Laser-Scan Ltd.

UILMENUS User Guide

Issue 1.7

Copyright (C) 2019 Laser-Scan Ltd
Science Park, Milton Road, Cambridge, England CB4 4FY tel: +44 (0) 1223 420414

Document "UILMENUS User Guide"		Category "USER"
Document Issue 1.1	Clarke Brunt	29-Oct-1990
Document Issue 1.2	Clarke Brunt	15-Feb-1991
Document Issue 1.3	Andy Brown	26-Feb-1991
Document Issue 1.4	Clarke Brunt	22-Jan-1992
Document Issue 1.5	Clarke Brunt	12-Feb-1992
Document Issue 1.6	Clarke Brunt	6-Apr-1992
Document Issue 1.7	Clarke Brunt	9-Jul-1992

CONTENTS

MODULE UILMENUS

FUNCTION

UILMENUS is a suite of two programs which together enable the user to produce a menu driven graphics application interface for a workstation running the DECwindows windowing system. The programs provide mechanisms for defining 'widgets' such as labelled buttons, icons, toggle switches, slider bars etc, and for associating actions with each widget. DCL symbols should be set up to run the programs, i.e.

```
      $ UILGEN == "$LSL$EXE:UILGENMOTIF" and
      $ UILMENUS == "$LSL$EXE:UILMENUSMOTIF"
```

The UILGEN program takes as input .UIM files created by the user, and produces as output a .UIL file. This is compiled by the VAX UIL compiler to produce a .UID file which is then used by the UILMENUS program to produce the menus on the screen. Note that the command UIL/MOTIF may be required unless it has been arranged that the MOTIF (rather than XUI) qualifier for UIL is the default. The generated .UIL files include some header files from the LSL$UIL directory. These files are supplied in LSL$PUBLIC_ROOT:[LITES2.UIL], so LSL$UIL will normally be defined as a search list, passing through LSL$SITE_ROOT:[LSL.UIL], LSL$PUBLIC_ROOT:[LITES2.UIL] and possibly other directories.

Used in conjunction with LAMPS utilities that accept mailbox input, such as LITES2, UILMENUS allows input from a screen-based set of menus in addition to the normal use of the keyboard.

UILMENUS has a mode of operation which can accept as input existing .DAT and .CDL files intended for the UISMENUS utility.

UILMENUS is restricted to use on VAX graphics workstations that are running DECwindows. Anyone who wishes to design a new menu interface using UILMENUS should first read the introductory sections of the DEC documentation on the DECwindows system, to familiarise themselves with the concepts of X windows and DECwindows.

FORMAT

```
$ UILGEN uim-file-name uil-file-name
```

Command qualifiers	**Default**
/CDL	/NOCDL
/CONVERT	/NOCONVERT
/DEFINE	/NODEFINE

PROMPTS

```
_UIM input file:  uim-input-file-name

_CDL Data file:  dat-file-name (if /CDL or /CONVERT used)

_UIL output file:  uil-file-name

_UIM output file:  uim-output-file-name (if /CONVERT used)
```

PARAMETERS

uim-input-file-name

- specifies an ASCII file that contains UILGEN commands. The name is parsed against LSL$UIL:---.UIM. This file may include other files using the FILE command.

dat-file-name

- specifies an ASCII file that contains UILGEN commands. This parameter replaces the uim-input-file-name parameter if the /CDL or /CONVERT qualifier is used. The file should be in the format expected for a .DAT file by the UISMENUS utility. The name is parsed against LSL$CDL:---.DAT. This file may include other files by using the FILE command.

uil-file-name

- specifies the name of the output UIL file. The name is parsed against LSL$UIL:---.UIL.

uim-output-file-name

- specifies the name of the output UIM file The name is parsed against LSL$UIL:---.UIM. This parameter replaces the uil-file-name parameter if the /CONVERT qualifier is used.

COMMAND QUALIFIERS

/CDL
/NOCDL **(default)**

- this qualifier is used to specify that the input files use the syntax of the UISMENUS program, in particular, coordinates are measured from the bottom left. It is intended to provide a degree of compatibility with menu files written for the UISMENUS utility. When the /CDL qualifier is used, different header files are included in the generated .UIL file, which attempt to adjust the appearance of the menus to match the UISMENUS style. It is normally recommended that the /CONVERT qualifier is used instead of /CDL (see below).

/CONVERT
/NOCONVERT **(default)**

- this qualifier is also used to specify that the input files use the syntax of the UISMENUS program, but in this case the input files are translated into a .UIM file (the native input file for UILGEN). Once the UILMENUS files have been translated in this way, the resulting UIM file may be further developed and used with UILGEN without the /CDL or /CONVERT qualifiers. This method of converting UISMENUS files may not initially give results which appear identical to when UISMENUS was used, but is preferred to using the /CDL qualifier, which does not allow any development of the menus except by altering the old .CDL and .DAT files.

/DEFINE=(symbol1,symbol2...)
/NODEFINE **(default)**

- this qualifier is used to define a list of symbols which may be used to conditionally process parts of the input file. The intention is that a single input file may be used to produce different menus depending on the values used in the /DEFINE qualifier.

FORMAT

```
$ UILMENUS uid-file-name[,uid-file-name...]
```

Command qualifiers	**Default**
/ABORT	/NOABORT
/CLASSNAME	/CLASSNAME=LSLUILMENUS
/COMMAND	/NOCOMMAND
/INPUT	/NOINPUT
/LOGICAL	/LOGICAL=LSL$UILMENUSTEXT
/OUTPUT	/OUTPUT=LSL$LITES2AUX:
/SYMBOL	/SYMBOL=LSL$UILMENUSTEXT

PROMPTS

```
_UID file:  uid-file-name
```

PARAMETERS

uid-file-name

- specifies a list of files (up to 16) containing the output of the UIL compiler. These are produced by running UIL on the .UIL file produced by UILGEN. The names are parsed against LSL$UIL:---.UID.

COMMAND QUALIFIERS

/ABORT[=abort_mailbox]

- this qualifier is used to specify the name of the file to which UILMENUS is to send commands using the 'ABORT' action. The default name, if /ABORT is given, is LSL$LITES2ABORT:. The name is most likely to be the logical name for a mailbox, though TT: (for example) may be used while testing to type the commands at the terminal.

