Laser-Scan Ltd.

LITES2

X-Windows (MOTIF) Workstation Guide

Issue 1.7

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

Document "LITES2 - X-Windows (MOTIF) Workstation Guide" Category "USER"
Document Issue 1.0 Clarke Brunt 2-Jul-1990
Document Issue 1.1 Clarke Brunt 24-Mar-1991
Document Issue 1.2 Clarke Brunt 21-May-1991
Document Issue 1.3 Clarke Brunt 6-Apr-1992
Document Issue 1.4 Clarke Brunt 17-Dec-1992
Document Issue 1.5 Clarke Brunt 21-Jul-1993
Document Issue 1.6 Clarke Brunt 25-Sep-1996
Document Issue 1.7 Ron Russell 19-Nov-2001

1 *Introduction*

This document describes the workstation dependent facilities available in the version of LITES2 for X-Windows workstations (image LITES2MOTIF.EXE). It is to be read as a supplement to the LITES2 Reference Manual and the LITES2 User's Guide.

2 *Workstation*

Although the output from this version of LITES2 may in principle be directed to any workstation or terminal running X-Windows, Laser-Scan only support interaction via a DEC VMS workstation running DECwindows and the Motif window manager (mwm). LITES2 may be run from a terminal emulating window on the display screen, or from a separate terminal, possibly on another network node, in which case an appropriate (DCL) SET DISPLAY command will be required to direct the output to the workstation.

X-Windows workstations normally run a window manager which is responsible for putting decorative borders and banners round the screen windows, and providing functionality for raising and lowering windows etc. This version of LITES2 is designed to cooperate with the Motif window manager (mwm) - if any other is used, some unexpected behaviour may be exhibited. It is advisable to edit the following line into the window manager defaults file DECW$MWM.DAT (in the user's login directory):

```
Mwm*focusAutoRaise:     false
```

This prevents windows from being raised when this is not required.

Up to 4 display windows are supported. GRAPHICS must be enabled. Primary and secondary displays are enabled using ENABLE PRIMARY and ENABLE SECONDARY. Both primary and secondary windows support cursor movement, highlighting, and interaction. If both primary and secondary displays are enabled, then it is likely that the SUPPRESS command will be used in order for instance to keep a full map view in one window while zooming in the other. The third and fourth display windows are used by the VIEW command, or the various ANNOTATION and DRAW commands.

The attributes (size, shape, number of colours etc.) of all four windows may be set using DISPLAY commands. It may be necessary to restrict the number of colours in each display (using DISPLAY COLOURS, and the mechanism described below) so as not to exceed the maximum number of colours which the workstation can display independently. The colours in the primary and secondary windows are taken initially from the colour lookup table on logical name LSL$DECW_COLOUR (see below), and the WORKSTATION COLOUR command affects both windows. Overlays may be created (by WORKSTATION OVERLAY) in the primary and secondary displays - the colours and overlay structure are shared between the two, so for instance creating an overlay in one also creates the same overlay in the other.

ENABLE SEGMENTS is not supported, so all redrawing is performed from the source IFF file.

By default, on an 8 plane system, the LITES2 cursor, and all highlighting of found features etc., is drawn using exclusive or mode, in a primary colour complementary to the underlying colour. Colours 0-63 may be used for the map

data (0 is the background). LITES2 allows some control over highlighting colours using its REFRESH BITS command, or logical name LSL$DECW_REFRESH_BITS may be defined as a bit-set integer to alter the default (which is to invert all the bits).

By default, on a 4 plane system, colours 0-7 may be used for the map. Highlighting also uses these colours in such a way that colour 7 is highlighted in colour 0, 6 in 1, 5 in 2 etc. The colour table should therefore be set up in such a way that the colours are distinct.

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

The number of colours used may be modified to some extent by the use of the WORKSTATION TYPE command. WORKSTATION TYPE 1 6000 is relevant only to the 8 plane system, and allows the user to define 128 rather than 64 colours. A side effect of this is that highlighting is done in one of the user-defined colours, rather than in a primary colour defined by LITES2. Colour 1 is highlighted as 126, 2 as 125 etc, unless the highlighting behaviour is altered as discussed above. WORKSTATION TYPE 1 7000 is relevant to both 4 and 8 plane systems. The number of colours used is then allowed to reach the maximum for the particular number of planes (16 and 256 respectively), the defaults being determined from the number that appear to be free, often around 240 on the 8 plane display. If logical name LSL$DECW_MAX_COLOUR is defined to be an integer, then this number of colours will be used instead. This could be used for example to use the full 256 colours of an 8 plane display.

