



Introduction

The CET W32 Extended Function Library provides over 200 additional functions that can be called from within W32 BASIC programs. These functions must are not contained within the standard W32 runtime executable (RUNW32), but can be linked to programs directly.

We would like to thank Omega Systems, the CET European distributor, for funding the development and testing of this function library, and producing the majority of this document.

Overview

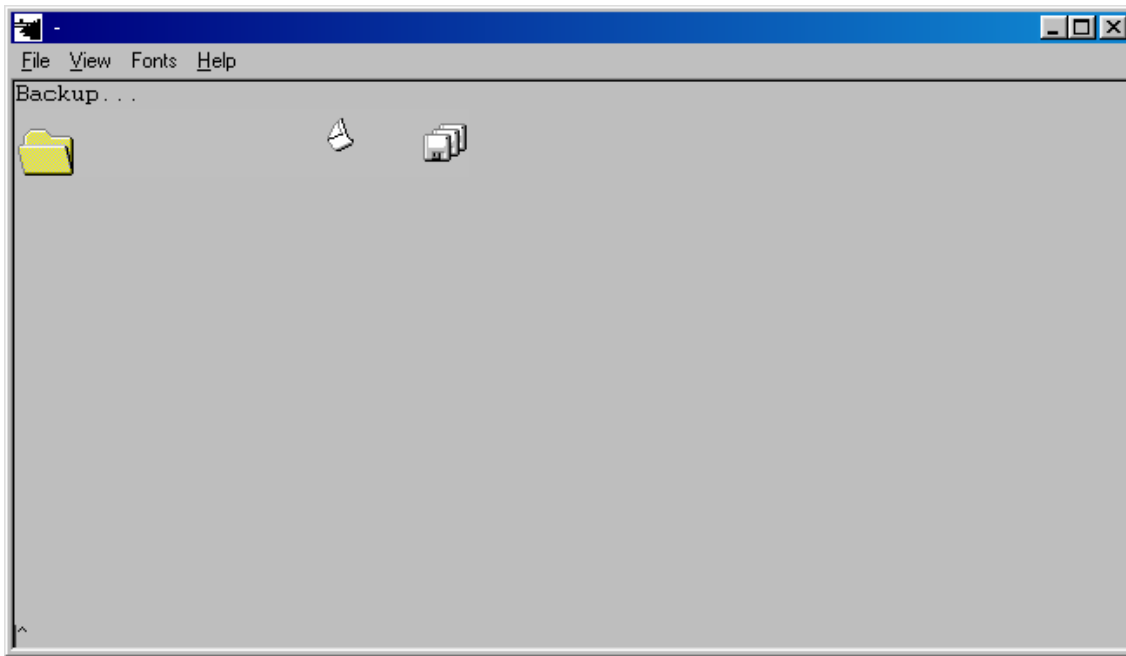
The following functions are available to CET BASIC developers by a set of OBJ files or a LIB file. In general, these OBJ files (or LIB file) must be saved to the folder where the source code files (.b) are.

You can execute these functions from your BASIC program using the *CALL* statement. When compiling your programs you must specify the name of the functions that you are calling.

For example: Here is a sample program (see function #63) that uses *CALL* sentences to execute several of the library functions. Suppose that the following code is saved to a file called backup.b that looks like this:

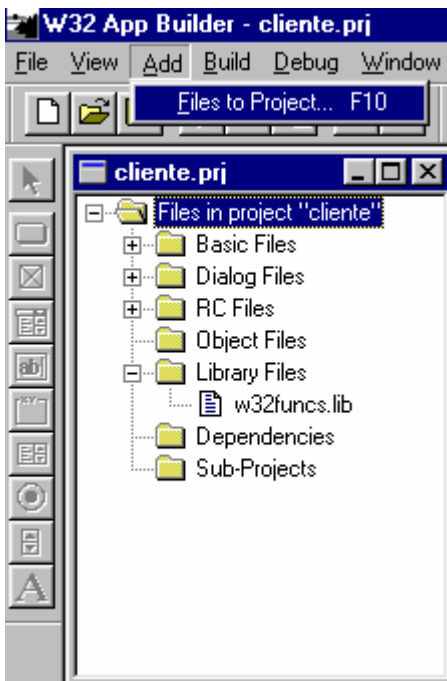
```
print at$(1,1);"Backup..."
call w32CreateAnimation(1,15,150,50,addrof(ani%))
call w32AnimationOpen(ani%,".\file.avi",addrof(ret%))
call w32AnimationPlay(ani%,addrof(ret%))
wait
call w32AnimationStop(ani%,addrof(ret%))
wait
call w32DestroyWindow(ani%)
print at$(1,1);"End Backup"
wait
```

This program shows how an AVI file can be used to create an animation (see illustration below)



To compile this program, you can use the command-line `obw32` command or the W32 Application Builder.

Building Programs Using the Extended Library

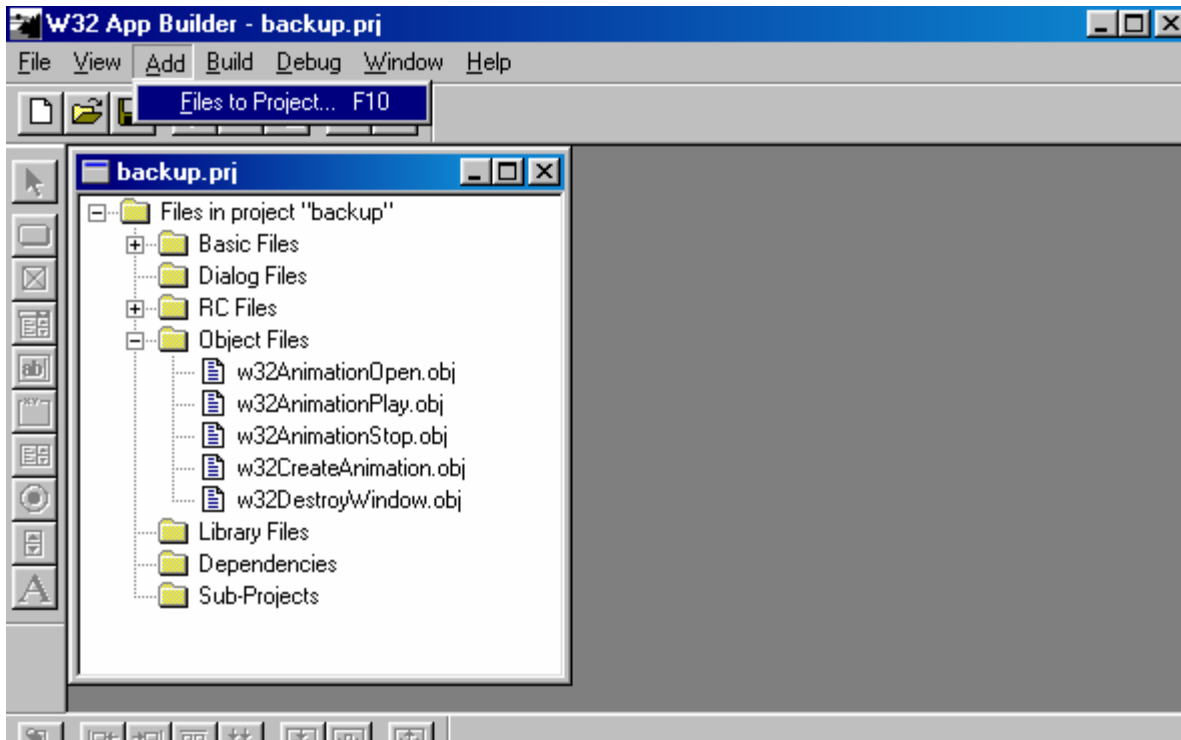


Command Line (or .bat file):

```
Obw32 -o backup backup.b w32CreateAnimation.obj  
w32AnimationOpen.obj w32AnimationPlay.obj  
w32AnimationStop.obj w32DestroyWindow.obj
```

Using CET W32 Application Builder:

You must use the option (Add /Files to Project... F10) to add all OBJ files you call from your program. This files will be showed in the 'Object Files' group.



You can also just the entire library, the file w32funcs.lib and all functions will be available.

The W32 Extended Library Functions

The following functions are available for use in W32 BASIC programs. Many of the functions have an associated test BASIC program that you can compile and link to see how the function works.

1. w32CreateCaret(nWidth%,nHeight%,addrof(retcode%))
2. w32ExitWindows (code%,addrof(retcode%))
3. w32PlaySound(filename\$,addrof(retcode%))
4. w32MessageBox(message\$,title\$,style%,addrof(retcode%))
5. w32SetWindowText(window\$,title\$,addrof(retcode%))
6. w32GetWindowSessions(window\$,addrof(retcode%))
7. w32LBoxResetContent()
8. w32LBoxDir(directories\$,addrof(retorno%))
9. w32BLoadImage(filename\$,addrof(retcode%))
10. w32BFLoadImage(fichero\$,addrof(retorno%))
11. w32ClipCopyData(text\$,addrof(retcode%))
12. w32ClipIsAvailable(addrof(result%))
13. w32ClipPasteData(addrof(text\$),addrof(retcode%))
14. w32CopyFile(source\$,target\$,addrof(error\$),addrof(retcode%))
15. w32CreateDirectory(dir\$,addrof(retcode%))
16. w32EditUndo()

17. w32FileSize(file\$,addrof(size%),addrof(retcode%))
18. w32FileTime(file\$,addrof(year%),addrof(month%),addrof(day%),
addrof(hour%),addrof(min%),addrof(sec%),addrof(retcode%))
19. w32FindFirstFile(pattern\$,addrof(firstfile\$),addrof(handle%))
20. w32FindNextFile(handle\$,addrof(nextfile\$),addrof(retcode%))
21. w32GetComputerName(addrof(name\$))
22. w32GetDiskFreeSpace(drive\$,addrof(bytesfree%),addrof(retcode%))
23. w32GetWindowsDirectory(addrof(path\$))
24. w32GetDriveType(drive\$,addrof(retcode%))
25. w32keybd_event(char\$,char%)
26. w32Skeybd_event_down(key%)
27. w32Skeybd_event_up(key%)
28. w32MoveWindow(title\$,x%,y%,width%,height%,addrof(retcode%))
29. w32RemoveDirectory(dir\$,addrof(retcode%))
30. w32SetForegroundWindow(title\$,addrof(retcode%))
31. w32BMultiline()
32. w32B3D()
33. w32BFlat()
34. w32GetDiskSpace (unit\$,addrof(space%),addrof(ret%))
35. w32SetFont(font\$,points%,angle%,bold%,italics%,underline%,strike_out%)
36. w32CreateProgress(x%,y%,width%,height%,smooth%,vertical%,addrof(hand%))
37. w32ProgressSetRange(hand%,min%,max%)
38. w32ProgressSetStep(hand%,step%)
39. w32ProgressStepIt(hand%)
40. w32ProgressSetColor(hand%,r%,g%,b%)
41. w32CreateCalendar(x%,y%,width%,height%,addrof(cal%))
42. w32CalendarGetSel(cal\$,addrof(year%),addrof(month%),addrof(day%))
43. w32CreateList(x%,y%,width%,height%,addrof(lis%))
44. w32ListInsertColumn(lis%,title\$,width%,alig%,index%,addrof(ret%))
45. w32ListInsertItem(lis%,text\$,item%,addrof(resu%))
46. w32ListSetText(lis%,text\$,item%,subitem%,addrof(resu%))
47. w32ListGetText(lis%,item%,subitem%,addrof(text\$),addrof(ret%))
48. w32ListGetSelItem(lis\$,addrof(item%))
49. w32ListSetSelItem(lis%,item%,addrof(ret%))
50. w32ListGetItemCount(lis\$,addrof(count%))
51. w32ListFindItem(lis%,key\$,addrof(item%))
52. w32ListSetColor(lis%,r%,g%,b%)
53. w32ListDeleteItem(lis%,item%,addrof(ret%))
54. w32ListDeleteAllItems(lis\$,addrof(ret%))
55. w32ListDeleteColumn(lis%,column%,addrof(ret%))
56. w32SetFocus(hand\$,addrof(ret%))
57. w32CreateAnimation(x%,y%,width%,height%,addrof(ani%))
58. w32AnimationOpen(ani%,file\$,addrof(ret%))
59. w32AnimationPlay(ani\$,addrof(ret%))
60. w32AnimationStop(ani\$,addrof(ret%))
61. w32MAPISendDocuments(FullPaths\$,filenames\$,addrof(ret%))
62. w32MAPIReadMail(directory\$,originator\$,addrof(files%),addrof(mes%))

