dL4 Version 4.3 GUI Training

INTRODUCTION:

The dL4 graphical user interface (GUI) provides standard GUI elements such as buttons and edit boxes to otherwise character based applications. The application creates GUI elements by printing mnemonics. Information is transferred to the application as strings of character input allowing the programs to accept this graphical input as if the user typed in the character strings. This document describes the GUI features available for dL4 version 4.3.

All of the graphical features are provided by the dL4 graphic window driver which is standard in dL4 for Windows and, when used with dL4Term, in dL4 for Unix. The features cannot be used if the character window driver is selected.

The graphical elements of dL4 are: Buttons, Boxes, Lists, drop downs and menus.

---

dL4 Graphical User Interface Programming Specifications:

Graphical elements in the form of buttons, edit boxes, list boxes, and mouse/pointer input are a standard feature of modern user interfaces. Such graphical interface programming is supported in dL4 4.3 by the following set of mnemonics (WC indicates window control):

'WCNUMBER'	Create and define numeric input box
'WCSTRING'	Create and define character input box
'WCPRIVATE'	Create and define character hidden input box
'WCBUTTON'	Create and define push button
'WCDEFAULTBTN'	Create and define default push button
'WCCHECK'	Create and define check box
'WCRADIO'	Create and define radio button (used with 'WCGROUP')
'WCLABEL'	Create and define a label for an input box
'WCTEXT'	Create and define multi-line character display box
'WCMEMO'	Create and define multi-line character input box
'WCLIST'	Create and define selection list box
'WCEDITLIST'	Create and define editable selection list box
'WCLISTDROP'	Create and define drop down selection list
'WCEDITDROP'	Create and define drop down editable selection list box
'WCMENU'	Create and define menu
'WCSUBMENU'	Create and define sub-menu
'WCMENUACTION'	Create and define menu action item
'WCMENUCHECK'	Create and define menu check box item
'WCMENURADIO'	Create and define menu radio button item
'WCMENUSEP'	Create menu separator
'WCENDMENU'	End menu or sub-menu definition
'WCDELETE'	Delete a graphical element
'WCQUERY'	Cause graphical element to send current value
'WCACTION'	Change the action performed by an input element

'WCGROUP'	Group graphical elements (especially radio buttons)
'WCSELECT'	Select current graphical element
'WCENABLE'	Enable user input/selection to/of graphical element
'WCDISABLE'	Disable user input/selection to/of graphical element
'WCMARK'	Select a list item in a list box
'WCUNMARK'	De-select a list item in a list box
'WCFOCUS'	Select current input element
'WCSETFONT'	Set font for use by all new graphical elements
'WCWHERE'	Sends current element's action string as input

Coordinates x,y are always column,row.

In addition to graphical input features, dL4 4.3 also offers new screen output features. In both dL4 for Windows and dL4 for Unix, dL4 and Dynamic windows are displayed as true windows with the standard Microsoft Windows appearance rather than being drawn with character graphics.

Besides adding the new GUI mnemonics, the set of output mnemonics supported in windows is enlarged in dL4 4.3 to now include all of the following. Details on these mnemonics can be found in the Windows Class driver document :

'FONTFACE'	Set current font name
'FONTSIZE'	Set current font size
'FONTCELL'	Sets the font size by specifying the line height
'FONTCOLOR'	Set current font color
'BACKCOLOR'	Set background color
'RESETCOLOR'	
'GRIDFONT'	Define window coordinates by font size
'GRIDENGLISH'	Define window coordinates by 1000ths of an inch
'GRIDMETRIC'	Define window coordinates by 100ths of a millimeter
'PENWEIGHT'	Set pen width for line drawing
'PENCOLOR'	Set pen color
'PENSTYLE'	Select solid, dashed, or dotted lines
'LINETO'	Draw a line from cursor to position x,y
'RECTTO'	Draw a rectangle from cursor to position x,y
'RECT'	Draw a rectangle from position a,b to position c,d
'ELLIPSE'	Draw an ellipse
'BACTFN'	Begin activate-on-function-key input
'EACTFN'	End activate-on-function-key input
'PGMFN'	Program function key text
'BEGIN'	Place cursor at start of current text box
'ONCLOSE'	Controls exiting dL4 for Windows or dL4Term

Colors & fonts

See Window Class document, Special Output Characters Controlling Text Drawing for a complete list of mnemonics and definition.

By default a window background is white and text is black or whatever combination the user has chosen in the dL4/dL4Term “Preferences”. You can change background color using the 'nBACKCOLOR' mnemonic, where n designates the RGB color parameter. The parameter is a 24-bit integer RGB value in which the most significant 8-bits specify the red component, the middle 8-bits specify the green component, and the least significant 8-bits specify the blue component.
An example of creating the RGB parameter :

If the color is light orange the RGB value would be 255 226 213.
Take 255 x (65536) + 226 x (256) + 213 = 16769749

You can change the font text color using the 'nFONTCOLOR' mnemonic, where n designates the RGB color parameter.

There are two special color codes in addition to RGB color values. Code -1 selects the standard system text color for dialogs. Code -2 selects the standard system background color for dialogs. For example, printing '-1FONTCOLOR' would select the system dialog text color as the current font color. These special color codes can be used to match the dialog color scheme selected by the user. (This is referring to the Windows Color Scheme selected by the PC user in the Windows, Display Properties, Appearance settings.)

To set the background color of the entire window you should open the window as a hidden style, set the background color and then show the window.

If you are using GUI buttons, check boxes, drop downs, etc., it is recommended that you use the '-1FONTCOLOR' statement, since GUI buttons, check boxes, drop downs, etc. inherit the Windows Color Scheme.

A mnemonic, 'INVERT' is available to invert colors within a specified region. The mnemonic has the following forms:

'INVERT' - Invert colors from the current position to the end of the current line.
'n INVERT' - Invert colors from the current position for 'n' characters.
'w,h INVERT' - Invert colors from the current position within a rectangle of 'w' characters width and 'h' characters height.
'x1,y1,x2,y2 INVERT' - Invert colors within rectangle defined by the points 'x1,y1' and 'x2,y2' of the current coordinate grid.

This mnemonic is currently defined in the dL4Term terminal definition file, but it may be possible to define this operation for other terminals by modifying their terminal definition files.

Drawing

See Window Class document, Special Output Characters for Graphic Drawing for a complete list of mnemonics and definition.

BOX and **LINE** statements can also be used. Refer to Language Reference Guide.

BOX statement

Synopsis

Draw a rectangular figure on display device.

Syntax

BOX {*chan.no*;} {*@x1,y1*;} [*TO @x2,y2*;] | [*SIZE w,h*]

Parameters

chan.no identifies a valid channel number.

x1,y1 are the column, row coordinates of the upper left corner.

x2,y2 are the lower right column, row coordinates.

w,h identify the width and height.

Remarks

Box drawing is a function of the window and printer drivers, and uses the **#,#RECTTO** and **#,#RECT** mnemonics.

If @x1,y1 is not specified, the current cursor position is used as the upper left corner.

Examples

```
Box @7,2; To @70,10;
Box @7,2; Size 70,19
Box To @70,10;
```

LINE statement

Synopsis

Draw a line on a display device.

Syntax

LINE {*chan.no*;} {*@x1,y1*;} **TO** *@x2,y2*; { **TO** *@x2,y2*; } ...

Parameters

chan.no identifies a valid channel number.

x1,y1 are the column, row coordinates of the start of a line.

x2,y2 are the ending column, row coordinates of a line.

Remarks

Line drawing is a function of the window and printer drivers. If running on a character terminal, your terminal description file must contain a definition for the mnemonic **#,#LINETO**.

If ***@****x1*,*y1* is not specified, the current cursor position is assumed.

TO is a keyword which must be followed by the ending coordinate position of the line segment.

