SUPPRESS, and ENABLE VECTOR (qv).

Other DRAW commands allow additional information to be added to the picture on the screen. This data is drawn with characteristics specified by the ANNOTATION command.

By default, LITES2 uses drawing buffers which can hold 8192 points, and this limits the size of areas that can be filled. Areas with more than this number of points will only be drawn in outline (perhaps with some invisible lines being visible). The size of the drawing buffer can be altered by setting the logical name LSL$FILL_POINTSMAX to the required size before LITES2 is started up. This logical name also controls the size of images that can be drawn; the error message

"Buffer too small to draw <type> - zoom in or increase LSL$FILL_POINTSMAX"

indicates that the buffers are too small, and should be increased.

When drawing fill areas there is a limit to the number of times a scan line can be cut. By default this is 100. If the message

"FILL_SIDE - Too many intersections found - ignored"

occurs, then this number is too small. It can be set by defining the logical name LSL$FILL_CUTSMAX before starting LITES2.

Note that memory has to be allocated in proportion to these numbers, so unnecessarily large values should be avoided.

Format: DRAW subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL SETUP

* **DRAW ABSOLUTE**
  Draws the symbol (specified by the ANNOTATION MARK command) at the current cursor position and a string of text containing the absolute position (ie full projection coordinates) of the cursor.

  The position of the text string is based on any ANNOTATION OFFSET commands that have been given, in conjunction with the symbol's bounding box. If the offset in X is positive, then the position of the text is this amount to the right of the bounding box, and if it is negative then the position is to the left. Similarly a positive or negative Y offset positions the text above or below the bounding box. When an offset of 0.0 is specified no allowance for the bounding box is made.

Format: DRAW ABSOLUTE

* **DRAW AREAREGIONS**
If a region is specified, it is displayed as a filled area.
If no region is specified, then all the defined regions are displayed.

This command uses the colour set with the ANNOTATION COLOUR command and the fill style set with ANNOTATION FILL command.

Format: DRAW AREAREGIONS [integer]

* **DRAW DTI**
Draws the specified DTI file in the current display.
The file is zoomed or subsampled as required to fill as much of the display as possible.

Format: DRAW DTI filename

* **DRAW FEATURE**
Draws the current found object with the current annotation characteristics.

Format: DRAW FEATURE

* **DRAW GEOGRAPHICAL**
Draws the symbol (specified by the ANNOTATION MARK command) at the current cursor position and a string of text containing the geographical position (ie latitude and longitude) of the cursor. This command is only available if at least one of the IFF files has a fully specified version 2 map descriptor. See the IMP utility ITRANS for more information.

The position of the text string is based on any ANNOTATION OFFSET commands that have been given, in conjunction with the symbol's bounding box. If the offset in X is positive, then the position of the text is this amount to the right of the bounding box, and if it is negative then the position is to the left. Similarly a positive or negative Y offset positions the text above or below the bounding box. When an offset of 0.0 is specified no allowance for the bounding box is made.

Format: DRAW GEOGRAPHICAL

* **DRAW GEOMETRY**
Draws geometries according to their type.

Point type geometries are displayed with a symbol at each of their locating points (the symbol being set by the ANNOTATION MARK command).
Line type geometries are displayed as lines.
Area type geometries are displayed as fill areas.

This command uses the drawing attributes set with the ANNOTATION command.

If a geometry is specified only that geometry is displayed.

If no geometry is specified, all the defined geometries are displayed.

Format: DRAW GEOMETRY [integer]
or DRAW GEOMETRIES

* **DRAW GRID**
Plots sector grid (program development aid).

This command uses the colour set for drawing labels. See the ANNOTATION COLOUR command.

Format: DRAW GRID

* **DRAW HARDCOPY**
Draws crosses at corner points of maps, then draws the screen on hard copy device.

Format: DRAW HARDCOPY

* **DRAW IFF**
Draws the specified IFF file in the current display.

Format: DRAW IFF filename

* **DRAW IMAGE**
Draws the raster image currently selected by the IMAGE NUMBER command into the current display (which must not be the primary or secondary display). An ANNOTATION OVERLAY must be specified. An area of the image centred on the cursor is drawn. The optional factor (default 1) specifies the pixel zoom factor e.g. DRAW IMAGE n means draw the image with n screen pixels for each image pixel. Numbers less than 1 mean to subsample the image, e.g. DRAW IMAGE 0.25 (or DRAW IMAGE 1/4) means that each screen pixel represents a 4 by 4 block of image pixels.

Format: DRAW IMAGE [factor]

* **DRAW LABEL**
Annotates all the features on the screen.
If the command LABEL FEATURE has been given (default) then the whole feature is labelled; if the command LABEL POINT has been given, then the individual points within the feature are labelled

The attributes to be used for labelling can be set with the LABEL command, and their appearance can be set with the ANNOTATION command.

Format: DRAW LABEL

* **DRAW LEGEND**
Draws a legend to indicate the meaning of the colours in the current image.

The size and shape of the legend boxes and the position of the legend is controlled by the ANNOTATION LEGEND command, and whether the background to the text is blanked out by the ENABLE BLANK command.
The legend boxes are drawn in the same overlay as the image; the text

is drawn in the annotation overlay (in the colour set by the ANNOTATION COLOUR command).

If the image is classified then one box is generated for each step/band in the range of the image. This may mean that the legend will not fit on the screen, and a message to this effect is output.
If the image is not classified, then a legend is always drawn, but if all the colours cannot be shown then a selection, evenly distributed through the colour range, is displayed.

Format: DRAW LEGEND

* **DRAW LSI**
Draws the specified LSI file in the current display.
The file is zoomed or reduced views are used as required to fill as much of the display as possible.

Format: DRAW LSI filename

* **DRAW LSR**
Draws the specified LSR file in the current display.
The file is zoomed as required to fill as much of the display as possible.

Format: DRAW LSR filename

* **DRAW MAP**
Draws the whole map on the screen.

Format: DRAW MAP
or DRAW

* **DRAW MARK**
Marks all the points in the features displayed on the screen.

The points to be marked can be selected with the LABEL ATTRIBUTE command; the symbol to be used to mark the points is selected with the ANNOTATION MARK command.

Format: DRAW MARK

* **DRAW MOVE**
Moves the drawing cursor in the annotation display to the specified point, without drawing a line.

If the current annotation display is the primary or secondary workstation, then the main LITES2 cursor is moved. By default the position is specified in IFF units (unless a UNITS command has been given).

Format: DRAW MOVE x y

* **DRAW NUMBERS**
Labels all the vertices in the features displayed on the screen with their point numbers.

The vertices to be labelled can be selected with the LABEL ATTRIBUTE command; the position of the labels are offset by the amounts specified by any ANNOTATION OFFSET command that has been given.

Format: DRAW NUMBERS

* **DRAW POSITION**
Draws the symbol (specified by the ANNOTATION MARK command) at the current cursor position and a string of text containing the position of the cursor in IFF units.

The position of the text string is based on any ANNOTATION OFFSET commands that have been given, in conjunction with the symbol's bounding box. If the offset in X is positive, then the position of the text is this amount to the right of the bounding box, and if it is negative then the position is to the left. Similarly a positive or negative Y offset positions the text above or below the bounding box. When an offset of 0.0 is specified no allowance for the bounding box is made.

Format: DRAW POSITION

* **DRAW REGIONS**
If a region is specified, displays the boundary of that region.
If no region is specified, displays the boundaries of all the defined regions.

This command uses the colour set for drawing labels. See the ANNOTATION COLOUR command.

Format: DRAW REGIONS [integer]

* **DRAW SCREEN**
Redraws the current window on the screen (default).

Format: DRAW SCREEN

* **DRAW SHEET**
Draws the symbol (specified by the ANNOTATION MARK command) at the current cursor position and a string of text containing the position of the cursor in sheet mm.

The position of the text string is based on any ANNOTATION OFFSET commands that have been given, in conjunction with the symbol's bounding box. If the offset in X is positive, then the position of the text is this amount to the right of the bounding box, and if it is negative then the position is to the left. Similarly a positive or negative Y offset positions the text above or below the bounding box. When an offset of 0.0 is specified no allowance for the bounding box is made.

Format: DRAW SHEET

* **DRAW TEXT**
Draws the symbol (specified by the ANNOTATION MARK command) at the current cursor position and the string of text specified in the

argument. If the text contains leading spaces, then it must be enclosed in double quotation marks.

Note that the argument may contain a substituted variable, eg DRAW TEXT '$IMAGEVALUE

The position of the text string is based on any ANNOTATION OFFSET commands that have been given, in conjunction with the symbol's bounding box. If the offset in X is positive, then the position of the text is this amount to the right of the bounding box, and if it is negative then the position is to the left. Similarly a positive or negative Y offset positions the text above or below the bounding box. When an offset of 0.0 is specified no allowance for the bounding box is made.

Format: DRAW TEXT text

* **DRAW TITLE**
  Draws the text specified in the argument at the current cursor position. If the text contains leading spaces, then it must be enclosed in double quotation marks.

  Note that the argument may contain a substituted variable, eg DRAW TITLE '$IMAGEVALUE

  The position of the text is based on the cursor position; it is not affected by any ANNOTATION OFFSET command

  Format: DRAW TITLE text

* **DRAW VECTOR**
  Draws a vector from the current cursor position to the specified position and moves the cursor to that position.

  By default the position is specified in IFF units (unless a UNITS command has been given).

  Format: DRAW VECTOR x y

7.15.2 **WINDOW**

Defines portion of map to be displayed on graphics screen for enlargement, etc. Use the DRAW command (qv) to redraw either the whole map or the current window. The action of WINDOW may be modified by SUPPRESS, and ENABLE VECTOR (qv).
The two subcommands only differ in the way the screen picture and cursor position are treated during the operation.

Format: WINDOW subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL WINDOW SETUP

* **WINDOW MAP**
  The current cursor position is taken as the lower left corner of the new window. This can be redefined by moving the cursor and using WINDOW or START.
  Then move the cursor to the top right corner of the new window and give command END.
  Differs from WINDOW SCREEN in that the screen cursor positions while the window is being defined are as if the whole map were displayed on the screen. This enables screen windowing onto a region not currently on the screen. The present window is drawn on the screen in the ANNOTATION COLOUR. It is possible to position the cursor on the screen and then ABANDON without performing the WINDOW. The cursor will still be in its new position, and one could, for example, ZOOM.

  Format: WINDOW MAP
  or   WINDOW

* **WINDOW SCREEN**
  The current cursor position is taken as the lower left corner of the new window. This can be redefined by moving the cursor and using WINDOW or START.
  Then move the cursor to the top right corner of the new window and give command END.

  Format: WINDOW SCREEN

7.15.3 **PICTURE**

Allows the IFF data to be transformed as it is displayed on the screen.

This command is only available on versions of LITES2 running on suitable hardware. This includes LITES2 running under VWS and MOTIF windowing systems.

Format: PICTURE subcommand

Valid in states READY

* **PICTURE ROTATE**
  Setup an orthogonal transformation that rotates the IFF data by the specified angle (in degrees) and introduces a scale change to allow the whole LITES2 coordinate space to be shown on the screen.

  The command PICTURE ROTATE 0 will cancel any existing setup and return to default behaviour, as will PICTURE SETUP followed by ABANDON.

  Format: PICTURE ROTATE angle

* **PICTURE SETUP**
  Setup an extended four point transformation to be used when IFF data is being displayed.

  Causes LITES2 to enter SETUP state, and to prompt for the user to digitise the 4 corner points of the first map with reference to points on the screen using START commands. (These will often be points in a raster image). WINDOW, DRAW, and ZOOM commands may be used to display the desired area on the screen. Once the setup is completed, LITES2 will return to READY state and subsequent drawing will distort the vector picture so as to match up the corner points with the selected points. ABANDON may be used at any time in SETUP state to abort the setup.

This command has the same effect as IMAGE SETUP.

Invoking PICTURE SETUP followed by ABANDON will cancel any existing setup and return to default behaviour.

Format: PICTURE SETUP

7.15.4 **SUPPRESS**

Modifies the action of DRAW, WINDOW, ZOOM, OVERLAY ERASE, and DELETE. SUPPRESS may be used to prevent the updating of a display during these operations. Displays may be suppressed during initial draw to speed things up. The action of SUPPRESS persists only until the next of these operations - the command must then be repeated if required.

Format: SUPPRESS subcommand

Valid in states READY LINE CIRCLE TEXT SYMBOL WINDOW SETUP PAINT

* **SUPPRESS ALL**
  All displays are suppressed.

  Format: SUPPRESS ALL

* **SUPPRESS CANCEL**
  No displays are suppressed (cancels the effect of previous SUPPRESS commands).

  Format: SUPPRESS CANCEL

* **SUPPRESS CLEAR**
  Stops the screen being cleared before the next DRAW MAP, DRAW SCREEN, WINDOW or ZOOM command draws vectors on the screen.

  This command has no effect if segments are being used.

  Format: SUPPRESS CLEAR

* **SUPPRESS PRIMARY**
  Primary display is suppressed (default).

  Format: SUPPRESS PRIMARY
  or SUPPRESS

* **SUPPRESS SECONDARY**
  Secondary display is suppressed.

  Format: SUPPRESS SECONDARY

* **SUPPRESS VECTOR**
  Display of vector data is suppressed.

  This command only has any effect when raster images are being displayed.

  Format: SUPPRESS VECTOR

7.15.5 **ZOOM**

Redraws the area around the cursor enlarged by a given zoom factor in the range 0.01 to 100.0. The default factor is 5.0.

If the subcommand IMAGE is given, then the zoom factor is interpreted instead as the pixel zoom of raster images, e.g. ZOOM n IMAGE means draw the image with n screen pixels for each image pixel. Numbers less than 1 mean to subsample the image, e.g. ZOOM 0.25 IMAGE (or ZOOM 1/4 IMAGE) means that each screen pixel represents a 4 by 4 block of image pixels.

The action of ZOOM may be modified by SUPPRESS, and ENABLE VECTOR (qv)

Format: ZOOM [real] [IMAGE]

```
eg       ZOOM
or       ZOOM 20
or       ZOOM 0.3
or       ZOOM 1 IMAGE
or       ZOOM 1/4 IMAGE
```

Valid in states READY LINE CIRCLE TEXT SYMBOL SETUP PAINT

7.15.6 **LABEL**

Sets the attributes to be used when annotating features or points with the DRAW LABEL command. Each LABEL command will set an additional attribute, and all these attributes (if present) will be used to annotate each feature that appears on the screen.

```
eg       LABEL FSN
         LABEL FC
         LABEL AC Height
```

will cause labels of the form "fsn fc [height]" to be attached to each feature on the screen. The height will only occur if the feature has a height AC (type 3) associated with it.

The LABEL NONE command should be used to cancel all labelling attributes.

The LABEL FEATURE command (default) allows features to be labelled by the next DRAW LABEL command; the LABEL POINT command allows the points within features to be labelled by the next DRAW LABEL command.

Where the AC or attribute to be labelled is explicitly specified, it can be referred to by its name or by the corresponding integer.
See the section on LITES2 command language for details of the format for this type of command argument.

Format: LABEL subcommand

Valid in all states

* **LABEL AC**
  Label features with the value of the specified AC.
  The format of the value displayed depends on the data type of the AC.

  See the section on LITES2 command language for details these data types.

  Format: LABEL AC type

* **LABEL ACINT**
  Label features with the value (treated as an integer) of the specified AC.
  This command is now redundant. LABEL AC should be used instead.

  Format: LABEL ACINT integer

* **LABEL ACREAL**
  Label features with the value (treated as a real value) of the specified AC.
  This command is now redundant. LABEL AC should be used instead.

  Format: LABEL ACREAL integer

* **LABEL ACTEXT**
  Label features with the text of the specified AC.

  Format: LABEL ACTEXT integer

* **LABEL ATTRIBUTE**
  Label points with the value of the specified point attribute. This command is also used to select which points to mark using the DRAW MARK command. If a value is given, then only points with that particular attribute value will be marked, and only values matching the given value will be included in labels.
  The format of the value displayed depends on the data type of the attribute.

  See the section on LITES2 command language for details these data types.

  Format: LABEL ATTRIBUTE type [value]

* **LABEL CONTOUR**
  Label features with the (integer) value of the contour AC. This is the same as LABEL AC Contour.

  Format: LABEL CONTOUR

* **LABEL FEATURE**
  The next DRAW LABEL command will label features with the required attributes.

  This is the opposite of the LABEL POINT command

  Format: LABEL FEATURE

* **LABEL FC**
  Label features with their feature code.

  Format: LABEL FC

* **LABEL FSN**
  Label features with their feature serial number.

  Format: LABEL FSN

* **LABEL HEIGHT**
  Label features with the (real) value of the HEIGHT AC. This is the same as LABEL AC Height.

  Format: LABEL HEIGHT

* **LABEL LH**
  Label features with the text of the Left Hand Boundary AC. This is the same as the LABEL ACTEXT LH_boundary command.

  Format: LABEL LH

* **LABEL NONE**
  Cancel all current labelling attributes.

  Format: LABEL NONE

* **LABEL POINT**
  The next DRAW LABEL command will label points with the required attributes.

  This is the opposite of the LABEL FEATURE command

  Format: LABEL POINT

* **LABEL RH**
  Label features with the text of the Right Hand Boundary AC. This is the same as the LABEL ACTEXT RH_boundary command.

  Format: LABEL RH

7.15.7 **ANNOTATION**

Sets the characteristics of graphical information, other than the map data drawn by LITES2. This includes annotation produced by the various DRAW commands.

When labelling features the following rules apply:

- For texts and symbols labels are drawn above and to the right of the bounding box around the feature. For composite texts, the first text component is so labelled.

- For linear features the label is drawn parallel to the feature each time it enters (or leaves) the current window. There are two possibilities in this case:

  a) By default, the label will be drawn at the end of (the part of) the feature nearest the left side or bottom of the window. This makes labels easily read, and this possibility is set with the ANNOTATE LEFT command.

  b) If the direction of the feature is important (eg for labelling with LH or RH codes) then the ANNOTATE START command should be used, when the feature will be labelled each time it enters the window.

Format: ANNOTATION subcommand

Valid in all states

* **ANNOTATION ANGLE**
  The labels (and symbol where appropriate) produced by the DRAW ABSOLUTE, DRAW GEOGRAPHICAL, DRAW POSITION, DRAW SHEET, DRAW TITLE and DRAW TEXT commands will be drawn at the given angle (specified in degrees).

  The offset of the label relative to the locating point is also rotated so that the label maintains its position relative to the symbol.

  Format: ANNOTATION ANGLE real

* **ANNOTATION BEARING**
  The labels (and symbol where appropriate) produced by the DRAW ABSOLUTE, DRAW GEOGRAPHICAL, DRAW POSITION, DRAW SHEET, DRAW TITLE and DRAW TEXT commands will be drawn at the given bearing (specified in degrees).

  The offset of the label relative to the locating point is also rotated so that the label maintains its position relative to the symbol.

  Format: ANNOTATION BEARING real

* **ANNOTATION COLOUR**
  Use the specified colour index for annotations.

  If -1 is specified, labels on features will be drawn in the same colour as the feature. In this case other annotations will be drawn in colour 1.

Note that by setting colour index 0 (background colour) with this command, existing labels may be deleted without redrawing the screen. (This is only available on raster displays)

Format: ANNOTATION COLOUR integer

* **ANNOTATION DISPLAY**
Draw annotations into the specified display. The default display (0) means draw on the primary and/or secondary workstation. This command sets the ANNOTATION OVERLAY to 0, so the ANNOTATION OVERLAY command must be given again if a particular overlay is to be used.

Format: ANNOTATION DISPLAY integer

* **ANNOTATION FILL**
Use the specified fill style value (as in FRT) to draw area annotations.

This value is an integer in the range -1 : 6 or in the range 101 - 106

```
-1 means draw solid areas
 0 means draw hollow (ie draw boundary)
 1 horizontal hatching
 2 vertical   hatching
 3 +45 degree hatching
 4 -45 degree hatching
 5 horizontal and vertical cross hatching
 6 +45 and -45 degree cross hatching
```

Values 101 - 106 are the same as 1 - 6 with the addition of the boundary.

Format: ANNOTATION FILL integer

* **ANNOTATION FONT**
Annotate features using the specified font.

Format: ANNOTATION FONT integer

* **ANNOTATION HARDWARE**
Use the specified hardware value (as in FRT) for annotations.
Provides the same facilities for annotations as setting the hardware field in the FRT does for features. The effect varies depending on the display device, but may typically be used to specify line cap and join styles. The default is 0.

Format: ANNOTATION HARDWARE integer

* **ANNOTATION HWTEXT**
Attempt to use hardware facilities to draw labelling text (rather than using the character shapes from the TRI file).

Hardware text is enabled if the integer is missing or 1, disabled if 0 and by default.

Format: ANNOTATION HWTEXT [integer]

* **ANNOTATION JOURNAL**
Allows commands which annotate the display to be stored in a macro. The commands journalled include most ANNOTATION, LABEL, and DRAW commands. The macro may be replayed later to draw the same set of annotations.

Format: ANNOTATION JOURNAL subcommand

- o **ANNOTATION JOURNAL CLOSE**
Closes a previously opened annotation journal macro. Annotation commands are no longer journalled. The macro still remains defined, and may be replayed as required.

  Format: ANNOTATION JOURNAL CLOSE

- o **ANNOTATION JOURNAL OFF**
Temporarily turns off the journalling of annotation commands.

  Format: ANNOTATION JOURNAL OFF

- o **ANNOTATION JOURNAL ON**
Turns back on the journalling of annotation commands after a previous ANNOTATION JOURNAL OFF command.

  Format: ANNOTATION JOURNAL ON

- o **ANNOTATION JOURNAL OPEN**
Subsequent annotating commands are journalled to a named macro. A macro with the given name is created, and subsequent annotating commands (most of ANNOTATION, LABEL, and DRAW subcommands) are automatically appended to it, unless a plot is currently on. The macro is intended to be replayed by giving its name in the normal way. The macro may be cancelled (using CANCEL MACRO) when no longer required.

  Format: ANNOTATION JOURNAL OPEN macroname

ANNOTATION LEFT
Annotate linear features as near to the bottom left corner of the screen as possible.
(This is the opposite of ANNOTATION START)

Format: ANNOTATION LEFT

* **ANNOTATION LEGEND**
Defines the size, shape and position of the legend drawn by the DRAW LEGEND command.
The first two arguments define the size of the boxes in the legend. They are specified as a proportion of the screen size.
The second two arguments, if present, define the position that the lower left corner of the legend should take up. These are also specified as a proportion of the screen. If the legend will not fit on the screen with its lower left corner at this position, then the legend

is moved so that it lies entirely on the screen.
Note that it may be useful to specify these real arguments as a vulgar fraction. For example if the DRAW LEGEND command gives an error because "n" boxes will not fit on the screen, then the command

    ANNOTATION LEGEND 0.05 1/n

will ensure that the legend will be drawn.

Format: ANNOTATION LEGEND xsize ysize [xprop yprop]

* **ANNOTATION LOCATION**
The labels produced by the DRAW ABSOLUTE, DRAW GEOGRAPHICAL, DRAW POSITION, DRAW SHEET, DRAW TITLE and DRAW TEXT commands will be drawn with the specified position within the text over the locating point for the label.

As for all texts, the location is an integer in the range 0 - 8.

Format: ANNOTATION LOCATION integer

* **ANNOTATION MARK**
Specify the symbol to be used to mark the points of a feature, with the DRAW MARK command. If a positive number is given, this is taken to be a symbol feature code from the FRT file, while a negative number (converted to positive) refers directly to a symbol number in the SRI file. Note that in either case, only the symbol number is used; the colour and size of the mark is controlled by the ANNOTATION SIZE and ANNOTATION COLOUR commands.

Format: ANNOTATION MARK fc
or   ANNOTATION MARK -sc

* **ANNOTATION OFFSET**
Offset the annotation from the locating point by the specified distances (in mm on the screen) in X and Y. These values of X and Y are in the direction of the feature.
The default values are 0.0 in X and 1.0 in Y.

Format: ANNOTATION OFFSET real real

* **ANNOTATION OVERLAY**
Use the specified display overlay for annotations. This overlay is also used for the DRAW AREAREGION, DRAW REGION, DRAW GRID, and WINDOW MAP commands. The specified overlay must have been created by a WORKSTATION OVERLAY command. If 0 is given, then annotations will revert to their default behaviour of being written to all planes of the display. This command may be used for example to ensure that annotations are written into the same display overlay as vector data and do not overwrite parts of an image backdrop. Alternatively, annotations could be written into an overlay reserved for the purpose which could then be REVEALed or CONCEALed as required.

Format: ANNOTATION OVERLAY integer

* **ANNOTATION SCREEN**
  Draw the annotation at the edge of the screen.
  (This is the opposite of ANNOTATION WINDOW)

  Format: ANNOTATION SCREEN

* **ANNOTATION SIZE**
  Draw the annotation at the specified size (in screen mm).