63. w32DestroyWindow(handle%)
64. w32ListEnsureVisible(lis%,item%,addrof(ret%))
65. w32ListGridLines(lis%,addrof(ret%))
66. w32ListClearGridLines(lis%,addrof(ret%))
67. w32CalendarSetSel(cal%,year%,month%,day%)
68. w32ListFullRowSel(lis%,addrof(return%))
69. w32ListDragDrop(lis%,addrof(return%))
70. w32CreateTip(text\$,addrof(tip%))
71. w32TipUpdateText(tip%,newtext\$,addrof(return%))
72. w32GetParent(addrof(parent%))
73. w32ResizeDialog(parent%,x%,y%, width%,height%,addrof(return%))
74. w32ListSetBkColor(lis%,r%,g%,v%)
75. w32EnableWindow(hand%,state%)
76. w32FtpGet(ftpfile\$,user\$,password\$,localfile\$,addrof(return%))
77. w32FtpPut(ftpfile\$,user\$,password\$,localfile\$,addrof(return%))
78. w32PrinterDlg(addrof(context%),addrof(printer\$),addrof(o%),addrof(copies%))
79. w32PrinterContext(printer\$,addrof(context%))
80. w32PrinterDeleteContext(context%)
81. w32PrinterOffsetX(context%,addrof(offsetx%))
82. w32PrinterOffsetY(context%,addrof(offsety%))
83. w32PrinterWidth(context%,addrof(totalwidth%))
84. w32PrinterHeight(context%,addrof(totalheight%))
85. w32ListSortItems(lis%,column%,alphanum%,order%,addrof(ret%))
86. w32CreateUpDown(min%,max%,addrof(updown%))
87. w32ListCheckBoxes(lis%,addrof(return%))
88. w32ListGetCheck(lis%,item%,addrof(sel%))
89. w32ListSetCheck(lis%,item%,state%,addrof(retcode%))
90. w32GetFont(addrof(font\$),addrof(points%),addrof(angle%),addrof(bold%),addrof(italics%),ad
drof(underline%),addrof(strike_out%))
91. w32GetWindow(parent%,ID%,addrof(hand%))
92. w32SetFontH(font\$,points%,angle%,bold%,italics%,underline%,strike_out%)
93. w32ResizeWindowH(hand%,x%,y%, width%,height%,addrof(return%))
94. w32CreateImageH(hand%,file\$,x%,y%, width%,height%,addrof(ret%))
95. w32EditENTERn(hand%,IDbuttonESC%)
96. w32SetCtrlColor(padre%,control%,rf%,gf%,bf%,rb%,gb%,bb%)
97. w32CreateListH(dialog%,x%,y%,width%,height%,addrof(lis%))
98. w32ListGetEventsn(dialog%,lis%,button%,addrof(event%),addrof(param1%),addrof(param2%
))
99. w32EditENTER(index%,hand%,IDbutton%,addrof(key%))
100. w32BrowseForFolder(Dialog%,title\$,addrof(dir\$),addrof(ret%))
101. w32FileCopyProgress(Dialog%,from\$,to\$,addrof(ret%))
102. w32FileDeleteProgress(Dialog%,from\$,addrof(ret%))
103. w32FileRenameProgress(Dialog%,from\$,to\$,addrof(ret%))
104. w32FileMoveProgress(padre%,from\$,to\$,addrof(ret%))
105. w32ListFindItemPartial(lis%,string\$,position%,addrof(when%))

106. w32ControlFlat(control%)
107. w32CreateImageList(width%,file\$,addrof(ima%))
108. w32ListImageList(lis%,ima%,addrof(ret%))
109. w32ListSetImage(lis%,item%,ima_index%,addrof(ret%))
110. w32ListInsertItemEx(lis%,text\$,item%,ima_index%,addrof(resu%))
111. w32CreateTreeH(dialog%,x%,y%,width%,height%,addrof(tree%))
112. w32TreeInsertItem(tree%,text\$,parent%,index%,childrens%,addrof(ret%))
113. w32TreeImageList(tree%,ima%,addrof(ret%))
114. w32TreeInsertItemEx(tree%,text\$,parent%,index%,childrens%,ima_index%,addrof(ret%))
115. w32TreeExpand(tree%,parent%,addrof(ret%))
116. w32TreeEnsureVisible(tree%,item%,addrof(ret%))
117. w32TreeDeleteItem(tree%,item%,addrof(ret%))
118. w32TreeDeleteAllItems(tree%,addrof(ret%))
119. w32TreeSetColor(tree%,r%,g%,b%)
120. w32TreeSetSel(tree%,item%,addrof(ret%))
121. w32TreeGetSelItem(tree%,addrof(ret%))
122. w32TreeGetItem(tree%,item%,addrof(text\$),addrof(index%))
123. w32TreeGetParent(tree%,item%,addrof(parent%))
124. w32TreeGetChild(tree%,parent%,addrof(child%))
125. w32TreeGetNextSibling(tree%,item%,addrof(sibling%))
126. w32TreeGetEventsn(dialog%,tree%,button%,addrof(etree%),addrof(ptree%))
127. w32DefineRegionn(dialog%,button%,addrof(control%),addrof(left%),addrof(top%),addrof(right%),addrof(bottom%))
128. w32ListSetHotItem(lis%,item%,addrof(ret%))
129. w32CreateProgressH(parent%,x%,y%,width%,height%,smooth%,vertical%,addrof(hand%))
130. w32ChooseColor(parent%,r%,g%,b%,addrof(red%),addrof(green%),addrof(blue%),addrof(ret%))
131. w32DeleteObject(objeto%)
132. w32GetBitmapDimensions(path\$, addrof(w%), addrof(h%))
133. w32CreateImageHEx(hand%,path\$,x%,y%,w%,h%,addrof(image%),addrof(control%))
134. w32B3DH(button%)
135. w32BFlatH(button%)
136. w32BMultilineH(button%)
137. w32BLoadImageH(button%,file\$,w%,h%,addrof(image%))
138. w32BFLoadImageH(button%,file\$,w%,h%,addrof(image%))
139. w32ComboGetCurSel(combo%,addrof(sel%))
140. w32ComboResetContent(combo%,addrof(ret%))
141. w32CreateCalendarH(parent%,x%,y%,w%,h%,addrof(cal%))
142. w32CreateProcess(command\$,addrof(retorno%))
143. w32CreateShortCut(program\$,access_ink\$,work_dir\$,comment\$,addrof(ret%))
144. w32CreateTipH(control%,text\$,addrof(tip%))
145. w32CreateWindowH(parent%,ID%,type%,style%,style_ex%,title\$,x%,y%,w%,h%,addrof(control%))
146. w32DTPSetSel(control%,null%,year%,month%,day%,hour%,min%,sec%,addrof(ret%))

147. w32DTPGetSel (control%, addrof(year%), addrof(month%), addrof(day%), addrof(hour%),
addrof(min%), addrof(sec%), addrof(ret%))
148. w32EditGetLineCount (control%,addrof(lines%))
149. w32EditGetLine (control%, line_number%, addrof(text\$), addrof(length%))
150. w32GetDiskFreeSpaceEx(unit\$,addrof(bytesfree\$),addrof(retcode%))
151. w32GetDiskSpaceEx (unit\$,addrof(espace\$),addrof(ret%))
152. w32GetVersion(addrof(ver%))
153. w32GetWindowID(control%,addrof(id%))
154. w32GetWindowRectEx(control%,addrof(left%),addrof(top%),addrof(right%),addrof(bottom%)
,addrof(ret%))
155. w32ImageBackgroundn(parent%,image\$,w%,h%,addrof(ret%))
156. w32LBoxResetContentH(control%)
157. w32ListGetRect(lis%,item%,subitem%,addrof(left%),addrof(top%),addrof(right%),addrof(bott
om%),addrof(ret%))
158. w32ListSetColumn(list%,column%,title\$,addrof(ret%))
159. w32ListSetFullBkColor(lis%,r%,g%,b%)
160. w32MAPISendMail(subject\$,content\$,destination_
address\$,name\$,number_of_files%,file\$,addrof(ret%))
161. w32CreateMenu(addrof(menu%))
162. w32MenuItemInsert(menu%, id%, type%, text\$, addrof(ret%))
163. w32MenuItemBitmap(menu%,ID%,image\$,addrof(image%))
164. w32MenuPopup(menu%, control%, left%, top%, addrof(sel%))
165. w32MenuDestroy(menu%,addrof(ret%))
166. w32SetWindowOrder(control%,control_ref%,addrof(ret%))
167. w32TabStopWindow(control%,yes_no%)
168. w32TipUpdateTextH(tip%, control%, new_text\$, addrof(retorno%))
169. w32TreeGetRect(tree%, item%, addrof(left%), addrof(top%), addrof(right%),
addrof(bottom%),addrof(ret%))
170. w32WinHelp(parent%,help_file\$,operation%,value\$,addrof(ret%))
171. w32ListGetColumnsOrder(lis%,number_of_cols%,addrof(columns_order\$),addrof(ret%))
172. w32CreateGrid(parent%,title\$,ID%,x%,y%,w%,h%,addrof(grid%))
173. w32GridDeleteAllItems(grid%)
174. w32GridDeleteColumn(grid%, col%)
175. w32GridDeleteItem(grid%, temp%)
176. w32GridEditCurCell(grid%)
177. w32GridEditing (grid%, addrof(editing%))
178. w32GridGetCellProtection(grid%, row%, col%, addrof(protected%))
179. w32GridGetCol(grid%, addrof(current_col%))
180. w32GridGetColCount(grid%, addrof(num_of_cols%))
181. w32GridGetEditString(grid%, addrof(txt\$))
182. w32GridGetEvent(grid%, addrof (event%))
183. w32GridGetEventP1(grid%, addrof (p1%))
184. w32GridGetEventP2(grid%, addrof (p2%))
185. w32GridGetItemCount(grid%, addrof (rows%))

```

186. call w32GridGetRect (grid%, row%, col%, addrof(left%), addrof(top%), addrof(right%),
addrof(bottom%))
187. w32GridGetRow(grid%, addrof (row%))
188. w32GridGetText(grid%, row%, col%, addrof(text$))
189. w32GridInsertColumn(grid%, position%)
190. w32GridInsertItem(grid%, position%)
191. w32GridIsHighlighted(grid%, addrof(yes_no%))
192. w32GridProtectCell(grid%, row%, col%, protect%)
193. w32GridSetAllowColResize(grid%, redim%)
194. w32GridSetBkColor(grid%, r%, g%, b%)
195. w32GridSetCellBkColor(grid%,row%,col%,r%,g%,v%)
196. w32GridSetColMaxChars(grid%,column%,max_char%)
197. w32GridSetColsNumbered(grid%, auto_numbered%)
198. w32GridSetColWidth(grid%, col%,width%)
199. w32GridSetCursorColor(grid%,r%,g%,b%)
200. w32GridSetCursorPos(grid%,row%, column%)
201. w32GridSetDim(grid%,num_rows%, num_columns%)
202. w32GridSetEditable(grid%, editable%)
203. w32GridSetEnterMode (grid%, mode%)
204. w32GridSetSelColor (grid%, r%, g%,b%)
205. w32GridSetSelTextColor (grid%, r%, g%, b%)
206. w32GridSetText(grid%, row%, column%, text$)
207. w32GridSetTitleFont(grid%, font$, points%, orientation%, bold%,
italic%,inderline%,strike_through%)
208. w32GridSetTitleHeight(grid%,height%)
209. w32GridShowHilight(grid%,yes_no%)
210. w32EditENTEREx(index%,hand%,ID%,addrof(key%))
211.
w32ListGetEventsExn(parent%,lis%,ID%,addrof(event%),addrof(param1%),addrof(param2%)
)
212. w32TreeGetEventsExn(parent%,tree%,ID%,addrof(etree%),addrof(ptree%))
213. w32TreeSetImage(tree%,ref%,image_index%,addrof(ret%))

```

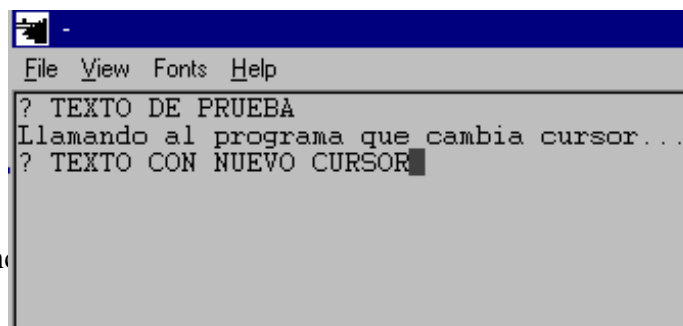
1. **w32CreateCaret(nWidth%,nHeight%,addrof(retcode%))**

This function creates a new shape for the system caret and assigns ownership of the caret to the specified window.

