Combobox

Portions of this topic are copied from the Liberty BASIC helpfile.

Table of Contents

Description

A combobox is a control that contains a list of items that is hidden until the user clicks on the arrow button, at which time the list drops down and allows the user to select an item.

Comboboxes include an associated textbox that allows the user to type text. The programmer can retrieve the text typed by the user.

After the user makes a selection from the dropdown list, it is displayed and highlighted in the textbox portion of the combobox.

Syntax

```
COMBOBOX #handle.ext, array$(), eventHandler, xPos, yPos, wide, high
```

Arguments Explained

#handle.ext

The #handle part of the combobox statement needs to be the same as the handle of the window containing the combobox. The .ext part needs to be unique so that the program can send commands to the combobox and get information from it later.

array$()

This is the name of the **single-dimensioned string** array associated with the combobox. Load the array with strings before opening the window. If it becomes necessary to change the contents of the combobox as the program is run, change the contents of the array and send a RELOAD command to the combobox. The index numbers of items in the array may not match the index numbers of the same items in the control. The control is loaded from the array, and the first index used in the control is "1". No empty strings are loaded into the control, so only the array items that contain text are loaded.

eventHandler

This is the branch label or subroutine where execution begins when the user selects an item from the combobox by clicking it.

xPos & yPos

These coordinates specify the the distance in x and y (in pixels) of the combobox from the upper-left corner of the window.

wide & high

These arguments set the width and height (in pixels) of the combobox. "Height" in this case refers to the length of the selection list when the combobox's button is clicked, not to the size of the initial selection window, which is dependant upon the size of the font.

Commands

```
#handle.ext "!contents"
```

This changes the contents of the **textbox** part of the combobox to be the string after the !.

```
#handle.ext "contents? text$"
```

This retrieves the contents of the **textbox** part of the combobox into the variable called text$. The variable can be any valid string variable name.

```
#handle.ext "locate x y width height"
```

This respositions the combobox in its window, if the window is of type "window". The combobox will not update its size and location until a REFRESH command is sent to the window.

```
#handle.ext "font facename pointSize"
```

This sets the control's font to the specified face and point size. If an exact match cannot be found, then Liberty BASIC will try to find a close match, with size taking precedence over face.

```
'Example:
#handle.ext "font times_new_roman 10"

#handle.ext "select string"

#handle.ext "select "; string$
```

This selects the item the same as "string" and updates the display. Note that "string" must be a valid item from the combobox array. If a variable is to be used in this command, it must be located outside the quotation marks, with the blank space preserved.

```
#handle.ext "selectindex i"

#handle.ext "selectindex "; i
```

This selects the item at index position i and updates the display. Note that "i" must be a valid index number for the combobox array. If a variable is to be used in this command, it must be located outside the quotation marks, with the blank space preserved

```
#handle.ext "selection? selected$"
```

This places the string selected by the user into the variable selected$. Any valid string variable name may be used. If there is no selected item, then selected$ will be a string of zero length (a null string).

```
#handle.ext "selectionindex? index"
```

This places the index of the selected string into the variable called index. Any valid numeric variable name may be used. If there is no selected item, then index will be set to 0.

```
#handle.ext "reload"
```

This reloads the combobox with the current contents of its array and updates the display.

```
#handle.ext "setfocus"
```

This causes the combobox to receive the input focus. This means that any keypresses are directed to the combobox.

```
#handle.ext "enable"
```

This causes the control to be enabled. The user can view and activate the combobox.

```
#handle.ext "disable"
```

This causes the control to be inactive and grayed-out. The user can view it, but it is grayed-out to indicate that it cannot be used.

```
#handle.ext "show"
```

This causes the control to be visible.

```
#handle.ext "hide"
```

This causes the control to be hidden or invisible.

Demos

Combobox demo with branch label event handler:

```
  nomainwin
  a$(1) = "one"
  a$(2) = "two"
  a$(3) = "three"
  a$(4) = "four"
  combobox #win.combo, a$(),[doCombo],10,10,120,200
  open "Combobox Demo" for window as #win
  #win "trapclose [quit]"
  #win.combo "selectindex 1"
  wait

[quit]
  close #win
  end

[doCombo]
  #win.combo "selection? sel$"
  notice "You chose ";sel$
  wait
```

Combobox demo with subroutine event handler:

```
  nomainwin
  a$(1) = "one"
  a$(2) = "two"
  a$(3) = "three"
  a$(4) = "four"
  combobox #win.combo, a$(),doCombo,10,10,120,200
  open "Combobox Demo" for window as #win
  #win "trapclose Quit"
  #win.combo "selectindex 1"
  wait

sub Quit handle$
```

```
  close #handle$
  end
end sub

sub doCombo handle$
  #handle$ "selection? sel$"
  notice "You chose ";sel$
end sub
```

TIPS

Here are some handy tips for dealing with comboboxes.

Array Indices

The combobox is filled from the associated **single-dimensioned string** array. If the array item is empty, the empty string is skipped and not loaded into the combobox. Array indices may start at 0. Combobox indices **always** start at 1. For these two reasons, the index given to an array item may not match the index given to the same item in the combobox.

Click items in the combobox demo below. The indices assigned are 0, 4, 6, 8. The combobox gives them indices of 1, 2, 3 and 4.

```
nomainwin
  a$(0) = "one"
  a$(4) = "two"
  a$(6) = "three"
  a$(8) = "four"
  combobox #win.combo, a$(),[doCombo],10,10,120,200
  open "Combobox Demo" for window as #win
  #win "trapclose [quit]"
  wait

[quit]
  close #win
  end

[doCombo]
  #win.combo "selectionindex? selindex"
  notice "You chose index ";selindex
  wait
```

Modified Array

If the associated array is changed in any way, the combobox must be issued a "reload" command to update the contents. Some changes that may be made to the array are:

- Redimensioning with REDIM.
- Assigning additional or different values to array contents.
- SORTing the array.

Array Gotchas

Arrays by default are dimensioned with indices 0 - 10. Larger arrays must be dimensioned before use.

```
DIM myArray$(13)
```

Comboboxes require a **string** array. A string array name includes the dollar sign character and must be filled with string literals or variables. They cannot be assigned numeric values.

```
myArray$(2) = "Antarctica"
var$ = "South America"
myArray$(3) = var$
'WRONG:
myArray$(4) = 17
```

Table of Contents

Array Indices

Modified Array

Array Gotchas