  Format: ANNOTATION SIZE real

* **ANNOTATION START**
  Annotate linear features where they enter the window.
  (This is the opposite of ANNOTATION LEFT)

  Format: ANNOTATION START

* **ANNOTATION WIDTH**
  Draw line work in the annotation at the specified thickness (in screen mm).
  In text and symbol components this value will be overridden if there is a non-zero entry in the WIDTH field of the SCT entry for that component in the current FRT.

  Format: ANNOTATION WIDTH real

* **ANNOTATION WINDOW**
  Draw the annotation at the edges of a window defined by the position of the cursor. By default, the size of this window is 0.5 times the size of the screen. The optional argument to this command can be used to alter the size of this window. It is the fraction of the full screen that should be used.
  (This is the opposite of ANNOTATION SCREEN)

  Format: ANNOTATION WINDOW [real]

7.16 **Exiting Commands**

7.16.1 **DUMP**

Finishes and saves workspace IFF file(s).
Renames .WRK workspace file as .DMP. if no filename given. Useful if just stopping for lunch and intend to resume LITES2 after: the session may be resumed with e.g. IFF LSL$LITES2WORK:name.DMP
If DISABLE EXIT (qv) was used, LITES2 will return to INITIAL state in preparation for reading in different map(s).

Format: DUMP [filename] (full command required)

Valid in state READY

7.16.2 **EXIT**

Finishes editor session and creates new version of IFF file(s).
If SELECT OUTPUT has been given, then only the selected features are output.
A filename for the output may be specified. If a filename is specified and more than one map has been selected for output, the selected files will be merged together to form one file. This merged file will have a valid type 2 map descriptor, having the scale of the first map that it contains. Its map header will be blank.

If DISABLE EXIT (qv) was used, LITES2 will return to INITIAL state in preparation for reading in different map(s).

Format: EXIT [filename] (full command required)

Valid in state READY

7.16.3 **QUIT**

Finishes and ignores all editing, or discards a particular map. Deletes the .WRK workspace file.

If used without an integer argument, then all maps are discarded. If DISABLE EXIT (qv) was used, LITES2 will return to INITIAL state in preparation for reading in different map(s). If already in INITIAL state, the session is always ended.

Used with the optional integer (a map number), only that map is discarded, and LITES2 remains in the same state. To save the edits made to a map before discarding it, use the WRITE command, possibly in combination with SELECT MAP and SELECT OUTPUT if more than one map is in use. The map being unloaded will still remain on the screen until a redraw operation is performed (except on a display with segment store and ENABLE SEGMENTS, in which case the map will be erased from the screen).

If the map unloaded is not the one with the highest map number, then a "hole" will remain in the list of map numbers in use. The system variable $MAPTOTAL still holds with highest map number in use, which may not be the same as the total number of maps in use. When a new map is read, it will use the lowest available map number. The system variable $MAPSTATUS may be used to determine the status of a particular map number.

If all maps are unloaded while still in READY state, then when the first map is read, the LITES2 working area will be set to its RANGE (plus the usual 5% all round), possibly modified by a RANGE LIMITS command, and the ranges of any raster images in use.

Format: QUIT [integer] (full command required)

Valid in states INITIAL READY

7.16.4 **WRITE**

Creates new version of IFF file(s) from the current workspace file(s) as for EXIT. The editing session is not terminated.
May be used to output the current state of editing, or to create subsets of a file using SELECT OUTPUT.
A filename for the output may be specified. If a filename is specified and more than one map has been selected for output, the selected files will be merged together to form one file. This merged file will have a valid type 2 map descriptor, having the scale of the first map that it contains. Its map header will be blank.

WRITE may be used to produce new versions of READONLY or INSITU files, possibly with selections, whereas EXIT has no effect on these.

Format: WRITE [filename] (full command required)

Valid in state READY

7.17 **Miscellaneous Commands**

7.17.1 **NULL**

The null operation. Nothing is done.

Format: NULL

Valid in all states

7.17.2 **DEBUG**

The LITES2 program debug collapse routine is entered. This is intended as a program development tool. If entered by accident then give command GO to continue.

Format: DEBUG integer (full command required)

Valid in all states

7.17.3 **SECTOR**

Sets the number of sectors in x and y directions for the spatial index. This is a program tuning aid and is not recommended for operator use.

Increasing the number of sectors (up to the point where, on average, there is one feature in each sector) will increase the speed of finding and windowing operations; however there will be a increase in the time taken to read maps into LITES2, and in the time taken to initialise searches and redrawing the entire LITES2 area.

Increasing the number of sectors uses more system resources, and LITES2 may exit with a "Insufficient virtual memory" error, while reading in the map(s) or carrying out editing operations. In this event either use fewer sectors, or see your system manager to get your quotas increased.

The default state is the equivalent of a SECTOR 30 30 command. When editing single standard map sheets, it is not usually productive to increase the sectors to more than SECTOR 50 50, and in fact a performance improvement may be noticed in some applications if the number of sectors is reduced below the default. The command DRAW GRID will display the sector grid as an aid to determining if the current setting is sensible.

Format: SECTOR integer integer

Valid in state INITIAL

7.17.4 **RANGE**

Specifies an area to be sectored which is not the default area. By default the area that will be sectored is the combined range of all maps and images specified in INITIAL state, plus 10% (i.e. 5% of the combined range is added to all edges).

Format: RANGE subcommand

Valid in states INITIAL READY

* **RANGE LIMITS**
  Specifies the area that will become the LITES2 coordinate space. The coordinates are specified as full projection coordinates i.e. coordinates that include any origin offset specified in the IFF files.

  This command is intended for use when further maps will be read in READY state. It may be used in READY state only when there are no maps currently in use.

  NOTE

  1. Used in INITIAL state, the specified range will be expanded by the range of maps and images also specified in INITIAL state. This expanded range will then be further expanded by 10%.

  2. Used in READY state, the specified range will be expanded by the range of the first map specified subsequently and any images in use when the map is specified. Again, this expanded range will be further expanded by 10%.

  3. Any expanded range will persist on return to INITIAL state.

  Format: RANGE LIMITS xmin xmax ymin ymax

* **RANGE SECTOR**
  Specifies an area to be sectored which is not the default area. The sector mechanism is used to provide the spatial index and fast find mechanism.

  The coordinates are specified as full projection coordinates i.e. coordinates that include any origin offset specified in the IFF files.

  This command is not yet implemented

  Format: RANGE SECTOR xmin xmax ymin ymax

7.17.5 SAVE

Saves requested information in a file or macro.

Format: SAVE subcommand

Valid in all states

* **SAVE COLOURS**
  Saves the colour lookup table from the specified LITES2 display as a LITES2 command file containing a series of OVERLAY COLOUR commands. Missing parts of the filename are filled in from LSL$LITES2CMD:---.LCM The intention is that the command file can be used to set up the same colours in a later LITES2 session, in order to re-display a saved picture (see also SAVE DISPLAY command). The overlay structure of the display is not preserved - in order to use the command file, a single overlay using all the planes of the display should be created.
  This command is only available with some versions of LITES2.

  Format: SAVE COLOURS n filename

  Not valid in INITIAL state

* **SAVE DISPLAY**
  Saves the screen picture from the specified LITES2 display in a DTI file. Missing parts of the filename are filled in from LSL$DTI:---.DTI The intention is that the DTI file can be drawn later by LITES2 or other Laser-Scan utilities (see also SAVE COLOURS and SAVE LUT commands).
  This command is only available with some versions of LITES2.

  Format: SAVE DISPLAY n filename

  Not valid in INITIAL state

* **SAVE LUT**
  Saves the colour lookup table from the specified LITES2 display as a file suitable for use with other Laser-Scan programs (e.g. ROVER in the TVES package). Missing parts of the filename are filled in from LSL$LITES2CMD:---.COL The intention is that the lookup table can be used to set up the same colours in other Laser-Scan utilities, in order to re-display a saved picture (see also SAVE DISPLAY command).
  This command is only available with some versions of LITES2.

  Format: SAVE LUT n filename

  Not valid in INITIAL state

* **SAVE MACRO**
  Writes the specified macro to a file called "LSL$LITES2CMD:macroname.LCM". On later runs of LITES2 this macro can be defined again by entering @macroname.

  Format: SAVE MACRO macroname

* **SAVE SECTORS**
  Writes sector information to file (Program development aid).

  Format: SAVE SECTORS

* **SAVE SELECTIONS**
  Create a macro with the given name containing LITES2 commands to restore the selections to the state when the command was given. If selection-list is given, it should be a list of one or more of MAPS, LAYERS, FCS, FSNS, ALL, to specify which selections are to be saved. The default is ALL. Only selections for these 4 items are saved.

  Format: SAVE SELECTIONS macroname [selection-list]

  eg SAVE SELECTIONS allsel
  or SAVE SELECTIONS codesel LAYERS FCS

7.17.6 **VIEW**

Viewing commands make use of a shared image pointed at by the logical name LSL$LITES2_VIEW_ROUTINES. This image is supplied by Laser-Scan. It is called LSL$EXE:LITES2VIEWSHR.EXE.

The VIEW command controls the generation and display of 3 dimensional pictures (called views) from DTI images which represent the terrain. Each view can consist of up to three levels; these levels may either contain terrain information or vector information (from the IFF files).

An appropriate licence is required to use this command. As this command depends on the existence of DTI images, the licence for the IMAGE command is also required.

This command is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware.

Before a view can be produced its spatial context and IFF content must be defined. The basis for the specification of the view spatial context is the view cone. Commands are provided to locate and define the dimensions of this view cone in reference to the DTI and IFF files containing terrain and vector data to be included in the view. LITES2 will not permit the user to render a view until the spatial context of the view is adequately defined via one of two routes:

1. The user specifies an observer position, a cone angle, and a target position to locate and orient the centreline of the view cone using the VIEW FROM x y z, VIEW CONE angle, and VIEW TO x y z commands.

2. The user specifies an observer position, a cone angle, a maximum distance of view, an elevation for the centreline of the cone of vision and a bearing along which the observer "looks".

In both cases, optional front and back clip planes for the cone of vision may be specified.

Format: VIEW subcommand

Valid in all states

* **VIEW AMBIENT**
  Specifies the characteristics of the ambient lighting of a view.

  Format: VIEW AMBIENT subcommand

  o **VIEW AMBIENT COLOUR**
    Specifies the colour and brightness of the ambient light. The colour value is specified as a red, green and blue triplet, all of which must lie in the range 0.0 - 1.0. Thus VIEW AMBIENT COLOUR simultaneously specifies the ambient light colour and brightness. By default VIEW AMBIENT COLOUR 0.0 0.0 0.0 is used, i.e. no ambient light

    Format: VIEW AMBIENT COLOUR r g b

* **VIEW BACK**
  Specifies the distance to the back clip plane of the view; i.e. the maximum distance that the observer can see assuming that obstacles to view or depth/fog effects do not intervene.

  The distance (specified in IFF units) is the horizontal distance from the observer's position measured along the view direction. Note that this clipping plane is a straight line across the viewing cone perpendicular to the viewing direction at this distance; it is NOT the arc of a circle about the observer's position. If this command is not given, the distance to the target position is used.

  Format: VIEW BACK distance

* **VIEW BEARING**
  Specifies the view bearing.

  The angle, measured clockwise (in degrees) from North, at the observer's position to the target position. If the observer's position and the target position are defined, this command will alter the target position.

  Format: VIEW BEARING angle

* **VIEW CLEAR**
  Clears the data from the specified level in the current view.
  This allows new data to be rendered into this level.
  If no level is specified, all levels are cleared.

  Format: VIEW CLEAR [level]

* **VIEW COLOUR**
  Specifies the colours to be used to display various parts of the view.

Format: VIEW COLOUR subcommand

- o **VIEW COLOUR FOREGROUND**
  Specifies the colour to be used to display the foreground in the view. This is that part of the view that lies in front of the front clipping plane.

  The colour value is specified as a red, green and blue triplet, all of which must lie in the range 0.0 - 1.0. By default VIEW COLOUR FOREGROUND 0.0 0.0 0.0 is used.

  Format: VIEW COLOUR FOREGROUND r g b

- o **VIEW COLOUR SEA**
  Specifies the colour to be used for representing sea. This is that part of the view terrain that lies at sea level (value 0).

  The colour value is specified as a red, green and blue triplet, all of which must lie in the range 0.0 - 1.0. By default VIEW COLOUR SEA 0.0 0.35 0.55 is used.

  Format: VIEW COLOUR SEA r g b

- o **VIEW COLOUR SKY**
  Specifies the colour to be used for representing sky. This is that part of the view that lies beyond the back clipping distance of the model.

  The colour value is specified as a red, green and blue triplet, all of which must lie in the range 0.0 - 1.0. By default VIEW COLOUR SKY 0.75 0.75 0.75 is used.

  Format: VIEW COLOUR SKY r g b

* **VIEW CONE**
  Specifies the field of view (in degrees). The angle must be less than 180 degrees. The view "cone" is actually a pyramid of rectangular section with the observer at the apex of the pyramid. It is not a true cone. A default value of 45 degrees is used.

  Format: VIEW CONE angle

* **VIEW CREATE**
  The current view is to be created with the specified number of levels. The maximum number is 3. By default the view is are created with one level.

  Format: VIEW CREATE [levels]

* **VIEW DELETE**
  Deletes the current view. This allows another picture to be rendered into the current view. This command is similar in its effects to VIEW CLEAR.

  Format: VIEW DELETE

* **VIEW DEPTH**
  Controls the effect of depth cueing. Depth cueing causes the interpolation of view pixel colours between a starting value and the sky colour. Depth cueing gives the effect of haze or greying with increasing distance from the observer.

  Format: VIEW DEPTH subcommand

  o **VIEW DEPTH OFF**
    Switches depth cueing off (default).

    Format: VIEW DEPTH OFF

  o **VIEW DEPTH ON**
    Switches depth cueing on.

    Format: VIEW DEPTH ON

  o **VIEW DEPTH DISTANCE**
    Specifies the distance, measured in IFF units, from the observer beyond which the interpolation of pixel colour between starting value and sky colour results in all pixels assuming sky colour. By default a distance of 6000.0 is assumed.

    Format: VIEW DEPTH DISTANCE distance

* **VIEW DISPLAY**
  Displays the current level of the current view in the specified display and overlay (default).

  Optionally, a specific level may be supplied which need not be the current view level.

  Alternatively, multiple levels of the current view may be supplied, either as specific view level numbers separated by commas (i,j,k) or as ranges (n-m). This enables the user to direct the contents of more than one view level into a single display overlay. Higher number view levels take precedence over low numbered levels to ensure that the user can determine how the display pixels are coloured. It is clearly important to place the rendered terrain image in view level 1 and rendered IFF selections in higher numbered levels.

  The current (or specified) level of the current view must have been generated before it can be displayed.

  Format: VIEW DISPLAY displaynumber overlaynumber [viewlevel [,...]]

* **VIEW DISTANCE**
  Specifies the horizontal viewing distance, i.e. the maximum distance that the observer can see assuming that obstacles to view or depth/fog effects do not intervene.

  The distance is measured in IFF units, from the observer's position to the target position. If the observer's position and the target position are defined, this command will alter the target position. If a VIEW

FROM and VIEW TO command is used and no VIEW DISTANCE command is specified, the view distance defaults to the distance between the observer and the target.

Format: VIEW DISTANCE value

* **VIEW DTI**
Outputs all levels of the current view to the specified DTI file.

Format: VIEW DTI filename

* **VIEW ELEVATION**
Specifies the elevation of the centreline of the cone of vision.

The angle, measured in degrees in a vertical plane, at the observer's position to the target position. The angle is positive if the target is above the observer and negative if the target is lower than the observer. If the observer's position and the target position are defined, this command will alter the height of the target position.

Format: VIEW ELEVATION angle

* **VIEW EXAGGERATE**
Specifies the view Z exaggeration.

The Z exaggeration factor to be used to emphasise the height of the terrain when creating a view. By default an exaggeration of 1.0 (i.e. no exaggeration) is assumed. Note that the exaggeration factor is also applied to IFF features rendered as solid objects.

Format: VIEW EXAGGERATION factor

* **VIEW FC_OFFSET**
Specifies a feature code offset to allow features to be rendered using a different style from when they are displayed in plan. The value specified must be between 0 and 32767.

When rendering features, if the feature code is less than the specified offset and there is an entry in the FRT file with a feature code of (current feature code + offset) that has the same graphical type as the original feature code, then the feature will be rendered using this offset feature code.

Format: VIEW FC_OFFSET offset

* **VIEW FOG**
Controls the use and effects of fog while generating a view. Commands are provided to enable the specification of a fog cube clipped to front, back, horizontal bottom and top planes, within which fog effects are applied. The fog "visibility", the rate at which the pixel colours are modified to the fog colour, is applied within this cube.

Format: VIEW FOG subcommand

- **VIEW FOG BACK**
  Specifies the (horizontal) distance, measured in IFF units, from the observer to the back clip plane of the fog cube within which fog effects are applied. By default a VIEW FOG BACK distance of 30000.0 is assumed.

  Format: VIEW FOG BACK distance

- **VIEW FOG BOTTOM**
  Specifies the height, measured in IFF units above the horizontal datum, of the bottom clip plane of the fog cube within which fog effects are applied. Note that it is NOT the height above the terrain. By default a VIEW FOG BOTTOM height of 100.0 is assumed.

  Format: VIEW FOG BOTTOM height

- **VIEW FOG DISTANCE**
  Specifies the distance, measured in IFF units, within which the interpolation of pixel colour between starting value and fog colour results in all pixels assuming fog colour. This enables the user to specify the fog "visibility". The interpolation uses an exponential decay function. It is applied within the fog cube with effect from the intersection between the line of view and the fog cube clip plane closest to the observer. By default a distance of 600.0 is assumed.

  Format: VIEW FOG DISTANCE distance

- **VIEW FOG FRONT**
  Specifies the (horizontal) distance, measured in IFF units, from the observer to the front clip plane of the fog cube within which fog effects are applied. By default a VIEW FOG FRONT distance of 100.0 is assumed.

  Format: VIEW FOG FRONT distance

- **VIEW FOG OFF**
  Switches off the effects of fog, (default).

  Format: VIEW FOG OFF

- **VIEW FOG ON**
  Switches on the effects of fog.

  Format: VIEW FOG ON

- **VIEW FOG TOP**
  Specifies the height, measured in IFF units above the horizontal datum, of the top clip plane of the fog cube within which fog effects are applied. Note that it is NOT the height above the terrain. By default a VIEW FOG TOP height of 5000.0 is assumed.

  Format: VIEW FOG TOP height

* **VIEW FROM**
  Defines the observer's position.

  This position is specified in IFF units in LITES2 coordinate space. It is possible to position the observer outside the area covered by any images or maps that have been read into LITES2. Note that the z argument is the observer's height above the horizontal datum, NOT the observer's height above the terrain.

  Format: VIEW FROM x y z

* **VIEW FRONT**
  Specifies the (horizontal) distance, measured in IFF units, from the observer to the front clip plane of the view. By default a VIEW FRONT distance of 1.0 is assumed.

  Format: VIEW FRONT distance

* **VIEW GENERATE**
  Generates the current level of the current view. The VIEW GENERATE command generates the colour for each pixel in the view on the basis of the intrinsic colour for that pixel. The intrinsic colour for each pixel representing terrain rendered from a DTI file is determined by the colour method specified by the VIEW METHOD command. The intrinsic colour for pixels representing an IFF feature is taken from the FRT colour definition for the feature code defining that IFF feature.

  Optionally, one or more of the following colour modifiers may be specified, the effects of which are computed by the VIEW GENERATE command:

      fog, activated by VIEW FOG ON

      depth, activated by VIEW DEPTH ON

      illumination, activated by VIEW ILLUMINATION ON

  If colour modifiers are selected VIEW COLOUR commands may be used to specify colour values for sea, sky, fog, constant land colour and foreground.

  VIEW GENERATE command execution is slowed by the application of one or more colour modifier options. If the user wishes merely to determine the content of a view it can be quickly produced by generating with no colour modifiers. In the absence of colour modifiers the VIEW INDEX commands may be used to specify colour indices for sea, sky, constant land colour and foreground.

  If FOG, ILLUMINATION or DEPTH have been switched on, VIEW GENERATE must be preceded by one or more VIEW PALETTE commands.

  The VIEW GENERATE command must be given after a level has been rendered, and before the view can be displayed with a VIEW DISPLAY command. Several views may be generated and displayed from one rendered view. Different VIEW AMBIENT, VIEW COLOUR, VIEW DEPTH, VIEW DISTANCE, VIEW FOG, VIEW ILLUMINATION, VIEW LIGHT, VIEW METHOD, VIEW PALETTE, and

VIEW SPHERE commands may be given between successive VIEW GENERATE commands without re-rendering the view.

Format: VIEW GENERATE

* **VIEW ILLUMINATION**
  Controls the effect of illumination on a view.

  Format: VIEW ILLUMINATION subcommand

  o **VIEW ILLUMINATION OFF**
    Switches the effect of illumination off (default).

    Format: VIEW ILLUMINATION OFF

  o **VIEW ILLUMINATION ON**
    Switches the effect of illumination on.

    Format: VIEW ILLUMINATION ON

* **VIEW LEVEL**
  Specifies the level of the view which is to be made "current" upon which following VIEW RENDER, VIEW GENERATE and VIEW DISPLAY commands will act. A single view can have up to three levels. It is strongly recommended that the terrain component of a view (rendered from the DTI file) is placed in level one and that overlay information (e.g. rendered from IFF file) is placed in higher view levels. This will enable the user to use the VIEW DISPLAY facility to draw multiple view levels into the same display overlay. Higher number view levels take precedence over low numbered levels to ensure that the user can determine how the display pixels are coloured.

  Format: VIEW LEVEL number

* **VIEW INDEX**
  Specifies which colour index from the output display colour table is to be used for explicit themes in the view, i.e. foreground, constant land colour, sea, and sky. These indices are applied only when a view is generated with no colour modifiers and their application results in very rapid view generation as no colour interpolation is required.

  Format: VIEW INDEX subcommand

  o **VIEW INDEX CONSTANT**
    Specifies which colour index from the output display colour table is to be used to represent the constant land colour when a view is generated with no colour modifiers. The absence of colour modifiers will result in all land appearing as a silhouette in the colour defined by the specified colour index. By default colour index 2 is used.

    Format: VIEW INDEX CONSTANT integer

- o **VIEW INDEX FOREGROUND**
  Specifies which colour index from the output display colour table is to be used to represent the foreground of a view when the view is generated with no colour modifiers. By default colour index 1 is used.

  Format: VIEW INDEX FOREGROUND integer

- o **VIEW INDEX SEA**
  Specifies which colour index from the output display colour table is to be used to represent sea when a view is generated with no colour modifiers. By default colour index 1 is used.

  Format: VIEW INDEX SEA integer

- o **VIEW INDEX SKY**
  Specifies which colour index from the output display colour table is to be used to represent sky when a view is generated with no colour modifiers. By default colour index 0 is used.

  Format: VIEW INDEX SKY integer

* **VIEW LIGHT**
  Allows the operator to specify up to 5 light sources to be used when rendering an image file.

  At present only directional light sources are implemented, so only a bearing and elevation is required to define a light. It is possible to define these by specifying the position of the light source and of its target point.

  A light is activated by giving the command VIEW LIGHT NUMBER and is deactivated with the VIEW LIGHT DELETE command.

  Format: VIEW LIGHT subcommand

- o **VIEW LIGHT BEARING**
  Specifies light bearing.

  The angle, measured clockwise (in degrees) from North, at the light source position to the target position. If the source position and the target position are defined, this command will alter the target position. A default light bearing of 135.0 is assumed.

  Format: VIEW LIGHT BEARING angle

- o **VIEW LIGHT COLOUR**
  Specifies the colour and brightness of the light. The colour value is specified as a red, green and blue triplet, all of which must lie in the range 0.0 - 1.0 . Default values of 1.0 1.0 1.0 are used.

  Format: VIEW LIGHT COLOUR r g b

- o **VIEW LIGHT CONE**
  Specifies the current light cone angle. The value is given in degrees in the range 0.0 - 180.0. As only directional light sources are used at the moment, this command has no effect.

  Format: VIEW LIGHT CONE angle

- o **VIEW LIGHT DELETE**
  Deselects the current light source, selected with a VIEW LIGHT NUMBER command, from the list of available light sources, but retaining its parameters for future reselection.

  Format: VIEW LIGHT DELETE

- o **VIEW LIGHT DISTANCE**
  Specifies the light distance.

  The distance from the light source position to the light target position. If the source position and the target position are defined, this command will alter the target position.

  Format: VIEW LIGHT DISTANCE value

- o **VIEW LIGHT ELEVATION**
  Specifies the light elevation.