Examples

```
Line @3,3; To @30,3;
Line @3,3; To @3,9; TO @30,9;
Line To @30,1;
```

Window Coordinate Grids

The ‘GRIDFONT’ mnemonic is particularly important because it allows a program to use coordinates that are fractions of a character. For example, after printing the mnemonic string ‘4GRIDFONT’, all coordinates will be defined in fourths of a character width or height. Without this feature, a program would have no choice, but to make an edit box 1 character high (too short) or 2 characters high (too tall).

You have a choice of defining the Window Coordinate Grid using the 'GRIDFONT', 'GRIDENGLISH' or 'GRIDMETRIC' mnemonic.
'GRIDFONT' is based on line height and character width of current font divided by the parameter.
'GRIDENGLISH' is based on thousandths of an inch times the parameter.
'GRIDMETRIC' is based on hundreds of a millimeter times the parameter.

Mnemonics 'n,d GRIDENGLISH', 'n,d GRIDMETRIC', and 'n,d GRIDFONT" have been created to provide a higher precision method of setting the coordinate grid. In these two parameter mnemonics, "n" and "d" are the numerator and denominator of a fraction which is passed to mnemonic as a parameter. Thus the mnemonic string '1000,72 GRIDENGLISH' is equivalent to 'x GRIDENGLISH' where "x" is equal to 1000/72. Since 'GRIDENGLISH' sets the coordinate grid in 1000ths of an inch, this sets the grid to 1/72 inches (points).

Function Key Activation

Because some of the graphical input elements send function key characters (‘F1’, ‘F2’, . . .) as input, an important mnemonic is ‘BACTFN’ (begin active function keys). This mnemonic enables activate-on-function-key input mode. When this mode is enabled, any function key character received as input will cause the input to terminate, just as if the ENTER or carriage return key had been pressed. The KEY option (“INPUT KEY K$;I$”) can be used within the INPUT statement to retrieve the input termination character. The ‘EACTFN’ mnemonic is used to disable active-on-function-key mode. Note that the ‘BACTFN’ mnemonic is overridden by the ‘IOBI’ (enable binary input mode) and ‘IOBC’ (enable activate-on-control-character mode) mnemonics (until an 'IOEI' or 'IOEC' is encountered).

A function key is any keyboard sequence such as 'F4' or 'PAGE DOWN' that generates a mnemonic character as input data. Function keys such as 'INSERT' or "right arrow" that are used as edit control keys will not terminate input in this mode. Example:

```
Print 'BACTFN';
Input Key TermChar$;"Prompt: "Response$
Print 'EACTFN';
If TermChar$ = 'F9' Goto DisplayHelp
If TermChar$ = 'MU' Goto EditHistory
```

A mnemonic, 'PGMFN' is available to control the text sent by the 'Fn' function keys. The mnemonic sequence

```
PChr$(n,"text");'PGMFN'
```

will program function key 'n' ('Fn') to send the character string "text" as input when pressed. This mnemonic is currently defined in the dL4Term terminal definition file only, but it can be implemented for other terminal types by modifying their terminal definition files. The 'RF' or 'XX' mnemonics can be used to reset all function keys to their original values of 'Fn'.

Note for dL4Term users: the 'PGMFN' mnemonic supports function keys 'F0' through 'F63', however, dL4Term's default configuration only defines keys for 'F1' through 'F12'.
If, for example, you want to use shift-F1 as 'F13' in dL4Term, use the dL4Term "Preferences->Keyboard" menu item to define the additional keyboard translation. The "Keyboard" dialog and methods of copying keyboard configurations are described in the dL4Term Reference Guide.

Element numbers

Graphical Input Elements and Windows:

All graphical elements belong to a window and **each window has an independent set of graphical element numbers**. The element numbers do not have to be assigned in any particular order or sequence.

The element number is essentially an identifier for referencing each particular element. Most GUI mnemonics are preceded by an assigned element number to designate the graphical element to be affected. **Element numbers are used in determining the tab sequence between elements that have the "tab stop" option**. The element number is also significant when using field labels (WCLABEL) with select keys in that when the selection key is activated by the user, the next element number after the field label's element number is selected. The element number is also significant in that elements can be placed into groups, particularly radio buttons, and element number ranges of different groups cannot overlap.

The display area of the window is the background for the graphical elements. Text printed on a window at the same location as a graphical element will be behind the element and hidden until the graphical element is deleted. Graphical elements are not effected by any non-GUI mnemonic printed to the window with the exception of ‘CS’ (clear screen) and ‘XX’ (reset all). For example, printing ‘CU’ (clear unprotected) to a window will clear all unprotected text on the window, but will not have any effect on list boxes in the window. Printing a ‘CS’ or ‘XX’ mnemonic to a window will delete all graphical elements. Graphical elements can be deleted by the ‘WCDELETE’ mnemonic or by closing the window that contains the elements. Printing an ‘XX’ mnemonic to a window will clear all text, delete all graphical elements, and reset the current font used by graphical elements to the system default font.

Input to any window can be read from any window. For example, if windows are open on channel 1 and channel 2, pressing a button in the window open on channel 1 will create input that can be read on either channel 1 or channel 2. This change makes it easier for programs to use more than one input window at a time. Note that when using multiple windows, all of the GUI elements should have unique element numbers even if the elements are in different windows. (Most applications would not use interrelated independent windows, but rather modal windows with child windows.)

Keyboard Navigation

Graphical input elements can always be selected by “clicking” on the element with a mouse or other pointer device. It is also possible to navigate between elements using the keyboard. If a graphical element is currently selected, the TAB and BACKTAB keys can be used to move sequentially between elements that have the “tab stop” option in element number order.

Because many navigation keys are also data characters, keyboard navigation can not be used if the main window has the input focus. The ‘WCFOCUS’ mnemonic can be used to initially select a graphical element and thus enable navigation via the keyboard. Typically the 'WCFOCUS' mnemonic is used in the GUI application after the GUI window is displayed.

ONCLOSE

When using dL4Term, the user can end their telnet session by selecting the DISCONNECT menu action, the exit button, or the "close" action of the system menu. The user can similarly terminate a dL4 for Windows session. Exiting in this manner prevents the application from cleaning up any partially written transactions. A new mnemonic, 'ONCLOSE', has been defined to allow applications to control such exits.
The 'ONCLOSE' mnemonic has the following forms:

```
PChr$(0,"text");'ONCLOSE'  -  Display "text" within a message
                              box and give the user a choice
                              of exiting dL4Term or continuing
                        the application.
PChr$(1,"text");'ONCLOSE'  -  Display "text" within a message
                              box and prevent the user from
                              exiting dL4Term.
PChr$(2,"text");'ONCLOSE'  -  Prevent the user from exiting
                               dL4Term and treat "text" as input
                        for the program.
PChr$(0,"");'ONCLOSE'      -  Disable any previously set ONCLOSE
                              action.
```

Programs using the 'ONCLOSE' mnemonic should output a null action ('PChr$(0,"");'ONCLOSE') or an 'XX' mnemonic before exiting to clear the ONCLOSE action.

Auxiliary Printing

When using dL4Term or dL4 for Windows, the 'BA', 'EA', 'BO', 'EO', 'AE', and 'AD' mnemonics are supported to direct output to an "auxiliary printer". When printer output is enabled, output will be sent to the printer selected via the dL4Term/dL4 "Printer" sub-menu of the "Preferences" menu. When sending output to a Windows printer, a document will be considered complete and all pages will be printed when auxiliary output is disabled by an 'EA', 'EO', 'AD', or 'XX' mnemonic. A program should not expect to be able to suspend output with 'EA', 'EO', or 'AD' and then later continue output on the same printer page.