When attempting to use the full number of planes on a display, then there is inevitably contention for the available colours, as the window manager and other applications will have already allocated some cells, for example for window borders and banners. LITES2's default action is to use as many cells as possible in the default colour map. This means that the LITES2 colours corresponding to the cells which were already allocated will be displayed incorrectly. LITES2 attempts to use the colours in such a way that the incorrect colours are the high-numbered ones, and hopefully the effect is not noticeable, or at least isn't serious. If you really must have more, or even all, colours displayed correctly, then logical name LSL$DECW_MAX_COLOUR may be set all the way up to 256.

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

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

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

black and white between background and foreground.

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

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

The interpretation of this information is: Command WORKSTATION TYPE 1 7000 has been used, so an attempt is being made to allocate all 8 planes, or 256 colours; The default colormap is tried first; There are enough free cells here for LITES2 to be able to use colours 0-235 and 240-241, so colours 236-239 and 242-255 will be displayed incorrectly; The 'first allocated colour' and 'plane offset' information gives which X-windows cells are being used - this need not concern the user; The colours are being used in inverted order to ensure that only high numbered ones are displayed incorrectly (again this need not concern the user).

The colours used for the picture are defined using a text file on logical name LSL$DECW_COLOUR (if this is set up). See Appendix A for the format of this file. Colours may be changed subsequently using the WORKSTATION COLOUR command.

The cursor may be small or large, and blinking or steady (ENABLE/DISABLE BIG/BLINK).

This version of LITES2 supports display overlays. The number of bit planes available for use as overlays is controlled by the WORKSTATION TYPE command and LSL$DECW_MAX_COLOUR logical name as described above. Workstations are usually more efficient at displaying images in rows starting at the top of the screen rather than in columns, thus IMAGE CORNER NW and IMAGE DIRECTION CLOCKWISE might give the most efficient display. Experimentation with these commands should show the effect, and if necessary, DTI files may be rotated to the best orientation using the MATRIX package module DTIROTATE. The speed of drawing is also affected by the presence of other windows on top of the LITES2 graphics window, and whether the window lies entirely on the screen - again it is worth experimenting by popping the graphics window to the top (DISPLAY POP may be used) while drawing an image to observe whether this gives a significant speed gain.

Note that raster images might not be drawn correctly if both the primary and secondary windows are active simultaneously, especially if the windows are of different sizes. If this is a problem, then use SUPPRESS to cause drawing only into one window at a time.

The setting of the hardware field in the FRT is used to indicate what style of line caps and joins are required. The CAP style is a number in the range 0-2. 0 means butt caps (square ends), 1 means round caps, and 2 means extended caps (square but extended by half the line thickness. The JOIN style is also a number in the range 0-2. 0 means miter joins (the sides of the thick lines

---

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

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

Several logical names may be used to control the detail of how LITES2 draws into the windows:

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

LITES2MOTIF provides support for hardware text, if the option is enabled (ENABLE HWTEXT) and the appropriate bit is set in the flags entry of the FRT file. By default, X-Windows text is used. X-Windows text is not very versatile - it can only be drawn horizontal and at a limited range of sizes. The intention is that Display PostScript is used, but not all X-Servers support it, so an alternative text rendering system, which also renders Adobe Type 1 fonts, is also supported. When using either of these systems the logical name LSL$DPS_FONTLIST is used to define which PostScript fonts are to be used. It need not be specified if hardware text is not used. The logical name should be defined to point to a file (default filespec LSL$FRT:*.PSFONTLIST) containing directives defining the fonts. The format of this file is documented in the FRT User Guide (MAPPING package). Previous releases of LITES2 allowed the logical name to be defined as a search list, specifying the fonts. This still works, but the new mechanism is preferred - it allows several facilities not previously available. Note that attempts to italicise a font using a negative font number in the FRT does not work for hardware text. If either Display PostScript or Text Rendering is used, then the character shapes and widths in the TRI file are ignored - a TRI file must still be supplied, but it need not contain any fonts.

1. To attempt to use Display PostScript, define the logical name LSL$DECW_DPS as 1. If you do this, but the X-Server does not support Display PostScript, then a message to that effect will be produced.