  The angle, measured in the vertical plane, at the light source position to the light target position. The angle is positive if the target position is above the source position. If the source position and the target position are defined, this command will alter the target position.

  A default light elevation of -22.5 degrees is assumed.

  Format: VIEW LIGHT ELEVATION angle

- o **VIEW LIGHT FROM**
  Specifies the position of a light source.

  The coordinates are given in IFF units and are specified in the LITES2 coordinate space.

  Format: VIEW LIGHT FROM x y z

- o **VIEW LIGHT NUMBER**
  Succeeding VIEW LIGHT commands refer to this light source which is added to the list of light sources for use in the illumination model with the VIEW ILLUMINATION ON command. By default, light source number 1 is available and selected, with the default parameters listed here.

  Format: VIEW LIGHT NUMBER n

- o **VIEW LIGHT TO**
  Specifies the target of a light source.

The coordinates are given in IFF units and are specified in the LITES2 coordinate space.

Format: VIEW LIGHT TO x y z

* **VIEW METHOD**
The intrinsic pixel colours for the terrain component of a view are derived using one of four colour allocation methods:

CONSTANT
HEIGHT
IMAGE
RANDOM

Optionally, the intrinsic pixel colours provided by these method may, in turn, be subjected to colour modification using one or more of the following colour modifier options:

DEPTH
FOG
ILLUMINATION

The view colour method must be specified before the VIEW GENERATE command. By default VIEW METHOD CONSTANT is assumed.

Format: VIEW METHOD subcommand

o **VIEW METHOD CONSTANT**
Specifies that a constant intrinsic colour should be used for terrain (i.e. DTI height > 0) rendered from a DTI file. The constant land colour applied by VIEW METHOD CONSTANT is specified using the VIEW COLOUR CONSTANT or VIEW INDEX CONSTANT commands. This is the default colouring method.

Format: VIEW METHOD CONSTANT

o **VIEW METHOD HEIGHT**
Specifies that the intrinsic colour used for terrain shall be derived from the height classification scheme applied to the source DTI file at the time of rendering. The source DTI file classification is defined using IMAGE STEP and IMAGE BAND commands.

Format: VIEW METHOD HEIGHT

o **VIEW METHOD IMAGE**
Specifies that the intrinsic colour used for the terrain be derived from the height classification scheme applied to specified DTI file(s). The source DTI file classification is defined using IMAGE STEP and IMAGE BAND commands. This command enables the user to "drape" DTI files over the rendered surface.

Format: VIEW METHOD IMAGE range

o **VIEW METHOD RANDOM**
Specifies that the intrinsic colour used for terrain shall be derived using a random colour generator.

Format: VIEW METHOD RANDOM

* **VIEW NUMBER**
Succeeding VIEW commands refer to this view. Currently LITES2 supports a maximum of two views.

Format: VIEW NUMBER n

* **VIEW PALETTE**
Specifies the colour palettes used by the VIEW GENERATE command.

Format: VIEW PALETTE subcommand

o **VIEW PALETTE AUTO**
Automatically re-samples the colour values of the pixels for the next picture to be generated. It uses the current view colour modifiers for the calculation of the colour values. By default all the colours available for the specific display overlay are calculated.

Alternatively, specific colours may be supplied either as individual colour index numbers separated by commas (i,j,k) or as ranges (n-m).

Format: VIEW PALETTE AUTO displaynumber overlaynumber [range]

o **VIEW PALETTE DISPLAY**
Specifies the colour palette to be used for the final display of the generated view. The colour palette should be loaded from the display and overlay in which the view is eventually to be displayed. The display colour palette is used as follows:

1. When no colour modifiers are selected the colour index derived for each pixel in the view is looked up directly in the display colour palette. This is very quick.

2. When colour modifiers are specified for the VIEW GENERATE command a starting rgb triplet is used for the intrinsic colour for each pixel. This rgb value is then subjected to the specified colour modification , e.g. depth, and the display colour palette is searched for the match to the resulting rgb value. It is the value identified in the palette that is actually used to display the pixel, not the calculated value which resulted from the colour modification process. It is important to load a display palette which contains colours relevant to the view options. It should, for example, contain some greys and white if the fog modifier is selected.

Format: VIEW PALETTE DISPLAY display overlay

- o **VIEW PALETTE IFF**
  Specifies the colour palette in which the FRT colour index of IFF features will be looked up to get their rgb values at the start of the view generation process. If IFF features are to be given the same intrinsic colours in the view display as in the planimetric display, the IFF colour palette should be loaded from the planimetric display and overlay in which the IFF data are drawn.

  Format: VIEW PALETTE IFF display overlay

- o **VIEW PALETTE IMAGE**
  Specifies the colour palette from which the colour index of terrain pixels will be looked up to get their rgb values at the start of the view generation process. It is also used to lookup the colour indices of DTI files which are to be used by the image colour method.

  If the terrain is to be given the same intrinsic colours in the view display as in the planimetric display, the image colour palette should be loaded from the planimetric display and overlay in which the DTI terrain data are drawn.

  Format: VIEW PALETTE IMAGE display overlay

* **VIEW PIXELS**
  Specifies the size of the view to be created.

  This will often be the size of a display that has already been created. The size of a display (in screen pixels) are available in the system variables $DISPLAYCOLUMNS and $DISPLAYROWS.

  Format: VIEW PIXELS columns rows

* **VIEW RENDER**
  Activate the rendering phase of view production.

  Format: VIEW RENDER subcommand

  - o **VIEW RENDER IFF**
    Render a view of IFF data into the current level of the current view.

    All IFF features that are currently selected and fall in the window specified by the VIEW WINDOW command, are rendered against all the images selected by the IMAGE SELECT command.

    Texts and symbols are positioned at their correct positions in the view. They are made to "stand up" or "lie down" according to the flags defined for their feature code in the FRT file.

    Symbols that form part of patterned lines and patterned fill areas however "lie down" on the terrain.

    Fill areas that are defined as "patterned fill areas" in the FRT, are rendered as a series of vectors. Other fill areas are all infilled with solid colour.

Optionally, IFF lines and area features may be rendered to make "solid objects" in the view. The feature must contain at least one height at each coordinate to offset the feature from the terrain surface. This height value may be interpreted in a number of different ways according to the feature code flags set in the FRT file, (see the FRTLIB user guide for further details of flag interpretation), and the z-value interpretation option selected in LITES2, (see VIEW Z_INTERPRETATION). Typically the height is used to raise the line (or the perimeter of an area) above the terrain surface. Vertical "fences" are then projected down from the raised linework to the terrain surface. The "fences" are then solid filled and in the case of an area a "lid" is rendered to give the object an impression of being "solid".

Format: VIEW RENDER IFF

o **VIEW RENDER IMAGE**
Generate a view of terrain.

All DTI images that have been opened by LITES2 and selected by the IMAGE SELECT command are used to create a view in the currently selected level.

Format: VIEW RENDER IMAGE

* **VIEW ROLL**
Specifies a roll angle, expressed in degrees from the horizontal, for the view. This has the effect of rotating the horizon of the view. By default a roll value of 0.0 is assumed.

Format: VIEW ROLL angle

* **VIEW SAMPLE**
Specifies a sampling interval in terms of columns and rows of the DTI images being rendered. This has the effect of speeding up the rendering, at the expense of the quality of the final image. By default all DTI columns and rows are used.

Format: VIEW SAMPLE columns rows

* **VIEW SPHERE**
Specifies a sphere constructed around the observer within which no colour modifiers are applied during the VIEW GENERATE phase of view production.

Format: VIEW SPHERE subcommand

o **VIEW SPHERE DISTANCE**
Specifies the radius of a sphere, expressed in IFF units, constructed around the observer within which no colour modifiers are applied during the VIEW GENERATE phase of view production. This enables the user to specify a zone of "perfect visibility" within the view. By default a sphere distance of 1.0 is assumed.

Format: VIEW SPHERE DISTANCE distance

* **VIEW TO**
  Defines the target position for the view.

  This position is specified in IFF units in LITES2 coordinate space. It is possible to position the target outside the area covered by any images or maps that have been read into LITES2. Note that the z argument is the target height above the horizontal datum, NOT the target height above the terrain.

  Format: VIEW TO x y z

* **VIEW WINDOW**
  Defines the area to be used when generating a view.

  Format: VIEW WINDOW subcommand

  o **VIEW WINDOW LIMITS**
    The window is explicitly specified in IFF units, in terms of the LITES2 coordinate system.

    Format: VIEW WINDOW LIMITS xmin xmax ymin ymax

  o **VIEW WINDOW MAP**
    The window is the whole of the LITES2 working area.

    Format: VIEW WINDOW MAP

  o **VIEW WINDOW SCREEN**
    The window is the area currently displayed in the LITES2 primary display. This is the default windowing method.

    Format: VIEW WINDOW SCREEN

* **VIEW Z_INTERPRETATION**
  Specifies how Z coordinates and differences in height are to be interpreted when displaying lines and areas as features that are to have a solid vertical component in perspective views.

  This command is also used to specify the two point attribute types that are to be used for this vertical information.

  Format: VIEW Z_INTERPRETATION subcommand acdtype1 acdtype2

  o **VIEW Z_INTERPRETATION ABT**
    The value of the attribute acdtype1 is the absolute height of the bottom of the feature, while the value of the attribute acdtype2 is the absolute height of the top of the feature.

  o **VIEW Z_INTERPRETATION AHD**
    The value of the attribute acdtype1 is the absolute height of the top of the feature, while the value of the attribute acdtype2 is

the difference in height from the top down to the bottom of the feature.

- **VIEW Z_INTERPRETATION AHU**
  The value of the attribute acdtype1 is the absolute height of the bottom of the feature, while the value of the attribute acdtype2 is the difference in height from the bottom up to the top of the feature. This is the default z interpretation, with ACD types 93 and 97.

- **VIEW Z_INTERPRETATION RBT**
  The value of the attribute acdtype1 is the height of the bottom of the feature relative to the ground level, while the value of the attribute acdtype2 is the height of the top of the feature relative to the ground level.

- **VIEW Z_INTERPRETATION RHD**
  The value of the attribute acdtype1 is the height of the top of the feature relative to ground level, while the value of the attribute acdtype2 is the difference in height from the top down to the bottom of the feature.

- **VIEW Z_INTERPRETATION RHU**
  The value of the attribute acdtype1 is the height of the bottom of the feature relative to ground level, while the value of the attribute acdtype2 is the difference in height from the bottom up to the top of the feature.

7.17.7 **SPAWN**

An appropriate licence is required to use this command.
Creates a sub-process to execute a DCL command (which may be an @file command). If doing several commands, then one can SPAWN @TT:, perform the commands, then LOGOUT or CTRL/Z. Unless the /NOWAIT qualifier is given, LITES2 will wait for the subprocess to complete before continuing. If /NOWAIT is used, then the command executed by the subprocess should not normally read from or write to the terminal, since this is likely to result in confusion.

```
Format:  SPAWN [/NOWAIT] command

eg       SPAWN DIRECTORY LSL$IF:
or       SPAWN/NOWAIT UILMENUS EXAMPLE
```

Valid in all states

7.17.8 **WORKSTATION**

Alters facilities on the workstations in use.

Format: WORKSTATION subcommand

* **WORKSTATION COLOUR**
Sets the colour which will be used for a particular colour index. This should not normally be used if display overlays are in use - the OVERLAY COLOUR command should be used instead.
The effect of this command may vary depending on which workstations are in use. Colour index 0 is the background colour, while 1 up to a maximum are foreground colours. Negative colour indices may be used to set the colour of (eg) highlighted features in certain implementations. The red, green, and blue values should be in the range 0.0 to 1.0 and control the fraction of the primary colours in the mixture. In the case of workstations with both primary and secondary colour displays, this command will affect both.

Format: WORKSTATION COLOUR index red green blue

eg WORKSTATION COLOUR 1 0.0 0.5 0.7

Valid in states:
READY LINE CIRCLE TEXT SYMBOL EDIT MODIFY ON WINDOW CONSTRUCT AC RECOVER SETUP

* **WORKSTATION OVERLAY**
Allows the definition of display overlays consisting of a subset of the pixel planes available on the primary or secondary displays. Once defined, the OVERLAY command may be used to select which features are drawn in a particular overlay, and how the overlay appears on the display. The bit planes used in an overlay must be consecutive, and a single plane may only be used in one overlay. The overlay number is used to identify the overlay using OVERLAY NUMBER commands, and must be an integer in the range 1-8. The offset argument specifies the first bit plane to be used (the first plane is 0). If omitted, the first available value allowing the requested number of planes is used. In the case of workstation with primary and secondary displays which both support overlays, the overlays are shared between the two - the creation of an overlay in one automatically creates the same overlay in the other.

This command is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware.

Format: WORKSTATION OVERLAY number overlay planes [offset]

eg WORKSTATION OVERLAY 1 2 4 3
Defines overlay number 2 on workstation 1 (primary) to consist of bit planes 3,4,5, and 6.

Valid in all states

* **WORKSTATION TYPE**
Allows the user to specify the type of workstation being used, where there is the possibility of using slightly different hardware.

This command is only available on some versions of the program, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware. On the other versions it is a null operation.

Format: WORKSTATION TYPE workstation type

eg WORKSTATION TYPE 1 4014
If given in the TEK_ARGS version of LITES2, this command is used to use a TEKTRONIX 4014 terminal (or an emulator) as the primary display, without the use of a MUART.

Valid in state INITIAL

* **WORKSTATION VIEWPORT**
Sets the size and position on the screen of the active graphics area of a particular workstation.
The viewport extents should be in the range 0.0 to 1.0, thereby defining the required proportion of the total screen.
This command may be used in conjunction with the SCROLL and DESCRIBE SCREENMENU commands to separate the text, graphics, and screen menu areas on the screen.

Format: WORKSTATION VIEWPORT workstation Xmin Xmax Ymin Ymax

eg WORKSTATION VIEWPORT 1 0.2 1.0 0.0 1.0

Valid in state INITIAL

7.17.9 **OVERLAY**

Specifies details of the appearance of display overlays. These must first be created using the WORKSTATION OVERLAY or DISPLAY OVERLAY commands. Several of the OVERLAY subcommands require that the overlay to which they are to apply is first specified in an OVERLAY NUMBER command.
For more information about overlays, see the separate section of the manual below.

Format: OVERLAY subcommand

Valid in all states except PAINT

* **OVERLAY ATTRIBUTE**
Sets the attributes of a particular colour index in the selected overlay, or all colours if index is not specified.
Possible attributes are:

TRANSPARENT - the underlying colour shows through. The colour of this colour index is not relevant.

OPAQUE - this colour obscures any underlying colour.

INVERSE - a colour complementary to the underlying colour is shown.
The colour of this colour index is not relevant.

ADD - The underlying colour is combined with this colour so as to produce a lighter colour.

SUBTRACT - The underlying colour is combined with this colour so as to produce a darker colour.

MERGE - The displayed colour is an average of the underlying colour and this colour.

Format: OVERLAY ATTRIBUTE [index] attribute

eg OVERLAY ATTRIBUTE 1 OPAQUE

* **OVERLAY BACKDROP**

Sets the backdrop colour. This is the background colour of the display which is considered to lie behind all the overlays. The colour may be set either using the RGB scheme (the proportions of the primary colours are specified in the range 0-1), the HLS scheme (hue is an angle 0-360 starting as red and moving through green then blue, lightness and saturation are in the range 0-1), or the HSV scheme (hue as for HLS, saturation and value in the range 0-1).

Format: OVERLAY BACKDROP RGB red green blue
or OVERLAY BACKDROP HLS hue lightness saturation
or OVERLAY BACKDROP HSV hue saturation value

* **OVERLAY BLANK**

Sets the colour index to use when blanking behind texts in an overlay. If no colour index is given, then the blanking colour for this overlay is cancelled, and the default colour is used.

See ENABLE BLANK for more details.

When blanking behind texts in an overlay, it is usual to use an opaque colour.

Format: OVERLAY BLANK [colour_index]

* **OVERLAY CLEAR**

Removes all selections of IFF features or raster images from an overlay.

Format: OVERLAY CLEAR

* **OVERLAY CLUT**

Reads in a colour look up table for the selected overlay. This command is not yet implemented.

Format: OVERLAY CLUT filename

* **OVERLAY COLOUR**
Sets the colour of a particular colour index in the selected overlay. Zero is the background colour of the overlay (the colour which will appear in areas where nothing is drawn). The colour may be set either using the RGB scheme (the proportions of the primary colours are specified in the range 0-1), the HLS scheme (hue is an angle 0-360 starting as red and moving through green then blue, lightness and saturation are in the range 0-1), or the HSV scheme (hue as for HLS, saturation and value in the range 0-1).

```
Format:  OVERLAY COLOUR RGB red green blue
or       OVERLAY COLOUR HLS hue lightness saturation
or       OVERLAY COLOUR HSV hue saturation value
```

* **OVERLAY CONCEAL**
Makes the selected overlay invisible (opposite of OVERLAY REVEAL).

Format: OVERLAY CONCEAL

* **OVERLAY DEFER**
Specifies that changes to overlay colours and attributes should not be performed until an OVERLAY UPDATE command is given. This speeds up operation if a number of changes to overlays are to be made.

Format: OVERLAY DEFER

* **OVERLAY DELETE**
Deletes the definition of an overlay, and makes the bit planes available for re-use in WORKSTATION OVERLAY or DISPLAY OVERLAY commands. OVERLAY CLEAR must be used before OVERLAY DELETE.

Format: OVERLAY DELETE

* **OVERLAY DESELECT**
Deselects categories of IFF features and/or raster images from the selected overlay. The format of the command is the same as the ordinary DESELECT command, but only subcommands ALL, FCS, LAYERS, and MAPS, plus the additional subcommand IMAGES are permitted.

Format: OVERLAY DESELECT subcommand

* **OVERLAY ERASE**
Erases (clears) the current overlay in the current display. When primary and secondary displays both support overlays, the overlay will be cleared in both unless a SUPPRESS command (qv) has been used.

Format: OVERLAY ERASE

* **OVERLAY NUMBER**
Selects an overlay number to be used in subsequent OVERLAY commands which operate on a specific overlay. The number must be in the range 1-8 for specific overlays, or 0 which is an 'unset' value and prevents the use of some OVERLAY commands until another number is chosen.

Format: OVERLAY NUMBER n

* **OVERLAY POP**
  Moves the selected overlay to the top of the stack of overlays, so that it will appear 'in front' of all other overlays. The other overlays retain their existing order.

  Format: OVERLAY POP

* **OVERLAY PUSH**
  Moves the selected overlay to the bottom of the stack of overlays, so that it will appear 'behind' all other overlays. The other overlays retain their existing order.

  Format: OVERLAY PUSH

* **OVERLAY REVEAL**
  Makes the selected overlay visible (opposite of OVERLAY CONCEAL).

  Format: OVERLAY REVEAL

* **OVERLAY SELECT**
  Selects categories of IFF features and/or raster images to appear in the selected overlay. The format of the command is the same as the ordinary SELECT command, but only subcommands ALL, FCS, LAYERS, and MAPS, plus the additional subcommand IMAGES are permitted. Features must still be selected by the normal SELECT command in order to appear in any overlays. Nothing will appear in an overlay until an OVERLAY SELECT command is used.

  Format: OVERLAY SELECT subcommand

* **OVERLAY UPDATE**
  Causes the screen display to be updated to reflect changes to overlay colours and attributes made since an OVERLAY DEFER command. After OVERLAY UPDATE, changes will be performed immediately until another OVERLAY DEFER command is given.

  Format: OVERLAY UPDATE

7.17.10 **IMAGE**

An appropriate licence is required to use this command.

Specifies details of raster images. These may be displayed in display overlays using the OVERLAY SELECT IMAGES command. Note that display overlays are not supported by all version of LITES2. Several of the IMAGE subcommands require that the image to which they are to apply is first specified in an IMAGE NUMBER command.
For more information about images, see the separate section of the manual below.

Format: IMAGE subcommand

Valid in all states except PAINT

* **IMAGE BACKGROUND**
Specifies a colour value to be used as background in image editing operations. This is the value that will be written by IMAGE CLEAR, IMAGE ERASE, and IMAGE SPECKLE CLEAR commands, and is the value that the IMAGE BURN_IN command will leave unaltered. It is also the colour of speckles for IMAGE SPECKLE FILL. Since editing is only valid for LSR files containing bit data, the value must be 0 or 1. The default is 0.
Format: IMAGE BACKGROUND integer

* **IMAGE BAND**
Specifies that a range of values in the current image are to be displayed in a particular colour index, overriding the colour obtained from IMAGE STEP. If the two values are the same, or the high value is omitted, then only the single value will be drawn in the given colour index.
Image classification must already have been enabled by an IMAGE STEP command.

The optional label argument is used to annotate the corresponding band in output from the SHOW IMAGE and DRAW LEGEND command. If the label is to start with a number, then the high_value argument may not be omitted.

Format: IMAGE BAND index low_value [high_value] [label]

* **IMAGE BITS**
Specifies which bits from the current image to display. The default is to display the 8 least significant bits i.e. IMAGE BITS 0 8. The first-bit must be in the range 0-31, and the number-of-bits must be in the range 1-8. Image classification is disabled by this command until an IMAGE STEP command is given.

Format: IMAGE BITS first-bit number-of-bits

* **IMAGE BRUSH**
Specifies the size and shape of brush used for IMAGE PAINT and ERASE operations. The default is a circle of size 1.

Format: IMAGE BRUSH subcommand

  o **IMAGE BRUSH CIRCLE**
  Specifies a circular brush. By default the diameter is specified in IFF units (unless a UNITS command has been given).

  Format: IMAGE BRUSH CIRCLE diameter

  o **IMAGE BRUSH CURSOR**
  Specifies whether the screen cursor assumes the shape of the BRUSH during IMAGE PAINT and ERASE operations. The integer argument is 0 to use the default cursor, or 1 to use the brush-shaped cursor.

  Format: IMAGE BRUSH CURSOR on

  o **IMAGE BRUSH RECTANGLE**
  Specifies a rectangular (or square) brush. If the height is omitted, it is taken to be the same as the width, giving a square

brush. By default the size is specified in IFF units (unless a UNITS command has been given).

Format: IMAGE BRUSH RECTANGLE width [height]

* **IMAGE BURN_IN**
Edits the current image by reading back the current screen picture, and changing any pixels which are not the image background colour to be the image foreground colour. The effect is to burn any vector detail, or annotations into the image. This command requires that the image be displayed such that 1 image pixel corresponds to 1 screen pixel, which can be achieved using the ZOOM 1 IMAGE command if required. Note that only the current screen area is affected - to edit other areas, they have to be drawn on the screen first. Valid only for bit LSR files which are open for edit.

Format: IMAGE BURN_IN

* **IMAGE CLEAR**
Fills the interior of the image region with image background colour in the current image. The command may be aborted by CTRL/C if it is taking too long, or has already completed enough of the operation. Valid only for bit LSR files which are open for edit.

Format: IMAGE CLEAR

* **IMAGE CLOSE**
The image file associated with the current image number is closed. The number becomes available for re-use.

Format: IMAGE CLOSE

* **IMAGE CONNECT**
Specifies whether image pixels are taken to be connecting when they touch along their sides (default), or also when they touch diagonally. This affects the commands REGION n IMAGE, and IMAGE SPECKLE.

Format: IMAGE CONNECT subcommand

  o **IMAGE CONNECT DIAGONAL**
  Pixels are connected when they touch either by their sides, or diagonally, so a pixel may be connected to all 8 neighbours.

  Format: IMAGE CONNECT DIAGONAL

  o **IMAGE CONNECT SIDE**
  Pixels are connected when they touch by their sides, and not diagonally, so a pixel may only be connected to 4 neighbours.

  Format: IMAGE CONNECT SIDE

* **IMAGE COPY**
Copies the interior of the current image region to a new position. This command enters EDIT state, attaches the region to the cursor, and

allows it to be moved around until END is given. ABANDON may be used to cancel the operation. The original data is retained. The command may be aborted by CTRL/C if it is taking too long, or has already completed enough of the operation. Valid only for bit LSR files which are open for edit.

Format: IMAGE COPY

Valid in state READY

* **IMAGE CORNER**
Specifies the corner of the image of the first pixel in a DTI file. This is normally taken from the file header. The corner must be one of SW, NW, NE, and SE.

Format: IMAGE CORNER corner

eg IMAGE CORNER NW

* **IMAGE DIRECTION**
Specifies the direction of the first row/column of a DTI file with respect to the image corner. This is normally taken from the file header. The direction must be one of CLOCKWISE, ANTICLOCKWISE.