/CLASSNAME=string

- this qualifier is used to specify the class name for this Motif application. The default is LSLUILMENUS. The class name is used in forming the name of the application specific resource file (name.DAT in DECW$USER_DEFAULTS or DECW$SYSTEM_DEFAULTS) and also as the first component of the resources specified in these or other resource files. A resource file may be used to customise the general appearance of the menu interface. See below for a description of resource files.

/COMMAND=string

- this qualifier is used to specify a string of commands which UILMENUS will obey initially. It may, for instance, be used to display a particular set of boxes initially.

/INPUT=input_mailbox

- this qualifier is used to specify the name of a device from which UILMENUS is to read commands. It is most likely to be the logical name for a mailbox, though TT: (for example) may be used while testing to in order to give commands at the terminal.

/LOGICAL=logical_name
/LOGICAL=LSL$UILMENUSTEXT **(default)**

- this qualifier is used to specify the name of the job logical name to be set using the 'DEFINE' action.

/OUTPUT=output_file
/OUTPUT=LSL$LITES2AUX: **(default)**

- this qualifier is used to specify the name of the file to which UILMENUS is to send commands using the 'SEND' action. It is most likely to be the logical name for a mailbox, though TT: (for example) may be used while testing to type the commands at the terminal.

/SYMBOL=symbol_name
/SYMBOL=LSL$UILMENUSTEXT **(default)**

- this qualifier is used to specify the name of the DCL symbol to be set using the 'SET' action.

DESCRIPTION

General

UILMENUS is a LAMPS utility that creates a tree of boxes on the screen. These boxes may contain menus, buttons, informational text, and several other widgets available in the DECwindows toolkit. When a button is probed one or more of several actions are performed.

- o another box in the tree is made visible
- o a command is sent to another process
- o a DCL symbol is set
- o the utility is terminated

Actions

The action which takes place when a button is pressed, or a widget activated, depends on the command string given by the user in a DO command. These same commands may also be sent to UILMENUS by writing

them to the /INPUT device (most likely a mailbox being written by another process). The command string consists of a series of commands separated by ';'. For widgets returning a value, the value will be substituted for any '?' characters e.g. the command typed into a command window, or the value from a scale widget. If '?' appears in the DO string for a widget that does not return a value, or in a command from the /INPUT device, then it will be ignored. The commands available are listed above. To get ';' or '?' into a command, duplicate the character. Note that commands which take a box name, or widget name, will not work until the box has been fetched from the UID file. Root boxes are fetched automatically, others may be fetched by the use of ADD, DISPLAY, FETCH, POSITION, or REMOVE commands.
The following commands are available:

- ABORT string: the string is sent to the /ABORT file, which will usually be a mailbox being read by another process.

- ADD name: The name given may be either the name of a box, or an individual named widget. The item is added to the display. Items may be added to boxes even when the box is not currently displayed.

- BOTH: following commands are obeyed unconditionally - cancels previous ON or OFF.

- COLOUR RGB r g b box: the original and new colours in the COLOUR_MIX box are set to the given red, green, and blue components (in the range 0.0 to 1.0).

- DEFINE string: the job logical name is defined to the specified string - this can be used as a secondary mechanism for communicating to an associated process which has established a mailbox link to UILMENUS.

- DISPLAY box: the given box is displayed. All those boxes above it in the tree and including the specified box will be displayed, all other boxes are removed.

- EXIT: UILMENUS terminates

- FETCH box: the given box is fetched from the UID file but not displayed.

- FILE SEARCH ["file_spec"] box: the file list in the given FILE_SELECT box is regenerated. If a file_spec is given, then the file filter is changed to this.

- LABEL LABEL "string" name: sets the label in the given named LABEL widget (created by ADD TEXT). This action may also be used to set the label in a push button (BUTTON TEXT), toggle button (TOGGLE TEXT), pulldown menu (PULLDOWN_MENU), or option menu (OPTION_MENU).

- LIST ADD "item_name" "do_string" name: a new item is added to the named LIST widget.

- LIST CLEAR name: all items are removed from the named LIST widget.

- LIST DESELECT ["item_name"] name: the given item (all items if not given) are deselected in the named LIST widget.

- LIST MOVE "item_name" name1 name2: the given item is removed from list name1, and added (at the end) to list name2. Its DO string, and current selected state, are preserved. The source and destination list may be the same, in which case the item is just moved to the end.

- LIST REMOVE "item_name" name: the given item is removed from the named LIST widget.

- LIST SELECT ["item_name"] name: the given item (all items if not given) are selected in the named LIST widget. This is not useful in a SINGLE type of LIST (only the last item will remain selected).

- TEXT VALUE ["string"] name: the current string in the given TEXT widget is changed. If the string is omitted, an empty string is used.

- OFF: following commands are only obeyed if the widget is turned off.

- ON: used as part of a DO string for a toggle button, a button within a multiple list, a BOX PROMPT, a BOX FILE_SELECT, or a BOX COLOUR_MIX - the following commands are only obeyed if the button is turned on, or in the case of the boxes, the OK (rather than CANCEL) button was pressed. If neither ON nor OFF appears, the commands are obeyed regardless.

- POSITION [x y] box: the given box is moved to the given position. If the position is omitted, the box is moved to the current position of the screen pointer. The box need not be visible at the time it is moved.

- PROMPT LABEL "string" box: sets the prompt label in the given prompt box.

- REMOVE name: The name given may be either the name of a box, or an individual named widget. The item is removed from the display. Items may be removed from boxes even when the box is not currently displayed. If a composite widget, such as a menu, is removed, then all the buttons contained in it will disappear also.

- RESET box: all the buttons in the given box are reset to their initial states. This is 'off' for all TOGGLEs, but 'on' for the first TOGGLE in a RADIO_BOX. An OPTION_MENU displays the name of its first button. The command strings for the buttons are not obeyed.

- SCALE VALUE value name: set the value of the named SCALE widget.