These mnemonics can be used without dL4Term if the mnemonics are defined in the appropriate terminal definition file and if the mnemonics are output to the main window. When using

Dynamic Windows ("WINDOW OPEN"), the "WINDOW OFF" statement or the CALL DUPCHANNEL intrinsic should be used to output the mnemonics to the main window.

Status Line

When using dL4Term or dL4 for Windows, the 'WS', 'ES', 'SO', and 'SF' mnemonics are supported to display a status line the bottom of a window. The 'WS' mnemonic redirects all output to the status line until an 'ES' mnemonic is output.
The 'SO' (status line on) and 'SF' (status line off) mnemonics control whether the status line is visible.
Note: the mnemonics and status line text must be output to the window that will contain the status line. When using Dynamic Windows ("WINDOW OPEN"), the "WINDOW OFF" statement or the CALL DUPCHANNEL intrinsic should be used to output status line mnemonics to the main window rather than the current window.

These mnemonics can be used without dL4Term if the mnemonics are defined in the appropriate terminal definition file.

133 character display
When using dL4Term or dL4 for Windows, the 'WD' and 'NR' mnemonics can be used to change the current font size to either the default font size or 6 tenths of the default font size. Assuming a default window width of 80 characters, the 'NR' mnemonic will set the font size to provide a width of 133 narrow characters. Both mnemonics clear the window with the equivalent of a 'CS' mnemonic.

Miscellaneous GUI Mnemonics

WCSELECT

In order to send mnemonics and data to an edit box, list box, or other graphical element, the element must be the current selected element. An element automatically becomes the current element when it is created. It can be re-selected at any time by printing a 'nWCSELECT' mnemonic to the channel of the window containing the graphical element. The parameter "n" is the element number specified when the graphical element was created. The window containing the elements is element number 0.

WCDELETE

The 'nWCDELETE' mnemonic is used to delete the graphical element number "n" in the window to which the mnemonic is output. A deleted element disappears from the display. Graphical elements are also deleted whenever another graphical element is created with the same element number.

Menu elements can be deleted only by deleting the entire menu to which they belong. Deleting a menu element or overwriting a menu element with another graphical element will cause the entire menu to be deleted.

WCQUERY

The 'nWCQUERY' mnemonic causes graphical element "n" of a window to send its current value as input characters to the window. The format of the input characters depends on the type of the graphical element.

WCACTION

Graphical input elements can perform an action when they are selected (if it is a button) or when their value is modified. By default, the action is to send a function key mnemonic, "Fn" where "n" is the element number, to the window containing the graphical input element when it is selected or modified. The 'WCACTION' mnemonic is used to change the default action:

'n,a,PChr$(text$) WCACTION'

n	element number of the input element
a	action number to be modified:
	0 send text$ when selected
	1 send text$ when value of element is modified
text$	string value used by the action

WCGROUP

The 'WCGROUP' mnemonic is used to group graphical elements, particularly radio buttons. A group is displayed as an enclosing and labeled rectangle. The 'WCGROUP' mnemonic has 6 parameters

'n,x1,y1,x2,y2,PChr$(label$) WCGROUP'

n	element number of the group
x1,y1	Upper left grid coordinates of the group rectangle (in relation to window)
x2,y2	Lower right grid coordinates of the group rectangle
label	title string displayed along the group box

WCENABLE/WCDISABLE

The ‘nWCENABLE’ and ‘nWCDISABLE’ mnemonics are used to enable and disable the graphical element “n” in the window to which the mnemonic is output. A disabled input element can not be selected by the user.

WCMARK/WCUNMARK

The 'nWCMARK' mnemonic is used to set the current values of a list box,
where ‘n’ is the zero based index of the value in the list. In a multiple selection list box, a selected list item can be unselected by sending an ‘nWCUNMARK’ mnemonic (or ‘WCUNMARK’ to set all items to an unselected state).

WCLABEL

The ‘WCLABEL’ mnemonic is used to display a label and define a selection key for a graphical input element. An ampersand in the label defines the next character to be a selection key associated with the next sequential graphical element. The ‘WCLABEL’ mnemonic provides programs with full control of label positioning.

PChr$(n,x1,y1,x2,y2,label$);’WCLABEL’

n	element number of the label
x1,y1	Upper left grid coordinates of the display rectangle
x2,y2	Lower right grid coordinates of the display rectangle
label	title string including selection key marked by an ampersand

WCFOCUS

The ‘n WCFOCUS’ mnemonic selects the graphical input element ‘n’ to receive the keyboard focus. The selected element receives keyboard input and is usually highlighted in some way. Performing input from a window automatically changes the focus to that window unless echo and the cursor are disabled (‘IOEE K0’).

WCSETFONT

The ‘WCSETFONT’ mnemonic selects the current window font and font size for use by all newly created graphical elements except for menu elements. An important use of this mnemonic is to select a font for use in edit boxes that is slightly smaller than the normal font. This can make it possible to fully display user input within an edit box that is only 1 character high (the overhead of borders leaves less than one character height available within the box). Such edit boxes are often used when adding edit boxes to an existing terminal oriented screen layout.

Typically GUI type displays require more display space for box drawings, etc. There are two ways to approach GUI displays. The recommended approach is to use the GRIDFONT or other GRID mnemonics to make input boxes slightly larger than the font. But if you are converting an existing text based display that is very crowded and you want all the information to fit within the old display constraints to avoid redesign, you can take the WCSETFONT approach of reducing the font size.

EXAMPLE:

```
10 Print #1;'10GRIDFONT';        !sets grid to 10ths
20 Print #1;'9FONTSIZE';         !reduce font size to 9/10ths normal
30 Print #1;'WCSETFONT';         !sets font size
40 Print #1;'10FONTSIZE';        !change font size back
50 Print #1;'1GRIDFONT';         !grid back to 1
```

WCWHERE

The mnemonic, 'n WCWHERE' causes the GUI input element that currently has the input focus to return the action 'n' string as input. If the window itself has the input focus, a 'CR' will be returned.
Can be used to provide context-sensitive help or to verify where the user currently is. (Use is not highly recommended because the user can change focus while statements are processing so that WCWHERE may in fact provide information on where the focus was!)

dL4 Boxes: Check, Number, String and Private

EDIT BOX (NUMBER, STRING and PRIVATE):
Edit boxes are graphical elements that contain string or numeric input values. The user can change the value of an edit box by selecting the box and typing characters. Edit boxes are created by outputting a 'WCNUMBER', 'WCSTRING', or 'WCPRIVATE' mnemonic which has up to 8 parameters as shown below:

PChr$(n,x1, y1, x2, y2, "label",options,l);'WCNUMBER'
PChr$(n,x1, y1, x2, y2, "label",options,l);'WCSTRING'
PChr$(n,x1, y1, x2, y2, "label",options,l);'WCPRIVATE'

n	element number of the edit box
x1,y1	Upper left grid coordinates of the rectangle containing the edit box and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the edit box and title
label	title string displayed near the edit box, see description below
options	an optional numeric parameter, see description below
l	optional limit on the number of characters accepted (maximum field length)

The "options" parameter is the sum of following option values:

1	Disable and "gray out" the edit box (user can't select the edit box)
2	Use the edit box as a tab stop (in element number order)
4	Send input to window when value changes
32	Any loss of input focus is reported as an input value change (option 4)

The default value of "options" is zero.

The "label" parameter defines the title string and the optional selection key. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the edit box. The user can select the edit box by typing "ALT" and the character.

The edit box's mnemonics differ in the handling of user input.
The 'WCNUMBER' mnemonic creates an edit box that accepts only numbers as valid input. 'WCNUMBER' produces an edit box that accepts signed floating point numbers. The numbers must be in the format <sign><digits><"."><digits> where the sign and decimal point are optional. Entering an illegal character will be rejected. Note: the edit box may return a return a value such as "-", ".", or "-." if the user starts to enter a legal number but never adds the expected digits.