Format: IMAGE DIRECTION direction

eg IMAGE DIRECTION CLOCKWISE

* **IMAGE DTI**
Specifies the name of a DTI file to be opened. The file is subsequently referred to by its number. The origin and pixelsize for the file are read from the projection record in its header if one is present, otherwise they are set to default values. The corner and direction are also taken from the file header. All these may be subsequently altered using IMAGE ORIGIN, PIXELSIZE, CORNER, and DIRECTION commands. The pixels are assumed to be point type, so the origin is at the centre of the bottom left pixel. Default filename is LSL$DTI:---.DTI;0

Format: IMAGE DTI filename

* **IMAGE EDIT**
Specifies that the current image should be opened for edit. This command should be given after IMAGE NUMBER, but before specifying the image file. Only one image may be opened for edit, and it must (at present) be an LSR type of image containing bit data.
An appropriate licence is required to use this command (in addition to the licence for the overall IMAGE command).
See also IMAGE READONLY.

Format: IMAGE EDIT

* **IMAGE ERASE**
Enters PAINT state, and draws the current image brush in the image background colour into the current image (which must be an editable LSR file), at the cursor position. As the cursor is moved, further copies of the brush are painted, until the END command is given. It is

possible to use the ZOOM command to display a different area while painting is in progress. The ABANDON command will end the operation and restore the image to its state before IMAGE ERASE was given.

Format: IMAGE ERASE

Valid in state READY

* **IMAGE FILL**
Fills the interior of the image region with image foreground colour in the current image. The command may be aborted by CTRL/C if it is taking too long, or has already completed enough of the operation. Valid only for bit LSR files which are open for edit.

Format: IMAGE FILL

* **IMAGE FIRSTCOLOUR**
Specifies the colour index to be used for the first step when image classification is in use. The default value is 0. The IMAGE STEP command must be given first.

Format: IMAGE FIRSTCOLOUR index

* **IMAGE FOREGROUND**
Specifies a colour value to be used as foreground in image editing operations. This is the value that will be written by IMAGE FILL, IMAGE PAINT, IMAGE BURN_IN, and IMAGE SPECKLE FILL commands. It is also the colour of speckles for IMAGE SPECKLE CLEAR. Since editing is only valid for LSR files containing bit data, the value must be 0 or 1. The default is 1.

Format: IMAGE FOREGROUND integer

* **IMAGE LSI**
Specifies the name of an LSI file to be opened. The file is subsequently referred to by its number. The origin and pixelsize for the file are set to default values, but may be subsequently altered using IMAGE ORIGIN, and PIXELSIZE commands. The pixels are taken to be area type, so the origin is at the bottom left of the bottom left pixel. Default filename is LSL$LSI:

Format: IMAGE LSI filename

* **IMAGE LSR**
Specifies the name of an LSR file to be opened. The file is subsequently referred to by its number. The origin and pixelsize for the file are read from its header but may be subsequently altered using IMAGE ORIGIN and PIXELSIZE commands. If the file has area type pixels, then the origin is taken to be the bottom left of the bottom left pixel. If the pixels are point type, then the origin is the centre of this pixel. Default filename is LSL$LSR:

Format: IMAGE LSR filename

* **IMAGE MOVE**
  Moves the interior of the current image region to a new position. This command enters EDIT state, attaches the region to the cursor, and allows it to be moved around until END is given. ABANDON may be used to cancel the operation. The original data is set to the image background colour. The command may be aborted by CTRL/C if it is taking too long, or has already completed enough of the operation. Valid only for bit LSR files which are open for edit.

  Format: IMAGE MOVE

  Valid in state READY

* **IMAGE NUMBER**
  Selects an image number to be used in subsequent IMAGE commands. The IMAGE DTI or IMAGE LSI command is used to open an image file, which is subsequently referred to by its number. The number must be in the range 1-8 for specific images, or 0 which is an 'unset' value and prevents the use of some IMAGE commands until another number is chosen.

  Format: IMAGE NUMBER n

* **IMAGE ORIGIN**
  Specifies the location in absolute units of the SW corner of the current image file. Default is 0.0 0.0 for a DTI file, or half the pixel size in each axis for an LSI file.

  Format: IMAGE ORIGIN x y

* **IMAGE PAINT**
  Enters PAINT state, and draws the current image brush in the image foreground colour into the current image (which must be an editable LSR file), at the cursor position. As the cursor is moved, further copies of the brush are painted, until the END command is given. It is possible to use the ZOOM command to display a different area while painting is in progress. The ABANDON command will end the operation and restore the image to its state before IMAGE PAINT was given.

  Format: IMAGE PAINT

  Valid in state READY

* **IMAGE PIXELSIZE**
  Specifies the size of each pixel in the current image in IFF units. If ysize is omitted, it is assumed to be the same as xsize. The default is IMAGE PIXELSIZE 1 1 for a DTI file, or the value read from the file header for an LSI file.

  Format: IMAGE PIXELSIZE xsize [ysize]

* **IMAGE RANGE**
  Sets the range of values in the current image to be used when image classification is enabled (see IMAGE STEP). Values outside the range will be drawn in colour 0. The range is initially set to the minimum and maximum values in the file by the IMAGE DTI command.

Format: IMAGE RANGE lower upper

* **IMAGE READONLY**
Specifies that the current image should be opened for reading only. This command should be given after IMAGE NUMBER, but before specifying the image file. It is the opposite of IMAGE EDIT, and is the default is neither command is specified.

Format: IMAGE READONLY

* **IMAGE RECOVER**
Restores the current image to its state before the last IMAGE FILL, CLEAR, MOVE, COPY, PAINT, ERASE, BURN_IN, or SPECKLE command. The command IMAGE RECOVER CLEAR may be used to clear out the recover information, reclaiming the memory used, and preventing accidental recovery of an edit that you know is to be permanent.

Format: IMAGE RECOVER
or IMAGE RECOVER CLEAR

* **IMAGE REGION**
Specifies a LITES2 region to be used for IMAGE FILL, IMAGE CLEAR, IMAGE MOVE, IMAGE COPY, and IMAGE SPECKLE commands. The region need not exist when the command is given. 0 means that no region is selected.

Format: IMAGE REGION integer

* **IMAGE SEA**
Specifies a colour index to be used for zero values in the current image. Exactly equivalent to the command IMAGE BAND index 0.0

The optional label argument is used to annotate the corresponding band in output from the SHOW IMAGE and DRAW LEGEND command.

Format: IMAGE SEA index [label]

* **IMAGE SELECT**
Specifies one or more images from which the value of the $IMAGEVALUE, $IMAGEGRADIENT, and $IMAGEASPECT system variables is taken. This is particularly useful when several images overlap. The selected images need not be currently displayed. If the range of image numbers is omitted, then the image selections become null, and another IMAGE SELECT command must be given before the variables can be used again.

Format: IMAGE SELECT [range]

eg IMAGE SELECT 3
or IMAGE SELECT 2,5-7

* **IMAGE SETUP**
Causes LITES2 to enter SETUP state, and to prompt for the user to digitise the 4 corner points of the first map with reference to a raster image on the screen using START commands. WINDOW, DRAW, and ZOOM commands may be used to display the desired area on the screen. Once the setup is completed, LITES2 will return to READY state and subsequent drawing will distort the vector picture so as to match up

the corner points with the raster image. ABANDON may be used at any time in SETUP state to abort the setup. Invoking IMAGE SETUP followed by ABANDON will cancel any existing setup and return to default behaviour. See chapter on Display Overlays and Raster Images.

Format: IMAGE SETUP

Valid in state READY

* **IMAGE SPECKLE**
  Scans the interior of the current image region removing speckles (areas of pixels of a given size or less). The command may be aborted by CTRL/C if it is taking too long, or has already removed the intended speckles. Valid only for bit LSR files which are open for edit.

  Format: IMAGE SPECKLE subcommand

  Valid in state READY

  o **IMAGE SPECKLE CLEAR**
    Replaces speckles of the current image foreground colour with the image background colour. By default the size is specified in IFF units (unless a UNITS command has been given).

    Format: IMAGE SPECKLE CLEAR size

  o **IMAGE SPECKLE FILL**
    Replaces speckles of the current image background colour with the image foreground colour. By default the size is specified in IFF units (unless a UNITS command has been given).

    Format: IMAGE SPECKLE FILL size

* **IMAGE STEP**
  Enables image classification for the current image and sets the step size to the value given (which must be greater than 0.0). If classification was not already enabled, then all bands are cancelled and first colour is set to 0.
  The colour index used for a particular value is given by:

      index = (value - low_end_of_range) / step + first_colour

  Format: IMAGE STEP step

* **IMAGE SUBSAMPLE**
  Specifies how subsampling of the current image should be performed, if the image is drawn at a scale such that 1 display pixel corresponds to more than one image pixel.

  Format: IMAGE SUBSAMPLE subcommand

  o **IMAGE SUBSAMPLE FAST**
    Specifies fast subsampling (default). For DTI and LSR files, this means that rows and columns of the image will be missed out when

drawing. LSI files are never subsampled - instead reduced views are used if they exist in the file.

Format: IMAGE SUBSAMPLE FAST

- **IMAGE SUBSAMPLE PRIORITY**
  Allows a colour value to be specified which takes priority if it occurs within the block of image pixels which correspond to a single display pixel. This can prevent break up of subsampled images which can occur with IMAGE SUBSAMPLE FAST. The command is only valid for LSR type images containing bit data, and the priority value must therefore be 0 or 1.

  Format: IMAGE SUBSAMPLE PRIORITY integer

7.17.11 DISPLAY

Specifies details of displays (additional windows on the display screen).

This command is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware. Displays 1 and 2 (primary and secondary) are created automatically when LITES2 moves from INITIAL to READY state - other displays are created by the DISPLAY CREATE command.

Format: DISPLAY subcommand

Valid in all states

* **DISPLAY BORDER**
  Specifies whether the display is to have a border. The value must be 1 (with border, default), or 0 (no border).
  This command must be given before DISPLAY CREATE.

  Format: DISPLAY BORDER n

* **DISPLAY COLOURS**
  Sets the number of colours to be used for the display (including the background colour). This command must be given before DISPLAY CREATE. If the argument is omitted, then the default number of colours (dependent on the particular LITES2 version) is used.

  Format: DISPLAY COLOURS [n]

* **DISPLAY CONCEAL**
  Makes the selected display invisible (opposite of DISPLAY REVEAL).

  Format: DISPLAY CONCEAL

* **DISPLAY CREATE**
  The selected display is created using the attributes which have been set. The display will appear on the screen, unless in INITIAL state, in

which case if will appear after the maps have been read in.

Format: DISPLAY CREATE

* **DISPLAY CURSOR**
Specifies whether displays 3 or 4 are to have a crosshair cursor. The value must be 1 (with cursor), or 0 (no cursor, default). This command may be given at any time. The command is intended primarily for use in conjunction with DRAW IMAGE, after which the coordinate system in the display will correspond to the image in the main display, enabling accurate pointing to image pixels. At all other times, the coordinate system in the display will correspond to the entire LITES2 working area.

Format: DISPLAY CURSOR n

* **DISPLAY DELETE**
Deletes a display. The display will disappear from the screen, and any definitions of overlays in it will be lost. The attributes remain set and the display may be re-created using DISPLAY CREATE.

Format: DISPLAY DELETE

* **DISPLAY ERASE**
Erases the contents of a display.

Format: DISPLAY ERASE

* **DISPLAY LIMITS**
Sets the coordinate range which will be used to draw into the current display. If the limits do not define an area with the same aspect ratio as the display, then part of the display area will not be used. If this command is not given, or the values are omitted, then the default limits (the same as the LITES2 working area) are used. The display limits may be changed at any time.

Format: DISPLAY LIMITS [xmin xmax ymin ymax]

* **DISPLAY NUMBER**
Selects a display number to be used in subsequent DISPLAY commands. The number must be in the range 1 up to a limit dependent on the version of LITES2 for specific displays, or 0 which is an 'unset' value and prevents the use of some DISPLAY commands until another number is chosen. Display numbers 1 and 2 refer to the primary and secondary workstations. Only some of the the display commands are valid for these.

Format: DISPLAY NUMBER n

* **DISPLAY OVERLAY**
Allows the definition of display overlays consisting of a subset of the pixel planes available on the display. Sets up display overlays using the graphics planes. Overlays in the primary or secondary displays are created using the WORKSTATION OVERLAY command (qv). Once defined, the OVERLAY command may be used to specify how the overlay appears on the display. The bit planes used in an overlay must be consecutive, and a

single plane may only be used in one overlay. The overlay number is used to identify the overlay using OVERLAY NUMBER commands, and must be an integer in the range 1-8. The offset argument specifies the first bit plane to be used (the first plane is 0). If omitted, the first available value allowing the requested number of planes is used.

Format: DISPLAY OVERLAY overlay planes [offset]

eg DISPLAY OVERLAY 2 4
Defines overlay number 2 in the current display to consist of 4 bit planes

* **DISPLAY POP**
Pop the selected display to the front (if it is obscured by other displays).

Format: DISPLAY POP

* **DISPLAY POSITION**
Specify the initial position of the display on the screen. The position of the bottom left hand corner is given as a fraction of the screen size, in the range 0.0 to 1.0.
This command must be given before DISPLAY CREATE.

Format: DISPLAY POSITION xpos ypos

* **DISPLAY PUSH**
Push the selected display to the back (behind any other displays).

Format: DISPLAY PUSH

* **DISPLAY REVEAL**
Makes the selected display visible (opposite of DISPLAY CONCEAL).

Format: DISPLAY REVEAL

* **DISPLAY SIZE**
Specify the size of the display on the screen. The size is given as a fraction of the screen size, in the range 0.0 to 1.0.
This command must be given before DISPLAY CREATE.

Format: DISPLAY SIZE xsize ysize

* **DISPLAY TITLE**
Specifies a title to appear at the top of the display (providing that it has a border). If the title is omitted, then the display will have no title. If the title is to have leading spaces, then it must be enclosed in double quotation marks. The default titles are "Graphics Window" for display 1, "Secondary Window" for display 2, and no title for any other displays.
This command must be given before DISPLAY CREATE.

Format: DISPLAY TITLE [string]

7.17.12 **PLOT**

Controls hardcopy plotting.

This command is only available with some versions of LITES2, and the user should refer to the hardware dependent reference manual for the possibilities available with his hardware.

Format: PLOT subcommand

Valid in all states

* **PLOT ADVANCE**
  Performs a clear operation on the plot device. Depending on the plotter in use, this will advance to a new sheet of paper, or load new film etc.

  Format: PLOT ADVANCE

* **PLOT ANNOTATION**
  Specifies whether sizes set using ANNOTATION SIZE are measured in mm on the plot (PLOT ANNOTATION PLOT, default), or are plotted in the same proportion to the rest of the picture as they would have been on the screen (PLOT ANNOTATION SCREEN).

  Format: PLOT ANNOTATION PLOT
  or PLOT ANNOTATION SCREEN

* **PLOT AUTOSCALE**
  Specifies that IFF and image data drawn into the plot is to be scaled to fit the available plotting area. This is the default. See PLOT SCALE and PLOT RATIO for details of setting an absolute plot scale.

  Format: PLOT AUTOSCALE

* **PLOT CLIP**
  Controls whether annotations sent to the plotter are clipped at the boundary of the drawing area. Specify 1 to clip (default), or 0 not to clip. The picture from the primary or secondary display is always clipped.

  Format: PLOT CLIP n

* **PLOT CLOSE**
  Finishes off a plot and closes the connection to the plot device.

  Format: PLOT CLOSE

* **PLOT ESCAPE**
  Used to communicate a device dependent function to the particular plot device in use. See the Plotters User Guide in the PLOTTING package documentation for details of what functions are available. For example...

Using GKSCAL5800SHR or GKSVRSVGSSHR - escid = 1 uses a drawing mode in which the things drawn subsequently are opaque (things drawn already do not show through). escid = 2 uses a drawing mode in which everything drawn is superimposed (the inks are mixed). If necessary, then draw the picture with some selections in force, then change the selections, give the PLOT ESCAPE command, and draw again.

Format: PLOT ESCAPE escid

* **PLOT HWTEXT**

Attempt to use hardware facilities to plot text (rather than using the character shapes from the TRI file) if the FRT includes a hardware bit in the flags entry for a text feature code. See the Plotters User Guide for details of whether a device supports hardware text.

Attempts to use hardware text on a device which does not support it may result in text not appearing at all.

Hardware text is enabled if the integer is missing or 1, disabled if 0 and by default.

Format: PLOT HWTEXT [integer]

* **PLOT LIMITS**

This command is not implemented - the limits for the display are used.

Sets the coordinate range for the plot when the various annotating DRAW commands are used. If the limits do not define an area with the same aspect ratio as the plot area, then part of the plot area will not be used. If this command is not given, or the values are omitted, then the default limits (the same as the LITES2 working area) are used. The plot limits may be changed at any time.

Format: PLOT LIMITS [xmin xmax ymin ymax]

* **PLOT LOAD**

Loads the specified GKS shareable image for hardcopy plotting. This command must be given before a plot can be started with PLOT OPEN.

GKS shareable images for different plot devices are supplied by Laser-Scan, and will normally reside in the directory LSL$PUBLIC_ROOT:[PLOTTING.EXE], which is included in the LSL$EXE search list, with names like GKSxyzSHR.EXE where xyz is some indication of the plotter device.

The specified filename may be either a logical name (which must translate to a device, directory, and filename, the file extension of .EXE being assumed), or an actual file name (in which case a default of LSL$EXE:---.EXE is applied).

For example, to plot on a Laserplot, one might give the command PLOT LOAD GKSLPSHR.

Format: PLOT LOAD filename

* **PLOT OFF**
Causes output from drawing commands to revert to its normal destination, rather than the plotter.

Format: PLOT OFF

* **PLOT ON**
Directs any output from drawing commands to the plotter.

Format: PLOT ON

* **PLOT OPEN**
Opens the plot device and begins a plot. Use PLOT ON and PLOT OFF commands to direct drawing output to the plotter. The plot is terminated by a PLOT CLOSE command.

Format: PLOT OPEN

* **PLOT ORIGIN**
Set the origin of the drawing area. The position of the drawing area specified using PLOT POSITION and PLOT SIZE commands is offset by the specified amount in mm.

Format: PLOT ORIGIN xoff yoff

* **PLOT PIXELS**
Sets the maximum number of image pixels in either x or y to be drawn into the plotting area when plotting images. Setting a low value will force subsampled views to be drawn, which might save time, or prevent pixels far too small for the plotter from being drawn. The default number depends on the particular plotter in use. The number is set to this default by the PLOT OPEN command, so any PLOT PIXELS commands should come after PLOT OPEN.

Format: PLOT PIXELS integer

* **PLOT POSITION**
Set the plotting position within the drawing area. Position 0 is bottom left, 1 is centre left, 2 is top left, 3 is bottom centre, and so on up to 8 which is top right. The default is position 0.

Format: PLOT POSITION n

* **PLOT RATIO**
Specifies the ratio between the scale of the plot and the true scale of the data, thus the command PLOT RATIO 1.0 will produce a plot at its true scale (provided that this has been set correctly by SCALE commands, or by default).

Format: PLOT RATIO real

* **PLOT SCALE**
Specifies the source scale of the plot to be produced. For this to work, the IFF data must have a source scale defined in its map descriptor. For example, if a map is a 1:1250, then the command PLOT SCALE 2500 will cause it to be plotted at half the size.

Format: PLOT SCALE real

* **PLOT SEPARATOR**
  Used to specify a PLOT ESCAPE function which is performed automatically when the priority changes and drawing sorted by priority is being used. It is used on plotters which by default draw transparently to ensure that each priority layer is drawn opaque. The default value (which is reset by PLOT LOAD) is 0, which means that no separator will be used. The value 1 is usually required for electrostatic plotters. See the PLOT ESCAPE command and the Plotters User Guide in the PLOTTING package documentation for details of what functions are available.

  Format: PLOT SEPARATOR ESCAPE escid

* **PLOT SIZE**
  Set the size of the plotting area in mm. The default is the full size of the available area on the plotter. The size is set to this default by the PLOT OPEN command, so any PLOT SIZE commands should come after PLOT OPEN.

  Format: PLOT SIZE xsize ysize

* **PLOT TYPE**
  Sets the workstation type for use in future PLOT OPEN commands. Appropriate numbers (if required) are given in the section of the FPP Plotters User Guide for the plot device concerned.

  Format: PLOT TYPE n

7.17.13 **WARP**

Control transformation (warping) of raster images.
A typical sequence of commands might be:
WARP TRANSFORM - specify type of transformation
WARP CLEAR - clear previously specified points if required
WARP POINT IMAGE/MAP - specify points (as many as required)
WARP FIT - calculate a fit
WARP DELETE/REPLACE - modify points if required, or add more
WARP FIT - fit again if required
WARP ON - turn on warping
WARP MAP/IMAGE - specify whether to warp the vectors or the image
Use drawing commands to view the warped picture.

Format: WARP subcommand

Valid in states INITIAL READY LINE CIRCLE TEXT SYMBOL

* **WARP CLEAR**
  Delete all warp points. Any warp transformation in use, or already fitted, remains active. Use WARP OFF to turn off an active transformation.
  Format: WARP CLEAR

* **WARP DELETE**
  Deletes a warp point. The point number must be between 1 and the number of points digitised. After this command, higher numbered points are re-numbered to fill the gap.

  Format: WARP DELETE n

* **WARP FIT**
  Calculate a fit to the current set of warp points. The type of fit is specified by the WARP TRANSFORM command. The goodness of the fit may be shown by the SHOW WARP command, and also appears in the various $WARP... system variables. Once fitted, the warp may be activated using the WARP ON command.

  Format: WARP FIT

* **WARP IMAGE**
  Warp the raster image to fit the vector data. This can be slower than WARP MAP. Because it causes the image to be resampled, a side-effect is that the image may be drawn at arbitrary zoom factors.

  Format: WARP IMAGE

* **WARP MAP**
  Warp the vector data to fit the raster image (default). This can be faster than WARP IMAGE. As when not warping at all, the raster image can only be zoomed to multiples of its original pixel size.

  Format: WARP MAP

* **WARP OFF**
  Turn off warping. The warp that was active may be turned back on using WARP ON.

  Format: WARP OFF

* **WARP ON**
  Turn on warping. If a new warp has been calculated using WARP FIT, then this is activated, otherwise the previously active warp is activated.

  Format: WARP ON

* **WARP POINT**
  Specify a data point for fitting a warp. If the coordinates are omitted, the current cursor position is used. Coordinates are specified in IFF units and identify a position either in the vector data (MAP) or the raster data (IMAGE). IMAGE and MAP points may be specified in any order - when a fit is performed, the first IMAGE point will map to the first MAP point etc.

  Format: WARP POINT IMAGE [x y]
  or WARP POINT MAP [x y]

* **WARP REPLACE**
  Replace a previously specified warp data point. The point number must be between 1 and the number of points digitised. If the coordinates are

omitted, the current cursor position is used. Coordinates are specified in IFF units and identify a position either in the vector data (MAP) or the raster data (IMAGE).

```
Format:  WARP REPLACE IMAGE  n [x y]
or       WARP REPLACE MAP    n [x y]
```

* **WARP TRANSFORM**
Specifies the type of warp transformation to be used. See the description of the SETUP TRANSFORM for details of the formulae used. The LINEAR transformation is just a shift if one point is supplied, or a shift plus a rotation with two points.

```
Format:  WARP TRANSFORM type

         where type = AFFINE (default)
                    = EXTENDED
                    = LINEAR
                    = ORTHOGONAL
                    = PROJECTIVE
```

8 Edgematching within LITES2

The EDGEMATCH command allows features on either side of a predefined line to be JOINed, TIEd or EXTENDed automatically on this line.
This section of the user manual explains in detail how this is achieved and how to use the command.

8.1 Defining A Base To EDGEMATCH To

Before features can be EDGEMATCHed a base must be defined. This base is a vector of a linear feature. To define it, a feature is found; the cursor is moved to the required vector (first vertex, or between the two vertices) with the VERTEX, NEXT, PREVIOUS, FRACTION (etc) commands; finally the command BASE EDGE is given. Any previously defined edgematching base will have been overwritten.

Details of the current edgematching base can be seen by giving the SHOW BASE command.

8.2 Controlling EDGEMATCHing

Default settings of things that control edgematching may need to be altered. These are:-

* TOLERANCE EDGE - for finding points within this distance of the line, and then within this distance of the projection of the aforementioned point on the line. The default is set to 1.5 sheet mms

* TOLERANCE PROPAGATE - any mismatch between edgematched features will be propagated this distance back along both features. If no propagation is required (ie the mismatch is to be taken out on the first vector of each feature) then this value should be set to 0.0. The default is 10 sheet mms.

* MATCH FSN - only features with the same FSN will be edgematched.
  OFF by default.

* MATCH FC - only features with the same FC will be edgematched.
  ON by default.

* MATCH MAP - only features within the same map will be edgematched.
  ON by default.

* MATCH LAYER - only features within the same layer will be edgematched.
  ON by default.

* MATCH PC - only features with the same process code will be

edgematched.
OFF by default.