NWidth%, caret width in logical units

NHeight%, caret height in logical units

If the function succeeds, the *retcode%* value is nonzero



Sample W32 BASIC Program: **cc.b**

2. **w32ExitWindows (code%,addrof(retcode%))**

either logs off, shuts down, or shuts down and restarts the system.

Code% values:

0 logs the user off

1 Shuts down the system to a point at which it is safe to turn off the power

2 Shuts down the system and then restarts the system

5 Shuts down the system .Forces processes to terminate

6 Shuts down the system and then restarts the system. Forces processes to terminate

If the function succeeds, the *retcode%* value is nonzero

Sample W32 BASIC Program: **ew.b**

3. **w32PlaySound(filename\$,addrof(retcode%))**

This function plays a sound specified by the given *filename\$*

filename\$, the name is interpreted as a audio filename

If the function succeeds, the *retcode%* value is nonzero

Sample W32 BASIC Program: **ps.b**

4. **w32MessageBox(message\$,title\$,style%,addrof(retcode%))**

function creates, displays, and operates a message box. The message box contains an application-defined message and title, plus any combination of predefined icons and push buttons

message\$, the message to be displayed

title\$, the dialog box title

style\$, determine the contents and behavior of the dialog box. This parameter can be a sum of values of flags from the following groups.

- 0 The message box contains one push button: OK. This is the default
- 1 The message box contains two push buttons: OK and Cancel
- 2 The message box contains three push buttons: Abort, Retry, and Ignore
- 3 The message box contains three push buttons: Yes, No, and Cancel
- 4 The message box contains two push buttons: Yes and No
- 5 The message box contains two push buttons: Retry and Cancel

- 16 A stop-sign icon appears in the message box
- 32 A question-mark icon appears in the message box.
- 48 An exclamation-point icon appears in the message box
- 64 An icon consisting of a lowercase letter *i* in a circle appears in the message box

- 0 The first button is the default button.
- 256 The second button is the default button
- 512 The third button is the default button

- 0 The user must respond to the message box before continuing work in the application window . However, the user can move to the windows of other applications and work in those windows
- 4096 Same as the previous one except that the message box is the top most of the windows stack .

The return value is zero if there is not enough memory to create the message box.
 If the function succeeds, the return value is one of the following menu-item values returned by the dialog box:

- 1 OK button was selected
- 2 Cancel button was selected
- 3 Abort button was selected
- 4 Retry button was selected
- 5 Ignore button was selected
- 6 Yes button was selected
- 7 No button was selected

Here is a sample program:

```
call cForceExit
call cAppTitle("Mi aplicacion")

arti$="00214020"
titulo$="Registro bloqueado"
linea1$="El articulo "&arti$&" no se puede actualizar"
linea2$="puesto que otro usuario lo esta modificando."
linea3$="Indique el procedimiento a seguir:"
linea4$="* ANULAR, para cancelar la actualizacion."
```

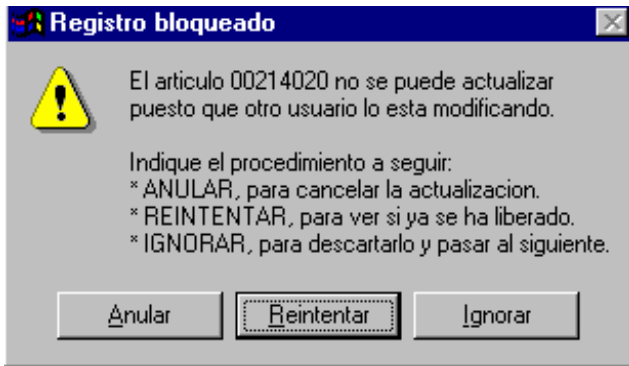
```

linea5$="* REINTENTAR, para ver si ya se ha liberado."
linea6$="* IGNORAR, para descartarlo y pasar al siguiente."

texto$=linea1$&chr$(10)&linea2$&chr$(10)&chr$(10)&linea3$&chr$(10)&linea4$&chr$(10)
&linea5$&chr$(10)&linea6$
call w32MessageBox(texto$,titulo$,4402,addrof(retcode%))

```

The result is the following:



The number 4402 is the sum of 2, 48, 256 and 4096 .

Sample W32 BASIC Program: *mb.b*

5. **w32SetWindowText(window\$,title\$,addrof(retcode%))**

search for the last opened window which title be equal to the text specified in the `window$` parameter, and then change the title of that window to the value of the `title$` parameter.

If the function succeeds, the *retcode%* value is nonzero

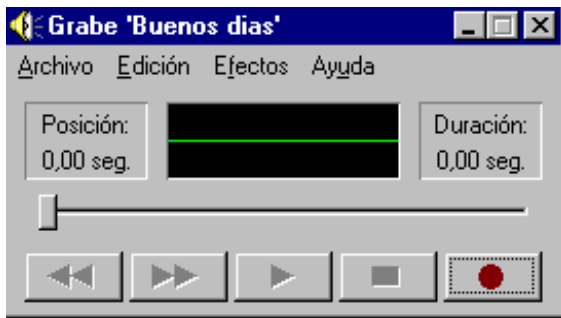
Here is an example:

```

call cForceExit
call cWinExec("sndrec32",1)

call w32SetWindowText("Sound - Sound Recorder","Grabe 'Buenos
dias' ",addrof(retcode%))
wait

```



Sample W32 BASIC Program: **swt.b**

6. **w32GetWindowSessions(window\$,addrof(retcode%))**

return the number% of opened window which title be equal to the text specified in the window\$ parameter.

```
call cAppTitle("Mi ventana")
call w32GetWindowSessions("Mi ventana",addrof(numero%))
print numero%
if numero% <5
then
call cWinExec(".\gws",1)
ifend
wait
```

Sample W32 BASIC Program: **gws.b**

7. **w32LBoxResetContent()**

remove all items from a list box.

Focus must be placed over the list box control. (Use cSetFocus function for this purpose).

Sample W32 BASIC Program: **lbrc.prj**

8. **w32LBoxDir(directories\$,addrof(retorno%))**

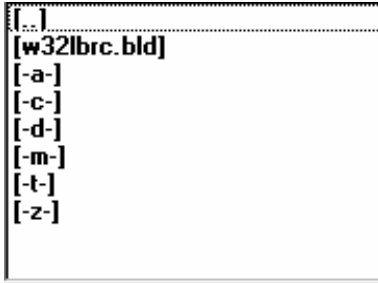
add a list of folders and drives to a list box.

Focus must be placed over the list box control. (Use cSetFocus function for this purpose).

Directories\$, specifies the directories to add to the list. If it contains wildcards (for example, *.*), all files that match the wildcards are added to the list.

Subdirectory names are showed enclosed in square brackets ([])
 Drives are listed in the form [-x-], where *x* is the drive letter.

The return value is the zero-based index of the last item added to the list.



Sample W32 BASIC Program: **lbrc.prj**

9. **w32BLoadImage(filename\$,addrof(retcode%))**

associate a new image (bitmap) with the button over which the focus is placed.

Filename\$, the path of the .bmp file.

Sample W32 BASIC Program: **lbrc.prj**

10. **w32BFLoadImage(fichero\$,addrof(retorno%))**

associate a new image (bitmap) with the button over which the focus is placed. Besides, the button becomes a flat button.



Sample W32 BASIC Program: **lbrc.prj**

11. **w32ClipCopyData(text\$,addrof(retcode%))**

This function places the string *text\$* on the clipboard in TEXT format. If the function succeeds, the *retcode%* value is nonzero.

12. **w32ClipsAvailable(addrof(result%))**

This function determines whether the clipboard contains data in TEXT format. *Result%*: Nonzero indicates that the clipboard format is available.

13. **w32ClipPasteData(addrof(text\$),addrof(retcode%))**

This function retrieves data from the clipboard and place the information on *text\$*. If the function succeeds, the *retcode%* value is nonzero

14. **w32CopyFile(source\$,target\$,addrof(error\$),addrof(retcode%))**

This function copies an existing file to a new file. If the *target\$* file already exists, the function overwrites the existing file and succeeds, that is return a nonzero value. If an error occurs the error message is returned in the *error\$* parameter.

```
call w32CopyFile("c:\windows\w32app.w32",
"\omega\publico\w32app.w32",addrof(error$),addrof(retcode%))
```

15. **w32CreateDirectory(dir\$,addrof(retcode%))**

This function creates a new directory.
Retcode%: Zero indicates failure.

16. **w32EditUndo()**

Undo the last edit control operation. Focus must be placed over the edit control. (Use `cSetFocus` function for this purpose).

17. **w32FileSize(file\$,addrof(size%),addrof(retcode%))**

Returns the *size%* in bytes of the specified *file\$*.
Retcode%: Zero indicates failure.

18. **w32FileTime(file\$,addrof(year%),addrof(month%),addrof(day%), addrof(hour%),addrof(min%),addrof(sec%),addrof(retcode%))**

Returns the time of the last write in the *file\$*.
Retcode%: Zero indicates failure.

19. **w32FindFirstFile(pattern\$,addrof(firstfile\$),addrof(handle%))**

The **w32FindFirstFile** function opens a search *handle%* and returns the name of the *firstfile\$* whose name matches the specified *pattern\$*. Once the search *handle%* is established, you can use the **w32FindNextFile** function to search for other files that match the same *pattern\$*.

20. **w32FindNextFile(handle%,addrof(nextfile\$),addrof(retcode%))**

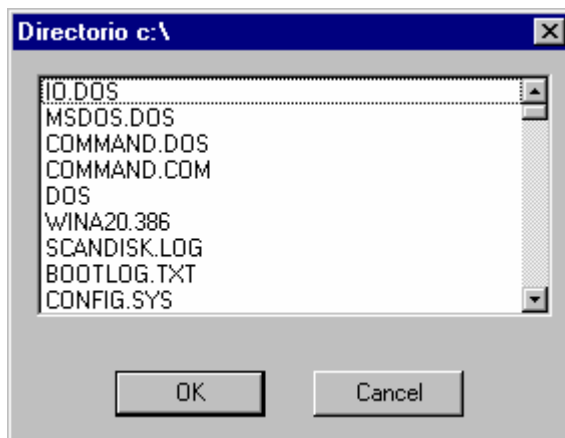
This function continues a file search from a previous call to the **w32FindFirstFile** function.
Handle%: Search handle returned by a previous call to the **w32FindFirstFile** function.
NextFile\$: The name of the found file or directory.
Retcode%: Zero indicates failure or that no more files can be found.

```
dim matriz$(500)
indice=1
call w32FindFirstFile("c:\*.*",addrof(fichero$),addrof(handle%))
if handle%<>0
```

```

then
    matriz$(indice)=fichero$
    call
w32FindNextFile(handle%,addrof(fichero$),addrof(retorno%))
    while retorno%<>0
        indice=indice+1
        matriz$(indice)=fichero$
        call
w32FindNextFile(handle%,addrof(fichero$),addrof(retorno%))
    wend
ifend
call cChoiceListNS("Directorio c:\",mataddrof(matriz$),addrof(sel$))

```



21. w32GetComputerName(addrof(name\$))

Retrieve the computer name.

22. w32GetDiskFreeSpace(drive\$,addrof(bytesfree%),addrof(retcode%))

Returns how much free space there is on a specified disk drive.
Retcode%: Zero indicates failure.

23. w32GetWindowsDirectory(addrof(path\$))

retrieve the path of the Windows directory.

24. w32GetDriveType(drive\$,addrof(retcode%))

Returns the type of the *drive\$*.

Retcode% can be:

0	ERROR
1	DRIVE DOES NOT EXIST, OR THERE IS NO MEDIA INSERTED
2	DRIVE REMOVABLE
3	DRIVE FIXED
4	DRIVE REMOTE
5	DRIVE CDROM
6	DRIVE RAMDISK

```
call cOpenBrowser(addrof(nombre$),"Imágenes para Botones
(*.bmp)|*.bmp|")
if sch(0,nombre$,"\")<>0
then
    for i=1 to len(nombre$)
        if nombre$[i:i]="\" then break
    next i
    call w32GetDriveType(nombre$[1:i],addrof(retorno%))
    SELECT retorno%
    CASE 2
        MsgBox "No se permiten las unidades removibles"
    CASE 3
        rem call cSetFocus...
        rem call w32BLoadImage(nombre$,addrof(valor%))
    CASE 4
        MsgBox "No se permiten las unidades de red"
    CASE 5
        MsgBox "No se permiten las unidades de CD-ROM"
    CASE 6
        MsgBox "No se permiten los discos RAM"

    CEND
endifend
```

25. w32keybd_event(char\$,char%)

This function may be used to put a character into the input buffer just as the key had been entered by the operator.

Char\$: The character to be entered.

Char%: The ASCII value of the character.

26. **w32Skeybd_event_down(key%)**

This function may be used to simulate a key press.