- o SEND string: the string is sent to the /OUTPUT file, which will usually be a mailbox being read by another process.
- o SET string: the DCL symbol is set to the specified string - this is normally followed by EXIT so that the symbol can be utilised.
- o TOGGLE ON name: turns on a named TOGGLE widget. If, instead, the name is the name of a box, then all the buttons defined as TOGGLEs in the given box are set to 'on'. This does not change the state of buttons within a RADIO_BOX. The command strings for the buttons are not obeyed.
- o TOGGLE OFF name: turns off a named TOGGLE widget. If, instead, the name is the name of a box, then all the buttons defined as TOGGLEs in the given box are set to 'off'. This does not change the state of buttons within a RADIO_BOX. The command strings for the buttons are not obeyed.

Hierarchy

Each box has a name, and a parent box. The hierarchy is used by the DISPLAY command to determine which boxes should be made visible. For example, a tree might be of the form:

```
                    box_1
                      |
         +----------+------------+
         |                       |
       box_2                   box_3
         |
 +-------+-------+
 |               |
box_4          box_5
```

This example has just one root box. It is possible to have several roots (each has the parent "none"). When UILMENUS starts up, just the root boxes will be displayed. The boxes may be defined in any order, but all boxes must ultimately be connected to a root box.

The first root box defined in any UID file is treated specially and will always have window manager borders (regardless of the use of BOX NOBORDER), and will include an iconise button which iconises all windows of the UILMENUS application. If several .UID files are to be used as input to UILMENUS, then only root boxes defined in the first UID file will be initially displayed - any root boxes defined in the other .UID files may be displayed by the use of DISPLAY or ADD commands. The first root box in the subsequent .UID files cannot be displayed at all - a good way to avoid a problem here is to include the same initial root box in all the .UID files - in this way the box will appear regardless of which of the .UID files are passed to UILMENUS.

Resource files

The general appearance of a Motif application may be tailored by specifications contained in a resource file. The description here is brief but there is DEC documentation and other books on the subject. The DEC Session Manager loads resources from several files (called DECW$*.DAT - these would be called .Xdefaults on some other systems) when it starts up, and these provide defaults for all applications. When UILMENUS starts up, it looks for a file called LSLUILMENUS.DAT in the directory DECW$SYSTEM_DEFAULTS or DECW$USER_DEFAULTS. The name LSLUILMENUS is the default application class for UILMENUS, and may be changed using the /CLASSNAME qualifier. The lines within this file consist of specifications such as

```
LSLUILMENUS*background:       red
```

which would specify that the background colour of all widgets not otherwise specified in the UIL file be red. The lists of resources which may be set may be found in the Motif documentation. In addition to widget resources, UILMENUS allows the user to override colour and font specifications produced by DEFINE FONT or DEFINE COLOUR in the UIM file. For example, if you had defined and used a colour named my_back, then the specification

```
LSLUILMENUS*my_back:          green
```

would override the definition in the UIM file and set it to green.

UIM FILES

The input to UILGEN is in the form of UIM (User Interface Menu) files. The entire input may be contained within one file, but if required, the initial file may contain FILE commands, to include others. An individual box is defined by BOX commands, followed by the other UIM commands, and terminated by an END command, or the end of the file. Each box must be contained entirely within one file. For all commands expecting a position, or a size, these are measured in pixels. For this purpose, the screen is assumed to be 1024 by 864 pixels. The position of boxes is given relative to a corner of the screen, while the position of other object are relative to a corner of their containing box. By default positions are measured from the top left of the screen or box, and give the position of the top left of the box or other item. If the /CDL qualifier is used, positions are measured from the bottom left, and give the position of the bottom left of the object. When displayed on screens with different numbers of pixels, the menus will scale proportionally to the number of screen pixels.

Conditional commands

These may appear anywhere within the file. They control conditional compilation of sections of the input file, and are modelled on a subset of the commands used by the preprocessor for the 'C' language. The commands all begin with the character '#', and may not be abbreviated. The symbols used in these commands are defined on the UILGEN command line using the /DEFINE qualifier, and are always converted to upper case. The conditional commands may be nested (up to a level of 8). Any UIM file must contain a matched set of conditional commands, i.e. an included file may not have #ELSE or #ENDIF commands which match #IFDEF or #IFNDEF commands in the outer file.

#IFDEF symbol

- the UIM code following (until an #ELSE or #ENDIF) is only processed if the given symbol is one of those defined on the command line.

#IFNDEF symbol

- the UIM code following (until an #ELSE or #ENDIF) is only processed if the given symbol was **not** defined on the command line.

#ELSE

- this command matches the last preceding #IFDEF or #IFNDEF. The UIM code following (until an #ENDIF) is only processed if the code before was not being processed (and vice-versa).

#ENDIF

- this command matches the last preceding #IFDEF or #IFNDEF. The processing of UIM code following reverts to that in force before the matching #IFDEF or #IFNDEF.

General commands

These may appear anywhere within the file.

FILE filename

- the given UIM file is included in the input. This command may only occur in the initial UIM file, and must not occur in the middle of the definition of a box. The filename is parsed against LSL$UIL:---.UIM, or LSL$CDL:---.CDL if the /CDL qualifier is used.

UIL "string"

- the string is written to the .UIL file, and should therefore be valid UIL. This could be used to include a user-written UIL file, or to define a value for future use.

DEFINE FONT name "string"

- allows a user-defined font. The name given may then be used in FONT commands. The string must be the name of a valid DECwindows font. This command actually defines the value font_'name' to be the given font. A font defined in this way may be redefined without rebuilding the menus by including the resource

  ```
  LSLUILMENUS*name:   string
  ```

  in a resource file. LSLUILMENUS is the default application class - it may be altered by using the /CLASSNAME qualifier on UILMENUS.

DEFINE COLOUR NAME name "string"

- allows a user-defined colour. The name given may then be used in COLOUR commands. The string must be the name of a valid DECwindows colour. This command actually defines the value color_'name' to be the given colour. The colour names available may be found by examining the file SYS$MANAGER:DECW$RGB.COM - just the parts of the name after DECW$RGB_. (COLOR is a synonym for COLOUR.) A colour defined in this way may be redefined without rebuilding the menus by including the resource

  ```
  LSLUILMENUS*name:   string
  ```

  in a resource file. LSLUILMENUS is the default application class - it may be altered by using the /CLASSNAME qualifier on UILMENUS.