* MATCH AC - only features which have exactly matching AC types 2, 3, 4 and 5 will be edgematched.
OFF by default.

Details of the current settings can be seen with the SHOW TOLERANCE and SHOW MATCH commands.

Note that MATCH settings are used by the TIE and JOIN command, and that the TOLERANCE PROPAGATE setting is used by the PROPAGATE command.

8.3 **Invoking EDGEMATCHing**

There are three commands to invoke edgematching - EDGEMATCH JOIN, EDGEMATCH TIE and EDGEMATCH EXTEND. These are based on the LITES2 JOIN, TIE and EXTEND commands.

LITES2 does not allow a feature to be JOINed (or TIEd) to itself. This means that EDGEMATCH JOIN will not produce closed features. To achieve this effect, it is necessary to use the EDGEMATCH TIE command first, to produce features whose ends are coincident, then EDGEMATCH JOIN will give the desired result.

8.4 **What EDGEMATCHing Does**

The program acts as follows (EDGEMATCH JOIN and EDGEMATCH TIE):-

1. It SEARCHes for the next end of a feature (graphical types 1 6 11 & 12) which lies within the edgematching tolerance of the edgematching base (and satisfies any current SELECTion criteria).
   It will not find the base itself.

2. It drops the cursor perpendicularly onto the base (or its extension) - position A

3. It FINDs the features (that satisfy the current SELECTion criteria) whose ends lie within the edgematching tolerance of the cursor. If there are several, only the four closest are considered; if there are none, then the program returns to step 1).

4. It looks at the (up to) four features that have been found and checks them against certain criteria (see below).
   The closest feature that satisfies all the criteria is accepted. If none of the four features satisfy all the criteria then the program returns to step 1).

5. A perpendicular is dropped from the end of this found feature onto the base (or its extension) - position B. The mean of position A and position B is computed - if it lies outside the base, it is moved onto the end of the base - and the two features are TIEd or JOINed, as appropriate, at this point. Any mismatch is propagated along each

feature the defined distance.

6. For TIE, one of the two features is allowed to be in a read-only map. In this case, the end of the editable feature is moved to the nearest position on the baseline to the end of the read only feature.

7. If the features are TIEd together, the second feature has its end marked, so that it will not be found in step 1).

8. The program returns to step 1).

When there are no more features found in the SEARCH, the edgematch is complete. If there were any features which were found in step 1, that were not ultimately matched with something, the user is given the chance of giving the REVIEW command. This runs a command file that:-

* takes the user to all the positions where a problem occurred

* lists the problems

* allows the user to make any manual edits in the area.

8.5 **Criteria For Accepting Matching Features.**

The criteria used to accept found features to match with the "searched for" feature are:-

* The found feature must not have a substantial part of it lying along the base line.

* The two features must have the second from end vertex, from the end of the feature in question, on opposite sides of the edgematch base.

* The found feature must not be the edgematch base

* If MATCH FC is set, then the two features must have the same feature code.

* If MATCH FSN is set, then the two features must have the same feature serial number.

* If MATCH MAP is set, then the two features must be in the same map.

* If MATCH LAYER is set, then the two features must be in the same layer.

* If MATCH PC is set, then the two features must have the same process code set.

* If MATCH AC is set, then the two features must have exactly the same AC types 2, 3, 4 and 5. (ie values and any strings must match)

* The found feature has not been used to match to anything else before.

8.6 EDGEMATCH EXTEND

EDGEMATCH EXTEND SEARCHes for the next end of a feature as with JOIN and TIE. The intersection of the last line segment of the feature and the baseline is calculated. The feature is then extended or truncated to the intersection point on the base line. The amount by which the feature may be extended is limited to three times the current edgematching tolerance.

REVIEW takes the user to those positions where no intersection with the baseline could be calculated. This may occur when:-

* the last line segment points away from the baseline or is parallel to it.

* the intersection lies outside the baseline.

* the length of the line which has been extended is greater than three times the edgematching tolerance.

9 Squaring within LITES2

The SQUARE command allows the geometry of features to be constrained so as to produce a "neat" appearance. LITES2 implements two different algorithms which are known as "angle squaring" and "OS squaring"
This section of the user manual explains in detail how squaring is achieved and how to use the commands.

9.1 Comparison Of Squaring Algorithms

The angle squaring command has only one tolerance (an angle), and is totally independent of the units or size of either the map (IFF data), or the sheet. It can however only sensibly cope with features having a single orientation of corners to be squared. As it has no distance tolerance, corners can be moved by significant amounts if the feature is off-square.

The OS squaring algorithm was a direct implementation of the D14 algorithm as used in the ICL mainframe DMB system at OS Southampton. It has been enhanced to include facilities from the algorithm used by OS in their later programs. These additional facilities are selected by the ENABLE FIXED command (on by default) and include:

* The ability to hold points with specific attributes fixed. These attributes are specified by the PRIVILEGE ATTRIBUTE command.

* When squaring to external base(s), parts of the feature not perpendicular or parallel to the base(s) are squared internally

* Redundant points are removed between nearly parallel adjacent lines.

It has several tolerances expressed as sheet mm, but has inbuilt assumptions about the map units (IFF data). It will only work sensibly if the IFF units are ground metres, and the map scale is in the range 1:1250 to 1:10000. It uses an iterative approach, and can cope with multiple squaring orientations. It also has a facility for based squaring of features to predetermined orientation lines.

9.2 Angle Squaring Algorithm

The angle squaring algorithm is invoked by the SQUARE ANGLE command. It requires that a linear feature has already been found at a corner vertex that is to become a right angle. This nominated corner defines the orientations for parallel and perpendicular for the rest of the feature. The actions of the algorithm are as follows:

1. The feature is scanned for sides which fall within angle tolerance (given by TOLERANCE DEGREES or TOLERANCE RADIANS of parallel and perpendicular as defined by the nominated corner. All sides are classified as Parallel, Perpendicular, or Other.

2. Refined orientations are defined for the orientations of parallel and perpendicular by taking the mean of the directions all such classified sides, weighted by length.

3. All sides previously classified as parallel and perpendicular are rotated about their midpoints to the new 'normal' orientations.

4. All sides are linearly extended/contracted as needed to intersect their neighbours.

5. The modified feature is highlighted and is deposited in place of the original feature when an END command is given.

9.3 **OS Squaring Algorithm**

The OS D14 squaring algorithm is invoked by the SQUARE WHOLE or SQUARE PART commands. Both these commands require that a linear feature has already been found. The position on the feature by which it was found is immaterial. The only difference between SQUARE WHOLE and SQUARE PART is that for SQUARE WHOLE the value of the tolerance parameter SQMT is ignored and assumed to be infinite.
The actions of the basic algorithm are as follows:

1. The feature is scanned for the longest side not yet processed. This defines the orientation base for this iteration. Note that for based squaring this step is skipped and the predetermined baseline defined by the BASE SQUARE command is used instead.

2. Each side is examined, and if it is an invisible step, or less than the minimum length tolerance (SQLT), then it is marked as processed and its orientation is fixed.

3. The feature is scanned and each unprocessed side is classified as parallel, perpendicular, or other, according to whether its endpoints would have to be moved more than tolerance SQMT if it were rotated about its midpoint to the orientation base.

4. When not using an external base, this base orientation is refined by taking the mean of the orientations of all parallel and perpendicular classified sides, weighted by length.

5. Each side is examined, and if it has not previously been processed, and fits the constraints of move tolerance (SQMT), then it is rotated to be parallel or perpendicular to the base. Any side so moved is marked as processed.

6. If there remain any sides which have not been processed after this pass, then loop back to 1 to get a new base orientation and start another pass. Note that this can only occur on SQUARE PART, as for SQUARE WHOLE all sides will have been processed on the single pass as SQMT is infinite.

7. Checks are made for closure of the feature using the tolerance SQCT, and if necessary, the start and endpoints are forced together.

8. All sides are linearly extended/contracted through their mid-point as needed to intersect their neighbours. If adjacent lines are parallel (within the parallel tolerance SQPL), then the original point between the lines is used. (This can alter the calculated orientation of the line).

   A warning is given if any vertex is moved by more than the warning tolerance (SQWT).

9. The modified feature is highlighted and is deposited in place of the original feature when an END command is given.

In the enhanced squaring algorithm (with ENABLE FIXED) the steps are modified as follows:

* If squaring to external bases, after all the bases have been used, the algorithm does loop through step 1

* In step 1, lines that have both end points fixed, are used as bases before the rest of the unsquared lines in a feature.

* If there are any fixed points in a feature, the orientation of the base is not refined in step 4

* Between steps 6, and 7, unfixed points that lie between sides that are within the parallel tolerance (SQPL) are removed from the feature.

* In step 7, if an unfixed point is forced onto a fixed one, the former inherits the attributes of the latter.

* In step 8, the sides are only extended through their mid-points if neither of their end points are fixed, otherwise they are extended through their fixed points.

NOTE

> This forcing of sides through fixed points may cause points to be finally moved by more than the movement tolerance SQMT, which is only used to select the sides to be squared.
>
> More significant movements of points can occur when the opening and closing sides of a feature are parallel. In this case the last point (and perhaps others) will be removed, and the first point may be moved a long way in the direction of the closing side, to close the feature.

9.4 **Controlling Squaring**

Default settings of things that control squaring may need to be altered. These are:-

```
TOLERANCE DEGREES real  - Angle squaring tolerance in degrees
TOLERANCE RADIANS real  - Angle squaring tolerance in radians
TOLERANCE SQDEF         - restore to default setting all OS SQxx parameters
TOLERANCE SBMT  real    - as SQMT but for based squaring
TOLERANCE SBLT  real    - as SQLT but for based squaring
TOLERANCE SQBT  real    - length of base must be longer than this distance (mm)
TOLERANCE SQCT  real    - distance (mm) to be used when comparing the end
                          points of a feature for closure
TOLERANCE SQMT  real    - distance (mm) to be used when deciding if a
                          side is to be included in this squaring pass.
                          It is the maximum lateral distance a point
                          might be expected to move
TOLERANCE SQLT  real    - minimum length of line to be moved (mm)
TOLERANCE SQPL  real    - angle (degrees) to be used in OS squaring
                          to test if two sides are parallel
TOLERANCE SQWT  real    - warning issued when point moved more than this (mm)
```

10 TEXT within LITES2

Text features in maps are different from other data, in that they generally have no physical equivalent on the ground.
This section of the user manual explains in detail how text is handled within LITES2.

10.1 Characteristics Common To All Text Features

Text features are handled in LITES2 by storing (at least) the following characteristics in the IFF file:

* the coordinates of the text's locating point

* the angle at which the text lies

* the text string (as ASCII characters)

* the feature code, which must be graphical type 10, in the current FRT

* the style of the text, which is an integer in the range 0 - 3

* the category of the text, which is an integer in the range 0-63

The style and category are characteristics that are specific to Ordnance Survey maps, and may be set to 0 (or even used for other purposes) with non-Ordnance Survey type data.

In addition, a flag is set to indicate that the feature is a text. This flag is set for the use of other programs which may not use the FRT mechanism.

10.2 Size Of Text

The area that a text occupies on the map depends on:

* the number and actual characters in the ASCII string

* the shape of the characters. These are defined in the TRI file that is being used. For details of TRI files, see the FRTLIB reference manual.

* the height of the text. The width of each character in a font is a fixed proportion of its height; all characters in a text string have the same height, which is called the height of the text.

The height of a text can either be stored in the IFF file or in the FRT file, and which method is chosen depends on the setting of the HEIGHT option. If this is enabled (using the ENABLE HEIGHT command), then the height is stored in the IFF file and as a consequence is variable, for any feature code. By default the HEIGHT option is disabled, and this means that the height of any text feature is defined by its feature code; the actual height is read from the FRT.

Heights are stored in the FRT as mm at map scale.

When heights are stored in the IFF file, there are two possibilities:

* if the POINT option is disabled, then the height is stored as mm at map scale.
  Note that the smallest discrete step in height in this case is 0.01mm.

* if the POINT option is enabled (the default case), then the height is stored as a point size. These are point sizes as defined by the Ordnance Survey, Southampton, and their relationship to mm on the map sheet are given in the following table:

Point Size	**Height in mm**
24	5.00
22	4.45
20	3.95
18	3.60
16	3.25
14	2.90
12	2.45
11	2.20
10	2.05
9	1.75
8	1.60
7	1.40
6	1.25
5	1.10
3	0.85

  Note that some point sizes are undefined.

10.3 **Justification Of Text**

By enabling the positioning option (ENABLE POSITION - enabled by default) a locating (or justification) position can be stored with each text. This is an integer in the range 0 - 8, indicating one of the nine possible locating points as shown in the following diagram:

```
2-------------5-------------8
|             |             |
1-------------4-------------7
|             |             |
0-------------3-------------6
```

This is the point of the text that will lie over its digitised point.

If the location option is disabled, LITES2 treats all texts as if locating position 0 has been chosen.

Each character in a font is usually terminated by some blank space, so that the characters in strings of text are separated. When the length of a text string is required (e.g. when drawing a box around it), this space must be subtracted from the last character.

The TOLERANCE JUSTIFY command sets the amount to be subtracted from a text string to represent this blank space. The argument is the proportion of the height of the text that is blank space. By default the value 0.333333 is used.

10.4 **Composite Text**

There is an additional (licensed) feature in LITES2 to allow the construction and manipulation of composite text features. Access to this feature is achieved by enabling the COMPOSITE option (disabled by default).

Composite texts are texts that consist of several text strings, all of which have their own characteristics. It is possible to either manipulate the text feature as a whole, or to edit each sub-text individually. When editing the whole feature, then LITES2 will be in MODIFY state; when editing the subtexts then LITES2 goes into MODIFY (part) state.

LITES2 commands that are specifically for manipulating composite texts are:

THIS, FIRST, NEXT, PREVIOUS, LAST,
WHOLE, COLLAPSE,
PARAGRAPH, SPLIT, BEND

Important Note

This feature uses the TS entry in the IFF file. While the new IMP suite of utility programs and the FPP plotting programs can deal with these entries, other older programs cannot.

In particular the DAMP utilities will not work on these files, and some data conversion programs such as I2MOSS etc. will not deal with composite texts.

11 **User Routines**

LITES2 provides the opportunity for users to write their own programs to implement any operations that they require, that are not provided by the LITES2 commands, and that cannot be provided by the use of the MACRO command facility.

This facility is achieved by the LITES2 USER command, which allows access to one such program, and the ROUTINE command which allows access to another 10.

The programs must be linked as shareable images (see below for details) and they are accessed by logical names as follows:

* The "USER" command uses the image pointed to by the logical name LSL$LITES2ROUTINES

* The "ROUTINE n" command uses the image pointed to by the logical name LSL$LITES2ROUTINES_n, where n is an integer either between 1 and 5 or between 101 and 105. The routines with numbers greater than 100 are reserved for user routines supplied by Laser-Scan.

NOTE

In the rest of this chapter reference is made to the USER command. In any particular case this can be substituted by the ROUTINE n command.

Before attempting to provide their own shared image, users should have an understanding of the concept of features, as used by LITES2; in particular the various entries that are associated with different graphical types of features. For further information see Laser-Scan's documentation on IFFLIB and FRTLIB.

11.1 **Calling The User Routines From LITES2**

When the USER command is given to LITES2 the subroutines that the user has written and included in the shareable image, are called **by LITES2**. LITES2 is designed to call up to 10 routines, but note that

* If a particular routine does not exist in the image, then it will not be called. This means that there is no need for the user to provide routines he does not require.

* In some cases there are alternative routines that can be supplied. For example there are several different routines that can be supplied to get the coordinates of the current feature, depending on the complexity of the data that is required - only X and Y; X, Y and Z; or X, Y and a full set of attributes for each point. In this case LITES2 will call the most complex routine that exists in the image, and will ignore any other routines in this group.

The user routine can control the order in which the subroutines are called by setting a return code in each routine to say what routine is to be called next.

For example, when the USER command is given to LITES2, the arguments (an integer and an optional string) are passed to the initialisation routine (USRINI) along with other pieces of information (see details of the subroutines below). On completion of the routine, LITES2 examines the return code that has been set within the routine, and

* If the return code = 0 will abort the USER command

* If the return code = 1 will call routines to obtain the Coordinates and the ACs of the current feature

* If the return code = 2 will call routines to obtain the Coordinates without the ACs of the current feature

* If the return code = 3 will call routines to obtain the ACs of the current feature without the coordinates

* If the return code = 4 will ignore the coordinates and ACs of the current feature and call the routine that does any processing required before constructing a new feature

* If the return code = 5 will call the completion routine, without any of the intervening routines

* If the return code > 100 will call the routine that obtains, and optionally writes back, the map header and the map descriptor for the a particular map (the map is specified as the difference between the return code and 100, for example is 101 is returned, the data from map 1 is used; 105 allows the data from map 5 to be examined and edited).

The flow of control can be determined on the completion of all the routines in a similar manner. The following diagram shows the various possibilities

Possible order in which routines are called.

Note that the USER command may be aborted after all subroutine calls if the return code 0 is returned.

The loop that runs from "Finish" back to "Do Processing" may be broken and the command aborted by pressing CTRL C, as can the loops that run from "Finish" through "Ancillary" back to "Finish" and the loop that allows repeated calls of the map header editing routine.

11.2 **Accessing Particular Shared Images.**

The shared images are only mapped when the first USER or ROUTINE n command is given. If the appropriate logical name is not set up then a warning message will be given at this stage.

11.3 **The Subroutines - General.**

The subroutines are listed, by functional group, in the order that they are called by the USER routine, and appear in the diagram above.

11.3.1 **Initialise**

This is the first subroutine to be called, whenever the USER command is given. There is only one routine that can be supplied in this group - USRINI

It is used to pass initial data to the shared image. This initial data consists of:-

* the integer argument that was given with the USER command.
  This integer is often used to define which of several different operations is to be carried out by the shared image.

* the optional string argument that may be given with the USER command. Any other information that is required by the user routines can be passed in this argument and subsequently decoded.

* the current cursor position.
  This is passed in the units that the data is stored in, in the IFF file(s). If there are multiple maps, any offset relative to the south western map will have been added.

* the current state

* a flag to say if there is a found object or not.

* some data about the found object.
  This is information that can be passed in a fixed format. Data about the feature that is of variable length (lists of ACs and coordinates) is dealt with by the routines get ACs and get coordinates (see below)

Having received this information, the subroutine must set the return code, to tell LITES2 which subroutine to call next. The possibilities are:-

"0" means that no more subroutines will be called.

"1" means call routines to get the ACs for the feature, and the coordinates.

"2" means call routines to get coordinates of the feature, but not the ACs.

"3" means call routines to get the ACs for the feature without the coordinates.

"4" means that the next subroutine to be called is to do the processing.

"5" means that the next subroutine to be called is the one that completes the command.

"> 100" means call the routine that allows the map header and map descriptor information to be edited.

Any other value will cause a fatal error (see routine USRERR below for details on error handling).
Entering 1,2 or 3 when there is no found feature will also cause a fatal error.
Attempting to get ACs from a feature that has none will cause the routine not to be called, and the next routine called will depend on the value of the return code supplied.
If the routine USRINI is not supplied in the shareable image, or if the return code is greater than 100, and the routine to get map header information is not supplied, USRDO will be the next routine that is called.

11.3.2 **Getting and writing file header information**

This routine is used to read and write information that is particular to the specified map. There is only one routine that can be supplied in this group - USRGMH

This subroutine is used to read any map header (MH) and map descriptor (MD) entries associated with the specified map. The map is specified by giving a return code > 100 - the map is the difference between the return code and 100.
It supplies the map header and the map descriptor (as arrays of 16 bit integers - INTEGER*2 in Fortran) and allows them to be altered and optionally written back to their files.

The possible return codes are:-

"0" means that no more subroutines will be called.

"1" means call routines to get the ACs for the feature, and the coordinates.

"2" means call routines to get coordinates of the feature, but not the ACs.

"3" means call routines to get the ACs for the feature without the coordinates.

"4" means that the next subroutine to be called is to do the processing.

"5" means that the next subroutine to be called is the one that completes the command.

"> 100" means call this routine again, with the data from the specified map (see above).

11.3.3 **Getting ACs**

This subroutine is used to read any AC entries associated with the found feature. There is only one routine that can be supplied in this group - USRGAC

This subroutine is used to read any AC entries associated with the found feature.
It reads one AC at a time and returns its type, value and any text associated with it.

The possible return codes are:-

"0" means that no more routines will be called.

"1" means call USRGAC again, if there are any more ACs, or else call the routine to get coordinates or to do the processing (depending on return code supplied by the Initialising routine)

"2" means don't call USRGAC again; call the routine to get coordinates or the routine to do the processing (depending on return code supplied by the Initialising routine)

"4" means don't call USRGAC again; call the routine to do the processing.

NOTES

1. TC entries are treated as ACs with a type of -1.
   They have no value associated with them, and a longer maximum text length.

2. CH entries are treated as ACs with a type of -2.
   They have no value associated with them, and a longer maximum text length.

11.3.4 **Getting Coordinates**

This group of subroutines are used to get the coordinates associated with the found feature. As the number of coordinates varies between feature and feature these subroutines pass the coordinates a block at a time. This means that arbitrarily long features can be processed by the USER command - although the user's shared image must of course have a method of dealing with them.

LITES2 decides how many points to pass to the routine; it will generally be 200, or the number of points remaining in the feature if that is less, but for composite texts the points will be passed one at a time.

The coordinates associated with each point in a feature can be simple or complex, according to the way in which LITES2 is being used. To avoid the user having to deal with unnecessary complexities, when he does not require the complex information, there are at several routines that may be called at this point. They are :

1. USRGCB - this routine passes, in addition to X and Y coordinates, an array of the attributes that are associated with each point, along with the arrays of header information for the attributes, that allow them to be used. This header information consists of a vector array of the codes used to identify each column of data and similar arrays that hold the names that are represented by the codes, and their data types.

   See the FRTLIB users guide for information about these codes and data types

   NOTES

   1. **Blocks of attribute information passed in repeated calls of this routine may not have their columns in the same order.**

   2. Attribute values may be null (i.e. unset). This may be tested for by comparing the ABSENT parameter (given in the template routine) with the INTEGER version of the required value.

2. USRGZS - this routine passes the X, Y and Z coordinates of each point of the feature.

3. USRGST - this routine is the simplest of the routines and passes a simple XY array.

4. USRGPT - this routine is similar to USRGST, but in addition it passes information about the text in text features. This can be used for simple (i.e. non-composite texts) but see below. It is included for upwards compatability with user-routines written for versions of LITES2 prior to version 2.8.

If two (or more) of these routines are included in the shared image, then the one higher up the list will be used.

In addition to passing the coordinates, the subroutines also pass a byte array of flags associated with each point. At present, only the bottom bit is significant, and it indicates whether the vector up to this point is to be visible (1) or invisible (0). In general, the flag for the first point has no significance.
The possible return codes are:-

"0" means that no more routines will be called.

"1" means call the routine again, if there are any more points, or else call the routine that will do the processing.

"4" means don't call the routine again - call the routine that will do the processing.

If the feature is a text feature, after the relevant routine has been called to get the coordinates, the routine USRGTX is called if it is present in the image. This returns data about the (sub)text associated with the coordinate information that has just been received. It does not require a return code to be supplied. The flow of control depends on the return code supplied in the coordinate passing routine.

11.3.5 **Processing**

There is only one routine that can be supplied in this group - USRDO

By the time this routine is called, all the information has been passed from LITES2 to the user routines. This routine is used to process this data. Its return code controls the transfer of data back to LITES2.

The possible return codes are:-

"0" means that no more user routines will be called.

"4" means do not pass a feature back to LITES2; call the completion routine straight away.

"5" means call routines to construct a new feature, keeping any old one.

"6" means call routines to construct a new feature, deleting any currently found feature.

NOTE

It is not possible to construct a new feature while LITES2 is in certain states. An attempt to do so, by setting the return code of this routine to 5 or 6, while in these states, will cause a fatal error. The states involved are:-

* Construct State
* Modify State
* AC State
* Edit State
* Initial State

11.3.6 **Start a New Feature**

There is only one routine that can be supplied in this group - USRSTO

If a new feature is to be constructed (code 5 or 6 returned by the processing routine), then the routine to start a new feature is called. This allows the shared image to define the type of feature that is to be constructed. Entering -1 as the value of the FSN, FC, MAP, LAYER or THICK allows default values of these to be used in the construction of the new feature (set TEXTF true if default text feature code is to be used), otherwise values can be explicitly set in these variables.

See IFFLIB documentation for the meaning of the 4 elements of the FC (or feature status entry)

The number of coordinated points and the number of ACs in the feature must be given at this stage.

The possible return codes are:-

"0" means that no more user routines are called.

"1" means that more data about the feature is to be passed; call either the routines to output an AC (if the number of ACs > 0), or else call the routines to output coordinates.