Key%: One of the following values:

BACKSPACE	8
TAB	9
RETURN	13
SHIFT	16
CONTROL	17
ALT	18
PAUSE	19
CAPITAL	20
ESCAPE	27
SPACE	32
PRIOR	33
NEXT	34
END	35
HOME	36
LEFT	37
UP	38
RIGHT	39
DOWN	40
INSERT	45
DELETE	46
'0' thru '9'	(48 - 57)
'A' thru 'Z'	(65 - 90)
Function Keys F1 thru F12	(112 - 123)

27. **w32Skeybd_event_up(key%)**

This function may be used to simulate a key release.

28. **w32MoveWindow(title\$,x%,y%,width%,height%,addrof(retcode%))**

Allows to change the position and size of the last opened window which title be equal to the text specified in the title\$ parameter.

Retcode%: Zero indicates failure.

29. **w32RemoveDirectory(dir\$,addrof(retcode%))**

This function remove the directory specified in the dir\$ parameter. The directory must be empty.

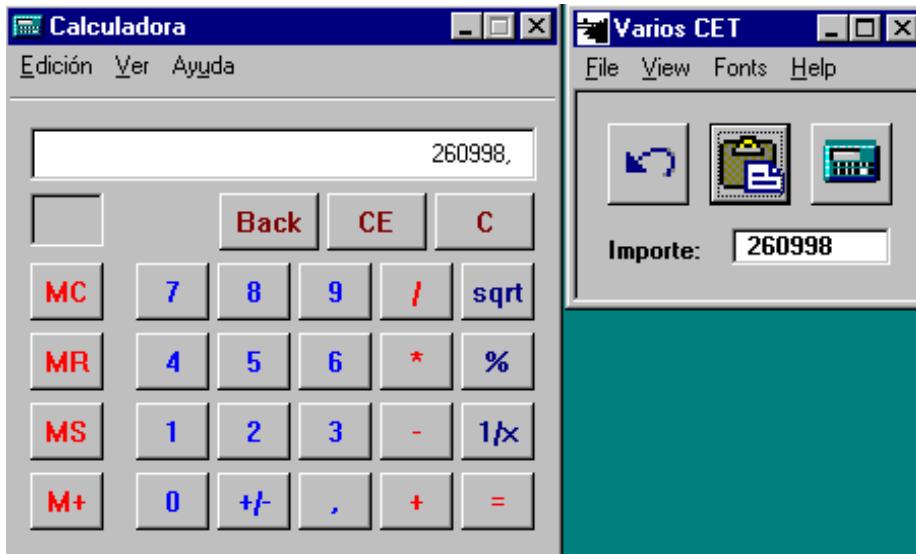
Retcode%: Zero indicates failure.

30. **w32SetForegroundWindow(title\$,addrof(retocod%))**

Set the last opened window which title be equal to the text specified in the title\$ parameter as the foreground window.

Retcode%: Zero indicates failure.

Here is a sample of the program **cal.prj** that shows the use of several functions.



```
$SUB dlg1BtnClick
REM      Button Click save area
REM@# btnstart
SELECT CMDID%
CASE 202
    Rem Undo- Boton Deshacer
    call cSetFocus(dlg1HDLG%,100)
    call w32EditUndo()
CASE 205
    Rem Paste- Boton Pegar
    call w32GetWindowSessions("Calculator",addrof(sesiones%))
    if sesiones%>0
    then
    call w32SetForegroundWindow("Calculator",addrof(retorno%))
    Rem - Simular CONTROL+C
    call w32Skeybd_event_down(17)
    call w32keybd_event("C",asc("C"))
    call w32Skeybd_event_up(17)
    call cAppWindowTop
    call w32ClipIsAvailable(addrof(ava%))
    if ava%<>0
    then
        call w32ClipPasteData(addrof(text$),addrof(retcode%))
        call cSetCtrlValue(dlg1HDLG%,100,text$)
    ifend
ifend
```

CASE 206

```
call w32GetWindowSessions("Calculator",addrof(sesiones%))
if sesiones%=0
then
    call cWinExec("calc.exe",1)
ifend
call cAppWindowTop
call w32MoveWindow("Calculator",0,0,276,272,addrof(retorno%))
call w32SetForegroundWindow("Calculator",addrof(retorno%))
Rem -Ver la calculadora Estandard
call w32Skeybd_event_down(18) \rem Pulsar ALT
call w32keybd_event("V",asc("V"))
call w32Skeybd_event_up(18) \rem Liberar ALT
call w32keybd_event("E",asc("E"))
```

CEND

REM@# btnend

GOSUB dlg1.BTNEND

\$EXIT

31. w32BMultiline()

Allow to show several text lines over the button in which the focus is placed.

32. w32B3D()

Set the state of the button in which the focus is placed as not pushed.

33. w32BFlat()

Set the state of the button in which the focus is placed as pushed.

34. w32GetDiskSpace (unit\$,addrof(space%),addrof(ret%))

Determine how much disk space a drive has.

Ret%: Zero indicates failure.

35. w32SetFont(font\$,points%,angle%,bold%,italics%,underline%,strike_out%)

Specify the font that the control in which the focus is placed is to use when drawing text.

Angle%.- Angle, in degrees, counterclockwise from the x-axis of the baseline of the character

36. **w32CreateProgress(x%,y%,width%,height%,smooth%,vertical%,addrof(hand%))**

Creates a progress bar control inside the window over which the focus be placed .

37. **w32ProgressSetRange(hand%,min%,max%)**

Sets the minimum and maximum values for a progress bar and redraws the bar to reflect the new range.

38. **w32ProgressSetStep(hand%,step%)**

Specifies the step increment for a progress bar. The step increment is the amount by which the progress bar increases its current position whenever the function w32ProgressStepIt is called. By default, the step increment is set to 10.

39. **w32ProgressStepIt(hand%)**

Advances the current position for a progress bar by the step increment and redraws the bar to reflect the new position.

When the position exceeds the maximum range value, this function resets the current position so that the progress indicator starts over again from the beginning.

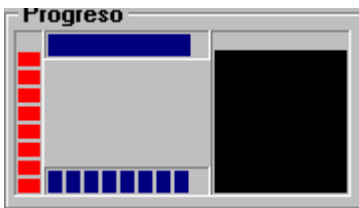
40. **w32ProgressSetColor(hand%,r%,g%,b%)**

Sets the color of the progress indicator bar in the progress bar control.

r % – r(ed) component value [0..255]

g% – g(reen) component value [0..255]

b% – b(lue) component value [0..255]



Sample W32 BASIC Program: Create.prj

41. **w32CreateCalendar(x%,y%,width%,height%,addrof(cal%))**

Creates a month calendar control inside the window over which the focus be placed



Example: Create.prj

42. w32CalendarGetSel(cal%,addrof(year%),addrof(month%),addrof(day%))

Retrieves the currently selected date.

43. w32CreateList(x%,y%,width%,height%,addrof(lis%))

Creates a list view control inside the window over which the focus be placed

44. w32ListInsertColumn(lis%,title\$,width%,alig%,index%,addrof(ret%))

Inserts a new column in a list view control

Title\$ -Title of the column showed in the list header

width% - Width of the column in pixels.

Alig% - Text alignment

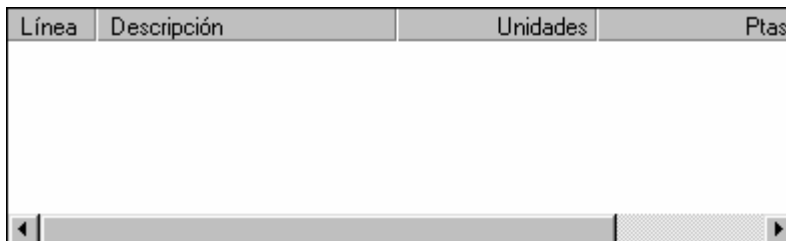
0.- Text is left-aligned.

1.- Text is centered

2.- Text is right-aligned

index% - Index of the new column

Ret% - Returns the index of the new column if successful, or -1 otherwise



Sample W32 BASIC Program: *Create.prj*

45. w32ListInsertItem(lis%,text\$,item%,addrof(resu%))

Inserts a new item in a list view control.

Returns the index of the new item if successful, or -1 otherwise.

46. w32ListSetText(lis%,text\$,item%,subitem%,addrof(resu%))

Changes the text of a list view subitem.

If the function succeeds, the *resu%* value is nonzero

47. w32ListGetText(lis%,item%,subitem%,addrof(text\$),addrof(ret%))

Retrieves the text of a list view item or subitem

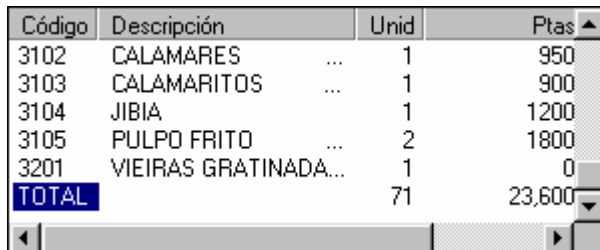
48. w32ListGetSelItem(lis%,addrof(item%))

Retrieves the index of the selected item.

49. w32ListSetSelItem(lis%,item%,addrof(ret%))

Sets the item #*item%* of the list as selected

If the function succeeds, the *ret%* value is nonzero



Código	Descripción	Unid	Ptas
3102	CALAMARES ...	1	950
3103	CALAMARITOS ...	1	900
3104	JIBIA	1	1200
3105	PULPO FRITO ...	2	1800
3201	VIEIRAS GRATINADA...	1	0
TOTAL		71	23,600

50. w32ListGetItemCount(lis%,addrof(count%))

Retrieves the number of items in a list view control

51. w32ListFindItem(lis%,key\$,addrof(item%))

Searches for a list view item which #0 field value be *key\$*.

Returns the index of the item if successful, or -1 otherwise

52. **w32ListSetColor(lis%,r%,g%,b%)**

Sets the color of the text of the list view control

r % – r(ed) component value [0..255]

g% – g(reen) component value [0..255]

b% – b(lue) component value [0..255]

53. **w32ListDeleteItem(lis%,item%,addrof(ret%))**

Removes an item from a list view control.

Item%

Index of the list view item to delete.

If the function succeeds, the *ret%* value is nonzero

54. **w32ListDeleteAllItems(lis%,addrof(ret%))**

Removes all items from a list view control

If the function succeeds, the *ret%* value is nonzero

55. **w32ListDeleteColumn(lis%,column%,addrof(ret%))**

Removes a column from a list view control.

Column%

Index of the column to delete

If the function succeeds, the *ret%* value is nonzero

56. **w32SetFocus(hand%,addrof(ret%))**

The function sets the keyboard focus to the specified window.

Hand%.- handle to window to receive focus. If this parameter is NULL, keystrokes are ignored.

If the function succeeds, the return value is the handle to the window that previously had the keyboard focus.

57. **w32CreateAnimation(x%,y%,width%,height%,addrof(ani%))**

Creates a animation control inside the window over which the focus be placed

58. **w32AnimationOpen(ani%,file\$,addrof(ret%))**

Opens an AVI *file\$* and displays its first frame in an animation control

You can only open silent AVI clips. W32AnimationOpen fails if *file\$* specifies an AVI clip that contains sound.

Returns nonzero if successful, or zero otherwise.

59. **w32AnimationPlay(ani%,addrof(ret%))**

Plays an AVI clip in an animation control. The control plays the clip in the background while the thread continues executing.



Returns nonzero if successful, or zero otherwise.

Sample W32 BASIC Program: Create.prj

60. **w32AnimationStop(ani%,addrof(ret%))**

Stops playing an AVI clip in an animation control

Returns nonzero if successful, or zero otherwise.

61. **w32MAPISendDocuments(FullPaths\$,filenames\$,addrof(ret%))**

The w32MAPISendDocuments function sends a standard mail message with one or more attached files and a cover note. The cover note is a dialog box that allows the user to enter a list of recipients and an optional message.