The 'WCSTRING' mnemonic creates an edit box that accepts any data character as input.
The 'WCPRIVATE' mnemonic creates an edit box that accepts any data character as input, but displays all input characters as asterisks.

The current (or default) value of an edit box is set by printing a value to the box. The value can be cleared by printing a 'CU' mnemonic to the box. The current value of a edit box is sent as CR terminated input to the window containing the edit box when a 'nWCQUERY' mnemonic is output with "n" equal to the element number of the edit box.

If option value 4 is used, the edit box sends a function key character as input to the window containing the edit box whenever the edit box value changes. The function key character is the

mnemonic ‘Fn’ where “n” is the element number of the edit box. The characters sent by the edit box can be changed by using the ‘WCACTION’ mnemonic.

The 'MH' mnemonic is supported in edit boxes to move the current position to the beginning of the text in the box.

The 'BEGIN' mnemonic can be sent to GUI edit boxes (such as 'WCSTRING') to place the cursor at the start of the current text and to select the current text for replacement. The user can tab or otherwise move to a different GUI element to leave the value unchanged or type new characters to completely replace the existing value. To send the 'BEGIN' mnemonic to an edit box, the edit box must be the currently selected GUI element (via 'nWCSELECT'). The following example could be used to initialize the contents of a newly created 'WCSTRING' edit box and select the contents for replacement:

```
Print '4WCSELECT';"Initial Data";'BEGIN';'0WCSELECT';
```

The 'n BEGIN' mnemonic can be used to perform the 'BEGIN' function on GUI element "n" and also set the input focus to that element:

```
Print '4BEGIN';
```

CHECK BOX:
Check boxes are graphical elements that have a true or false value. The user can toggle the value of a check box by selecting the check box. Check boxes are created by outputting a ‘WCCHECK’ mnemonic which has up to 7 parameters as shown below:

```
PChr$(n,x1,y1,x2,y2, “label”, options);’WCCHECK’
```

n	element number of the check box (must be unique in window)
x1,y1	Upper left grid coordinates of the rectangle containing the check box and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the check box and title
label	title string displayed near the check box, see description below
options	an optional numeric parameter, see description below

The “options” parameter is the sum of following option values:

1	Disable and “gray out” the check box (user can’t select the check box)
2	Use the check box as a tab stop (in element number order)
4	Send input to window when value changes
32	Any loss of input focus is reported as an input value change (option 4)

The default value of “options” is zero.

The “label” parameter defines the optional selection key. If an ampersand character (“&”) is included in the label string, the character following it will be used as a selection key for the check box. The user can select the check box by typing “ALT” and the character.

The current value of a check box is set by printing a “1” (selected) or a “0” (not selected) to the check box. The current value of a check box is sent as CR terminated input to the window

containing the check box when a ‘nWCQUERY’ mnemonic is output with “n” equal to the element number of the check box.

If option value 4 is used, the check box sends a function key character as input to the window containing the check box whenever the check box value changes. The function key character is the mnemonic ‘Fn’ where “n” is the element number of the check box. The characters sent by the check box can be changed by using ‘WCACTION’ mnemonic.

EXAMPLES

The Check box is similar to a radio button in functionality, but with check boxes you can select more than one at a time.
Here is a check box program example:

```
 10 Dim a$[1],b$[100]
 20 Print 'XX';
 30 Print "Please choose one of the following:"
 40 Print PChr$(2,20,5,32,6,"Item 1",0);'WCCHECK';
 50 Print PChr$(3,20,7,32,8,"Item 2",0);'WCCHECK';
 60 Print @0,12;
 70 Print "Please press the 'OK' button."
 80 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
 90 Print 'IOEE K0' ! Disable echo and cursor on input
100 Input Len (1);""a$
110 If a$ = 'F1'
120   Print "OK button pressed!"
130 Else
140   Print "Error on input"
150 End If
160 Print PChr$(2);'WCQUERY';
170 Input ""b$
180 If b$ = "1" Print "Item 1 selected"
190 Print '3WCQUERY';
200 Input ""b$
210 If b$ = "1" Print "Item 2 selected"
220 Print '0WCFOCUS K3';
230 End
```

The number box is a single line user input box for numeric data only.
Here is a number box program example:

```
 10 Dim a$[1],b$[100]
 20 Print 'XX 4GRIDFONT'; ! use quarter character coordinates
 30 Print "Please enter a number value:"
 40 Print PChr$(2,80,20,128,25,"",2);'WCNUMBER';
 50 Print @0,48;
 60 Print "Please press the 'OK' button."
 70 Print PChr$(1,144,60,176,68,"OK",0);'WCBUTTON'; ! display 'ok' button
 80 Print 'IOEE K0' ! Disable echo and cursor on input
 90 Input Len (1);""a$
100 If a$ = 'F1'
110   Print "OK button pressed!"
120 Else
130   Print "Error on input"
```

```
140 End If
150 Print PChr$(2);'WCQUERY';
160 Input ""b$
170 Print "Number box input = ";b$
180 Print '0WCFOCUS K3';
190 End
```

The string box is a single line user input box for alpha/numeric data.
Here is a string box program example:

```
 10 Dim a$[1],b$[100]
 20 Print 'XX 4GRIDFONT';
 30 Print "Please enter a string value:"
 40 Print PChr$(2,80,20,128,25,"",2);'WCSTRING';
 50 Print @0,48;
 60 Print "Please press the 'OK' button."
 70 Print PChr$(1,144,60,176,68,"OK",0);'WCBUTTON'; ! display 'ok' button
 80 Print 'IOEE K0' ! Disable echo and cursor on input
 90 Input Len (1);""a$
100 If a$ = 'F1'
110   Print "OK button pressed!"
120 Else
130   Print "Error on input"
140 End If
150 Print PChr$(2);'WCQUERY';
160 Input ""b$
170 Print "String box input = ";b$
180 Print '0WCFOCUS K3';
190 End
```

The private box is a single line user input box for hidden alpha/numeric data.
The private box can be used for passwords, as the user enters characters asterisk (*) is display.
Here is a private box program example:

```
 10 Dim a$[1],b$[100]
 20 Print 'XX';
 30 Print "Please enter a password:"
 40 Print PChr$(2,20,5,32,6,"",2);'WCPRIVATE';
 50 Print @0,12;
 60 Print "Please press the 'OK' button."
 70 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
 80 Print 'IOEE K0' ! Disable echo and cursor on input
 90 Input Len (1);""a$
100 If a$ = 'F1'
110   Print "OK button pressed!"
120 Else
130   Print "Error on input"
140 End If
150 Print PChr$(2);'WCQUERY';
160 Input ""b$
170 Print "Private box input = ";b$
180 Print '0WCFOCUS K3';
190 End
```

dL4 Buttons: Standard and Radio

STANDARD BUTTON:
Standard buttons are graphical elements that perform an action when selected. Buttons are created by outputting a 'WCBUTTON' mnemonic which has up to 7 parameters as shown below:

PChr$(n,x1, y1, x2, y2, "label",options);'WCBUTTON'

n	element number of the button (must be unique in window)
x1,y1	Grid coordinates of the upper left corner of the button
x2,y2	Grid coordinates of the lower right corner of the button
label	title string displayed on button, see description below
options	an optional numeric parameter, see description below

The "options" parameter is the sum of the following option values:

1	Disable and "gray out" the button (user can't select button)
2	Use the button as a tab stop (in element number order)

The default value of "options" is zero.

The "label" parameter defines the optional selection key. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the button. The user can select the button by typing "ALT" and the character.

When selected, a button sends a function key character as input to the window containing the button. The function key character is the mnemonic 'Fn' where "n" is the element number of the button. The characters sent by the button can be changed by using the 'WCACTION' mnemonic.