2. To use Text Rendering, define the logical name LSL$DECW_TEXT_RENDER as 1. This setting will indicate that the new text rendering is to be used, even if LSL$DECW_DPS is set to 1. When making use of Text Rendering the following points should be noted:

   (a) Text Renderng renders the same Adobe Type 1 fonts as used by PostScript and Display PostScript. It also relies on the associated Adobe Font Metric (AFM) files for these fonts. In this

system, these AFM files are of more significance in positioning the individual characters of a text and must be present. Details are provided below of the steps that are taken to ensure that the most appropriate AFM file is associated with a particular LITES2 font. When interpreting the file referred to by the logical name LSL$DPS_FONTLIST, only a subset of the directives are significant when using the text rendering system. These are:

1. FONT - mandatory

   the font filename is built up from the name specified in this directive, the search list LSL$PS_FONT_OUTLINE: and the extension .XDPS$OUTLINE.

   If this file cannot be found, or if the text rendering software has difficulty reading it, then the default font name (see below) is used.

2. SCALE - defaults to 1.0

3. KERNING - defaults to no kerning

4. DIRECTION - defaults to left-to-right

5. AFM - highly recommended, but defaults to the font name.

   NOTE: as the ENCODING directive is ignored, the AFM file is the only way of specifying the encoding vector to be used to relate the character codes in the text strings to the name of the glyphs in the font file.

   The AFM filename is built up from the name specified in this directive, the search list LSL$PS_FONT_METRICS: and the extension .AFM. If the logical name LSL$PS_FONT_METRICS has not been set up then the search list SYS$PS_FONT_METRICS: is used instead. This latter name may disappear with Display PostScript.

   If the file with this filename cannot be found, then the software defaults to using either the font name, or if that is not successful then the default font name (see below). In both of these default scenarios any '-' in the font name or the default font name is replaced by '_'.

(b) To mimic the Display PostScript behaviour, the default font name is "Courier". However, if this font does not exist on the system, or the text rendering system cannot interpret the file, this can be overridden by defining the logical name LSL$DECW_DEFAULT_FONT to be the name of a font which is known to exist and can be read by the system.

(c) The text rendering system supports the use of composite characters. These are characters made up by superimposing two or more existing characters in the font, such as accented letters. This facility is described more fully in the FRT User Guide in the section describing the PostScript Font List File. To use the facility the

logical name LSL$COMPOSITE_CHARACTERS should be defined to be 1.

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

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

3 *Interactive devices*

In addition to the keyboard, this version of LITES2 is capable of interpreting commands from a digitising table on a separate serial line, the workstation mouse or bitpad, a screen menu, a Laser-Scan button box connected on a separate serial line, and (as a separately licensed option) a KERN DSR photogrammetric plotter with optional KRISS image superimposition system. See the "LITES2 - KERN DSR Workstation Guide" for details of using the DSR.

3.1 *Digitising table*

The digitising table is activated by the commands ENABLE TABLE and ENABLE MONITOR (both are required). The table puck may be programmed using the PUCK command on device 3. The digitising table input is interpreted either using the Table Monitor system, or by reading the table directly. The former allows the table to be set in stream mode, giving smooth cursor tracking.

To use the Table Monitor, a table monitor process must be started, using program STARTMON. If the 'named monitor' option is used, then logical name LSL$MONITOR_TABLE must point to the serial line. In addition, if the table is anything other than a standard ALTEK, then logical table LSL$TABMON_ROUTINE (or LSL$TABMON_ROUTINE_<terminal> for named monitor) must point to a suitable decoding shareable image. This logical name must be available to the table monitor process, and so should be in the group or system logical name table. If stream mode is used, to allow smooth tracking using the lowest numbered button, then the lowest acceptable stream rate above 4 points per second should be used. If set too high, then the table monitor will use large quantities of system resources, if too low, then buttons other than the 'tracking button' will repeat

if held down.

If logical name LSL$MONITOR_TABLE is set up, but LITES2 determines that no table monitor process exists, the table will be accessed directly. This does not allow stream mode or smooth tracking.

3.2 *Mouse or bitpad*

The workstation is fitted with either a mouse or bitpad to control the windowing system. The commands ENABLE BITPAD or ENABLE BALL activate this device for use by LITES2. LITES2 does not distinguish between a mouse and a bitpad - either command will enable either device. The only difference is that ENABLE BITPAD will allow a menu to be set up on the bitpad.

LITES2 will respond to mouse or bitpad button presses only when the mouse cursor is within the graphics window. The LITES2 cursor will follow the mouse cursor whenever the lowest numbered button (left mouse button) is pressed or held down within the LITES window. The remaining buttons may be programmed as a puck on device 2 (ENABLE BITPAD) or device 4 (ENABLE BALL). If both BITPAD and BALL are enabled, then the bitpad puck takes precedence.