FullPaths\$: This list is formed by concatenating correctly formed file paths separated by the character ‘;’. An example of a valid list is:

```
C:\TMP\TEMP1.DOC;C:\TMP\TEMP2.DOC
```

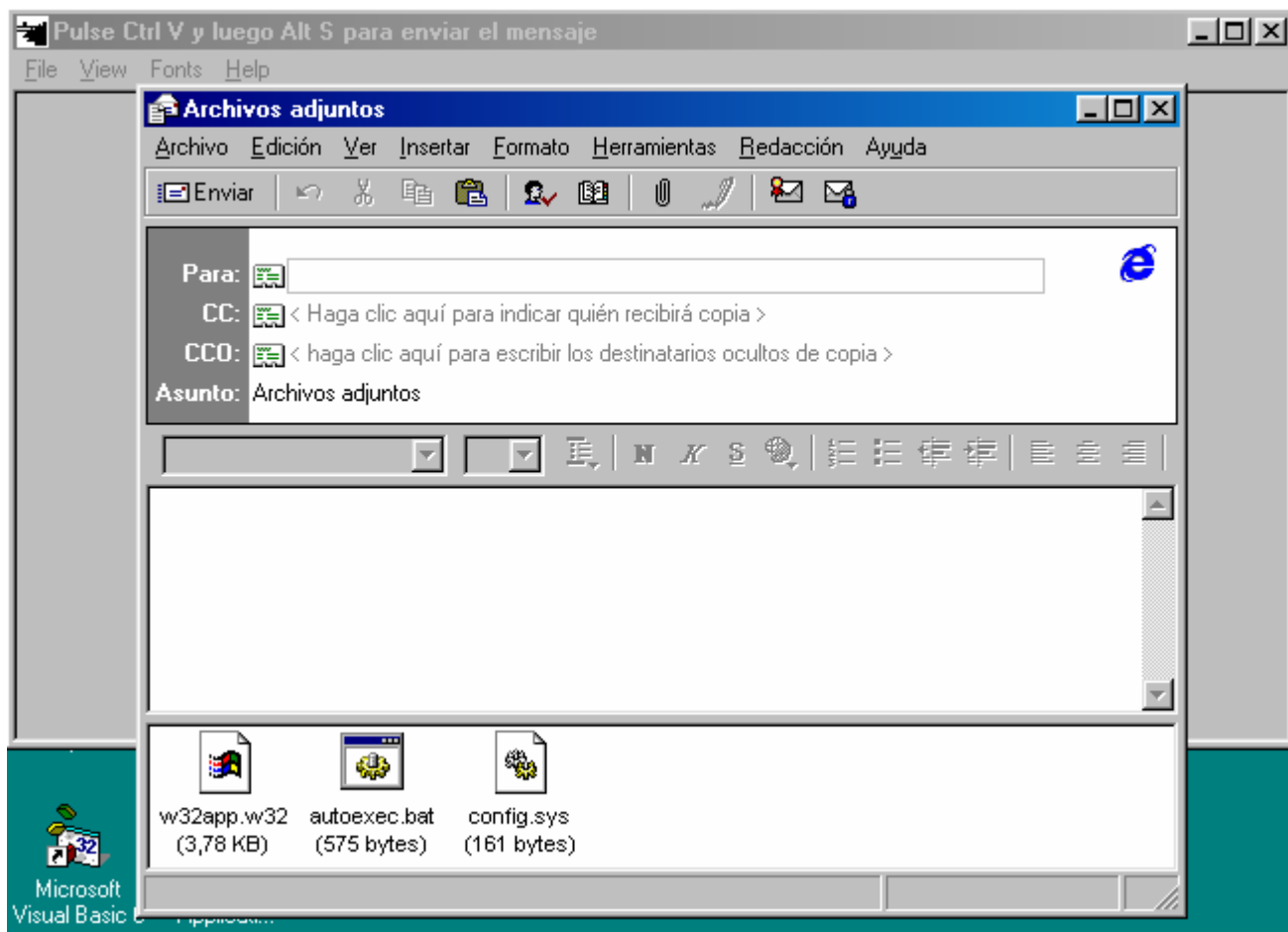
The files specified in this parameter are added to the message as file attachments. If this parameter contains an empty string, the Send Note dialog box is displayed with no attached files.

FileNames\$: List of the original filenames as they should appear in the message. When multiple names are specified, the list is formed by concatenating the filenames separated by the character ‘:’. An example is:

```
TEMP3.DOC;TEMP4.DOC
```

Sample W32 BASIC Program:

```
call
w32MAPISendDocuments ("c:\windows\w32app.w32;c:\autoexec.bat;c:\config
.sys" , "w32app.w32;autoexec.bat;config.sys" , addrof(ret%))
```



62. w32MAPIReadMail(directory\$,originator\$,addrof(files%),addrof(mes%))

Process all messages found in the 'Inbox' folder in which originator field occurs the text specified in the originator\$ parameter, file attachments are saved to folder specified in the *directory\$* paramater, and finally deletes the message.

files% returns the number of attached files that have been saved.

mes% returns the number of processed messages, or -1 in error case.

Sample W32 BASIC Program:

```
call
w32MAPIReadMail ("c:\cet\mail\" , "@activanet.es" , addrof(files%) , addrof(
ret%))
```

63. w32DestroyWindow(handle%)

Destroy the control created by the call to the function **w32Create** [Calendar|Progress|List|Animation|...]

Sample W32 BASIC Program:


```
print at$(1,1);"Backup..."
call w32CreateAnimation(1,15,150,50,addrof(ani%))
call w32AnimationOpen(ani%,".\file.avi",addrof(ret%))
call w32AnimationPlay(ani%,addrof(ret%))
wait
call w32AnimationStop(ani%,addrof(ret%))
wait
call w32DestroyWindow(ani%)
print at$(1,1);"End Backup"
wait
```

64. w32ListEnsureVisible(lis%,item%,addrof(ret%))

Call this function to ensure that a list view *item%* is fully visible. The list view control is scrolled if necessary.

65. w32ListGridLines(lis%,addrof(ret%))

Displays gridlines around the items and subitems of the list view control.



Cod.	Descripcion
02	N. COMENSAL
03	CAFES
04	CAVAS
05	CERVEZA
06	BRANDY
07	GINEBRAS
08	COMENTAR
09	LICORES
10	RON
11	VINOS TINTOS

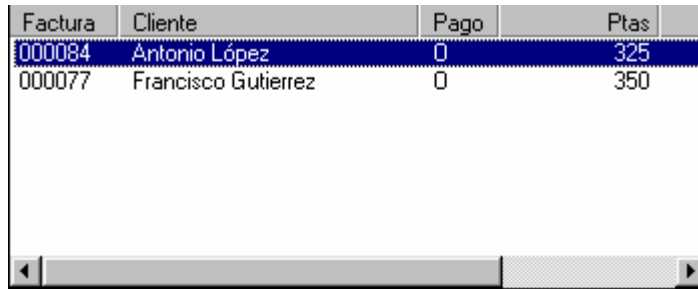
66. w32ListClearGridLines(lis%,addrof(ret%))

67. w32CalendarSetSel(cal%,year%,month%,day%)

Sets the currently selected date for the month calendar control *cal%* created with **w32CreateCalendar**. A four digit number must be used as *year%* parameter.

68. w32ListFullRowSel(lis%,addrof(return%))

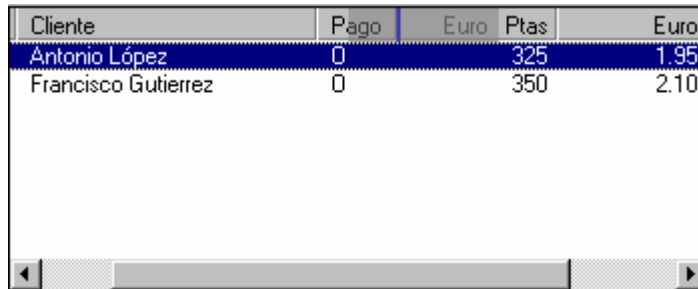
When an item is selected, the item and all its subitems are highlighted.



Factura	Cliente	Pago	Ptas
000084	Antonio López	0	325
000077	Francisco Gutierrez	0	350

69. w32ListDragDrop(lis%,addrof(return%))

Enables drag-and-drop reordering of columns in a list view control.



Cliente	Pago	Euro	Ptas	Euro
Antonio López	0	325		1.95
Francisco Gutierrez	0	350		2.10

70. w32CreateTip(text\$,addrof(tip%))

Create a small pop-up window that displays several lines of *text* appearing only when the user puts the cursor on the control and leaves it there for approximately one-half second.

Just before calling this function the focus must be placed over the desired control. (Use `cSetFocus` function for this purpose).

This function returns a reference for the Tip. This is the value that must be used if an update of the *text* is needed. (See `w32TipUpdateText` function)



71. w32TipUpdateText(tip%,newtext\$,addrof(retorn%))

Update the text of the Tip Control *tip%*.

Just before calling this function the focus must be placed over the desired control. (Use `cSetFocus` function for this purpose).

72. w32GetParent(addrOf(parent%))

Returns a reference to the parent windows of the control over which the focus is placed.
This function can be used to get a reference to the Dialog when the focus is over a control.

73. w32ResizeDialog(parent%,x%,y%, width%,height%,addrOf(return%))

This routine may be used to specify a new location and size for a Dialog Box.
The first parameter must be the value returned by the function w32GetParent.

74. w32ListSetBkColor(lis%,r%,g%,v%)

Sets the background color of text in a list view control.

r % – r(ed) component value [0..255]

g% – g(reen) component value [0..255]

b% – b(lue) component value [0..255]

75. w32EnableWindow(hand%,state%)

Enables (state%=1) or disables (state%=0) a control.

The first parameter must be the value returned by the function **w32Create**
[Calendar|Progress|List|Animation|...]

A disabled control is grayed out and cannot obtain the input focus.

76. w32FtpGet(ftpSite\$,user\$,password\$,ftpfile\$,localfile\$,addrOf(return%))

Downloads a file from a ftp site and save it as a local file.

The file cetftp.dll is required.

The return value can be:

- 1 Success.
- 1 Error when connecting to the ftp site.
- 2 Error when saving the file.

Sample W32 BASIC Program:

```
call  
w32FtpGet("ftp.microsoft.com","omega","pass","/usr/www/users/omega/license.txt","c:\microsoft\lice  
nse.txt",addrOf(return%))
```

77. w32FtpPut(ftpSite\$,user\$,password\$,ftpfile\$,localfile\$,addrOf(return%))

Sends a local file to a ftp site. The file cetftp.dll is required.

The return value can be:

- 1 Success.
- 1 Error when connecting to the ftp site.
- 2 Error when saving the file.

78. w32PrinterDlg(addrOf(context%),addrOf(printer\$),addrOf(o%),addrOf(copies%))

Shows a dialog box that allows the user to select a printer and to specify printer options such as number of copies and orientation. The cetprint.dll file is required.

- context%*, Reference to the printer object.
- printer\$*, name of the printer selected by the user.
- o%*, orientation (1-Portrait, 2-Landscape)
- copies%*, number of copies.

Sample W32 BASIC Program: prt.b

79. w32PrinterContext(printer\$,addrOf(context%))

Creates a reference to the printer *printer\$*. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

80. w32PrinterDeleteContext(context%)

Destroy the printer reference. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

81. w32PrinterOffsetX(context%,addrOf(offsetx%))

Returns the distance from the left edge of the physical page to the left edge of the printable area of the printer associated with *context%*, in pixels. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

82. w32PrinterOffsetY(context%,addrOf(offsety%))

Returns the distance from the top edge of the physical page to the top edge of the printable area of the printer associated with *context%*, in pixels. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

83. w32PrinterWidth(context%,addrOf(totalwidth%))

Returns the width of the physical page, in pixels. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

84. w32PrinterHeight(context%,addrof(totalheight%))

Returns the height of the physical page, in device pixels. The cetprint.dll file is required.

Sample W32 BASIC Program: prt.b

85. w32ListSortItems(lis%,column%,alphanum%,order%,addrof(ret%))

Sorts the items of a list view control. The index of each item changes to reflect the new sequence.

lis%, Reference to the List View Control

column%, number of column that states the relative order of list items

alphanum%, 0-alphabetic order, 2-numerical order

order%, 1-ascending, 2-descending.

86. w32CreateUpDown(min%,max%,addrof(updown%))

Create an *up-down control* consisting of a pair of arrow buttons that the user can click to increment or decrement an edit control's value.



Just before calling this function the focus must be placed over the desired edit control. (Use cSetFocus function for this purpose).

The parameters *min%* and *max%* set the minimum and maximum positions (range) for an up-down control.

The maximum position can be less than the minimum position. Clicking the up arrow button moves the current position closer to the maximum position, and clicking the down arrow button moves towards the minimum position.

Sample W32 BASIC Program:

```
call cSetFocus(dlg2HDLG%,100)
call w32CreateUpDown(1,10,addrof(updown%))
```

87. w32ListCheckBoxes(lis%,addrof(return%))

Enables check boxes for items in a list view control

Nombre	Descripción	Modificado	Tamaño	Registros
<input checked="" type="checkbox"/> FAMI	FAMILIA ARTICULOS	27/06/2000 19:08	2 KB	35
<input type="checkbox"/> LaPR	LINEA ALBARAN PR...	28/06/2000 12:12	2 KB	7
<input type="checkbox"/> LPPR	LINEA PEDIDO PRO...	27/06/2000 19:08	2 KB	29
<input checked="" type="checkbox"/> LTIC	LINEA TICKETS	28/06/2000 12:19	26 KB	794
<input checked="" type="checkbox"/> MESA	MESAS	27/06/2000 18:54	5 KB	306
<input type="checkbox"/> PARA	PARAMETROS	28/06/2000 12:12	2 KB	9
<input type="checkbox"/> PART	PROVEEDOR/ARTI...	27/06/2000 19:08	2 KB	7

88. **w32ListGetCheck(lis%,item%,addrof(sel%))**

Determine the state of the check box for a given *item%*.
Returns nonzero if the given item is selected, or zero otherwise.