DEFAULT BUTTON:
A mnemonic, 'WCDEFAULTBTN', is available to create default push buttons. These buttons are identical to those created by the 'WCBUTTON' mnemonic except that the button is marked as the default and will be triggered by the user typing ENTER if the focus is on the button or on another non-pushbutton GUI input element. The arguments to the 'WCDEFAULTBTN' mnemonic are identical to those of the 'WCBUTTON' mnemonic.
Note: if the 'nWCFOCUS' mnemonic is used to set focus to a non-default pushbutton, that button becomes the default.

RADIO BUTTON:
Radio buttons are graphical elements that have a true or false value. They are similar to check boxes except that they are used as a group in which only one radio button can have the value "true". The user can toggle the value of a particular radio button by selecting (clicking) the button. If the value is toggled to true, any other button in the same group with a true value is set to false. A radio button is considered to be in the group defined by the nearest enclosing rectangle defined by a 'x1,y1,x2,y2,n WCGROUP' mnemonic (see Misc. mnemonics). If there is no enclosing rectangle, then the button is part of group including all radio buttons in the window that are not part of an enclosing group rectangle. The element number ranges of different groups cannot overlap. Radio buttons are created by outputting a 'WCRADIO' mnemonic which has up to 7 parameters as shown below:

PChr$(n , x1, y1, x2, y2, “label”, options),’WCRADIO’

n	element number of the button (must be unique in window)
x1,y1	Upper left grid coordinates of the rectangle containing the button and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the button and title
label	title string displayed near the button, see description below
options	an optional numeric parameter, see description below

The “options” parameter is the sum of following option values:

1	Disable and “gray out” the radio button (user can’t select the button)
2	Use the button as a tab stop (in element number order)
4	Send input to window when value changes

The default value of “options” is zero.

The “label” parameter defines the title string and the optional selection key. If an ampersand character (“&”) is included in the label string, the character following it will be used as a selection key for the radio button. The user can select the button by typing “ALT” and the character.

The current value of a radio button is set by printing a “1” (selected) or a “0” (not selected) to the button.

The current value of a radio button is sent as CR terminated input to the window containing the button when a ‘nWCQUERY’ mnemonic is output with “n” equal to the element number of the button.

If option value 4 is used, the radio button sends a function key character as input to the window containing the button whenever the button value changes. The function key character is the mnemonic ‘Fn’ where “n” is the element number of the button. The characters sent by the button can be changed by using the ‘WCACTION’ mnemonic.

EXAMPLES

The standard button is a boxed area that can be pressed and will cause a character string to be sent to the application’s input handler. An example is the ‘OK’ button to complete a screen based input.
Here is a standard button example program.

```
 10 Dim a$[1]
 20 Print 'XX';
 30 Print "Please press the 'OK' button."
 40 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
 50 Print 'IOEE K0' ! Disable echo and cursor on input
 70 Input Len (1);""a$
 80 If a$ = 'F1'
 90   Print "OK button pressed!"
100 Else
110   Print "Error on input"
120 End If
130 Print '0WCFOCUS K3';
140 End
```

The pressing of the ‘OK’ button causes an input of a “function key character” by default, this can be changed to any character string.

A radio button is more of a selection button, but only one button can be selected at a time (like a radio tuner button).
Here is a radio button example program.

```
 10 Dim a$[1],b$[100]
 20 Print 'XX';
 30 Print "Please choose one of the following:"
 40 Print PChr$(2,20,5,32,6,"Item 1",0);'WCRADIO';
 50 Print PChr$(3,20,7,32,8,"Item 2",0);'WCRADIO';
 60 Print @0,12;
 70 Print "Please press the 'OK' button."
 80 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
 90 Print 'IOEE K0' ! Disable echo and cursor on input
100 Input Len (1);""a$
110 If a$ = 'F1'
120   Print "OK button pressed!"
130 Else
140   Print "Error on input"
150 End If
160 Print PChr$(2);'WCQUERY';
170 Input ""b$
180 If b$ = "1" Print "Item 1 selected"
190 Print '3WCQUERY';
200 Input ""b$
210 If b$ = "1" Print "Item 2 selected"
220 Print '0WCFOCUS K3';
230 End
```

dL4 Multi-line Boxes: Text and Memo

MULTI-LINE TEXT and MEMO BOX:
Multi-line boxes are graphical elements that contain one or more lines of text. A 'WCTEXT' box is used to display text and cannot be changed by the user.

The user can change the value of a 'WCMEMO' edit box by selecting the box and typing characters. Both types of boxes provide scroll bars if the current value is too wide or has too many lines to be fully displayed in the box. Boxes are created by outputting a 'WCTEXT' or 'WCMEMO' mnemonic which have up to 8 parameters as shown below:

PChr$(n,x1, y1, x2, y2, "label", options);'WCTEXT'
PChr$(n,x1, y1, x2, y2, "label", options,l);'WCMEMO'

n	element number of the edit box (must be unique in window)
x1,y1	Upper left grid coordinates of the rectangle containing the edit box and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the edit box and title
label	title string displayed near the edit box, see description below
options	an optional numeric parameter, see description below
l	optional limit on the number of characters accepted

The "options" parameter is the sum of following option values:

1	Disable and "gray out" the edit box (user can't select the edit box)
2	Use the edit box as a tab stop (in element number order)
4	Send input to window when value changes
32	Any loss of input focus is reported as an input value change (option 4)

The default value of "options" is zero.

The "label" parameter defines the title string and the optional selection key. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the edit box. The user can select the edit box by typing "ALT" and the character.

The current value of an edit box is set by printing one or more lines to the box. The value can be cleared by printing a 'CU' mnemonic to the box. Multi-column output can be formatted by using the 'n ST' mnemonic to set tab positions and then using horizontal tabs ('HT') between column values. All lines in the edit box must use the same tab settings.

The current value of a edit box is sent as input to the window containing the box when a 'nWCQUERY' mnemonic is output with "n" equal to the element number of the box. Each line of input is terminated by a 'CR'. If an input line is empty, a single space will be sent followed by a 'CR'. The final input line is followed by 'CR CR'.

If option value 4 is used, the edit box sends a function key character as input to the window containing the edit box whenever the edit box value changes. The function key character is the mnemonic 'Fn' where "n" is the element number of the edit box. The characters sent by the edit box can be changed by using the 'WCACTION' mnemonic.

EXAMPLES

The text box is a multi-line user display box for alpha/numeric data.
The user can not enter or change the text that is display in the box.
Here is a text box program example:

```
 10 Dim a$[1],b$[100]
 20 Print 'XX';
 30 Print "Please read the following:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCTEXT';
 50 Print '2WCSELECT';
 60 Print "This is a test of the text box display."
 70 Print "If this was a real text box display, you"
 80 Print "would be instructed to tune to your local"
 90 Print "radio station."
100 Print '0WCSELECT';
110 Print @0,12;
120 Print "Please press the 'OK' button."
130 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
140 Print 'IOEE K0' ! Disable echo and cursor on input
150 Input Len (1);""a$
160 If a$ = 'F1'
170   Print "OK button pressed!"
180 Else
190   Print "Error on input"
200 End If
210 Print '0WCFOCUS K3';
220 End
```