"4" means call the completion routine straight away (no new feature will be constructed).

NOTES

1. Any attempt to construct a feature in a non-existent map or layer, a read only map, with an non-existent feature code, with an invalid feature code in the circumstances, or with an inappropriate number of points will result in a fatal error.

2. The FC entries are truncated to INTEGER*2 values before they are interpreted by LITES2.

3. The top two bits of the truncated value of FC(3) will be set by LITES2, depending on the graphical type of the feature.

4. Invalid values of THICK (when the HEIGHT and POINT options have been enabled) will cause a non-fatal error. The value of THICK is ignored for non-text features, or when the HEIGHT option is disabled.

5. ROTAT is only meaningful for oriented symbols (and text, if USRPPT is being used to output a text). It is entered as a value in radians.

6. Orientation of scaled symbols is achieved by entering two points.

7. There are OPERATION USER_FEATURE and OPERATION USER_POINT commands that will allow any ACs and point that are constructed by the user routine to be automatically updated.

11.3.7 **Outputting an AC**

This subroutine is used to output any AC entries to be associated with the ne construction. There is only one routine that can be supplied in this group - USRPAC

This routine is called after the routine that started a new feature, if the return code was set to 1 and there were some ACs to add to the feature. As long as its return code is set to 1, it is called NACS (as returned by the start feature routine) times.

The possible return codes are:-

"0" means that no more user routines are called

"1" means call this again if there are more ACs, otherwise call a routine to output some coordinates

"2" means don't call this routine again, call a routine to output some coordinates

"4" means call the completion routine straight away.

NOTES

1. Trying to construct an AC, TC or CH entry with a text that is too long, will result in a non-fatal error and the text being truncated.

2. A return code of 2 calls the relevant coordinate output routine straight away.
Any AC data passed in a call of this routine with a return code of 2 is ignored by LITES2.

11.3.8 **Outputting coordinates**

This group of subroutines are used to output the coordinates of the new feature.

The coordinates are passed to LITES2 in blocks, in repeated calls of one of the output routines, in a similar manner to the way in which coordinates are passed out of LITES2 by the routines that get coordinates. The size of each block of coordinates is determined by the shared image - **but it must not be larger than the value that is passed to the user image in the argument SIZE.**

When constructing non-text features, for efficiency, the largest possible blocks of data should be supplied to this routine. For texts however, only one point should be supplied at a time, so that a call of USRPTX can be associated with each coordinate

In a manner similar to the routines that get coordinates, depending on the amount of data required, there are several routines that may be called at this point. They are :

1. USRPCB - this routine passes, in addition to X and Y coordinates, an array of the attributes that are associated with each point, along with an array of header information for the attributes.

2. USRPZS - this routine passes the X, Y and Z coordinates of each point of the feature.

3. USRPST - this routine is the simplest of the routines and passes a simple XY array.

4. USRPPT - this routine is similar to USRPST, but in addition it passes information about the text in text features. This can be used for simple (i.e. non-composite texts) but see below. It is included for upwards compatability with user-routines written for versions of LITES2 prior to version 2.8.

If two (or more) of these routines are included in the shared image, then the one higher up the list will be used.

In addition to passing the coordinates, the subroutines also pass a byte array of flags associated with each point. At present, only the bottom bit is significant, and it indicates whether the vector up to this point is to be visible (1) or invisible (0). In general, the flag for the first point has no significance.

The possible return codes are:-

"0" means abort the construction of the feature, and don't call any more routines

"1" means write these coordinates, or if LITES2 has already completed the feature, call the completion routine

"4" means abandon the construction of the feature, if it is not yet completed; call the completion routine.

NOTES

1. When one of these routines has been called, LITES2 will ultimately revert to READY state, either on completion of the feature, or when the construction is abandoned.

2. Any error in constructing the feature will result in a fatal error, and the construction will be abandoned.

3. Trying to construct a text with a zero length string will result in a fatal error

4. Trying to construct a feature with coordinates that fall outside the limits of the map(s) will also cause a fatal error.

5. Supplying the return code 4, before all the points have been passed (with calls with return code 1) will result in the construction being abandoned.

If the feature is a text feature, after the relevant routine has been called to get the coordinates, the routine USRPTX is called if it is present in the image. This provides data about the (sub)text associated with the coordinate information that has just been sent. It does not require a return code to be supplied. The flow of control depends on the return code supplied in the coordinate passing routine.

11.3.9 **Completing the USER command**

There are four routines that can be supplied in this group - USRRET and the ancillary routines, USRDEF, USRANO and USRDRW.

USRRET is the last user routine to be called by the USER command. It allows the condition flag (used for determining the flow of control in macros) to be set and also allows a LITES2 command string to be entered. This command string will be obeyed immediately after the current USER command is completed.

By supplying a suitable return code, LITES2 can be instructed to call an ancillary completion routine : USRDEF, USRANO or USRDRW. After this ancillary routine has completed, USRRET is called again.

The possible return codes to USRRET are:-

"0" do not set the condition flag or obey the command string

"1" set the condition flag and obey the command string

"2" call the processing routine again

"3" call the ancillary completion routine USRDEF

"4" call the ancillary completion routine USRANO

"5" call the ancillary completion routine USRDRW

NOTE

This flow of control, that may cause a (infinite) loop to be set up, if this routine always sets the return code to be greater than 1, can be broken and the user routine aborted by entering CTRL C at the keyboard, or through the abort auxiliary input.

USRRET must exist if an ancillary routine is to be called.

There is no return code supplied to these ancillary routines; after they have completed successfully USRRET is called again. If an error occurs while carrying out the action required by the routine (for example if a variable to be set does not exist, or a drawing command is given in an inappropriate state) the user routine aborts.

1. USRDEF is used to set the value of LITES2 variables.

   It takes arguments of all possible data types, but only the one that relates to the type of the LITES2 variable that has been specified is used.

2. USRANO is used to set the characteristics for drawing annotation from the user routine with the USRDRW routine. Annotation characteristics thus set are lost when the user routine completes.

   It takes as arguments a string, and the length of the string. The routine should return the required secondary command (and any arguments) for an ANNOTATION command. The string length, on input is the maximum length of the string to be returned. If 0 is returned as the string length then no ANNOTATION action is performed.

3. USRDRW is used to draw something on the screen from the user routine.

   It takes as arguments a string, and the length of the string. The routine should return the required secondary command (and any arguments) for a DRAW command. The string length, on input is the maximum length of the string to be returned. If 0 is returned as the string length then no DRAWing is performed.

11.3.10 **Error handling**

There is only one routine that can be supplied in this group - USRERR.

If the FATAL argument is TRUE, then the USER command will be aborted, as soon as control returns from this routine - abandoning any feature that is being constructed.
If the FATAL argument is FALSE, then the USER command will continue after outputting a warning message.

See the details of the routine for the meaning of the error numbers.

11.4 **The Subroutines - Specifications.**

There are template files supplied in LSL$PUBLIC_ROOT:[LITES2.ROUTINES.TEMPLATE] for all the following routines

11.4.1 **Initialise**

11.4.1.1 **USRINI**

```
        SUBROUTINE USRINI(ACTION,STRL,STR,CURSOR,CNDFLG,STATE,
     &                    GOTFO,NCOORD,NACS,FSN,FC,MAP,LAYER,GT,
     &                    ROTAT,THICK,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
C Arguments
C
        INTEGER*4       ACTION          ! action to carry out
        INTEGER*4       STRL            ! number of characters in STR
        CHARACTER*(*)   STR             ! string passed to USER command
        REAL            CURSOR(2)       ! coordinates of cursor
        LOGICAL         CNDFLG          ! condition flag.
        CHARACTER*(*)   STATE           ! current state
        LOGICAL         GOTFO           ! TRUE if there is a found
                                        ! object, FALSE otherwise, when
                                        ! the next 4 arguments are
                                        ! undefined
        INTEGER*4       NCOORD          ! number of coords
        INTEGER*4       NACS            ! number of ACS
        INTEGER*4       FSN             ! number of feature
        INTEGER*4       FC(4)           ! feature status
        INTEGER*4       MAP             ! map
        INTEGER*4       LAYER           ! layer
        INTEGER*4       GT              ! graphical type
        REAL            ROTAT           ! rotation if text or oriented
                                        ! symbol (in radians)
        INTEGER*4       THICK           ! size of text
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call
                                        !     processing routine
                                        ! = 1 for get coords and ACs
                                        ! = 2 for get coords without ACs
                                        ! = 3 for get ACs without coords
                                        ! = 4 for call processing without
                                        !     coords or ACs
                                        ! = 5 for call completion
                                        !     routine
                                        ! > 100 - call the routine to get
                                        !  map header information (for map
                                        !  RETCOD - 100)
```

```
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
```

11.4.2 **Getting file header information**

11.4.2.1 **USRGMH**

```
        SUBROUTINE USRGMH(MH_LEN,MH,WRITE_MH,MD_LEN,MD,WRITE_MD,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
C Arguments
C
        INTEGER*4       MH_LEN          ! input  - length of original MH
                                        ! output - length of updated MH
        INTEGER*2       MH(MH_LEN)      ! map header - NOTE INTEGER*2
        LOGICAL*4       WRITE_MH        ! input  - TRUE if MH is writable
                                        ! output - TRUE if MH is to be written
        INTEGER*4       MD_LEN          ! input  - length of original MD
                                        ! output - length of updated MD
        INTEGER*2       MD(MD_LEN)      ! map descriptor - NOTE INTEGER*2
        LOGICAL*4       WRITE_MD        ! input  - TRUE if MD is writable
                                        ! output - TRUE if MD is to be written
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call processing
                                        !     routine
                                        ! = 1 for get coords and ACs
                                        ! = 2 for get coords without ACs
                                        ! = 3 for get ACs without coords
                                        ! = 4 for call processing routine
                                        !     without
                                        !     coords or ACs
                                        ! = 5 for call completion routine
                                        !     without coords or ACs
                                        ! > 100 - call this routine again
C
C       All these arguments may be considered writable but note that
C       the new lengths of the arrays MUST not be longer than the
C       original arrays
C
C       Trying to write a map header or a map descriptor to a file that
C       has been opened for READing only will cause an error.
C
C       Writing a new map header or map descriptor will not affect the
C       idea that LITES2 has of the origin and scale, and the system,
C       variables $MHARR, $MHLEN, $MDARR and $MDLEN although the output
C       file will contain the changes.
C
C       Changes to the projection parts of the map descriptor (ie apart
C       from the origin and scale) require an intimate knowledge of
C       Laser-Scan's projection software. It is recommended that such
C       changes are made using the program ITRANS.
C
```

11.4.3 **Getting ACs**

11.4.3.1 **USRGAC**

```
        SUBROUTINE USRGAC(ACTYPE,ACIVAL,ACTXTL,ACTXT,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
C Arguments
C
        INTEGER*4       ACTYPE          ! type of AC
        INTEGER*4       ACIVAL          ! AC value
                                        ! note: to read a real AC value,
                                        ! a copy of this will have to
                                        ! be equivalenced to a real
        INTEGER*4       ACTXTL          ! number of characters in ACTXT
        CHARACTER*(*)   ACTXT           ! text (maximum of 80 chars)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call USRDO
                                        ! = 1 for get more ACs if there
                                        !     are any, or start getting
                                        !     coords if reqd, or call
                                        !     USRDO if coords not reqd
                                        ! = 2 stop getting ACs, start
                                        !     getting coordinates
                                        ! = 4 for call USRDO right away
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
```

11.4.4 **Getting coordinates**

11.4.4.1 **USRGCB**

```
        SUBROUTINE USRGCB(SIZE,USERXY,USRFLG,
     &                   MAX_ATTR,USERNATT,USERATTC,
     &                   USERDATATYPES,USERNAMELENS,USERNAMES,USERIATTV,
     &                   USERRATTV,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! number of coords passed
                                        ! with this call
        REAL            USERXY(2,SIZE)  ! coordinates
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER         MAX_ATTR        ! maximum number of attributes
        INTEGER         USERNATT        ! number of attributes present
        INTEGER         USERATTC(MAX_ATTR)      ! attribute codes
        INTEGER         USERDATATYPES(MAX_ATTR) ! datatypes of attributes
        INTEGER         USERNAMELENS(MAX_ATTR)  ! name lengths
        CHARACTER*(*)   USERNAMES(MAX_ATTR)     ! names of attributes
C
C the following two arrays are equivalenced in the calling routine
        INTEGER         USERIATTV(MAX_ATTR,*)   ! integer values
        REAL            USERRATTV(MAX_ATTR,*)   ! real values
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call
                                        !     processing routine
                                        ! = 1 for get more coords
                                        !     or call processing routine,
                                        !     if no more
                                        ! = 4 for abort, but call
                                        !     processing routine
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
C
C       the following parameter is for testing if an attribute value
C       is present for a particular point
C
C       NOTE: this must be tested against an integer, which has
C       been equivalenced onto the real value to be tested
C
        INTEGER*4       IABSENT
        PARAMETER       (IABSENT = '80000000'X)
```

11.4.4.2 **USRGZS**

```
        SUBROUTINE USRGZS(SIZE,USERXYZ,USRFLG,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! number of coords passed
                                        ! with this call
        REAL            USERXYZ(3,SIZE) ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call
                                        !     processing routine
                                        ! = 1 for get more coords
                                        !     or call processing routine,
                                        !     if no more
                                        ! = 4 for abort, but call
                                        !     processing routine
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
C
C       the following parameter is for testing if a Z coordinate value
C       is present for a particular point
C
C       NOTE: this must be tested against an integer, which has
C       been equivalenced onto the real value to be tested
C
        INTEGER*4       IABSENT
        PARAMETER       (IABSENT = '80000000'X)
```

11.4.4.3 **USRGST**

```
        SUBROUTINE USRGST(SIZE,USERXY,USRFLG,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
                                          ! with this call
        REAL            USERXY(2,SIZE)    ! coords
        LOGICAL*1       USRFLG(SIZE)      ! flags (visibility only)
        INTEGER*4       RETCOD            ! return code
                                          ! = 0 abort, don't call
                                          !     processing routine
                                          ! = 1 for get more coords
                                          !     or call processing routine,
                                          !     if no more
                                          ! = 4 for abort, but call
                                          !     processing routine
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
```

11.4.4.4 **USRGPT**

```
        SUBROUTINE USRGPT(SIZE,USERXY,USRFLG,TEXTL,TEXT,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! number of coords passed
                                        ! with this call
        REAL            USERXY(2,SIZE)  ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       TEXTL           ! number of characters in TEXT
        CHARACTER*(*)   TEXT            ! text string, if text feature
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, don't call
                                        !     processing routine
                                        ! = 1 for get more coords
                                        !     or call processing routine,
                                        !     if no more
                                        ! = 4 for abort, but call
                                        !     processing routine
C
C       All these arguments, apart from RETCOD, should be considered
C       as read only
C
```

11.4.4.5 **USRGTX**

```
        SUBROUTINE USRGTX(TEXT,TEXTL,TS,HEIGHT,ROTAT,AUX)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       TEXTL           ! number of characters in TEXT
        CHARACTER*(*)   TEXT            ! text string, if text feature
        INTEGER*4       TS(4)           ! text component status
        REAL            ROTAT           ! rotation of text component
        INTEGER*4       HEIGHT          ! height of text (in points or
                                        ! 0.01mm)
        REAL*4          AUX(8)          ! data about text (in IFF units)
                                        ! AUX(1) = angle (in radians)
                                        ! AUX(2) = cosine of angle
                                        ! AUX(3) = sine of angle
                                        ! AUX(4) = height (in IFF units)
                                        ! AUX(5) = minimum X value
                                        !         (rel to locating point)
                                        ! AUX(6) = maximum X value
                                        ! AUX(7) = minimum Y value
                                        ! AUX(8) = maximum Y value
C
```

11.4.5 **Processing**

11.4.5.1 **USRDO**

```
        SUBROUTINE USRDO(RETCOD)
C
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call completion
                                        !     routine
                                        ! = 4 for abort, call completion
                                        !     routine
                                        ! = 5 to create a new feature
                                        !     (and keep old one if there
                                        !      was one)
                                        ! = 6 to create a new feature
                                        !     and delete old one
C
```

11.4.6 **Starting a new feature**

11.4.6.1 **USRSTO**

```
        SUBROUTINE USRSTO(FSN,FC,MAP,LAYER,TXTF,NOPTS,NAC,
     &                   ROTAT,THICK,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       FSN             ! feature serial number to use
                                        ! (set to -1 for unknown)
        INTEGER*4       FC(4)           ! feature status to use
                                        ! (set FC(I) to -1 for unknown)
        INTEGER*4       MAP             ! map to put feature in
                                        ! (set to -1 for unknown)
        INTEGER*4       LAYER           ! layer to use
                                        ! (set to -1 for unknown)
        LOGICAL         TXTF            ! .TRUE. if FC =-1 and want
                                        ! to create a text feature
        INTEGER*4       NOPTS           ! number of points in feature
        INTEGER*4       NAC             ! number of ACs in feature
        REAL            ROTAT           ! rotation if text or oriented
                                        ! symbol (in radians)
        INTEGER*4       THICK           ! size of text
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call completion
                                        !     routine
                                        ! = 1 for ask for data
                                        ! = 4 abort, call completion
                                        !     routine
C
C all the arguments in this subroutine are writable
C
```

11.4.7 **Outputting an AC**

11.4.7.1 **USRPAC**

```
        SUBROUTINE USRPAC(ACTYPE,ACIVAL,ACTXTL,ACTXT,RETCOD)
C
C       Copyright Laser-Scan Laboratories Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C       Dummy routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       ACTYPE          ! type of AC
        INTEGER*4       ACIVAL          ! AC value
                                        ! note: to read a real AC value,
                                        ! a copy of this will have to
                                        ! be equivalenced to a real
        INTEGER*4       ACTXTL          ! number of characters in ACTXT
        CHARACTER*(*)   ACTXT           ! text (maximum of 255 chars)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, no call completion
                                        !     routine
                                        ! = 1 for write another AC if
                                        !     there are any, or else
                                        !     start writing coords
                                        ! = 2 for start writing coords
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable
C
```

11.4.8 **Outputting coordinates**

11.4.8.1 **USRPCB**

```
        SUBROUTINE USRPCB(SIZE,USERXY,USERFLG,
     &                MAX_ATTR,USERNATT,USERATTC,
     &                USERIATTV,USERRATTV,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! input:  maximum number of
                                        !         coords to pass back
                                        ! output: actual number of
                                        !         coords passed back
                                        ! with this call
        REAL            USERXY(2,SIZE)  ! coords
        LOGICAL*1       USERFLG(SIZE)   ! flags (visibility only)
        INTEGER         MAX_ATTR        ! maximum number of attributes
        INTEGER         USERNATT        ! number of attributes present
        INTEGER         USERATTC(MAX_ATTR)      ! attribute codes
C
C the following two arrays are equivalenced in the calling routine
        INTEGER         USERIATTV(MAX_ATTR,*)   ! integer values
        REAL            USERRATTV(MAX_ATTR,*)   ! real values
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call completion
                                        !     routine
                                        ! = 1 for write more coords, if
                                        !     there are any, or else
                                        !     call USRRET
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable.
C
C don't send more than maximum number of attributes -- most important
C ============================================================
C
```

11.4.8.2 **USRPZS**

```
        SUBROUTINE USRPZS(SIZE,USERXYZ,USRFLG,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! input:  maximum number of
                                        !         coords to pass back
                                        ! output: actual number of
                                        !         coords passed back
                                        ! with this call
        REAL            USERXYZ(3,SIZE) ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call
                                        !     completion routine
                                        ! = 1 for write more coords, if
                                        !     there are any, or else
                                        !     call USRRET
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable.
```

11.4.8.3 **USRPST**

```
        SUBROUTINE USRPST(SIZE,USERXY,USRFLG,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! input:  maximum number of
                                        !         coords to pass back
                                        ! output: actual number of
                                        !         coords passed back
                                        ! with this call
        REAL            USERXY(2,SIZE)  ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call
                                        !     completion routine
                                        ! = 1 for write more coords, if
                                        !     there are any, or else
                                        !     call USRRET
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable.
```

11.4.8.4 **USRPZS**

```
        SUBROUTINE USRPZS(SIZE,USERXYZ,USRFLG,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C       Dummy user routine
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! input:  maximum number of
                                        !         coords to pass back
                                        ! output: actual number of
                                        !         coords passed back
                                        ! with this call
        REAL            USERXYZ(3,SIZE) ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call
                                        !     completion routine
                                        ! = 1 for write more coords, if
                                        !     there are any, or else
                                        !     call USRRET
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable.
```

11.4.8.5 **USRPPT**

```
        SUBROUTINE USRPPT(SIZE,USERXY,USRFLG,TEXTL,TEXT,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C
        IMPLICIT NONE
C
C Arguments
C
        INTEGER*4       SIZE            ! input:  maximum number of
                                        !         coords to pass back
                                        ! output: actual number of
                                        !         coords passed back
                                        ! with this call
        REAL            USERXY(2,SIZE)  ! coords
        LOGICAL*1       USRFLG(SIZE)    ! flags (visibility only)
        INTEGER*4       TEXTL           !  input: max size of TEXT
                                        ! output: actual size of TEXT
        CHARACTER*(*)   TEXT            ! text string, if text feature
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 abort, dont call completion
                                        !     routine
                                        ! = 1 for write more coords, if
                                        !     there are any, or else
                                        !     call completion routine
                                        ! = 4 for abort, call completion
                                        !     routine
C
C       All these arguments are writable.
C
```

11.4.8.6 **USRPTX**

```
        SUBROUTINE USRPTX(TEXT,TEXTL,TS,THICK,ROT)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
C Arguments
C
        CHARACTER*(*)   TEXT            ! text string, if text feature
        INTEGER*4       TEXTL           !  input: max size of TEXT
                                        ! output: actual size of TEXT
        INTEGER*4       TS(4)           ! feature status for texts
        INTEGER*4       THICK           ! height of text
        REAL            ROT             ! angle of text
C
C       All these arguments are writable.
C
```

11.4.9 **Completing the USER command**

11.4.9.1 **USRRET**

```
        SUBROUTINE USRRET(CNDFLG,RTSTRL,RTSTR,RETCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines.
C
        IMPLICIT NONE
C
C Arguments
C
        LOGICAL         CNDFLG          ! LITES2 conditional flag
        INTEGER*4       RTSTRL          !  input: maximum size of RTSTR
                                        ! output: size of RTSTR
        CHARACTER*(*)   RTSTR           ! LITES2 command line, to be
                                        ! executed before any other
                                        ! command
        INTEGER*4       RETCOD          ! return code
                                        ! = 0 for abort
                                        ! = 1 for CNDFLG to be set
                                        !     and command to be executed
                                        ! = 2 for call processing routine
                                        !     again
                                        ! = 3 for call USRDEF before calling
                                        !     this routine again
                                        ! = 4 for call USRANO before calling
                                        !     this routine again
                                        ! = 5 for call USRDRW before calling
                                        !     this routine again
C
C all these arguments are writable
C
```

11.4.9.2 **USRDEF**

```
        SUBROUTINE USRDEF(VARNAM_LEN,VARNAM,INDEX,INTVAL,REALVAL,
     &                       DBLVAL,CHARVAL_LEN,CHARVAL)
C
C       Copyright Laser-Scan Laboratories Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
C Arguments
        INTEGER          VARNAM_LEN      ! input: maximum size of VARNAM
                                         ! output: size of VARNAM
        CHARACTER*(*)    VARNAM          ! variable name to set
        INTEGER          INDEX           ! element if VARNAM is array
        INTEGER          INTVAL          ! integer value to set
        REAL             REALVAL         ! real value to set
        REAL*8           DBLVAL          ! double value to set
        INTEGER          CHARVAL_LEN     ! input: maximum size of CHARVAL
                                         ! output: size of CHARVAL
        CHARACTER*(*)    CHARVAL         ! character value to set
C
C all these arguments are writable.
C
```

11.4.9.3 **USRANO**

```
        SUBROUTINE USRANO(RTSTRL,RTSTR)
C
C       Copyright Laser-Scan Laboratories Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
        INTEGER*4       RTSTRL          ! input: maximum size of RTSTR
                                        ! output: size of RTSTR
                                        ! if 0 is returned no ANNOTATION
                                        ! command is executed.
        CHARACTER*(*)   RTSTR           ! secondary command (with
                                        ! arguments) for the ANNOTATION
                                        ! command
C
C all these arguments are writable.
C
```

11.4.9.4 **USRDRW**

```
        SUBROUTINE USRDRW(RTSTRL,RTSTR)
C
C       Copyright Laser-Scan Laboratories Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
        INTEGER*4       RTSTRL          ! input: maximum size of RTSTR
                                        ! output: size of RTSTR
                                        ! if 0 is returned no DRAW command
        CHARACTER*(*)   RTSTR           ! secondary command (with
                                        ! arguments) for the DRAW command
C
C all these arguments are writable.
C
```

11.4.10 **Error Handling**

11.4.10.1 **USRERR**

```
        SUBROUTINE USRERR(FATAL,ERRCOD)
C
C       Copyright Laser-Scan Ltd., Cambridge, England.
C
C Description
C
C       LITES2 cartographic editor user command routines .
C
        IMPLICIT NONE
C
C Arguments
C
        LOGICAL         FATAL           ! .TRUE. if user routine is
                                        !        about to be aborted
                                        ! .FALSE. if only a warning
        INTEGER*4       ERRCOD          ! error code
C
C  Error numbers passed to USRERR
C  (errors marked with * are fatal)
C
C* trying to get ACs or coordinates when there is no found feature
        PARAMETER       USR_NOFEATURE   =  1
C
C trying to get ACs from feature with none
        PARAMETER       USR_NOACS       =  2
C
C* tried to create a feature, while in an invalid state to do so
        PARAMETER       USR_NONEWCONSTR =  3
C
C* tried to create a feature in an non-existant map
        PARAMETER       USR_MAPNOTEXIST =  4
C
C* tried to create a feature in a read only map
        PARAMETER       USR_MAPREADONLY =  5
C
C* tried to create a feature in an non-existant layer
        PARAMETER       USR_LAYNOTEXIST =  6
C
C* tried to create a feature with an non-existant feature code
        PARAMETER       USR_BADCODE     =  7
C
C* tried to create a generated feature with an impossible feature code
        PARAMETER       USR_INVALFC     =  8
C
C* tried to create a feature with the wrong number of points
        PARAMETER       USR_WRNGNOPTS   =  9
C
C trying to create text feature with a height of an illegal point size
C (defaulted to 24)
        PARAMETER       USR_UNKPTSIZ    = 10
C
C trying to create an AC that is too long. It has been truncated
```

```
        PARAMETER       USR_ACTOOLONG   = 11
C
C* trying to construct a feature with a coordinate outside the limits
C  of the map
        PARAMETER       USR_PTOUTRANGE  = 12
C
C* trying to create a feature with zero length text
        PARAMETER       USR_TEXTTOOSHORT= 13
C
C* other error while constructing feature.
C The feature has been abandoned.
        PARAMETER       USR_FTABANDONED = 14
C
C* unrecognised return code returned by a USR* routine
        PARAMETER       USR_UNKRETCOD   = 15
C
C  trying to create text feature with an illegal height (in mms). Default
C  value used.
        PARAMETER       USR_UNKHTSIZ    = 16
C
C* error while setting a variable by a call of USRDEF
        PARAMETER       USR_VARIABLEERR = 17
C
C* error while calling USRGMH
        PARAMETER       USR_MH_MD_ERR   = 18
C
C* error while calling USRANO
        PARAMETER       USR_ANNO_ERR    = 19
C
C* error while calling USRDRW
        PARAMETER       USR_DRAW_ERR    = 20
```

11.5 Compiling And Linking A Shared Image.

For detailed information about shared images, reference should be made to the appropriate VAX VMS manuals.
This section is intended only as a guide, to allow users to compile and link their own simple images.