COLOUR BACKGROUND [name]
COLOUR FOREGROUND [name]
COLOUR HIGHLIGHT [name]

- specifies the colours to be used for future objects. The name should have been defined in a DEFINE COLOUR command. The actual colour name used is color_'name'. COLOR is a synonym for COLOUR, and HILITE for HIGHLIGHT. If the name is omitted, the default colour is used. The HIGHLIGHT colour is used when a button is activated.

FONT [name]

- specify the font to be used to display subsequent informational and button texts. The name should have been defined by a DEFINE FONT command. If name is omitted, the default font will be used. The actual name used is font_'name'. Fonts 0 to 11 are predefined in a header file - these are monospaced fonts and are included for compatibility with CDL and UISMENUS. The default is FONT 0 if the /CDL qualifier is used, otherwise the default font.

Box definition commands

These commands must come at the start of the definition of a box. The box is defined once all four of SIZE, POSITION, NAME, and PARENT have been given.

The default type of 'box' is (in DECwindows terms) a dialog box, which may contain such sub-menus, buttons, etc as as required. UILGEN presently supports three other types of 'box' - the file selection box, the prompt box, and the colour mixing box - if one of these is required, the BOX FILE_SELECT, BOX PROMPT, or BOX COLOUR_MIX, and any DO command pertaining to it must be given before the four compulsory commands are completed.

The BOX NAME and BOX PARENT commands are used to define the structure of the tree. The box names so defined are used to establish connections between boxes and some applications use them to indicate which box is to be made visible.

BOX COLOUR_MIX

- the next box is to be a colour mixing box. It will display an original colour and a new colour, together with various means of choosing the new colour, and 5 buttons - OK, APPLY, RESET, CANCEL, and HELP. RESET sets the new colour to be the original colour again. HELP gives information on how to use the box. Any action specified after ON in a DO command will be obeyed when the OK or APPLY buttons are pressed. Any '?' characters in the DO string will be replaced by a string containing 3 numbers, the red, green, and blue components of the new colour in the range 0.0 to 1.0. Any commands after OFF will be obeyed when the CANCEL button is pressed. The box is automatically removed when OK or CANCEL are pressed, but remains displayed if APPLY is used. In the absence of ON or OFF, the commands will be obeyed when whichever button is pressed. The colour mixing box may contain other buttons, labels, etc. as required.

BOX DIALOG

- the next box is to be a dialog box. This is the default.

BOX FILE_SELECT "file_spec"

- the next box is to be a file selection box. It will display all files fitting the file-spec (which may include wildcards). Any action specified after ON in a DO command will be obeyed when a file is selected (by double-clicking on a filename, or clicking on a file name and then using the OK button). Alternatively, a filename may be typed in. Any '?' characters in the DO string will be replaced by the filename. Any commands after OFF will be obeyed when the CANCEL button is pressed. In the absence of ON or OFF, the commands will be obeyed when either button is pressed. The file-selection box may contain other buttons, labels, etc. as required.

BOX PROMPT "prompt_text"

- the next box is to be a prompt box. It allows the user to enter a text string. Any action specified after ON in a DO command will be obeyed when the text is terminated by carriage return, with '?' characters being replaced by the text. The box includes an OK button (which functions the same as carriage return), and a CANCEL button which obeys any commands after OFF. In the absence of ON or OFF, the commands will be obeyed when either button is pressed. The box is automatically removed from the screen when either button is operated. The prompt box may contain one other user defined button if required.

BOX MODAL
BOX MODELESS

- specifies whether the next box is to be modal, or modeless (default). If a modal box is displayed, then only this box will accept button presses and keyboard input until the box is removed.

BOX NOBORDER

- indicate that the next box is not to have a frame around it. This means that the box cannot be moved, or pushed behind other windows. The first box to be defined will always have a frame regardless.

BOX POSITION xpos ypos

- specify the position of the box.

BOX SIZE xsize ysize

- specify the size of the box window.

BOX NAME "text"

- give a name to the box that is being defined. This name will appear as the title of the box, and is used in some of the DO commands.

BOX PARENT "text"

- give the name to the box that is this box's parent in the tree. Note that if this is a root box, then the parent name "none" should be given instead. Note that the name "cdl$none" is interpreted as "none" for compatibility with .CDL files.

Box contents commands

Commands defining the contents of a box - with certain limitations, these may be given in any order. Positions are relative to the containing dialog box. Any items included in a MENU, RADIO_BOX, PULLDOWN_MENU, or OPTION_MENU will be positioned automatically, and any position arguments should normally be omitted.

ADD LINE [xstart ystart xend yend]

- add a line to the box. The line is drawn from a start position to an end position. If the line is part of a MENU, RADIO_BOX, PULLDOWN_MENU, or OPTION_MENU, then the positions are not required - the line will be positioned automatically. The line is actually a 'separator' widget, which restricts it to being either horizontal or vertical.

ADD TEXT [xoff yoff] "text"

- add the specified informational text, in the current font. The text is displayed using a 'label' widget.

ADD ICON [xoff yoff] [name]

- draw an icon at the given position. If name is given, then icon_'name' is used as the name of the icon. This should have been defined elsewhere, possibly using the UIL command. For compatibility with CDL files and UISMENUS, if the name is omitted, the name taken from an ICON FILE command is used.

BUTTON TEXT [xoff yoff] "text"

- create a button, labelled with the specified text.

BUTTON ICON [xoff yoff] name

- create a button represented by an icon. If name is given, then icon_'name' is used as the name of the icon. This should have been defined elsewhere, possibly using the UIL command. For compatibility with CDL files and UISMENUS, if the name is omitted, the name taken from an ICON FILE command is used.

COMMAND_WINDOW xoff yoff width lines "prompt"

- creates a command input window. The width in pixels, number of command lines displayed, and a prompt are specified. The DO commands for this widget may use the '?' character in place of the command input.

DO "string"

- gives a list of commands to be obeyed when buttons or other widgets are activated. The string consists of a series of commands separated by ';'. For widgets returning a value, the value will be substituted for any '?' characters. The commands available are listed above. To get ';', '?', or '"' into a command, duplicate the character.
  e.g. DO "SEND %tol find ?" for a scale widget
  or   DO "ON;SEND %select fc 10;OFF;SEND %deselect fc 10" for a toggle button
  These same strings of commands may be sent to UILMENUS from an external source by writing them to the device specified in the /INPUT qualifier.