The memo box is a multi-line user input box for alpha/numeric data.
Here is a memo box program example:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please enter or edit text in the box:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCMEMO';
 50 Print '2WCSELECT';
 60 Print "This is a test of the text box display."
 70 Print "If this was a real text box display, you"
 80 Print "would be instructed to tune to your local"
 90 Print "radio station."
100 Print '0WCSELECT';
110 Print @0,12;
120 Print "Please press the 'OK' button."
130 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
140 Print 'IOEE K0' ! Disable echo and cursor on input
150 Input Len (1);""a$
160 If a$ = 'F1'
170   Print "OK button pressed!"
180 Else
190   Print "Error on input"
200 End If
210 Print PChr$(2);'WCQUERY';
220 Loopinput: Input ""b$
230 If b$[1,1] = "" Goto exitloop
240 l = l + 1
250 Print "Memo line #";l;"= ";b$
260 Goto Loopinput
270 exitloop: Print '0WCFOCUS K3';
280 End
```

dL4 Multi-line List Boxes: LIST and EDITLIST

MULTI-LINE LIST and EDITLIST BOX:
List boxes are graphical elements that contain string input values. The user can change the value of a list box by selecting one of the values displayed in the list box. If a 'WCEDITLIST' list box is used, the user can also type a value directly into the list box. List boxes are created by outputting a 'WCLIST' or 'WCEDITLIST' mnemonic which has up to 8 parameters as shown below:

PChr$(n,x1, y1, x2, y2, "label", options);'WCLIST'
PChr$(n,x1, y1, x2, y2, "label", options,l);'WCEDITLIST'

n	element number of the list box (must be unique in window)
x1,y1	Upper left grid coordinates of the rectangle containing the list box and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the list box and title
label	title string displayed near the list box, see description below
options	an optional numeric parameter, see description below
l	optional limit on the number of characters accepted

The "options" parameter is the sum of following option values:

1	Disable and "gray out" the list box (user can't select the box)
2	Use the list box as a tab stop (in element number order)
4	Send input to window when value changes
8	Allow selection of multiple items (WCLIST only)
16	Don't display first field of list, but return only it as value (WCLIST only)
32	Any loss of input focus is reported as an input value change (option 4)

The default value of "options" is zero.

The "label" parameter defines the title string and the optional selection key. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the list box. The user can select the list box by typing "ALT" and the character.

The values listed in a list box are set by printing one or more lines to the box with each line (separated by a CR) considered a list value. Multi-column output can be formatted by using the 'n ST' mnemonic to set tab positions and then using tab characters ('HT') between column values. All lines in the list box must use the same tab settings. The list can be cleared by printing a 'CU' mnemonic to the box.

The current values of the list box are set by printing an 'nWCMARK' mnemonic to the box where 'n' is the zero based index of the value in the list. In a multiple selection list box, a selected list item can be unselected by sending an 'nWCUNMARK' mnemonic (or 'WCUNMARK' to set all items to an unselected state). The current value of a list box is sent as CR terminated input to the window containing the box when a 'nWCQUERY' mnemonic is output with "n" equal to the element number of the list box. If the list box options allow selection of multiple items, the input list will be terminated by a final line of 'CR'.

If option value 4 is used, the list box sends a function key character as input to the window containing the box whenever the list box value changes. The function key character is the

mnemonic ‘Fn’ where “n” is the element number of the list box. The characters sent by the list box can be changed by using the ‘WCACTION’ mnemonic.

EXAMPLES

The list box is a multi-line selection box. The user can not enter or change the lines that are display in the box, they can only select a line.
Here is a list box program example:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please choose one of the following:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCLIST';
 50 Print '2WCSELECT';
 60 For i = 1 To 20
 70   Print "Item #";i
 80 Next i
 90 Print '0WCSELECT';
100 Print @0,12;
110 Print "Please press the 'OK' button."
120 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
130 Print 'IOEE K0' ! Disable echo and cursor on input
140 Input Len (1);""a$
150 If a$ = 'F1'
160   Print "OK button pressed!"
170 Else
180   Print "Error on input"
190 End If
200 Print PChr$(2);'WCQUERY';
210 Input ""b$
220 Print "List selection = ";b$
230 Print '0WCFOCUS K3';
240 End
```

The editlist box is a multi-line selection box with user input. The user can not enter or change the lines that are display in the box, they can select a line or enter their own line.
Here is a editlist box program example:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please choose one of the following or enter your own line:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCEDITLIST';
 50 Print '2WCSELECT';
 60 For i = 1 To 20
 70   Print "Item #";i
 80 Next i
 90 Print '0WCSELECT';
100 Print @0,12;
110 Print "Please press the 'OK' button."
120 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
130 Print 'IOEE K0' ! Disable echo and cursor on input
140 Input Len (1);""a$
150 If a$ = 'F1'
160   Print "OK button pressed!"
```

```
170 Else
180   Print "Error on input"
190 End If
200 Print PChr$(2);'WCQUERY';
210 Input ""b$
220 Print "Editlist selection = ";b$
230 Print '0WCFOCUS K3';
240 End
```

This list box example uses the hidden field option (16) to return codes (“Code n”) rather than the displayed text of the selected line. The example also uses the ‘nST’ mnemonic to align the displayed text in two columns:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please choose one of the following:"
 40 Print PChr$(2,10,5,72,10,"",16+2);'WCLIST';
 50 Print '2WCSELECT';
 60 Print '12ST'; ! setup the second column at the 12th character
 70 For i = 1 To 20
 80   Print "Code";i;'HT';"Item #";i;'HT';"Descriptive text"
 90 Next i
100 Print '0WCSELECT';
110 Print @0,12;
120 Print "Please press the 'OK' button."
130 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
140 Print 'IOEE K0' ! Disable echo and cursor on input
150 Input Len (1);""a$
160 If a$ = 'F1'
170   Print "OK button pressed!"
180 Else
190   Print "Error on input"
200 End If
210 Print PChr$(2);'WCQUERY';
220 Input ""b$
230 Print "List selection = ";b$
240 Print '0WCFOCUS K3';
250 End
```

dL4 Multi-line Drop Down Boxes: LISTDROP and EDITDROP

MULTI-LINE LISTDROP and EDITDROP BOX:
Drop down boxes are graphical elements that contain string input values. Like list boxes, drop down boxes allow the user to select an input value from a list, but, in a drop down box, the list is only visible when the drop down list is selected. If a 'WCEDITDROP' drop down box is used, the user can also type a value directly into the box. The user can change the value of a drop down box by selecting one of the values displayed in the list. Drop down boxes are created by outputting a 'WCLISTDROP' or 'WCEDITDROP' mnemonic which have up to 8 parameters as shown below:

PChr$(n,x1, y1, x2, y2, "label", options);'WCLISTDROP'
PChr$(n,x1, y1, x2, y2, "label", options,l);'WCEDITDROP'

n	element number of the drop down box (must be unique in window)
x1,y1	Upper left grid coordinates of the rectangle containing the drop down box and title
x2,y2	Lower right grid coordinates of the rectangle enclosing the drop down box and title
label	title string displayed near the drop down box, see description below
options	an optional numeric parameter, see description below
l	optional limit on the number of characters accepted

The display rectangle of a drop down box must provide space for the drop down list. For this reason, the rectangle should be larger than the reserved display area and the drop down list portion will often overlap other input elements.

The "options" parameter is the sum of following option values:

1	Disable and "gray out" the drop down box (user can't select the box)
2	Use the drop down box as a tab stop (in element number order)
4	Send input to window when value changes
16	Don't display first field of list (WCLISTDROP only)
32	Any loss of input focus is reported as an input value change (option 4)

The default value of "options" is zero.

The "label" parameter defines the title string and the optional selection key. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the drop down box. The user can select the drop down box by typing "ALT" and the character.

The values listed in the value list of a drop down box are set by printing one or more lines to the box with each line considered a list value. The list can be cleared by printing a 'CU' mnemonic to the box. The current value of a drop down box can be set by printing an 'nWCMARK' mnemonic to the box where 'n' is the zero based index of the value in the list. The current value of a drop down box is sent as CR terminated input to the window containing the box when a 'nWCQUERY' mnemonic is output with "n" equal to the element number of the box.