89. **w32ListSetCheck(lis%,item%,state%,addrof(retcode%))**

Sets the state of the item checkbox.
State%, 0-unchecked, 1-checked

90. **w32GetFont(addrof(font\$),addrof(points%),addrof(angle%),addrof(bold%),addrof(italics%),addrof(underline%),addrof(strike_out%))**

The routine *w32GetFont* retrieve the font values of the control over which the focus is placed.

(See function *w32SetFont* to get an explanation of the parameters)

91. **w32GetWindow(parent%,ID%,addrof(hand%))**

Returns a handle to a dialog box control. The first parameter must be the value returned by the function *w32GetParent*. The second parameter must be the ID of the control.

92. **w32SetFontH(font\$,points%,angle%,bold%,italics%,underline%,strike_out%)**

Specify the font that the control referenced by *hand%* is to use when drawing text.

Sample W32 BASIC Program:

```

$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Rem Set the focus over a control to get the dialog handle

```



```

Call cSetFocus(dlg1HDLG%,203)

call w32GetParent(addrOf(padre%))

Rem Get the handle of the control 100
Call w32GetWindow(padre%,100,addrOf(ed100%))

Rem Set the font
Call w32SetFontH(ed100%,"Tahoma",13,0,0,0,0,0)

REM@# iniend

```

93. **w32ResizeWindowH(hand%,x%,y%, width%,height%,addrOf(retorno%))**

This routine may be used to specify a new location and size for the control referenced by *hand%*.

94. **w32CreateImageH(hand%,file\$,x%,y%, width%,height%,addrOf(ret%))**

Allows to show an image (bitmap) in the dialog/control referenced by *hand%*.
File\$, the path of the .bmp file.

Sample W32 BASIC Program:

```

$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Rem Set the focus over a control to get the dialog handle
Call cSetFocus(dlg1HDLG%,200)

call w32GetParent(addrOf(padre%))

...

call w32CreateImageH(padre%,"logo.bmp",15,10,270,135,addrOf(ret%))

REM@# iniend

```

95. **w32EditENTERn(hand%,IDbuttonESC%)**

Where *n* is a value between 1 and 9. Calling this group of functions allows the user to move the focus to the next control by pressing ENTER key.

This special behaviour only affects to the control referenced by the first parameter.

Hand%, reference to the control.

IDbuttonESC%, value between 200 and 299 that indicates the ID of the button (real or dummy) which code in *BtnClick Subroutine* will be executed each time the user press ESC key.

In all of the dialog boxes that make up a .exe program, it must be used a different function name for each control. So, to associate this behaviour to three controls the functions to use would be ...

Call w32EditENTER1...

Call w32EditENTER2...

Call w32EditENTER3...

Sample W32 BASIC Program:

```
$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Call cSetFocus(dlg1HDLG%,200)

call w32GetParent(addrOf(padre%))
...

Call w32GetWindow(padre%,100,addrOf(ed100%))
Call w32GetWindow(padre%,101,addrOf(ed101%))

...

call w32EditENTER1(ed100%,201)
call w32EditENTER2(ed101%,201)
REM@# iniend
```

96. **w32SetCtrlColor***n*(padre%,control%,rf%,gf%,bf%,rb%,gb%,bb%)

Where *n* is a value between 1 and 9. Calling this group of functions allows to specify the foreground and background colors of Edit, List Box and Combo Box (Drop down list), text, radio, check and group controls.

This special behaviour only affects to the control referenced by the second parameter.

Parent%, handle to the Dialog Box. (Value returned by w32GetParent)

Control%, reference to the control.

rf % – r(ed) component value for foreground color. (-1 to use the default color)

gf% – g(reen) component value for foreground color.

bf% – b(lue) component value for foreground color.

rb % – r(ed) component value for background color. (-1 to use the default color)

gb% – g(reen) component value for background color.

bb% – b(lue) component value for background color.

In all of the dialog boxes that make up a .exe program, the same function must be called once only. So, to change the color properties of the controls three times the functions to use would be ...

Call w32SetCtrlColor1...

Call w32SetCtrlColor2...

Call w32SetCtrlColor3...



Sample W32 BASIC Program:

```

$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Call cSetFocus(dlg1HDLG%,200)

call w32GetParent(addrOf(padre%))
...

call w32GetWindow(padre%,701,addrOf(c701%)) \rem Caja Combo
call w32GetWindow(padre%,504,addrOf(c504%)) \rem Importe
call w32GetWindow(padre%,101,addrOf(c101%)) \rem Año
call w32GetWindow(padre%,800,addrOf(c800%)) \rem Facturas
call w32GetWindow(padre%,801,addrOf(c801%)) \rem presupuestos
call w32GetWindow(padre%,900,addrOf(c900%)) \rem Por fecha
call w32GetWindow(padre%,901,addrOf(c901%)) \rem Por importe
call w32GetWindow(padre%,300,addrOf(c300%)) \rem List Box

call w32SetCtrlColor1(padre%,c701%,255,0,0,-1,0,0)
call w32SetCtrlColor2(padre%,c504%,-1,0,0,255,255,0)
call w32SetCtrlColor3(padre%,c101%,128,0,0,-1,0,0)
call w32SetCtrlColor4(padre%,c800%,255,0,255,-1,0,0)
call w32SetCtrlColor5(padre%,c801%,128,0,128,-1,0,0)
call w32SetCtrlColor6(padre%,c900%,0,0,255,-1,0,0)
call w32SetCtrlColor7(padre%,c901%,0,0,128,-1,0,0)

```

```
call w32SetCtrlColor8(padre%,c300%,0,128,0,-1,0,0)
```

```
...
```

```
REM@# iniend
```

97. **w32CreateListH(dialog%,x%,y%,width%,height%,addrof(lis%))**

Creates a list view control in the Dialog Box referenced by *dialog%*. (Use the value returned by *w32GetParent* as the first parameter)

Use this function to create the control if it is necessary to trap events generated by list view controls. (See *w32ListGetEventsn*)

98. **w32ListGetEventsn(dialog%,lis%,button%,addrof(event%),addrof(param1%),addrof(param2%))**

Where *n* is a value between 1 and 9. Calling this group of functions allows to trap events generated by list view controls.

Dialog%, reference to the Dialog Box. (Value returned by *w32GetParent*).

lis%, reference to the list view control (Value returned by *w32CreateListH*).

button%, value between 200 and 299 that indicates the ID of the button (real or dummy) which code in *BtnClick Subroutine* will be executed each time an event occurs.

Event%, event number. (See table below)

Param1%, first property for the event. (See table below)

Param2%, second property for the event. (See table below)

<i>Event</i>	<i>Description</i>	<i>Property 1</i>	<i>Property 2</i>
1	The user clicks an item with the left mouse button	Item	Subitem
2	An item has been selected or checked	Item	-1
3	The user double-clicks an item with the left mouse button	Item	Subitem
4	The user clicks an item with the right mouse button	Item	Subitem
5	A key has been pressed	Code for the key (See <i>w32Skeybd_event_down</i>)	-1
6	a column was clicked	-1	Column

In all of the dialog boxes that make up a .exe program, the same function must be called once only. So, to define event processing for three list view controls the functions to use would be ...

Call w32ListGetEvents1...
Call w32ListGetEvents2...
Call w32ListGetEvents3...

Sample W32 BASIC Program:

```
$SUB dlg1BtnClick
REM      Button Click save area
REM@# btnstart
SELECT CMDID%
...
CASE 250
    Rem Notifications for list view lis%
    Rem evento% -> Event number
    Rem l_item% -> Property 1
    Rem l_sub% -> Property 2
    SELECT evento%
    CASE 1
        Rem Click
        rem MsgBox "Click: "&str$(l_item%)&","&str$(l_sub%)

    CASE 2
        Rem Item selected or checked
        rem MsgBox "Item changed: "&str$(l_item%)
        call w32ListGetSelItem(lis%,addrof(item%))
        if item%<>-1
        then
        call
w32ListGetText(lis%,item%,0,addrof(text$),addrof(ret%))
        call cSetCtrlValue(dlg1HDLG%,100,text$)
        ifend

    CASE 3
        Rem Double-Click
        MsgBox "Double-Click: "&str$(l_item%)&","&str$(l_sub%)

    CASE 4
        Rem Right mouse button click
        MsgBox "Right mouse button click:
"&str$(l_item%)&","&str$(l_sub%)

    CASE 5
        Rem A key has been pressed
        MsgBox "Key code: "&str$(l_item%)

    CASE 6
```

```

Rem Column click
Msgbox "Column: "&str$(l_sub%)
call w32ListSortItems(lis%,l_sub%,0,1,addrof(ret%))

```

```

CEND

```

```

...
CEND
REM@# btnend
GOSUB dlg1.BTNEND
$EXIT

```

```

$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Call cSetFocus(dlg1HDLG%,200)

call w32GetParent(addrof(padre%))
...
call w32CreateListH(padre%,10,195,475,180,addrof(lis%))
...
call
w32ListGetEvents1(padre%,lis%,250,addrof(evento%),addrof(l_item%),add
rof(l_sub%))
...
REM@# iniend

```

99. **w32EditENTER(index%,hand%,IDbutton%,addrof(key%))**

Calling this function allows the user to move the focus to the next control by pressing ENTER key. Besides, every key% the user press can be processed.

This special behaviour only affects to the control referenced by the first parameter.

Index%, an integer value between 1 and 99

Hand%, reference to the control.

IDbutton%, value between 200 and 299 that indicates the ID of the button (real or dummy) which code in *BtnClick Subroutine* will be executed each time the user press a key when the focus be over that control.

Key%, Code for the key that has been pressed. (See w32Skeybd_event_down function for the code list)

In all of the dialog boxes that make up a .exe program, it must be used a different index% value for each control. So, to associate this behaviour to three controls the syntax of the functions to use would be ...

```
Call w32EditENTER(1,...
Call w32EditENTER(2,...
Call w32EditENTER(3,...
```

Sample W32 BASIC Program:

```
$SUB dlg1BtnClick
REM      Button Click save area
REM@# btnstart
SELECT CMDID%
...
CASE 230
    SELECT tecla100%
        CASE 112
            Rem Handling F1 key pressing

        CEND

CASE 231
    SELECT tecla101%
        CASE 33
            Rem Pg. Down
        CASE 34
            Rem Pg. Up
    CEND
CEND
REM@# btnend
GOSUB dlg1.BTNEND
$EXIT
```

```
$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
Call cSetFocus(dlg1HDLG%,200)
call w32GetParent(addrOf(padre%))
...

Call w32GetWindow(padre%,100,addrOf(ed100%))
Call w32GetWindow(padre%,101,addrOf(ed101%))

...

call w32EditENTER(1,ed100%,230,addrOf(tecla100%))
call w32EditENTER(2,ed101%,231,addrOf(tecla101%))

REM@# iniend
```

100. **w32BrowseForFolder(Dialog%,title\$,addrof(dir\$),addrof(ret%))**

Creates a dialog box that allows the user to select a folder
Dialog%, reference to the Dialog Box. (Value returned by w32GetParent).
title\$, Help text to be showed in the dialog box.
dir\$, Path selected by the user
ret%, Return Value.(0 if Cancel button was pressed)

Sample W32 BASIC Program:

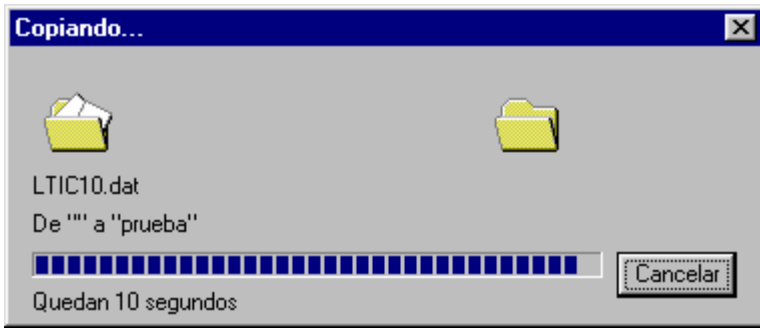
```
call w32BrowseForFolder(padre%,"Seleccione el directorio destino para la copia de seguridad",addrof(texto$),addrof(ret%))
```



101. **w32FileCopyProgress(Dialog%,from\$,to\$,addrof(ret%))**

Copies the files specified in the *from\$* parameter to the location specified in the *to\$* parameter displaying a progress dialog box.