END

- terminates the definition of a MENU, RADIO_BOX, PULLDOWN_MENU, OPTION_MENU, LIST, or the box itself. Any text after the END command is ignored, so it is permitted to put e.g. END MENU for clarity.

LIST type xoff yoff [width] visible_items

- creates a list widget within the box. Only BUTTON TEXT widgets may occur within the list. They will be placed in order in the list (their positions should not be given). If the width argument is given, the list will be a fixed width, with a horizontal scroll bar being added if the items are longer than the width. If omitted, the list will vary in size to accomodate long items. The type of list may be either SINGLE or MULTIPLE. In a SINGLE list, only one item may be selected at a time - if another is selected, the first selected item is deselected. In a MULTIPLE list, any number of items may be selected, and a selected item may be deselected by clicking on it again, so it is useful to include ON and OFF commands in the DO string for the buttons. The list need not contain any items initially, if the intention is to add them at run time using the UILMENUS action LIST ADD. The definition of the list is terminated by an END command, or the end of the file.

MENU orientation xoff yoff
RADIO_BOX orientation xoff yoff

- creates a menu within the box. The menu is a container for buttons. Following buttons will be placed in order within the menu (their positions should not be given). A RADIO_BOX is intended to contain a series of TOGGLE buttons, only one of which is on at any given time - when a button is activated, any other button is turned off, and both obey their command strings. VERTICAL or HORIZONTAL may be given for orientation. The definition of the menu is terminated by an END command, or the end of the file.

MENU BAR xoff yoff

- creates a menu bar within the box. A menu bar is a container for pulldown menus, which are always arranged horizontally within it.

OFFSET xoff yoff

- specify an offset which is applied to all subsequent positions in this file.

OPTION_MENU [xoff yoff] "text"

- creates an option menu labelled with the given text. This must occur either as a separate item in the box (when a position must be given), or be in a horizontal or vertical MENU. Subsequent buttons become part of the option menu. The option menu will initially display the name of its first button. The definition of the OPTION_MENU is terminated by an END command, or the end of the file.

PULLDOWN_MENU "text"

- creates a pulldown menu labelled with the given text. This must occur within a MENU BAR, OPTION_MENU, or another PULLDOWN_MENU. Subsequent buttons become part of the pulldown menu. The definition of the PULLDOWN_MENU is terminated by an END command, or the end of the file.

SCALE orientation xoff yoff width decimal_points min max "title"

- creates a scale widget. The width in pixels (height for a vertical scale), number of decimal places in the value, minimum and maximum values (real numbers if required), and a title are specified. The DO commands for this widget may use the '?' character in place of the value. VERTICAL or HORIZONTAL may be given for orientation.

TEXT xoff yoff width ["initial_string"]

- creates a text input widget. The width in pixels, and an optional initial string are specified. The widget is triggered by pressing the return key. The DO commands for this widget may use the '?' character in place of the text input.

TOGGLE TEXT [xoff yoff] "text"

- create a toggle button. The button contains a marker to indicate whether it is currently on or off. The default is off except for the first button in a RADIO_BOX.

TOGGLE ICON [xoff yoff] name

- create a toggle button represented by an icon. If name is given, then icon_'name' is used as the name of the icon. This should have been defined elsewhere, possibly using the

UIL command. For compatibility with CDL files and UISMENUS, if the name is omitted, the name taken from an ICON FILE command is used.The button contains a marker to indicate whether it is currently on or off. The default is off except for the first button in a RADIO_BOX.

WIDGET NAME "string"
- gives a name to the next widget to be defined. The name can then be used in some of the DO commands to identify the widget.

UISMENUS and CDL compatibility commands

The following commands have been superceded by other UILGEN commands but are available for compatibility with .CDL files and the UISMENUS utility.

DEFINE COLOUR RGB name r g b
- allows a user-defined colour given in terms of its red, green, and blue components (in the range 0.0 to 1.0). The name given may then be used in COLOUR commands. This command actually defines the value color_'name' to be the given colour. (COLOR is a synonym for COLOUR.) This command will only work in the first of a list of .UID files given to UILMENUS. The DEFINE COLOUR NAME command is preferred.

FOREGROUND r g b
BACKGROUND r g b
HILITE r g b
- These combine the function of a DEFINE COLOUR RGB command and a COLOUR command, which should be used instead. They use colour names color_'n', where n is a number incremented for each command.

WINDOW POSITION xpos ypos
- synonym for BOX POSITION.

WINDOW SIZE xsize ysize
- synonym for BOX SIZE.

WINDOW NOFRAME
- synonym for BOX NOBORDER

NAME CONSOLE "text"
- synonym for BOX NAME.

NAME PARENT "text"
- synonym for BOX PARENT.

ICON FILE "filename"
- specify the name of the file defining an icon. The filename is parsed against LSL$CDL:---.ICON. This file is converted to LSL$UIL:name.ICON_UIL which is included in the output .UIL file. Subsequent ADD ICON and BUTTON ICON commands without an icon name will use this icon. This command may be used to convert a UISMENUS .ICON file to UIL. Users are expected to design new icons using UIL from scratch.

ON BORDER
OFF BORDER

- these have no effect in UILGEN.

CHOICE TEXT [xoff yoff] "text"

- create a toggle button, labelled with the specified text. The button forms one of a group of at least two where probing a button will turn it on, and turn all other buttons in the group off. The buttons turned off will not obey their DO commands. One button in a group of choices must always be on and when the box is first invoked this will be the first button defined in the group. A series of CHOICE commands must be indicated by a prior GROUP command. Similar functionality is now provided by the RADIO_BOX and TOGGLE TEXT commands.

CHOICE ICON [xoff yoff] name

- create a toggle button represented by an icon, otherwise as for CHOICE TEXT. Similar functionality is now provided by the RADIO_BOX and TOGGLE ICON commands.

GROUP

- indicate that the subsequent CHOICE buttons are to be considered as a group. Additional groups of choices are indicated by subsequent GROUP commands. A GROUP command must prefix at least one CHOICE commands. Similar functionality is now provided by the RADIO_BOX command.