If option value 4 is used, the drop down box sends a function key character as input to the window containing the box whenever the drop down box value changes. The function key character is the mnemonic ‘Fn’ where “n” is the element number of the drop down box. The characters sent by the drop down box can be changed by using the ‘WCACTION’ mnemonic.

EXAMPLES

The listdrop box is a multi-line selection box, that can drop down from the defined position. The user cannot enter or change the lines that are displayed in the box, they can only select a line.
Here is a listdrop box program example:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please choose one of the following:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCLISTDROP';
 50 Print '2WCSELECT';
 60 For i = 1 To 20
 70   Print "Item #";i
 80 Next i
 90 Print '0WCSELECT';
100 Print @0,12;
110 Print "Please press the 'OK' button."
120 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
130 Print 'IOEE K0' ! Disable echo and cursor on input
140 Input Len (1);""a$
150 If a$ = 'F1'
160   Print "OK button pressed!"
170 Else
180   Print "Error on input"
190 End If
200 Print PChr$(2);'WCQUERY';
210 Input ""b$
220 Print "Listdrop selection = ";b$
230 Print '0WCFOCUS K3';
240 End
```

The editdrop box is a multi-line selection box, that can drop down from the defined position. The user can not enter or change the lines that are display in the box, they can enter their own line or select a line.
Here is a editdrop box program example:

```
 10 Dim a$[1],b$[2000]
 20 Print 'XX';
 30 Print "Please choose one of the following or enter your own line:"
 40 Print PChr$(2,10,5,72,10,"",2);'WCEDITDROP';
 50 Print '2WCSELECT';
 60 For i = 1 To 20
 70   Print "Item #";i
 80 Next i
 90 Print '0WCSELECT';
100 Print @0,12;
110 Print "Please press the 'OK' button."
120 Print PChr$(1,36,15,44,17,"OK",0);'WCBUTTON'; ! display 'ok' button
130 Print 'IOEE K0' ! Disable echo and cursor on input
140 Input Len (1);""a$
```

```
150 If a$ = 'F1'
160   Print "OK button pressed!"
170 Else
180   Print "Error on input"
190 End If
200 Print PChr$(2);'WCQUERY';
210 Input ""b$
220 Print "Editdrop selection = ";b$
230 Print '0WCFOCUS K3';
240 End
```

dL4 Menu Controls

MENUS:
Menus are graphical elements that perform an action when selected. They are similar to buttons, but they are arranged in a group and, often, in a tree-like hierarchy. The first level of a menu hierarchy is the menu bar which lists the top level elements of a menu. A window can have only one menu bar and thus only one menu. A menu or sub-menu of a menu is created by outputting a 'WCMENU' or 'WCSUBMENU' mnemonic. The menu is then populated with menu items by outputting 'MENUACTION', 'WCMENUCHECK', 'WCMENURADIO', or 'WCMENUSEP' mnemonics. When the menu is complete, a 'WCMENUEND' mnemonic must be output. A sub-menu is created by issuing a 'WCSUBMENU' mnemonic while populating a higher level menu. Only main windows can have menus; child windows and windows with the "DIALOG" style cannot have menus.

The menu mnemonics are defined as follows:

PChr$(n,"label","key",options);'WCMENU'
PChr$(n,"label","key",options);'WCSUBMENU'
PChr$(n,"label","key",options);'WCMENUACTION'
PChr$(n,"label","key",options);'WCMENUCHECK'
PChr$(n, "label","key",options);'WCMENURADIO'
'WCMENUSEP'
'WCMENUEND'

n	element number of the menu item (must be unique in window)
label	title string displayed near the edit box, see description below
key	reserved for future use
options	an optional numeric parameter, see description below

The "options" parameter is the sum of following option values:

1	Disable and "gray out" the menu item (user can't select the item)

The default value of "options" is zero.

The "label" parameter defines the menu item name. If an ampersand character ("&") is included in the label string, the character following it will be used as a selection key for the menu item. The user can select a menu or submenu by pressing Alt and the selection key simultaneously. The user can select a menu item by typing the character after the menu bar or sub-menu containing the item is selected.

Menu items are defined by the 'WCMENU', 'WCSUBMENU', 'WCMENUACTION', 'WCMENUCHECK', and 'WCMENURADIO' mnemonics. All menu items except for sub-menus define actions and, when selected, send characters as input to the window containing the menu. The 'WCMENUCHECK' and 'WCMENURADIO' also have a displayed true/false state which can be changed by the program. This state is displayed as a check mark for 'WCMENUCHECK' and as a bullet for 'WCMENURADIO'. The current value of a check or radio menu item is set by first selecting the element number of the menu item (WCSELECT) and then printing a "1" (selected) or a "0" (not selected). The current value of a menu item is sent as CR terminated input to the

window containing the menu control when a ‘nWCQUERY’ mnemonic is output with “n” equal to the element number of the menu item.

Each radio menu item is a member of a group. The group is all other radio items in the menu unless a sub-group is created using ‘MENUSEP’ separators. Only one radio menu item in a group can have the value of true.

When selected, a menu item sends a function key character as input to the window containing the item. The function key character is the mnemonic ‘Fn’ where “n” is the element number of the item. **The characters sent by the item can be changed by using the ‘’WCACTION’ mnemonic.**

EXAMPLES

The menu selection allows a menu list to be added to the Windows menu bar. Here is a menu program example:

```
10 Print PChr$(1,"&Menu1","");'WCMENU';
20 Print PChr$(2,"&Help","");'WCMENUACTION';
30 Print PChr$(3,"&Bye","");'WCMENUACTION';
40 Print 'WCENDMENU';
50 Print PChr$(2,1,"Help\15\");'WCACTION';
60 Print PChr$(3,1,"BYE\15\");'WCACTION';
70 End
```

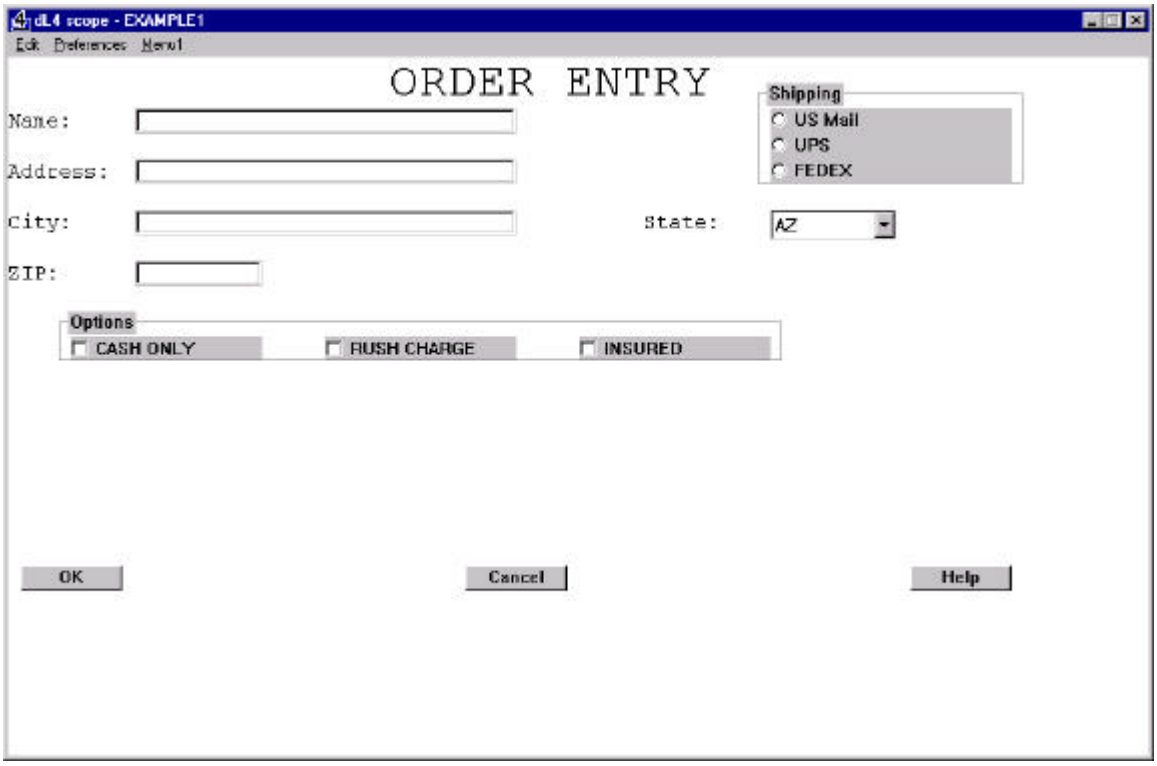
EXAMPLE PROGRAM #1:

This example program that shows how some of the controls can be use together on a page.