Dialog%, reference to the Dialog Box. (Value returned by w32GetParent).
If the path specified in *from\$* ends with *.* a recursive copy will be make.
Ret%, Zero if successful, or nonzero otherwise



102. w32FileDeleteProgress(Dialog%,from\$,addrof(ret%))

103. w32FileRenameProgress(Dialog%,from\$,to\$,addrof(ret%))

104. w32FileMoveProgress(padre%,from\$,to\$,addrof(ret%))

105. w32ListFindItemPartial(lis%,string\$,position%,addrof(when%))

Searches, from a *position%*, for a list view item whose first column text begins with the *string\$* specified as the second parameter.

lis%, reference to the list view control.

string\$, string to compare with the item text.

position%, starting position to search from.

when%, returns the index of the item if successful, or -1 otherwise

106. w32ControlFlat(control%)

Specifies that a *control%* has a border with a sunken edge.

Control%, reference to the control

Descripción	Unidad	Largo	Ancho	Alto	Precio Unidad
Zona A	1	2.25	1.35	0	423
Zona B	1	2.25	15	0	423
Zona A	1	2.25	1.35	0	423

Supr. Eliminar Insert. Insertar F2. Editar Registro F12. Ir a Árbol

107. w32CreateImageList(width%,file\$,addrof(ima%))

Creates a reference to a .bmp *file\$* that must contain an image list.

The *width%* in pixels of each image in the list must be specified as the first parameter.

A reference to the image list is returned. Calling to this function has no visual effect, only creates a reference to a image list to be used with List View and Tree View controls.

108. **w32ListImageList(lis%,ima%,addrof(ret%))**

Assigns an image list (*ima%*) to a list view control (*lis%*).
ret%, returns the handle to the image list previously associated with the control if successful, or 0 otherwise.

109. **w32ListSetImage(lis%,item%,ima_index%,addrof(ret%))**

Set the image (# *ima_index%*) of an *item%* of the list view (*lis%*) from the list's image list.

lis%, reference to the list view control.
item%, zero-based index of the item of the list whose image is to be assigned.
ima_index%, zero-based index of the item's icon in the control's image list

110. **w32ListInsertItemEx(lis%,text\$,item%,ima_index%,addrof(resu%))**

This function adds a new paramater to **w32ListInsertItem** to allow to specify a zero-based index (*ima_index%*) of the item's icon in the control's image list.

Sample W32 BASIC Program:

To contain all the icons that will be showed in a list view control the following file, called image.bmp, has been created:

Each icon is 20 pixels width.



```
call w32CreateImageList(20, ".\image.bmp", addrof(ima%))
call w32ListImageList(lis%, ima%, addrof(ret%))
call w32ListSetImage(lis%, 0, 1, addrof(ret%)).
```

Here is the result:

Capítulo	Descripción
 01	ACONDICIONAMIENTO DEL TERRENO

111. **w32CreateTreeH(dialog%,x%,y%,width%,height%,addrof(tree%))**

Creates a tree view control in the Dialog Box referenced by *dialog%*. (Use the value returned by **w32GetParent** as the first parameter)

112. **w32TreeInsertItem(tree%,text\$,parent%,index%,childrens%,addrof(ret%))**

Inserts a new item in alphabetical order in the tree view control referenced by *tree%*.

Tree%, reference to the tree view control (Value returned by `w32CreateTreeH`)

Text\$, string that contains the item text.

Parent%, handle to the parent item. If this member is the 0, the item is inserted at the root of the tree view control. (This value must be get by a previous call to `w32TreeInsertItem`)

Index%, a integer value to associate with the item.

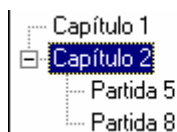
Childrens%, Flag that indicates whether the item has associated child items. This member can be one of the following values:

- `zer` The item has no child items.
- `o`
- `one` The item has one or more child items

Ret%, returns a handle to the just inserted item, that should be used in any function that requieres it.

Sample W32 BASIC Program: The following lines produce the tree below.

```
call w32TreeInsertItem(tree%,"Capítulo 1",0,0,0,addrOf(cap1%))
call w32TreeInsertItem(tree%,"Capítulo 2",0,0,1,addrOf(cap2%))
call w32TreeInsertItem(tree%,"Partida 5",cap2%,002001,0,addrOf(ret%))
call w32TreeInsertItem(tree%,"Partida 8",cap2%,002002,0,addrOf(ret%))
```



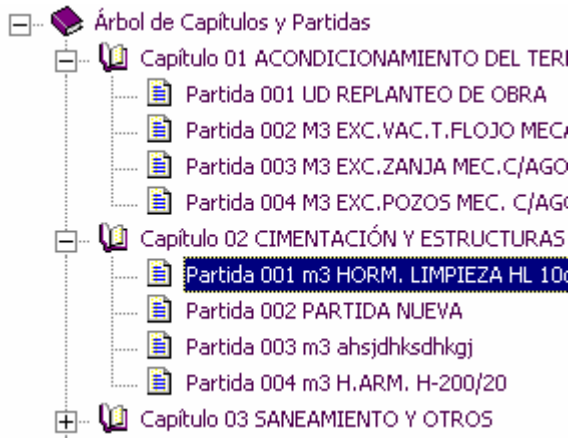
113. `w32TreeImageList(tree%,ima%,addrOf(ret%))`

Assigns an image list (*ima%*) to a tree view control (*tree%*).

ret%, returns the handle to the image list previously associated with the control if successful, or 0 otherwise.

114. `w32TreeInsertItemEx(tree%,text$,parent%,index%,childrens%,ima_index%,addrOf(ret%))`

This function adds a new paramater to `w32TreeInsertItem` to allow to specify a zero-based index (*ima_index%*) of the item's icon in the control's image list.



115. **w32TreeExpand(tree%,parent%,addrof(ret%))**

Expands the list of child items associated with the specified parent item, if any.

Tree%, reference to the tree view control.

Parent%, handle of the item to be expanded

Ret%, returns nonzero if a change took place, or zero otherwise.

116. **w32TreeEnsureVisible(tree%,item%,addrof(ret%))**

Ensures that a tree view item is visible, expanding the parent item or scrolling the tree view control, if necessary

Tree%, reference to the tree view control.

item%, handle of the item.

Ret%, returns nonzero if the system scrolled the items in the tree view control and no items were expanded. Otherwise, the message returns zero.

117. **w32TreeDeleteItem(tree%,item%,addrof(ret%))**

Removes an item from a tree view control.

Tree%, reference to the tree view control.

item%, handle of the item.

Ret%, returns 1 if successful, or 0 otherwise.

118. **w32TreeDeleteAllItems(tree%,addrof(ret%))**

Removes all items from a tree view control.

Tree%, reference to the tree view control.

Ret%, returns 1 if successful, or 0 otherwise.

119. **w32TreeSetColor(tree%,r%,g%,b%)**

Sets the text color of the control.

Tree%, reference to the tree view control.

r % – r(ed) component value [0..255]
g% – g(reen) component value [0..255]
b% – b(lue) component value [0..255]

120. w32TreeSetSel(tree%,item%,addrof(ret%))

Selects the specified tree view item.
Tree%, reference to the tree view control.
item%, handle of the item.
Ret%, returns 1 if successful, or 0 otherwise.

121. w32TreeGetSelItem(tree%,addrof(ret%))

Retrieves the currently selected item in the tree view control
Tree%, reference to the tree view control
Ret%, returns the handle to the item if successful, or 0 otherwise

122. w32TreeGetItem(tree%,item%,addrof(text\$),addrof(index%))

Returns the item text and the integer value associated with the item (index%)
Tree%, reference to the tree view control.
item%, handle of the item.
Text\$, buffer that receives the item text.
Index%, integer value associated with the item

123. w32TreeGetParent(tree%,item%,addrof(parent%))

Tree%, reference to the tree view control.
item%, handle of the item.
Parent%, handle to the item's parent.

124. w32TreeGetChild(tree%,parent%,addrof(child%))

Retrieves the first child item of the item specified by the *parent%* parameter

125. w32TreeGetNextSibling(tree%,item%,addrof(sibling%))

Retrieves the next sibling item of a specified *item%* in a *tree%* view control

126. w32TreeGetEventsn(dialog%,tree%,button%,addrof(etree%),addrof(ptree%))

Where *n* is a value between 1 and 9. Calling this group of functions allows to trap events generated by tree view controls.

Dialog%, reference to the Dialog Box. (Value returned by w32GetParent).
tree%, reference to the tree view control (Value returned by w32CreateTreeH).

button%, value between 200 and 299 that indicates the ID of the button (real or dummy) which code in *BtnClick Subroutine* will be executed each time an event occurs.

Etree%, event number. (See table below)

Ptree%, property for the event. (See table below)

<i>Event</i>	<i>Description</i>	<i>Property</i>
1	The user clicks an item with the left mouse button	Handle to the item
2	An item has been selected	Handle to the item
3	The user double-clicks an item with the left mouse button	Handle to the item
4	The user clicks an item with the right mouse button	Handle to the item
5	A key has been pressed	Code for the key (See w32Skeybd_event_down)
6	A parent item's list of child items has expanded	Handle to the item

In all of the dialog boxes that make up a .exe program, the same function must be called once only. So, to define event processing for three tree view controls the functions to use would be ...

Call w32TreeGetEvents1...

Call w32TreeGetEvents2...

Call w32TreeGetEvents3...

127. w32DefineRegion(*n*,*dialog%*,*button%*,*addrof(control%)*,*addrof(left%)*,*addrof(top%)*,*addrof(right%)*,*addrof(bottom%)*)

Where *n* is a value between 1 and 9. Calling this group of functions allows users to draw a rectangle on a dialog.

This function specifies an ID of a dummy button which code in *BtnClick Subroutine* will be executed each time the user releases the left mouse button.

Besides returning the coordinates of the rectangle, the last four parameters can be use as *input* parameters to delimit the region where user can drag the mouse to draw.

Dialog%, reference to the Dialog Box. (Value returned by w32GetParent).

button%, value between 200 and 299 that indicates the ID of the button (real or dummy) which code in *BtnClick Subroutine* will be executed each time the user releases the left mouse button.

Control%, name of the variable which value will determine if the user is allowed to drag the mouse to draw. A value of 0 enables and a value of 1 disables. Each time the user finish the draw releasing the left mouse button this variable will be set to 0

In all of the dialog boxes that make up a .exe program, the same function must be called once only.

w32ListSetFocusItem(lis%,item%,addrof(ret%))

Set the input focus to the desired item of the list, so it is surrounded by a standard focus rectangle.

<input type="checkbox"/> ? c4	Rest. Don Quijote	4
<input type="checkbox"/> ? c5	Rest. Don Quijote	5
<input type="checkbox"/> ? c6	Rest. Don Quijote	6

128. w32ListSetHotItem(lis%,item%,addrof(ret%))

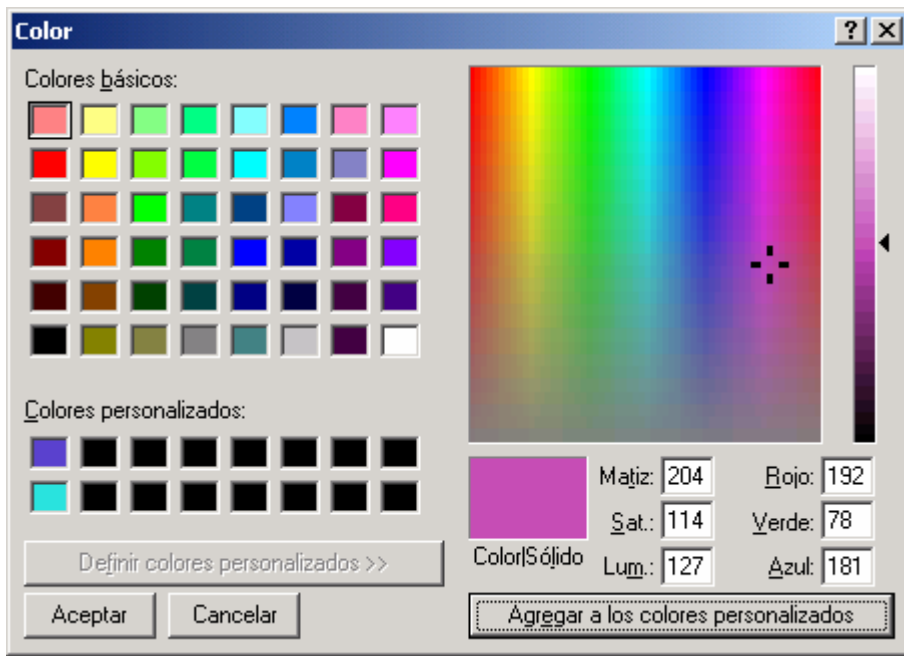
Sets the hot item in a list view control. Each list can have only one hot item simultaneously. To disable this property, call this function setting the second parameter to -1.