The required routines should be compiled normally, and then linked into a shareable image by using the /SHARE qualifier in the link command.

To allow LITES2 to locate the entry points of the routines that it calls it is necessary to include an options file that declares the required routines as UNIVERSAL. (Other routines used within the image should not be declared UNIVERSAL).

If the user routine image contains any common blocks, then they must be set to be non-shareable using the PSECT_ATTR linker command.

NOTE

> **If this is not done then the error message "Writable shareable images must be installed" will be given when the routine is first invoked.**
> The solution to this error is to set the common blocks non-shareable; do not try to install the shareable image!

There is an example user image supplied with LITES2 in the directory LSL$PUBLIC_ROOT:[LITES2.ROUTINES.EXAMPLE]. This may be built with the command file EXAMPLE_ROUTINE.COM which calls USRLNK.COM.
These illustrate the use of a library, USRLIB, which contains the specified user routines and any other subroutines that they call. They also show how common blocks must be included as non-shareable PSECTs in the link file.

EXAMPLE_ROUTINE.COM

```
$       SET ON
$       ON ERROR THEN GOTO EXIT
$       SET VERIFY
$       LIBR/CREATE USRLIB
$       FORTRAN/NOOPT/DEBUG USRDO,USRERR,USRGAC,USRINI,USRPAC
$       FORTRAN/NOOPT/DEBUG USRRET,USRSTO,USRGCB,USRGTX
$       FORTRAN/NOOPT/DEBUG USRPCB,USRPTX,USRDEF
$       LIBR/REPL USRLIB USRDO,USRERR,USRGAC,USRINI,USRPAC,-
                         USRRET,USRSTO,USRGCB,USRGTX,USRPCB,USRPTX,USRDEF
$!
$! and link it
$       @USRLNK
$EXIT:
$       PURGE USR*.*
$       SET NOVER
```

USRLNK.COM

```
$       LINK/DEBUG/SHARE=EXAMPLE_ROUTINE SYS$INPUT:/OPT
USRLIB/INCLUDE=(USRDO,USRGAC,USRINI,USRPAC,USRRET,-
USRSTO,USRGCB,USRGTX,USRPCB,USRPTX,USRERR,USRDEF)/LIB
UNIVERSAL = USRDO,USRGAC,USRINI,USRPAC,USRRET,USRSTO
UNIVERSAL = USRERR,USRPTX,USRGCB,USRGTX,USRPCB,USRDEF
PSECT_ATTR=USRKEEP,NOSHR
PSECT_ATTR=USRKEEPC,NOSHR
PSECT_ATTR=USRFEAT,NOSHR
PSECT_ATTR=USRFEATC,NOSHR
$!
```

To use this routine the logical name LSL$LITES2ROUTINES_1 should be set to point to the shared image. Then the command ROUTINE 1 1 can be given from LITES2, which gives instructions on how to use the rest of the shared image.

It should be noted that the above command files link a debugged version of this image. It is possible to use the debugger to examine the working of the image by giving the LITES2 command DEBUG. To set a break point on USRINI, for example, give the commands

```
DBG> SET IMAGE LSL$LITES2ROUTINES_1
DBG> SET BREAK USRINI
DBG> GO
```

NOTE

This will only be possible after the image has been mapped by at least one entry of the LITES2 command ROUTINE 1 x.

12 **Flagging of edited features in LITES2**

This section of the manual describes the commands which enable and utilise the flagging of edited and deleted features in LITES2.

Those features which have been edited and deleted may be flagged as such in the FS entry, and then may be selected for finding, display or output.

Edited (or constructed) features may additionally have a "date of edit" attached to them which can be updated to today's date at the end of a LITES2 session.

NB. The edited features are flagged in the fourth word of the FS entry, which has been designated for customer use. At present Laser-scan programs which process IFF files may not preserve information stored in the fourth word. There is therefore a danger of the edit flags being cleared if the files are processed by programs other than LITES2.

12.1 **The FLAGS OPTION**

The FLAGS option controls the way in which edited and deleted features are dealt with in LITES2. The three phases, input,editing and output, are all dependent on whether the FLAGS option has been set. These phases are described below.

12.1.1 **LITES2 Input**

If ENABLE FLAGS is set then on read in of the IFF file:-

> Edited and deleted flags are preserved. Deleted features may then be RECOVERed

If FLAGS is disabled then:-

> Edited flags are cleared and deleted features are not read in. This is the default mode of operation and is also a means by which the entire file may be cleared of edit flags.

12.1.2 **Editing Features**

During editing operations, if ENABLE FLAGS is set then:-

> Deleted and edited (or constructed) features are flagged. Features may be selected for finding or drawing by means of their flags (ie. SELECT EDITED,SELECT DELETED,SELECT UNEDITED)

If FLAGS is disabled then:-

> Deleted features only are flagged.

12.1.3 **Output**

Flags settings on output depend on whether SELECT OUTPUT is specified. If SELECT OUTPUT is specified then those features output will have edit and deleted flags cleared in the file produced. The WRITE command best utilises this facility as it will enable separate files containing deleted features, edited features or unedited features (or combinations) to be produced. The following command sequence would produce three files containing deleted,edited and unedited features. eg.

```
* SELECT OUTPUT
* SELECT DELETED
* WRITE DELFILE
* SELECT ALL
* SELECT EDITED
* WRITE EDITFILE
* SELECT ALL
* SELECT UNEDITED
* WRITE UNEDFILE
* DESELECT OUTPUT
```

It should be noted that:-

1. The flag selections are only implemented if the FLAGS option is enabled.

2. The SELECT ALL command automatically deselects deleted features. If they are required they should always be explicitly selected.

If SELECT OUTPUT is not specified then if the FLAGS option is enabled before exiting:-

Deleted features are output (if specifically selected) with deleted flags preserved and edited features with edited flags preserved.

If FLAGS is disabled then:-

Deleted features are not output and edited flags are still preserved.

12.2 **The DATE OPTION**

Each feature in the IFF file may have an ancillary code (AC) attached which holds the "date of last edit" in the value field. If the DATE option is enabled then those features flagged as edited or deleted will have this date updated at the end of a LITES2 session. If no such AC exists (eg. if a feature has been constructed during a session) then it will be inserted. The option is only valid if ENABLE FLAGS has already been specified. The format of the AC is:-

AC actype acdate

eg.

AC 110 870213 denotes a feature last edited on 13th February 1987

By default the AC type is 110 but can be specified by an optional argument to the ENABLE DATE command . If an AC type is specified which is different from the date ACs already existing in the IFF file then these will not be updated. Instead a new AC with the specified type will be added. It is advised, however, that customers should only use AC types which have been allocated by Laser-Scan.

12.3 **CHANGE EDITED**

The CHANGE EDITED gives explicit user control of the edit flag settings. When a feature is found, CHANGE EDITED 1 will set the edit flag while CHANGE EDITED 0 will clear it. Thus a macro could be written which would clear all flags if necessary (in a similar way in which deleted features can be recovered). It is not necessary for FLAGS to be enabled for this command to be used.

13 **Display Overlays and Raster Images**

These two topics are described together because they are related in some ways.

Display overlays allow the bit planes of a graphics display to be divided into independent groups called 'overlays'. Subsets of vector IFF data and raster images can be displayed in each overlay, with the advantage that the overlays can be made visible/invisible and be moved to foreground/background etc. independently and instantaneously. Overlays are created by the WORKSTATION OVERLAY command and are subsequently manipulated by the OVERLAY command. Although these commands are present in all version of LITES2, the ability to display overlays is confined to certain models of graphics workstation. See the appropriate Workstation Guide for details.

The ability to read in raster images is a licensed option in LITES2. It is only available in versions of LITES2 for displays capable of displaying overlays.

13.1 **Display Overlays**

Display overlays are created using the WORKSTATION OVERLAY command. This command allocates a number to the overlay - this number is used to refer to the overlay in future. The number of separate overlays which may be created is limited only by the number of pixel planes on the display, since each overlay must consist of a distinct contiguous set of planes. The particular planes to use is normally allowed to default, but an optional argument to the command allows the user to specify this if required. Remember that the number of colours (including a background colour) which may be displayed in an overlay is 2**number-of-planes i.e. 8 for 3 planes, 16 for 4 planes etc. In the case of workstation with primary and secondary displays both supporting overlays, then the overlays are shared between the two displays - creating an overlay, or setting its attributes, will automatically perform the same operation in the other.

Once an overlay has been created, it is advisable to set all colours in it. In addition to a colour value, specified by the OVERLAY COLOUR command, each colour has an attribute to specify how it interacts with the colours of overlays beneath it. It may, for instance, be TRANSPARENT, in which case the underlying colour will show through unchanged, and the setting of this colour will be irrelevant. Another attribute is OPAQUE, in which case the underlying colour will be totally obscured by this colour. The INVERSE attribute produces a complimentary colour to the underlying colour - the setting of this colour is irrelevant. Other attributes are ADD, SUB, and MERGE, which allow the underlying colour to combine with this colour in various ways. Attributes are set by the OVERLAY ATTRIBUTE command, which may be used to set the attributes of all the colours in an overlay simultaneously or individually. Note that if the background colour (colour 0) is made opaque, it will never be possible to see any overlays beneath this one.

An overlay may be made invisible or visible. This is done using the OVERLAY CONCEAL and OVERLAY REVEAL commands. An overlay is CONCEALed when created - it must be REVEALed before anything will be visible in it.

The final colour appearing at a given point on the screen is determined as follows. A ray of light is assumed to set off in the colour set by the OVERLAY BACKDROP command (this is not associated with any particular overlay). It is then modified as it passes through each visible overlay in turn, the colour of the ray interacting with the colour in the overlay according to the OVERLAY ATTRIBUTE for the colour in the overlay.

The exact behaviour of the different colour attributes is as follows. Assume that 'C' represents the amount of red, green, or blue in the ray of light passing through an overlay, and 'c' represents the colour in that overlay. The way that 'C' is modified on passing though the overlay is:

TRANSPARENT	C = C
OPAQUE	C = c
INVERSE	C = 1 - C
ADD	C = C + c - C*c
SUB	C = C*c
MERGE	C = (C + c)/2

The behaviour of TRANSPARENT and OPAQUE is fairly obvious. INVERSE produces a complimentary colour, for instance black<->white, yellow<->blue. ADD behaves like additive colours (rays of light) so that for instance red + green would give yellow. In ADD mode, the resulting colour is always brighter and nearer to white than the two component colours so that adding black to a colour leaves it unchanged, while adding white to a colour always produces white. SUB behaves like subtractive colours (paint pigments) so that for instance red + green would give black, while yellow + cyan would give green. In SUB mode, the resulting colour is always darker and nearer to black than the two component colours so that adding black to a colour produces black, while adding white to a colour leaves the colour unchanged. MERGE produces an effect intermediate to ADD and SUB - the two colours are averaged producing something intermediate in colour and brightness to the originals.

The order in which the overlays are traversed may be altered at will - it is not fixed by the particular planes allotted to the overlays. The OVERLAY POP command moves an overlay to the front of all the others, while OVERLAY PUSH moves it to the back. Using these two commands, it is possible to arrange any number of overlays in any order.

The default state of an overlay when created is that it is CONCEALed, POPped to the foreground, has a TRANSPARENT black colour 0 (background), and has all other colours OPAQUE white. The minimum which must be done to see anything in the overlay is to OVERLAY SELECT something to be drawn in it, OVERLAY REVEAL it, and re-draw.

Any changes to overlays affecting the appearance of the screen normally take place at once (this is actually achieved by re-computing a colour table and writing it to the display). If many changes are being made, it is inefficient to update the display after each, so the OVERLAY DEFER command prevents updating of the display. Once the changes have been made, the OVERLAY UPDATE command will update the display to the current state.

If no overlays are created, then LITES2 will continue to function as previously - the IFF vector data will be written into all available planes of the workstation. Colours are specified initially using a colour table file on disc, and subsequently using the WORKSTATION COLOUR command. Once any overlays have been created, LITES2 will use the overlays in preference to its default behaviour. In this case, the colour table file is irrelevant, and the WORKSTATION COLOUR command should not be used (it will still have an effect, but this will be overridden by the next OVERLAY command). Nothing will be displayed in an overlay until it has been explicitly selected. This is achieved using the OVERLAY SELECT/DESELECT commands. The original SELECT/DESELECT commands still function as before - they control which features may be found, drawn, or output. A feature must be selected both by ordinary SELECT commands, and OVERLAY SELECT commands to be drawn into an overlay. Although it is possible to select a feature for display in more than one overlay, it will only appear in the lowest numbered overlay. Having made appropriate selections, it is necessary to DRAW, WINDOW, ZOOM etc. to create the new display.

Raster images may also be displayed in overlays. These are selected using OVERLAY SELECT/DESELECT IMAGE commands.

The SHOW OVERLAYS command may be used to display details of all overlays, including the current selections for them.

To remove all selections from an overlay use OVERLAY CLEAR. It is then possible to delete the definition of the overlay using OVERLAY DELETE, freeing its planes for use by new overlays. If all overlays are deleted, then LITES2 will revert to its original non-overlay behaviour.

13.2 **Raster Images**

Up to 8 DTI files may be accessed simultaneously by LITES2. Only one LSI file may be accessed - it uses two of the slots. Up to 8 LSR files may be accessed, but depending on the format of the LSR file, it may require 1 or 2 slots, which could reduce the limit to 4. These are specified at any time using the IMAGE DTI, IMAGE LSI, or IMAGE LSR commands. An image is allocated a number from a preceding IMAGE NUMBER command - this number is used to refer to the image in future. An image may be closed (possibly to make room for another) using the IMAGE CLOSE command. Images should not be opened and closed unless absolutely necessary - this may eventually result in LITES2 running out of computer memory.

The SHOW IMAGES command may be used to display details of images.

In the case of DTI files, information on the pixelsize and origin of the file is taken from the projection record in the file header (if present). If this information is not available (this includes LSI files), then the information must be supplied using IMAGE ORIGIN and IMAGE PIXELSIZE commands. For DTI files, the orientation of the file may be specified using IMAGE CORNER and IMAGE DIRECTION commands. This information is normally also obtained from the file header. Because some displays are more efficient at displaying raster images in certain directions, it may be advisable to use the MATRIX package utility DTIROTATE to alter the file to the optimum orientation.

The origin of a DTI file is taken to be at the centre of the bottom left pixel, while for LSI files it is at the bottom left of this pixel. LSR files can behave either way, according to whether they have point or area type pixels.

LITES2 will proceed from INITIAL to READY state when the required number of IFF files have been specified, or the MAPS 0 command is given (in this case, at least one image file must be open). Any image files open at this time will be used in the calculation of the 'working area' (the maximum x and y extents of image and IFF files + 5% all round). Although image files lying outside this area may be specified subsequently, it will not be possible to access any parts of these files lying outside the working area. The RANGE LIMITS command may be used to set a larger working area initially. Images are not drawn during initial read-in. A DRAW, WINDOW, or ZOOM command must be given before they appear.

Images must be displayed in overlays. The OVERLAY SELECT/DESELECT IMAGE commands are used to specify which images are to be displayed in which overlays. Several images may be displayed in a single overlay (this is not useful if the images overlap - the higher numbered image will overwrite the others), and the same image may be displayed in several overlays (uses of this are not obvious, but there may be some!).

The colour displayed for a given pixel value is obtained by one of two methods. The default method, which is the fastest, is to take the pixel value as the colour index directly. The only control is provided by the IMAGE BITS command, which allows a set of bits to be selected from the pixel value. Another way of looking at the IMAGE BITS command is that the displayed colour is obtained from the data value as follows:

display_colour = (data_value/2**first_bit) modulo 2**number_of_bits

For instance, if a 16 bit (WORD format) DTI file was to be displayed in a 4 plane overlay, the default action would be to display the 4 least significant bits. The IMAGE BITS command could be used to display some other set of contiguous bits instead. The number of bits specified need not be the same as the number of planes in the overlay: extra high order bits are ignored, or the value is padded with 0 at the high order end if insufficient bits are specified (thus one could for instance display a single bit in a multi-plane overlay).

The second method of producing a colour index from the pixel value is called 'classification'. This is switched on by the IMAGE STEP command, and further controlled by the IMAGE RANGE, IMAGE FIRSTCOLOUR, IMAGE BAND, and IMAGE SEA commands. Although slower than the IMAGE BITS method, classification allows greater versatility is displaying the image.

In order for several image files to be displayed simultaneously, they must have the same pixelsize. If files with different pixel sizes are selected for display, then any which do not have the same pixel size as the lowest numbered file will not be displayed.

It is not possible to display image files at arbitrary zoom factors, since each image pixel must correspond to an integer number of screen pixels. In particular, the maximum number of pixels which may be displayed is limited by the number of pixels on the display screen. If the working area is larger than this, it will be impossible to display the whole area. In this case, for DTI or LSR files, the image is subsampled, taking every n'th pixel, but otherwise the window will be automatically reduced. When WINDOWing or ZOOMing into the picture, the nearest integer zoom factor displaying at least the required area will be chosen. A result of this is that the area drawn may not correspond exactly to the area expected. The SHOW WINDOW command may always be used to determine the area currently on the screen. LSI files may contain 'reduced views', with less pixels at a degraded resolution compared to the original view. These will be used whenever possible to allow a larger area to be displayed than would otherwise be possible. A reduced view may only be used if all selected image files contain the particular reduced view. Note that DTI and LSR files do not contain reduced views - subsampling is used instead.

The command ZOOM n IMAGE may be used to attempt to draw the image at a particular zoom factor, for instance ZOOM 2 IMAGE will draw with 2 screen pixels (in each direction) for each image pixel, while ZOOM 0.5 IMAGE (or ZOOM 1/2 IMAGE) will sub-sample with each screen pixel representing 2 image pixels in each direction.

For LSR files containing bit data, the command IMAGE SUBSAMPLE PRIORITY n may be used to specify that rather than missing out pixels completely when subsampling, pixel value 0 or 1 be given priority. This can help prevent the break up of subsampled images, with the loss of fine detail. IMAGE SUBSAMPLE FAST reverts to the normal behaviour.

The image data value at the current cursor position may be obtained using the $IMAGEVALUE system variable. An estimate of the gradient and aspect (direction of maximum gradient) may be obtained using the $IMAGEGRADIENT and $IMAGEASPECT system variables. It is not necessary for the image to be displayed in order to do this, indeed it is not necessary to have a graphics display at all - the 'cursor' may be positioned by keyboard commands. Since several images may cover the same point, it is necessary to specify the set of images to consider for this purpose using the IMAGE SELECT command. More than one image may usefully be specified in the case when they 'tile' together to form a single image. In the case when selected images overlap, the value will be taken from the highest numbered image.

When producing hardcopy plots which include LSR images, then logical name LSL$LSR_PLOT_MODE provides some control over the way the image data is written to the plot file. The default (equivalent to defining the logical name as 0) is to plot non-subsampled images in the tiles in which they are stored in the LSR file. For very large files, this has sometimes resulted in more tiles than the plotter or rasteriser can cope with. Subsampled images are plotted in swathes the full width of the image, and as tall as allowed by the memory buffers, which are allocated in proportion to logical name LSL$FILL_POINTSMAX (default 8192). If LSL$LSR_PLOT_MODE is defined as 1, then swathes are used even for non-subsampled images, hence the number of swathes can be controlled by the size of the buffers. Long thin swathes have sometimes caused problems if the plotter tries to rasterise the plot with scan lines running in the opposite direction to the swathes. If LSL$LSR_PLOT_MODE is defined as 2, then the plot is drawn as square tiles as large as will fit in the buffer, so rasterising in either direction should be equivalent. If LSL$LSR_PLOT_MODE is defined to be greater than 100, then square tiles with the specified number of pixels on each side will be used (provided they will fit in the buffer). For bit images, tile sizes which are a multiple of 8 may give better performance.

13.3 **Raster Editing**

LITES2 supports some editing of raster images. This facility is at present restricted to LSR type files containing bit data. To enable editing, the command IMAGE EDIT must be given (after IMAGE NUMBER, but before IMAGE LSR). IMAGE READONLY is the opposite of IMAGE EDIT, and is the default.

Once the image is open, its background and foreground values may be specified using IMAGE BACKGROUND (default 0) and IMAGE FOREGROUND (default 1). Since only bit images are supported, the values must be either 0 or 1. These commands control the behaviour or the rest of the image editing commands.

Most of the image editing commands operate on the interior of one of LITES2's REGIONs. The IMAGE REGION command specifies which region number is to be used. The region need not exist when this command is given, and may be redefined subsequently. The region may be created in all the usual ways, plus the more recently implemented REGION WINDOW (quickly constructs rectangular regions using the cursor), and REGION IMAGE (a region constructed from the image data itself, around pixels of the same colour).

IMAGE CLEAR and IMAGE FILL fill the interior of the region with background and foreground colour respectively.

IMAGE COPY and IMAGE MOVE both allow the image data within the region to be attached to the cursor, and moved until an END or ABANDON command is given. IMAGE COPY retains the original data, while IMAGE MOVE fills the original region with background.

IMAGE PAINT and IMAGE ERASE allow the image to be 'painted' in foreground and background colour respectively using the cursor. PAINT state is entered and copies of the current brush are deposited each time the cursor is moved until END is given (or ABANDON, which undoes any painting). The shape and size of the brush is controlled by the IMAGE BRUSH CIRCLE/RECTANGLE commands.

IMAGE SPECKLE CLEAR and IMAGE SPECKLE FILL search the region for speckles of foreground or background colour respectively, smaller than a given size, and remove them by painting with background and foreground colour respectively.

When searching for speckles, or constructing a region with REGION IMAGE, pixels may be taken as connecting if the touch by their sides (IMAGE CONNECT SIDE, the default), or also diagonally (IMAGE CONNECT DIAGONAL).