```
 10 External Sub Help()
 20   Dim a$[1]
 30   Rem Open a new window for help-note that is it sized using the current
 40   Rem coordinate definition of '4GRIDFONT'
 50   Open #99,{"Help","Titl",128,40} As "Window"
 60   Rem The new window does not inherit the current coordinate definitions
 70   Rem and has its own set of GUI element numbers
 80   Print #99;'4GRIDFONT';
 90   Print #99;PChr$(2,40,28,80,33,"Continue",2);'WCBUTTON';
 91   Rem set font size to 6/4ths or 1.5 times normal size & begin bold
100   Print #99;'6FONTSIZE BBOLD'; " This is an example\15\ of help text";
110   Print #99;'IOEE K0' ! Disable echo and cursor on input
120   Print #99;'2WCFOCUS'; ! Put focus on element #2, the button
130   Input #99;Len (1);a$
140   Close #99
150 End Sub
160 Dim a$[1],b$[2000]
170 Rem Clear the screen, existing GUI elements, and existing menus
180 Print 'XX';
190 Rem Define screen coordinates in quarter characters so that edit boxes
200 Rem can be sized just a little taller than the characters contained
210 Rem in the boxes.  Otherwise, the GUI element font should be set (via
220 Rem 'WCSETFONT') to be smaller than normal characters.  Note that all
230 Rem screen coordinates or character size after this are multiplied by 4.
240 Print '4GRIDFONT';
250 Rem Initialize the menus
260 Print PChr$(101,"&Menu1","");'WCMENU';
270 Print PChr$(102,"&Help","");'WCMENUACTION';
280 Print PChr$(103,"&Bye","");'WCMENUACTION';
290 Print 'WCENDMENU';
300 Print PChr$(102,1,'F4');'WCACTION';
310 Print PChr$(103,1,'F3');'WCACTION';
320 Rem Print the form title in double high characters
330 Print @120,0;'8FONTSIZE';"ORDER ENTRY";'4FONTSIZE';
340 Rem Print some labels
350 Print @0,8;"Name:";@0,16;"Address:";@0,24;"City:";@200,24;"State:";
360 Print @0,32;"ZIP:";
370 Rem Create buttons and edit boxes for input
380 Print PChr$(2,4,80,36,85,"OK",2);'WCBUTTON'
390 Print PChr$(3,144,80,176,85,"Cancel",2);'WCBUTTON'
400 Print PChr$(4,284,80,316,85,"Help",2);'WCBUTTON'
410 Print PChr$(11,40,8,160,13,"",2);'WCSTRING'; !Name
420 Print PChr$(12,40,16,160,21,"",2);'WCSTRING'; !Address
430 Print PChr$(13,40,24,160,29,"",2);'WCSTRING'; !City
440 Print PChr$(14,240,24,280,68,"",2);'WCLISTDROP'; !State
450 Print PChr$(15,40,32,80,37,"",2);'WCSTRING'; !Zip
460 Print PChr$(16,240,8,308,12,"US Mail",2);'WCRADIO'; !Ship Via's
470 Print PChr$(17,240,12,308,16,"UPS",2);'WCRADIO';
480 Print PChr$(18,240,16,308,20,"FEDEX",2);'WCRADIO';
490 Print PChr$(19,236,4,312,22,"Shipping");'WCGROUP' ; !enclose in a group
500 Print PChr$(20,20,44,80,48,"CASH ONLY",2);'WCCHECK'; !Shipping options
510 Print PChr$(21,100,44,160,48,"RUSH CHARGE",2);'WCCHECK';
520 Print PChr$(22,180,44,240,48,"INSURED",2);'WCCHECK';
```

```
 530 Print PChr$(23,16,40,244,52,"Options");'WCGROUP' ; !enclose in a group
 540 States: Data "AK","AL","AR","AS","AZ","CA","CO","CT","DC","DE", "FL"
 550Data"GA","GU","HI","IA","ID","IL","IN","KS","KY","LA","MA","MD","ME","MH"
 560Data"MI","MN","MO","MP","MS","MT","NC","ND","NE","NH","NJ","NM","NV","NY"
 570Data"OH","OK","OR","PA","PR","PW","RI","SC","SD","TN","TX","UT","VA","VT"
 580 Data "VI","WA","WI","WV","WY",""
 590 Restor States
 600 Print '14WCSELECT'; !select element #14, State, to load drop down
 610 Do
 620   Read b$
 630   If b$ = "" Exit Do
 640   Print b$
 650 Loop
 660 Print '0WCMARK'; !default state to first one
 680 WaitForEvent: Print '0WCSELECT'; !select element #0, window itself
 685 Print @0,88;
 690 Print 'IOEE K0' ! Disable echo and cursor on input
 700 Print '11WCFOCUS'; !put focus on name box
 710 Input Len (1);""a$
 720 If a$ = 'F3' Goto ProgramExit !bye menu or cancel button
 730 If a$ = 'F4' Call Help() \ Goto WaitForEvent !help menu or button
 750 If a$ = 'F2' !OK button
 760   Rem Get input values from each of the GUI input elements
 770   For i = 11 To 22
 780     If i <> 19
 790       Print PChr$(i);'WCQUERY';
 800       Input ""b$[(i - 11) * 30 + 1,(i - 11) * 30 + 30]
 810     End If
 820   Next i
 830 Else
 840   Print "Error on input"
 850   Goto ProgramExit
 860 End If
 870 Pause 5
 880 Print 'CS';
 890 Print @0,0;"Order Information:"  !look at input
 900 Print @0,8;"Name:";@80,8;b$[1,30]
 910 Print @0,12;"Address:";@80,12;b$[31,60]
 920 Print @0,16;"City:";@80,16;b$[61,90]
 930 Print @0,20;"State:";@80,20;b$[91,120]
 940 Print @0,24;"ZIP:";@80,24;b$[121,150]
 950 Print @0,28;"Shipping:";@80,28;
 960 If b$[151,151] = "1" Print "US Mail"
 970 If b$[181,181] = "1" Print "UPS"
 980 If b$[211,211] = "1" Print "FedEx"
 990 Print @0,32;
1000 If b$[271,271] = "1" Print "Options:"; Tab (80);"Cash Only"
1010 If b$[301,301] = "1" Print "Options:"; Tab (80);"Rush Charge"
1020 If b$[331,331] = "1" Print "Options:"; Tab (80);"Insured"
1030 Pause 50
1040 ProgramExit: Print '0WCFOCUS K3';
1045 Rem put keyboard focus on element #0, window, & bring back cursor
1050 Print 'XX'; !clear window
1060 End
```

Here is the first page as generated by example #1 program:


dL4 scope - EXAMPLE1
Edit Preferences Menu1
ORDER ENTRY
Name:
Address:
City:
State:
AZ
ZIP:
Shipping
US Mail
UPS
FEDEX
Options
CASH ONLY
RUSH CHARGE
INSURED
OK
Cancel
Help