A menu may be set up on the workstation bitpad (if a mouse is fitted, there is little point to setting up a menu unless the menu is attached to the screen). When prompted to set up the menu, any bitpad puck button may be used to digitise the corners. If the tracking button is used, care must be taken not to move while pressing it, or several points will be entered. As usual, the setup may be retained/aborted using the last two buttons on any defined puck (this includes the screen menu). Note that the cursor must be within the graphics window in order to access the bitpad menu. The terminal emulation window and screen menu should be positioned so as not to prevent access to parts of the menu, and care must also be taken if the graphics window is moved.

If a menu is used on the bitpad, then attention should be drawn to the PRIORITY PUCK command. The lowest numbered button is always used for cursor tracking, so for the 4 button puck, 2 other buttons may be given priority so that their puck function is obeyed even if they are pressed over the menu. At least one button must not be given priority, otherwise it will be impossible to access the menu.

Whilst it is possible to set up a tracking area on the bitpad, this is not recommended, since the LITES2 cursor would not move to the position of the mouse cursor. In any case, the lowest numbered button tracks anyway.

3.3 *Screen menu*

A facility for a single screen menu is provided within LITES2. The lowest numbered button (left button on the mouse) must be used on the screen menu.

Before a screen menu can be displayed on the screen, it must be defined as a PUCK on device 1 (the screen). This also defines the total number of boxes in the menu. The boxes of the screen menu can be programmed using MACRO commands as usual.

The size, layout, screen position and title of the screen menu are defined by the DESCRIBE SCREENMENU command. The titling of the boxes is achieved by the DESCRIBE MACRO command.

The menu is displayed on the screen by the ENABLE SCREENMENU command. If this command is given in INITIAL state, then a screen menu window will be created just after the graphics window appears. When a map has been read in, the existence of the screen menu can be controlled by the ENABLE, DISABLE and TOGGLE SCREENMENU commands.

Like any other window, the menu can be picked up and moved.

3.4 *UILMENUS*

A utility program UILMENUS is supplied which can generate a hierarchical tree of menus for sending commands to LITES2. The utility is described in full in the UILMENUS User Guide. This menu system is most useful when LITES2 is being used as a graphical enquiry tool as part of a captive command procedure, and the menu system is used as the controlling interaction stream.

UILMENUS runs as a separate program from LITES2 but communicates via a mailbox using the auxiliary input to LITES2 referred to by the logical name LSL$LITES2AUX.

UILMENUS may be started from within LITES2, e.g. by using the commands:

```
CREATE MAILBOX 1        ! create LSL$LITES2AUX mailbox
SPAWN/NOWAIT UILMENUS uid-file-name
```

As UILMENUS runs as a separate process, it can outlive LITES2 unless care is taken to ensure that the only route out of LITES2 also stops UILMENUS.

3.5 *Button box*

A Laser-Scan button box (Mapstation console) can be used as an input device. This is connected to a separate serial line, which must have the logical name LSL$CONSOLE. Before the buttons can be used, they must be defined as a PUCK on device 5. This defines the number of buttons that are to be used. The individual buttons can be programmed using MACRO commands as usual.

The buttons are activated by the ENABLE BUTTONS command. If this is given in INITIAL state, and a suitable PUCK command has also been given, then the buttons that are available for use on the button box will light up when the map is read in. When a map has been read in, the availability of the button box can be controlled by the ENABLE, DISABLE and TOGGLE BUTTONS commands.

APPENDIX A

Colour Table

The following is an example of a file describing the colours to be used. It should be set up on logical name LSL$DECW_COLOUR. The character ';' introduces a comment. The colours are specified as proportions of red, green, and blue, in hexadecimal in the range 0-FF. An example file is in LSL$LITES2CMD:VAX.COL

```
; system constants file for VAXstation GPX
; COLOUR DEFINITIONS
;       Red     Green   Blue    Blink   Comment
16                                      ; Number of colours
         0       0       0       0      ; 0
         0       FF      FF      0
        FF       0       0       0
        FF       0       FF      0
         0      FF       0       0
         0      80      FF       0
        FF      97       0       0
        FF      86      FF       0
         0      AF       0       0
         0      BE      FF       0
        FF      C0       0       0
        FF      B3      FF       0
         0      FF       0       0
         0      FF      FF       0
        FF      FF       0       0
        FF      FF      FF       0
```