RETURN CODE number

- specify the 'return code' for subsequent buttons. The DO command should now be used to specify actions instead.

RETURN TEXT "text"

- specify the 'return text' for subsequent buttons. The DO command should now be used to specify actions instead.

ICON FILES

Users requiring to use icons (bitmap pictures) in the ADD ICON, BUTTON ICON, and TOGGLE ICON commands should define the icon named in these commands using the UIL language. The definition may be put in a separate file, which may be included in the UIL file generated by UILGEN using the UILGEN command 'UIL "include file 'filename';"'. Alternatively, the entire definition may be included in the UIL file using a series of UIL commands. Note that in simple use, single and double quotes are equivalent in UIL (see UIL manual for details), but double quote must be duplicated if it is to be included in a UILGEN command string which is already in double quotes.

An example of the UIL commands required to define a simple icon follows:

```
value
   icon_square : icon (
      "          ",
      " ******** ",
            " *      * ",
            " *      * ",
            " *      * ",
            " *      * ",
            " *      * ",
            " *      * ",
            " ******** ",
            "          "
         );
```

The name of the icon (as used in the ADD ICON, BUTTON ICON, or TOGGLE ICON commands) is 'square'. The actual name must be prefixed by 'icon_'. Each character in the quoted strings represents a pixel on the screen. By default " " (space) is background colour, and "*" is foreground colour. Note that this icon is only 10 pixels square, which is very small.

In order to use other colours, some other UIL commands are required. The colours used must be defined using the UIL color function - those defined defined using the UILGEN command DEFINE COLOUR may not be used. The following example defines a coloured icon:

```
value
   rg : color_table( background color = " ",
                      color("red")     = "r",
                      color("green")   = "g");

   icon_color_square : icon ( color_table = rg,
      "          ",
      " rrrrrrrr ",
      " rggggggr ",
      " rggggggr ",
      " rggggggr ",
      " rggggggr ",
      " rggggggr ",
      " rggggggr ",
      " rrrrrrrr ",
      "          "
   );
```

"rg" is your chosen name for a color_table. This defines which characters are to represent which colours. "background color" and "foreground color" may always be used.

To convert a UISMENUS icon file (.ICON) to the new format (by default .ICON_UIL), run UILGEN just giving an ICON FILE command in its input (.UIM or .DAT) file.

EXAMPLES

The following is a simple example of a UIM file for input to UILGEN.

```
! LSL$UIL:SIMPLE_EXAMPLE.UIM
! Simple example of a .UIM file for UILMENUS documentation
! Author Clarke Brunt, Laser-Scan       19-Dec-1989

BOX DIALOG                              ! create a new box
BOX PARENT "none"                       ! at the root level
BOX NAME "Example"                      ! with an obvious name
BOX POSITION 200 200                    ! positioned here
BOX SIZE 300 200                        ! of this size

   MENU VERTICAL 50 50                  ! put a menu box in it

      DO "SEND %find"                   ! next button does this
      BUTTON TEXT "Find"                ! and it is labelled "Find"

      DO "SEND %start"                  ! next button does this
      BUTTON TEXT "Start"               ! etc etc

      DO "SEND %end"
      button text "End"

      DO "SEND %abandon"
      BUTTON TEXT "Abandon"

   END MENU                             ! end of the menu box

   DO "SEND %quit;EXIT"                 ! next button does this
   BUTTON TEXT 200 50 "Quit"            ! button positioned here

   DO "SEND %exit"                      ! a final command
   BUTTON TEXT 200 100 "Exit"           ! for a final button

END BOX                                 ! and end the whole thing

! That is the end of SIMPLE_EXAMPLE.UIM
```