IMAGE BURN_IN allows any data currently displayed on the screen to be 'burned into' the raster image. This operation affects just the screen area, and the image must be displayed with one screen pixel to one image pixel at the time (the command ZOOM 1 IMAGE facilitates this). Any pixels displaying a colour other than the image background are written into the image in the foreground colour.

IMAGE RECOVER allows the effect of the last image editing command to be undone, restoring the image to its previous state.

In order to point accurately to image pixels, without having to zoom in on the main image, the DRAW IMAGE command may be used to draw the image into the annotation display (3 or 4, if available), and pointing in this display will then track the LITES2 cursor. A cursor may be displayed in the annotation display also using DISPLAY CURSOR.

13.4 **Image Registration**

LITES2 allows an IFF file map to be registered on the screen with a raster image when the two are not in exactly the same coordinate space. One or more images, and an IFF map, are specified and read in as normal. It is advisable that the pixel size and the origin of the images is set so as to make the data line up as well as possible before performing the registration. Once in READY state, the command IMAGE SETUP should be given, and LITES2 will enter SETUP state and prompt for the position of the NW corner point of the IFF file to be digitised on the screen. WINDOW, DRAW, and ZOOM commands may be given to get the correct area of the raster image on the screen. The cursor should be moved to the position on the image where the NW corner point of the IFF file should be - the command START will digitise the point and move on to the SW corner point. When all 4 corners have been successfully digitised, LITES2 will return to READY state and the next redraw will register the corner points of the IFF file with the image. ABANDON may be used at any time in SETUP state to abort the SETUP and return to default behaviour.

The EXTENDED transformation (see SETUP TRANSFORM) is used for the registration, which means that the 4 corner points will fit exactly to the positions digitised, with the rest of the map being distorted as required.

14 Alternative Text and Symbol Drawing Routines

There are facilities in LITES2 to allow sites to have their own text and symbol drawing routines. This is a specialised facility and it is not envisaged that users would write their own routines without help from Laser-Scan.

The interface described below is liable to be altered at the discretion of Laser-Scan. The main purpose of the fragments of Fortran source code below is to document the arguments passed to the routines. The actual code merely draws and blanks the features in exactly the same way that LITES2 would have done if the routine had not been supplied.

The examples below, together with command files to build them, are in LSL$PUBLIC_ROOT:[LITES2.ROUTINES.DRAWING]. The routines are passed several routines from the FRT library as arguments. Since LITES2 uses the shareable image version of the FRT library, the routines may also be linked with this and will then share the library and associated data in common blocks. If for any reason variables in the FRT common blocks are changed, then they should be restored to their previous values afterwards, otherwise LITES2 could behave unpredictably. Similar caution should be exercised in calling any FRT routines other than those passed as arguments.

14.1 Text drawing

If the logical name LSL$TEXT_ROUTINE points to a shared image which has a UNIVERSAL entry point called DRAW_TEXT, the routine supplied will be called rather than the LITES2 internal routine. The specification of DRAW_TEXT is

```
        SUBROUTINE DRAW_TEXT(
     &          TEXT,FC,SC,XPOS,YPOS,AUX,TRITXT,SRIPLT,TRISCN,FS,HW)
C
C Copyright Laser Scan Laboratories Ltd., Cambridge, England.
C
C The routine is expected to be used as a shareable image by LITES2
C and FPP, on logical name LSL$TEXT_ROUTINE.
C
        IMPLICIT NONE
C
C define layout of AUX array
C
        PARAMETER AUXLEN = 8    ! length of auxiliary array
        PARAMETER ANGI = 1      ! angle
        PARAMETER COSI = 2      ! cosine
        PARAMETER SINI = 3      ! sine
        PARAMETER SIZI = 4      ! size
        PARAMETER MINXI= 5      ! minimum x
        PARAMETER MAXXI= 6      ! maximum x
        PARAMETER MINYI= 7      ! minimum y
        PARAMETER MAXYI= 8      ! maximum y
C
C arguments
        CHARACTER*(*) TEXT              ! text to draw
        INTEGER FC                      ! feature code
        INTEGER SC                      ! secondary code (font) from FRT
```

```
        REAL    XPOS,YPOS                ! coordinate
        REAL    AUX(AUXLEN)              ! array of extra information
        LOGICAL TRITXT                   ! FRTLIB routine to draw text
        LOGICAL SRIPLT                   ! FRTLIB routine to draw symbol
        LOGICAL TRISCN                   ! FRTLIB routine to size text
        INTEGER*2 FS(4)                  ! Feature status for feature.
        LOGICAL HW                       ! use h/w text?
C
C local variables
        REAL    X,Y
C
C This example code draws the text in exactly the same way
C as LITES2 or FPP themselves would have done.
C
C calculate position of bottom left of text (allow justification)
        X = XPOS+AUX(MINXI)*AUX(COSI)-AUX(MINYI)*AUX(SINI)
        Y = YPOS+AUX(MINXI)*AUX(SINI)+AUX(MINYI)*AUX(COSI)
        CALL TRITXT(TEXT,SC,X,Y,AUX(SIZI),AUX(ANGI),HW)
C
        RETURN
        END
```

14.2 **Symbol drawing**

If the logical name LSL$SYMBOL_ROUTINE points to a shared image which has a UNIVERSAL entry point called DRAW_SYMBOL, the routine supplied will be called rather than the LITES2 internal routine. The specification of DRAW_SYMBOL is

```
        SUBROUTINE DRAW_SYMBOL(
     &          FC,SC,NATT,ATTC,ATTV,XPOS,YPOS,AUX,TRITXT,SRIPLT,
     &          TRISCN,DRAW_TEXT,FS)
C
C Copyright Laser Scan Laboratories Ltd., Cambridge, England.
C
C The routine is expected to be used as a shareable image by LITES2
C and FPP, on logical name LSL$SYMBOL_ROUTINE.
C
C
        IMPLICIT NONE

C define layout of AUX array (copy of LITES2 CMN:AUXDEF.PAR)
C
        PARAMETER AUXLEN = 8    ! length of auxiliary array
        PARAMETER ANGI = 1      ! angle
        PARAMETER COSI = 2      ! cosine
        PARAMETER SINI = 3      ! sine
        PARAMETER SIZI = 4      ! size
        PARAMETER MINXI= 5      ! minimum x
        PARAMETER MAXXI= 6      ! maximum x
        PARAMETER MINYI= 7      ! minimum y
        PARAMETER MAXYI= 8      ! maximum y
C
C arguments
        INTEGER FC                       ! feature code
```

```
C
        INTEGER         SC              ! secondary code from FRT
        INTEGER         NATT            ! number of attributes
        INTEGER         ATTC(*)         ! the column headers
        INTEGER         ATTV(*)         ! the attributes
        REAL            XPOS,YPOS       ! coordinate
        REAL            AUX(AUXLEN)     ! array of extra information
        LOGICAL         TRITXT          ! FRTLIB routine to draw text
        LOGICAL         SRIPLT          ! FRTLIB routine to draw symbol
        LOGICAL         TRISCN          ! FRTLIB routine to size text
C
        EXTERNAL        DRAW_TEXT       ! user's text drawing routine
        INTEGER*2       FS(4)           ! feature status for feature
C
C This example code draws the symbol in exactly the same way
C as LITES2 or FPP themselves would have done.
C
        CALL SRIPLT(SC,XPOS,YPOS,AUX(SIZI),AUX(ANGI))
C
        RETURN
        END
```

14.3 **Text blanking**

The same shared image on logical name LSL$TEXT_ROUTINE may have a second UNIVERSAL entry point called BLANK_TEXT. The routine supplied will be called to supply a blanking polygon for a text rather than the LITES2 internal routine. The specification of BLANK_TEXT is

```
        SUBROUTINE BLANK_TEXT(
     &          TEXT,FC,SC,N,XY,OFF,CMPLX,
     &          XPOS,YPOS,AUX,
     &          TRI_BOUND,SRI_BOUND,TRISCN,SRI_OFFSET_POLYGON,
     &          FS,HW)
C
        IMPLICIT NONE
C
C Copyright Laser Scan Laboratories Ltd., Cambridge, England.
C
C The routine is expected to be used as a shareable image by LITES2
C and FPP, on logical name LSL$TEXT_ROUTINE.
C It is called to return an outline polygon for a text feature when
C a line or area code is used as part of the prioritised representation
C of a text.
C If N is returned zero or negative, then the programs will supply the
C default blanking.
C
C define layout of AUX array (copy of LITES2 CMN:AUXDEF.PAR)
C
        PARAMETER AUXLEN = 8    ! length of auxiliary array
        PARAMETER ANGI = 1      ! angle
        PARAMETER COSI = 2      ! cosine
        PARAMETER SINI = 3      ! sine
        PARAMETER SIZI = 4      ! size
```

```
        PARAMETER MINXI= 5      ! minimum x
        PARAMETER MAXXI= 6      ! maximum x
        PARAMETER MINYI= 7      ! minimum y
        PARAMETER MAXYI= 8      ! maximum y
C
C arguments
        CHARACTER*(*) TEXT              ! text to draw
        INTEGER FC                      ! feature code
        INTEGER SC                      ! secondary code (font) from FRT
        INTEGER N                       ! number of points in/out
        REAL    XY(2,*)                 ! the blanking points
        REAL    OFF                     ! offset from FRTSIZ
        LOGICAL CMPLX                   ! complex blanking?
        REAL    XPOS,YPOS               ! coordinate
        REAL    AUX(AUXLEN)             ! array of extra information
        LOGICAL TRI_BOUND               ! FRTLIB routine to get text bounds
        LOGICAL SRI_BOUND               ! FRTLIB routine to get symbol bounds
        LOGICAL TRISCN                  ! FRTLIB routine to size text
        EXTERNAL SRI_OFFSET_POLYGON     ! FRTLIB routine to offset polygon
        INTEGER*2 FS(4)                 ! feature status
        LOGICAL HW                      ! use h/w text?
C
C local variables
        INTEGER I
        REAL    TXY(2)
        REAL SIZE,SINANG,COSANG,OFFXY(2)
        INTEGER NIN
C
        NIN = N         ! save input size of array
C
        SIZE = AUX(SIZI)
        IF (CMPLX) THEN
C
C Use TRI_BOUND to get a region around the text.
C Just for example, use an offset of 0.0, then offset the
C region ourselves. We are passed OFF as a fraction of the size.
           IF (TRI_BOUND(TEXT,SC,N,XY,0.0,HW)) N = 0
C
C Offset the region
           CALL SRI_OFFSET_POLYGON(NIN,XY,N,OFF)
C
C Save bottom left position (e.g. for text which is not bottom-
C left justified)
           OFFXY(1) = AUX(MINXI)
           OFFXY(2) = AUX(MINYI)
        ELSE
C
C Simple blanking - just generate a box
           N = 4
           XY(1,1) = AUX(MINXI)-OFF*SIZE
           XY(2,1) = AUX(MINYI)-OFF*SIZE
           XY(1,2) = AUX(MAXXI)+OFF*SIZE
           XY(2,2) = XY(2,1)
           XY(1,3) = XY(1,2)
           XY(2,3) = AUX(MAXYI)+OFF*SIZE
           XY(1,4) = XY(1,1)
```

```
           XY(2,4) = XY(2,3)
           SIZE = 1.0           ! already scaled to size
           OFFXY(1) = 0.0
           OFFXY(2) = 0.0
        ENDIF
C
C Scale, offset, and rotate the region to its final position
        COSANG = AUX(COSI)
        SINANG = AUX(SINI)
        DO 40 I=1,N
           TXY(1) = XY(1,I)*SIZE + OFFXY(1)
           TXY(2) = XY(2,I)*SIZE + OFFXY(2)
           XY(1,I) = XPOS + TXY(1)*COSANG - TXY(2)*SINANG
           XY(2,I) = YPOS + TXY(1)*SINANG + TXY(2)*COSANG
40      CONTINUE
C
        RETURN
        END
```

14.4 **Symbol blanking**

The same shared image on logical name LSL$SYMBOL_ROUTINE may have a second UNIVERSAL entry point called BLANK_SYMBOL. The routine supplied will be called to supply a blanking polygon for a symbol rather than the LITES2 internal routine. The specification of BLANK_SYMBOL is

```
        SUBROUTINE BLANK_SYMBOL(
     &          FC,SC,N,XY,OFF,CMPLX,
     &          NATT,ATTC,ATTV,XPOS,YPOS,AUX,
     &          TRI_BOUND,SRI_BOUND,TRISCN,SRI_OFFSET_POLYGON,
     &          BLANK_TEXT,FS)
C
        IMPLICIT NONE
C
C Copyright Laser Scan Laboratories Ltd., Cambridge, England.
C
C The routine is expected to be used as a shareable image by LITES2
C and FPP, on logical name LSL$SYMBOL_ROUTINE.
C It is called to return an outline polygon for a symbol feature when
C a line or area code is used as part of the prioritised representation
C of a symbol.
C If N is returned zero or negative, then the programs will supply the
C default blanking.
C
C define layout of AUX array (copy of LITES2 CMN:AUXDEF.PAR)
C
        PARAMETER AUXLEN = 8    ! length of auxiliary array
        PARAMETER ANGI = 1      ! angle
        PARAMETER COSI = 2      ! cosine
        PARAMETER SINI = 3      ! sine
        PARAMETER SIZI = 4      ! size
        PARAMETER MINXI= 5      ! minimum x
        PARAMETER MAXXI= 6      ! maximum x
        PARAMETER MINYI= 7      ! minimum y
```

```
        PARAMETER MAXYI= 8      ! maximum y
C
C arguments
        INTEGER FC                          ! feature code
        INTEGER SC                          ! secondary code (font) from FRT
        INTEGER N                           ! number of points in/out
        REAL    XY(2,*)                     ! the blanking points
        REAL    OFF                         ! offset from FRTSIZ
        LOGICAL CMPLX                       ! complex blanking?
        INTEGER NATT                        ! number of attributes
        INTEGER ATTC(*)                     ! the columns headers
        INTEGER ATTV(*)                     ! the attributes
        REAL    XPOS,YPOS                   ! coordinate
        REAL    AUX(AUXLEN)                 ! array of extra information
        LOGICAL TRI_BOUND                   ! FRTLIB routine to get text bounds
        LOGICAL SRI_BOUND                   ! FRTLIB routine to get symbol bounds
        LOGICAL TRISCN                      ! FRTLIB routine to size text
        EXTERNAL SRI_OFFSET_POLYGON         ! FRTLIB routine to offset polygon
        EXTERNAL BLANK_TEXT                 ! user's text blanking routine
        INTEGER*2 FS(4)                     ! feature status
C
C local variables
        INTEGER I
        REAL    TXY(2)
        INTEGER BOUND_TYPE
        REAL SIZE,SINANG,COSANG
        INTEGER NIN
C
        NIN = N         ! save input size of array
C
        SIZE = AUX(SIZI)
C
C Set BOUND_TYPE for SRI_BOUND
        IF (CMPLX) THEN
           BOUND_TYPE = 3                   ! convex hull
        ELSE
           BOUND_TYPE = 2                   ! bounding box
        ENDIF
C
C Just for example, use an offset of 0.0, then offset the
C region ourselves. We are passed OFF as a fraction of the size.
        IF (SRI_BOUND(SC,N,XY,0.0,BOUND_TYPE)) N = 0
C
C Offset the region
        CALL SRI_OFFSET_POLYGON(NIN,XY,N,OFF)
C
C Scale, offset, and rotate the region to its final position
        COSANG = AUX(COSI)
        SINANG = AUX(SINI)
        DO 40 I=1,N
           TXY(1) = XY(1,I)*SIZE
           TXY(2) = XY(2,I)*SIZE
           XY(1,I) = XPOS + TXY(1)*COSANG - TXY(2)*SINANG
           XY(2,I) = YPOS + TXY(1)*SINANG + TXY(2)*COSANG
40      CONTINUE
C
```

```
        RETURN
        END
```

15 **Auxiliary Input to LITES2**

LITES2 accepts interactive input from a variety of devices. The devices available depend on the hardware that LITES2 is being run on and always include a terminal and possibly one or more of:

* bitpad (attached to the graphics device)
* button box (on a separate serial line)
* digitising table (either on a separate serial line or multiplexed on the graphics device serial line with a LSL muart)
* keyboard (attached to the graphics device)
* mouse (attached to the graphics device)
* screen menu
* tracker ball (attached to the graphics device)
* Kern DSR photogrammetric plotter

In addition facilities are available to accept LITES2 commands from additional serial devices. These could be additional terminals, mailboxes or even some other physical device attached by a serial line. At present there are 4 of these auxiliary input lines available, and they are accessed by the logical names:

* LSL$LITES2AUX
* LSL$LITES2AUX_2
* LSL$LITES2AUX_3
* LSL$LITES2AUX_4

If any of these logical names are set up when LITES2 is activated, then when interactive input is required, the program will accept LITES2 commands (as ASCII strings of up to 255 characters) from any of these devices.

Reading of interactive input is initiated by the triggering of an event flag, at which point all the interactive devices are checked to see if there has been any input. The auxiliary input devices are checked last, and they are all serviced before any more input is accepted. This means that no one input device can have exclusive control of LITES2; the input from the devices is generally interleaved.
There are obviously situations where it is necessary for a series of LITES2 commands to be executed sequentially, with no possibility of interruption from other devices. In this case the commands should be concatenated (with the concatenation character #) into one line. If a series of commands is required that when concatenated would make more than 255 characters, then a command file must be written, and then this can be read by using the file reading command @filename.

16 **Using LITES2 as a 3 dimensional editor**

16.1 **LITES2 as a 2 dimensional or a 3 dimensional editor**

In its default mode of operations, LITES2 is a two dimensional editor. This means that its geometrical treatment of features only relates to X and Y coordinates. Z information can be associated with individual points, but it is treated as an attribute of that point, not as a coordinate. Any manipulation of Z values is the responsibility of the operator, and is carried out by means of the EDIT ATTRIBUTE Z command.

NOTE

> The Z point attribute is unique, in that it is always available whether or not the logical name LSL$IFF_OUTPUT_REVISION has been set to 1 or not.

By use of the (licensed) command ENABLE Z, LITES2 can be made to operate as a 3 dimensional editor. This means that Z is treated as a 3rd coordinate of all the points in the IFF files, and can be edited graphically in the same way as the X and Y coordinates. This graphical editing is carried out using the 3 dimensional nature of the LITES2 cursor.

16.2 **The 3 dimensional cursor in LITES2**

The LITES2 cursor can be positioned in 3 dimensions. This is achieved by the use of the optional 3rd argument in the POSITION and ABSOLUTE commands. If this argument is not given then the height of the cursor is undefined.

Digitising devices (e.g. digitising tables, bitpads or mice) have no height information and only position the cursor in two dimensions (X and Y), however when LITES2 is using the KERN DSR photogrammetric plotter as an input device, it positions the cursor in three dimensions (X, Y and Z).

NOTE

> **Great care must be taken when using digitising tables, bitpads and mice to position the cursor when using LITES2 with Z enabled. It is very easy to lose the z coordinates of points under these conditions.**

The height of the cursor can be ascertained at any time by the SHOW POSITION command. If ENABLE Z has been given then all three coordinates will be displayed; if the cursor height is undefined then a "?" is displayed in place of the Z coordinate. If Z is disabled (the default state) then the Z coordinate of the cursor is only displayed if it is set to a definite value.

These values are available in the system variables $CURSX, $CURSY and $CURSZ. There is an additional variable $CURSZ_EXIST which is true if the Z value of the cursor is defined.

16.3 **Positioning the cursor on a feature**

When using LITES2 as a 2 dimensional editor and moving the cursor to a position defined by a feature (by FIND, SEARCH, FIRST, LAST, FRACTION etc) the height of the cursor is unaltered.

However when using LITES2 as a 3 dimensional editor, and moving the cursor to a position defined by a feature (by FIND, SEARCH, FIRST, LAST, FRACTION etc) the height of the cursor depends on the Z value of each point in the feature. If the cursor is positioned between two points, the Z value is interpolated in exactly the same way as the plan values are.

If, while the cursor is constrained on a feature, another feature is intersected, then the height of the cursor is determined by the height of the points in the original feature.

Optional Z values

> While each point in an IFF file **must** have an X and Y value, Z values need not be present. If the cursor is moved to between two points that do not have Z values, the cursor height will not be altered; if however one of the adjacent points is heighted, then the cursor will take its value.
>
> It should also be noted that Z values which are not present are undefined. **They are not 0.0.** 3 dimensional displays (eg the Kern KRISS image injection system) may however treat undefined values as being 0.0.

It is strongly recommended that when using LITES2 as a 3D editor, all existing data has proper 3D coordinates.

NOTE

> The 3 dimensional editing capability of LITES2 uses the Z attribute of individual points. It does not use any height information held in ancillary codes (ACs) 2 or 3 (contour or real height).

16.4 **Constructing new 3 dimensional features.**

Linear 3 dimensional features are simply created by positioning the LITES2 cursor and giving the START command. If the Z position is defined then this value will be added to the feature. To complete the feature the END command is given at the last point.

It should be noted that when operating in 3 dimensional mode, it is possible to have two adjacent points on a feature with the same plan position, as long as their heights are different. If such a feature is subsequently read into LITES2 when Z is disabled, then the second point will be lost.

If points are interpolated in a linear feature using the CURVE command, then the heights of the interpolated points are equally spaced between the heights of the master points. This is true whether Akima or McConalogue interpolation in X and Y is being used.

The heights of points generated by the CIRCLE, POLYGON, ARC, POLARC and RECTANGLE commands are determined by projecting the generated plan positions onto the plane through the master points. If there are only two master points (eg CIRCLE CENTRED) then the plane whose direction of maximum slope goes through these points is used. The affect of this is to generate features that have the expected shape on the map, not in 3 dimensional space.

Features of graphical types 2,3,4 and 5 (various types of circles and arcs) are constructed using appropriate START and END commands in the usual way, and a feature that is circular in plan is created.

Each of these features has a 3 dimensional plane associated with it, and this is used to determine the height of any point on it (e.g. this plane is used to position the cursor at the correct height during editing operations).

Symbols and texts take their height from the cursor when START is given. The height of the cursor at any second orienting or scaling point is ignored. Separate subtexts in composite texts can each have a different height.

16.5 **Editing 3 dimensional features**

Most of the 2 dimensional editing operations are available when using LITES2 as a 3 dimensional editor. The only exception to this is the FILTER command, which is not available when Z has been enabled.

There follows some comments on the manipulation of Z during various editing operations.

16.5.1 **EDIT, EXTEND and INSERT**

These commands allow the Z value of a point to be altered (as well as its plan position) depending on the position of the cursor. The Z value can also be edited explicitly using the EDIT ATTRIBUTE Z command.

While using these commands the commands FORCE FLAT and FORCE SLOPE can be used to position the Z value of the cursor relative to the vector being edited. FORCE FLAT will return Z to the value it was at when the editing command was given; FORCE SLOPE will force Z to lie on the plane whose direction of maximum slope lies through the two points defining the vector being edited.

NOTE

> When using the EXTEND command, the cursor is constrained to move in the line defined by the end vector of the feature. This constraint only applies to the plan position. The cursor will maintain any Z value it is set to.

16.5.2 JOIN, TIE, MEND and LOOP

These commands position the cursor between the two points being joined together. This applies to the Z coordinate in the same way as it does to X and Y. If the Z value of one point is undefined, then the Z value of the other point is used.

If the PROPAGATE command is given while JOINing or TIEing, the difference in Z between the selected point and the ends of the features is smoothed out along the same distance as the differences in plan position are.

16.5.3 CLOSE

CLOSE (NORMAL) positions the cursor to the X, Y and Z of the first point in the construction.

CLOSE SQUARE inserts a point on the line between the last digitised point and the cursor position, when the CLOSE SQUARE command is given. The height of this point is interpolated between the last digitised point and the cursor position, in the same way as the plan position. The cursor is finally positioned to the first point in the construction.

16.5.4 Part Operations, SPLIT, CLIP, INCLUDE and BRIDGE

Editing operations that generate new points based on existing features linearly interpolate a Z value for these points in the same way as the X and Y are calculated. When circle arcs are clipped or split, the resulting features all lie on the same plane as the original feature.

16.5.5 ORIENT, TRANSFORM and OFFSET

When the plan positions of features are altered using these commands the Z values of the points are not altered.

16.5.6 MOVE

When the plan positions of features are altered using this command, the Z values of all the points are altered by the difference in height between the point on the feature that the cursor was at when MOVE was given and the cursor height when END is given.

16.5.7 EDGEMATCH

While edgematching, heights are treated as in the primitive JOIN and TIE commands. As with the differences in plan position, the misclosure in Z is smoothed out back along both features if the propagating tolerance is greater than 0.0.

NOTE

When selecting features to edgematch, no account is taken of the Z coordinates. Only the plan position is considered.

16.5.8 **Composite Text manipulation**

When the PARAGRAPH command is given, all the resulting subtexts acquire the height of the first subtext.

When using the according to its position on the locating feature.

INDEX