APPENDIX B

Screen menu

The following is an example of a file that sets up a screen menu. It should be called LSL$LITES2CMD:SCREEN.LCM

```
! SCREEN.LCM
!
! Definition of screen menu
! ========================
!
! first define puck
!
%PUCK 1 32 SCREEN
!
! and describe what SCREEN is to look like
!
%DESCRIBE SCREEN 1 32 6 25.0 200.0 1.0 0.0 screen LSL
!
! now define the contents of each button (box) of SCREEN
!
%MACRO SCREEN2  #%START          #%ENDM
%MACRO SCREEN3  #%START#%END     #%ENDM
%MACRO SCREEN4  #%CURVE          #%ENDM
%MACRO SCREEN5  #%FIND           #%ENDM
%MACRO SCREEN6  #%END            #%ENDM
%MACRO SCREEN7  #%CLOSE          #%ENDM
%MACRO SCREEN8  #%CLOSE SQUARE   #%ENDM
%MACRO SCREEN9  #%MOVE           #%ENDM
%MACRO SCREEN10 #%ROTATE         #%ENDM
%MACRO SCREEN11 #%GET 3          #%ENDM
%MACRO SCREEN12 #%GET 4          #%ENDM
%MACRO SCREEN13 #%INVISIBLE      #%ENDM
%MACRO SCREEN14 #%REPEAT         #%ENDM
%MACRO SCREEN15 #%WINDOW MAP#%PING      #%ENDM
%MACRO SCREEN16 #%ABANDON        #%ENDM
%MACRO SCREEN17 #%SELECT ALL     #%ENDM
%MACRO SCREEN18 #%ZOOM           #%ENDM
%MACRO SCREEN19 #%ZOOM .5        #%ENDM
%MACRO SCREEN20 #%DRAW SCREEN    #%ENDM
%MACRO SCREEN21 #%DRAW MAP       #%ENDM
%MACRO SCREEN22 #%DELETE         #%ENDM
%MACRO SCREEN23 #%EDIT           #%ENDM
%MACRO SCREEN24 #%REMOVE         #%ENDM
%MACRO SCREEN25 #%BRIDGE         #%ENDM
```

```
%MACRO SCREEN26 #%SPLIT         #%ENDM
%MACRO SCREEN27 #%USER 1        #%ENDM
%MACRO SCREEN28 #%USER 2        #%ENDM
!%MACRO SCREEN29        #               #%ENDM
!%MACRO SCREEN30        #               #%ENDM
!%MACRO SCREEN31        #               #%ENDM
%MACRO SCREEN32 #%ABANDON       #%ENDM
!
! now what is to be written in each box
!
%DESCRIBE MACRO SCREEN2 Start
%DESCRIBE MACRO SCREEN3 Symbol
%DESCRIBE MACRO SCREEN4 Curve
%DESCRIBE MACRO SCREEN5 Find
%DESCRIBE MACRO SCREEN6 End
%DESCRIBE MACRO SCREEN7 Close
%DESCRIBE MACRO SCREEN8 Close Squ
%DESCRIBE MACRO SCREEN9 Move
%DESCRIBE MACRO SCREEN10 Rotate
%DESCRIBE MACRO SCREEN11 Get 3
%DESCRIBE MACRO SCREEN12 Get 4
%DESCRIBE MACRO SCREEN13 Invisible
%DESCRIBE MACRO SCREEN14 Repeat
%DESCRIBE MACRO SCREEN15 Window
%DESCRIBE MACRO SCREEN16 Abandon
%DESCRIBE MACRO SCREEN17 Sel all
%DESCRIBE MACRO SCREEN18 Zoom
%DESCRIBE MACRO SCREEN19 Zoom .5
%DESCRIBE MACRO SCREEN20 Draw screen
%DESCRIBE MACRO SCREEN21 Draw map
%DESCRIBE MACRO SCREEN22 Delete
%DESCRIBE MACRO SCREEN23 Edit
%DESCRIBE MACRO SCREEN24 Remove
%DESCRIBE MACRO SCREEN25 Bridge
%DESCRIBE MACRO SCREEN26 Split
%DESCRIBE MACRO SCREEN27 User 1
%DESCRIBE MACRO SCREEN28 User 2
!%DESCRIBE MACRO SCREEN29 MacroA
!%DESCRIBE MACRO SCREEN30 MacroB
!%DESCRIBE MACRO SCREEN32 MacroC
%DESCRIBE MACRO SCREEN32 Abandon
```