XXXXXX Replace this page by Figure 1 (Simple Example) XXXXXX

EXAMPLE 2

The following is a more complex example of a UIM file for input to UILGEN.

```
! LSL$UIL:COMPLEX_EXAMPLE.UIM
! More complex example of a .UIM file for UILMENUS documentation
! Author Clarke Brunt, Laser-Scan       19-Dec-1989

! define a colour with name 'my_gold' to be the DECwindows colour 'goldenrod'
DEFINE COLOUR NAME my_gold "goldenrod"

! define a colour with name 'my_red' in terms of rgb
DEFINE COLOUR RGB my_red 1.0 0.0 0.0

! define a font with name 'times' to be a supplied DECwindows font
DEFINE FONT times "*-Times-Bold-R-Normal--24-*"

! include a UIL file defining the icon with name 'LSL_LOGO'
UIL "include file 'lsl$uil:lsl_logo.icon_uil';"

! define a root dialog box to be called "Example".
! everything until the matching END BOX will go into it
BOX DIALOG                              ! create a new box
BOX PARENT "none"                       ! at the root level
BOX NAME "Example"                      ! with an obvious name
BOX POSITION 100 100                    ! position here
BOX SIZE 300 200                        ! of this size

! put an LSL_LOGO icon in the top left corner
ADD ICON 5 5 LSL_LOGO

! put a title near the top using the font defined above
FONT times
ADD TEXT 70 5 "UILmenus example"

! put a line below the title
ADD LINE 0 60 300 60

! start a horizontal menu in the lower part of the box
! everything until the matching END MENU will go into it
MENU HORIZONTAL 40 100

   ! set background to our defined colour my_gold, and
   ! set highlight to our defined colour my_red
   COLOUR BACKGROUND my_gold
   COLOUR HIGHLIGHT my_red

   ! return to the default font
   FONT

   ! put some buttons in the menu

   ! this button displays and resets the menu called "Select"
   DO "SEND %select all %over num 1;DISPLAY Select;RESET Select;TOGGLE ON
Select"
```

```
   BUTTON TEXT "Select"

   ! this button displays the option menu
   DO "DISPLAY Options"
   BUTTON TEXT "Options"

   ! this button displays the command menu
   DO "DISPLAY Commands"
   BUTTON TEXT "Commands"

! that's all for this horizontal menu
END MENU

! place a button at lower right to exit from UILMENUS
! first set the colours back to default
COLOUR BACKGROUND
COLOUR HIGHLIGHT
DO "EXIT"
BUTTON TEXT 230 160 "Quit"

! that's all for dialog box "Example"
END BOX

! ******************************************************************************

! define a new box called Select
BOX DIALOG                              ! a new box
BOX PARENT "Example"                    ! this is a child of "Example"
BOX NAME "Select"                       ! called "Select"
BOX POSITION 200 350
BOX SIZE 400 300

! put a title near the top using the font defined above
FONT times                              ! choose a named font
ADD TEXT 10 5 "Selections"
FONT                                    ! reset to default font

! put a vertical menu on left hand side
MENU VERTICAL 20 90
   DO "SEND %select all;TOGGLE ON Select;"
   BUTTON TEXT "All"
   ADD LINE
   DO "ON;SEND %select fc roads;OFF;SEND %dese fc roads"
   TOGGLE TEXT "Roads"
   DO "ON;SEND %select fc rail;OFF;SEND %dese fc rail"
   TOGGLE TEXT "Railways"
   DO "ON;SEND %select fc rivers;OFF;SEND %dese fc rivers"
   TOGGLE TEXT "Rivers"
end menu

! put an option menu at top right
OPTION_MENU 200 50 "Current overlay"
   DO "SEND %over num 1"
   BUTTON TEXT "1"
   DO "SEND %over num 2"
   BUTTON TEXT "2"
   DO "SEND %over num 3"
   BUTTON TEXT "3"
   DO "SEND %over num 4"
   BUTTON TEXT "4"
END OPTION_MENU

! with a menu below it
MENU VERTICAL 200 90
   DO "SEND %over rev"
   BUTTON TEXT "Reveal"
   DO "SEND %over con"
   BUTTON TEXT "Conceal"
   DO "SEND %over pop"
   BUTTON TEXT "Pop"
   DO "SEND %over push"
   BUTTON TEXT "Push"
END MENU

! and a one button menu bar
MENU BAR 200 210
   PULLDOWN_MENU "Attributes..."
      DO "SEND %over att opa"
      BUTTON TEXT "Opaque"
      DO "SEND %over att add"
      BUTTON TEXT "Add"
      DO "SEND %over att merge"
      BUTTON TEXT "Merge"
      DO "SEND %over att sub"
      BUTTON TEXT "Sub"
      DO "SEND %over att inv"
      BUTTON TEXT "Inverse"
   END PULLDOWN_MENU
END MENU BAR

! place a button at lower right to remove select box
DO "REMOVE Select"
BUTTON TEXT 320 260 "Return"

! that's all for dialog box "Select"
END BOX

! ****************************************************************************

! define a new box called Options
BOX DIALOG                              ! a new box
BOX PARENT "Example"                    ! this is a child of "Example"
BOX NAME "Options"                      ! called "Options"
BOX POSITION 200 350
BOX SIZE 400 300

! put a title near the top using the font defined above
FONT times
ADD TEXT 10 5 "Options"
FONT

! Use radio boxes for some on/off things
ADD TEXT 20 50 "Cursor"
RADIO_BOX HORIZONTAL 20 80
```

```
   DO "ON;SEND %disable big;"
   TOGGLE TEXT "Small"
   DO "ON;SEND %enable big;"
   TOGGLE TEXT "Big"
END RADIO_BOX
RADIO_BOX HORIZONTAL 20 120
   DO "ON;SEND %disable blink;OFF;SEND %enable blink"
   TOGGLE TEXT "Steady"
   DO " "
   TOGGLE TEXT "Blinking"
END RADIO_BOX

! And a set of TOGGLEs for ordinary options
MENU VERTICAL 200 80
   DO "ON;SEND %enable status;OFF;SEND %disable status;"
   TOGGLE TEXT "Status"
   DO "ON;SEND %enable table;OFF;SEND %disable table;"
   TOGGLE TEXT "Table"
END MENU

! place a button at lower right to remove option box
DO "remove Options"
BUTTON TEXT 320 260 "Return"

! that's all for dialog box "Options"
END BOX

! ****************************************************************************

! define a new box called "Commands"
BOX DIALOG
BOX PARENT "Example"
BOX NAME "Commands"
BOX POSITION 200 350
BOX SIZE 400 200

! Add a command window at lower left
! ? is replaced by any command entered
DO "SEND ?"
COMMAND_WINDOW 20 20 150 6 "LITES2> "

! Add a button to bring up a file_select box
DO "display Files"
BUTTON TEXT 300 20 "Files"

! Add a scale widget. The ? character is replaced by the value from the scale
DO "SEND %tol find ?"
SCALE HORIZONTAL 200 50 150 2 0.0 10.0 "Find tolerance"

! Include a UIL file defining the icon with name 'lsl_exit'
UIL "include file 'lsl$uil:lsl_exit.icon_uil';"

! place a icon button at lower right to remove commands box
DO "display Example"
BUTTON icon 350 160 lsl_exit

! that's all for dialog box "Commands"
```

```
END BOX

! ***************************************************************************

! Define the file box. The character ? is replaced by the selected file name
DO "ON;SEND %iff ?;OFF;REMOVE Files"    ! this is the action to do
BOX FILE_SELECT "LSL$IF:*.IFF"          ! file select using this specification
BOX PARENT "Commands"                   ! and it is a child of "Commands"
BOX NAME "Files"
BOX POSITION 400 400
BOX SIZE 400 350
END BOX

! ***************************************************************************

! That is the end of COMPLEX_EXAMPLE.UIM
```

XXXXXX Replace this page by Figure 2 (Complex Example) XXXXXX

LITES2 EXAMPLE

The following gives some hints on using UILMENUS with LITES2. LITES2 commands are preceded by '*' in the example.

LITES2 is usually started first. Any mailboxes to be used are created using LITES2 commands, and then UILMENUS is started as a subprocess. e.g.

```
$ LITES2 ...  ! start LITES2
!
! create LSL$LITES2AUX for LITES2 command input
* CREATE MAILBOX 1
!
! if required, create a mailbox to send commands to UILMENUS
* CREATE MAILBOX LSL$UILMENUSINPUT
!
! create an abort mailbox (default name LSL$LITES2ABORT)
* CREATE ABORT_MAILBOX
!
! run UILMENUS (output and abort default to our mailboxes)
* SPAWN/NOWAIT UILMENUS/INPUT=LSL$UILMENUSINPUT/ABORT MY_UID
!
! if you want to send commands from LITES2 to UILMENUS then...
! open UILMENUS input mailbox as a LITES2 file
* FILE APPEND 1 LSL$UILMENUSINPUT:
!
! send any required command to UILMENUS e.g. add a box
* FILE WRITE ADD MY_BOX
```

MESSAGES (INFORMATIONAL)

These messages give information only, and require no immediate action by the user. They are used to provide information on the current state of the program, or to supply explanatory information in support of a warning or error message.

CVTICON, converting ICON file 'filename' to UIL file 'filename'

> **Explanation:** UILGEN is processing the specified ICON file.
>
> **User action:** none.

IGNLIN, line not vertical/horizontal - ignored

> **Explanation:** UILGEN only supports vertical/horizontal lines in the ADD LINE command.
>
> **User action:** Remove the line, or make it vertical or horizontal.

READFILE, reading 'file type' file 'filename'

> **Explanation:** UILGEN is processing the input file.
>
> **User action:** none.

WRITFILE, producing 'file type' file 'filename'

> **Explanation:** UILGEN is producing the output file.
>
> **User action:** none.

MESSAGES (ERROR)

These messages indicate an error in processing which will cause the program to terminate. The most likely causes are a corrupt or otherwise invalid input file, or an error related to command line processing and file manipulation.

BADARGS, error in arguments in UIM file

> **Explanation:** the arguments to a command in the UIM file are missing or incorrect.
>
> **User action:** edit the UIM file to ensure that it only contains legal commands.

BADCOND, mismatched ELSE or ENDIF, or missing ENDIF

> **Explanation:** A #ELSE or #ENDIF has been found with no matching #IFDEF or #IFNDEF, or the end of file has been reached without #ENDIF commands to match all #IFDEF and #IFNDEF commands being found.
>
> **User action:** Correct the structure of the #IFDEF, #IFNDEF, #ELSE, and #ENDIF commands. Remember that #ELSE and #ENDIF can only match #IFDEF and #IFNDEF commands in the same file.

BADFILE, bad filename in FILE or ICON FILE command

> **Explanation:** the filename given after a FILE or ICON FILE command is invalid.
>
> **User action:** give a correct filename after the file command.

BADINPUT, error with input on line 'integer' of file 'filename'

> **Explanation:** there was some unexpected form of input on the specified line of the file.
>
> **User action:** examine the input file and correct the offending line.

BADROOT, root box must not be a FILE_SELECT box

> **Explanation:** The first box to be defined must be DIALOG type (the default).
>
> **User action:** Define the FILE_SELECT box later in the hierarchy.

BOXINBOX, BOX command found while box definition in progress

> **Explanation:** The BOX commands begin the definition of a box, and cannot be given again until the definition ends (with an END command, or end of file). useraction
>
> **User action:**

COMNOTMEN, a command_window must not be within a menu

Explanation: The command_window widget cannot be part of a menu. It is a separate item in the box.

User action: End the menu before specifying the command_window.

IFNESTEX, attempt to next more than 'integer' IFDEF or IFNDEF

Explanation: The conditional compilation #IFDEF and #IFNDEF statements may only be nested up to the specified limit.

User action: Reduce the level of nesting of #IFDEF and #IFNDEF statements.

ININCL, already in included file - FILE command not allowed

Explanation: a FILE command was found in a file included by another FILE command.

User action: included files may not themselves use the FILE command. Re-arrange the input files so that the FILE command is not required.

LISTNOTMEN, a list widget must not be within a menu

Explanation: The list widget cannot be part of a menu. It is a separate item in the box.

User action: End the menu before specifying the list widget.

MENNOTMEN, a menu or radio_box must not be within a menu

Explanation: A menu or radio_box must occur as a separate item in the box.

User action: Do not attempt to nest menus or radio boxes.

MISSEND, misplaced END command

Explanation: an END command was found in an unexpected place.

User action: ensure that END commands match up correctly.

OPNFIL, error opening file 'filename'

Explanation: there was an error when attempting to open the specified file.

User action: check that the file exists, or for an output file, that its directory exists.

OPTNOTMEN, an option menu must not be in a pulldown menu, option menu, or menu bar

Explanation: The option menu must occur either as a separate item in the box, or in an ordinary menu. It must not occur in a menu bar, pulldown menu, or option menu.

User action: Specify the option menu either as a separate item in the box, or in an ordinary menu.

PULLMEN, a pulldown menu must be in a menu bar, pulldown menu, or option menu

Explanation: A pulldown menu can only occur in a menu bar, another pulldown menu, or an option menu. It cannot occur in an ordinary menu.

User action: Define a menu bar to contain the pulldown menu entry, or place it within another pulldown menu, or an option menu.

SCANOTMEN, a scale widget must not be within a menu

Explanation: The scale widget cannot be part of a menu. It is a separate item in the box.

User action: End the menu before specifying the scale widget.

TEXTNOTMEN, a text widget must not be within a menu

Explanation: The text widget cannot be part of a menu. It is a separate item in the box.

User action: End the menu before specifying the text widget.

MESSAGES (FATAL)

These messages indicate a severe error in processing, or some form of system failure, which has caused the program to terminate.

ABORT, previous errors invalidate run - UILGEN aborting

Explanation: UILGEN has failed as indicated by a previous error. There is no point in continuing so UILGEN will terminate.

User action: Fix the problem(s) that gave rise to the earlier error messages.

MESSAGES (OTHER)

In addition to the above messages which are generated by the program itself, other messages may be produced by the command line interpreter (CLI) and by Laser-Scan libraries. In particular, messages may be generated by the IFF library and by the Laser-Scan I/O library, LSLLIB. IFF library messages are introduced by '%IFF' and are documented in the IFF library users' guide. In most cases IFF errors will be due to a corrupt input file, and this should be the first area of investigation. If the cause of the error cannot be traced by the user, and Laser-Scan are consulted, then the output file should be preserved to facilitate diagnosis. LSLLIB messages are introduced by '%LSLLIB' and are generally self-explanatory. They are used to explain the details of program generated errors.

INDEX