<input type="checkbox"/> ? c6	Rest. Don Quijote	6
<input type="checkbox"/> ? c7	Rest. Don Quijote	7

129. w32CreateProgressH(parent%,x%,y%,width%,height%,smooth%,vertical%,addrof(hand%))

Creates a progress bar control over the form specified as the first parameter.

130. w32ChooseColor(parent%,r%,g%,b%,addrof(red%),addrof(green%),addrof(blue%),addrof(ret%))



131. w32DeleteObject (objeto%)

This function frees the memory used by program objects (images and fonts)

132. w32GetBitmapDimensions(path\$, addrof(w%), addrof(h%))

133. w32CreateImageHex(hand%,path\$,x%,y%,w%,h%,addrof(image%),addrof(control %))

134. w32B3DH(button%)

135. w32BFlatH(button%)

136. w32BMultilineH(button%)

137. w32BLoadImageH(button%,file\$,w%,h%,addrof(image%))

138. w32BFLoadImageH(button%,file\$,w%,h%,addrof(image%))

139. w32ComboGetCurSel(combo%,addrof(sel%))

140. w32ComboResetContent(combo%,addrof(ret%))

141. w32CreateCalendarH(parent%,x%,y%,w%,h%,addrof(cal%))

142. w32CreateProcess(command\$,addrof(retorno%))

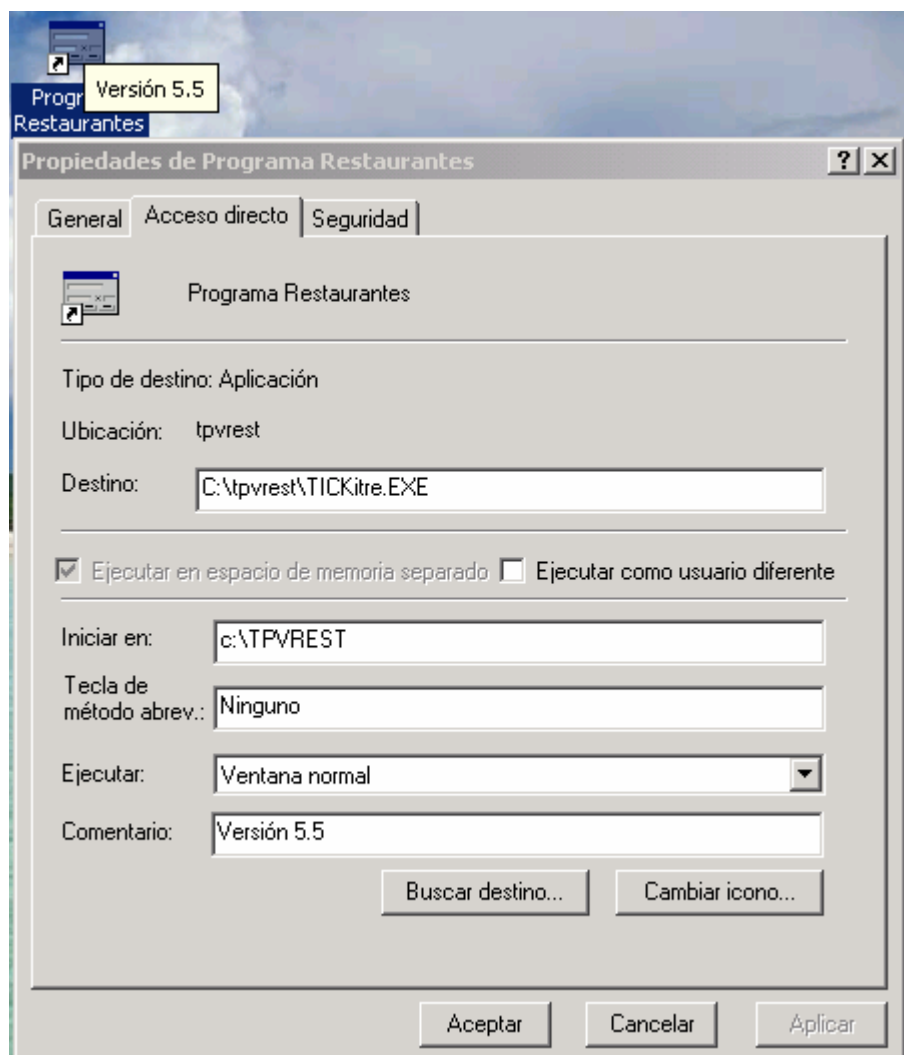
Sample W32 BASIC Program: call w32CreateProcess("command.com /c sound 900 3",addrof(ret%))

143. w32CreateShortCut(program\$,access_Ink\$,work_dir\$,comment\$,addrof(ret%))

Sample W32 BASIC Program:

```
Rem Find Desktop path... (See registry functions)
call cRegOpenKey(80000001H,"Software",addrof(sub%),0,addrof(ret%))
call cRegOpenKey(sub%,"Microsoft",addrof(sub1%),0,addrof(ret%))
call cRegOpenKey(sub1%,"Windows",addrof(sub2%),0,addrof(ret%))
call cRegOpenKey(sub2%,"CurrentVersion",addrof(sub3%),0,addrof(ret%))
call cRegOpenKey(sub3%,"Explorer",addrof(sub4%),0,addrof(ret%))
call cRegOpenKey(sub4%,"Shell Folders",addrof(sub5%),0,addrof(ret%))
call cRegQueryValue(sub5%,"Desktop",addrof(donde$),addrof(ret%))

call w32CreateShortCut("c:\TPVREST\tickitre.exe",donde$&"\Programa
Restaurantes.lnk","c:\TPVREST","Versión 5.5",addrof(ret%))
```



144. w32CreateTipH(control%,text\$,addrof(tip%))

145. w32CreateWindowH(parent%,ID%,type%,style%,style_ex%,title\$,x%,y%,w%,h%,ad drof(control%))

Sample W32 BASIC Program:

```
call w32CreateWindowH(parent%,106,"SysDateTimePick32",WS_BASE% OR  
WS_TABSTOP% OR WS_GROUP% OR DTS_SHOWNONE%, WS_EX_CLIENTEDGE%, "",  
470, 60, 300, 30, addrof(r106%))
```

```
call cSetFocus(dlg1HDLG%,106)
```



Types of controls and styles:

```
Rem Generic styles  
WS_OVERLAPPED% = 00000000H  
WS_POPUP% = 80000000H  
WS_CHILD% = 40000000H  
WS_VISIBLE% = 10000000H  
WS_DISABLED% = 08000000H  
WS_MINIMIZE% = 20000000H  
WS_MAXIMIZE% = 01000000H  
WS_CAPTION% = 00C00000H  
WS_BORDER% = 00800000H  
WS_DLGFRAME% = 00400000H  
WS_VSCROLL% = 00200000H  
WS_HSCROLL% = 00100000H  
WS_SYSMENU% = 00080000H  
WS_MINIMIZEBOX% = 00020000H  
WS_MAXIMIZEBOX% = 00010000H  
WS_GROUP% = 00020000H  
WS_TABSTOP% = 00010000H  
WS_BASE%=(WS_CHILD% OR WS_VISIBLE%)
```

```
Rem Extended styles
```

WS_EX_CLIENTEDGE% = 00000200H

Rem Static Controls. Type: "Static"

SS_LEFT% = 00000000H
SS_CENTER% = 00000001H
SS_RIGHT% = 00000002H
SS_ICON% = 00000003H
SS_BLACKRECT% = 00000004H
SS_GRAYRECT% = 00000005H
SS_WHITERECT% = 00000006H
SS_BLACKFRAME% = 00000007H
SS_GRAYFRAME% = 00000008H
SS_WHITEFRAME% = 00000009H
SS_SIMPLE% = 0000000BH
SS_LEFTNOWORDWRAP% = 0000000CH
SS_NOPREFIX% = 00000080H
SS_NOTIFY% = 00000100H

Rem Buttons. Type: "Button"

BS_PUSHBUTTON% = 00000000H
BS_DEFPUSHBUTTON% = 00000001H
BS_CHECKBOX% = 00000002H
BS_AUTOCHECKBOX% = 00000003H
BS_RADIOBUTTON% = 00000004H
BS_3STATE% = 00000005H
BS_AUTO3STATE% = 00000006H
BS_GROUPBOX% = 00000007H
BS_USERBUTTON% = 00000008H
BS_AUTORADIOBUTTON% = 00000009H
BS_OWNERDRAW% = 0000000BH
BS_LEFTTEXT% = 00000020H
BS_MULTILINE% = 00002000H
BS_VCENTER% = 00000C00H

Rem Edit Controls. Type: "Edit"

ES_LEFT% = 00000000H
ES_CENTER% = 00000001H
ES_RIGHT% = 00000002H
ES_MULTILINE% = 00000004H
ES_UPPERCASE% = 00000008H
ES_LOWERCASE% = 00000010H
ES_PASSWORD% = 00000020H
ES_AUTOVSCROLL% = 00000040H
ES_AUTOHSCROLL% = 00000080H
ES_NOHIDESEL% = 00000100H
ES_OEMCONVERT% = 00000400H
ES_READONLY% = 00000800H
ES_WANTRETURN% = 00001000H
ES_NUMBER% = 00002000H

Rem Scroll Bar. Type: "ScrollBar"
SBS_HORZ% = 0000H
SBS_VERT% = 0001H
SBS_TOPALIGN% = 0002H
SBS_LEFTALIGN% = 0002H
SBS_BOTTOMALIGN% = 0004H
SBS_RIGHTALIGN% = 0004H
SBS_SIZEBOXTOPLEFTALIGN% = 0002H
SBS_SIZEBOXTOTTOMRIGHTALIGN% = 0004H
SBS_SIZEBOX% = 0008H

Rem List Box. Type: "ListBox"
LBS_NOTIFY% = 0001H
LBS_SORT% = 0002H
LBS_NOREDRAW% = 0004H
LBS_MULTIPLESEL% = 0008H
LBS_OWNERDRAWFIXED% = 0010H
LBS_OWNERDRAWVARIABLE% = 0020H
LBS_HASSTRINGS% = 0040H
LBS_USETABSTOPS% = 0080H
LBS_NOINTEGRALHEIGHT% = 0100H
LBS_MULTICOLUMN% = 0200H
LBS_WANTKEYBOARDINPUT% = 0400H
LBS_EXTENDEDSEL% = 0800H
LBS_DISABLENOSCROLL% = 1000H
LBS_NOSEL% = 4000H
LBS_STANDARD% = (LBS_NOTIFY% OR WS_VSCROLL% OR WS_BORDER% OR
WS_HSCROLL% OR LBS_USETABSTOPS%)

Rem Combo Box. Type: "ComboBox"
CBS_SIMPLE% = 0001H
CBS_DROPDOWN% = 0002H
CBS_DROPDOWNLIST% = 0003H
CBS_OWNERDRAWFIXED% = 0010H
CBS_OWNERDRAWVARIABLE% = 0020H
CBS_AUTOHSCROLL% = 0040H
CBS_OEMCONVERT% = 0080H
CBS_SORT% = 0100H
CBS_HASSTRINGS% = 0200H
CBS_NOINTEGRALHEIGHT% = 0400H
CBS_DISABLENOSCROLL% = 0800H

Rem Date Time Picker Control. Type: "SysDateTimePick32"
DTS_TIMEFORMAT% = 0009H
DTS_SHOWNONE% = 0002H

146. w32DTPSetSel(control%,null%,year%,month%,day%,hour%,min%,sec%,addrof(ret%))

Set the value for a Date Time Picker control .

147. w32DTPGetSel (control%, addrof(year%), addrof(month%), addrof(day%), addrof(hour%), addrof(min%), addrof(sec%), addrof(ret%))

Get the current value for a Date Time Picker control.

148. w32EditGetLineCount (control%,addrof(lines%))

149. w32EditGetLine (control%, line_number%, addrof(text\$), addrof(length%))

150. w32GetDiskFreeSpaceEx(unit\$,addrof(bytesfree\$),addrof(retcode%))

151. w32GetDiskSpaceEx (unit\$,addrof(espace\$),addrof(ret%))

152. w32GetVersion(addrof(ver%))

Return value:

1 – 95/98/Me

2 - NT/2000/XP

153. w32GetWindowID(control%,addrof(id%))

154. w32GetWindowRectEx(control%,addrof(left%),addrof(top%),addrof(right%),addrof(bottom%),addrof(ret%))

155. w32ImageBackground n (parent%,image\$,w%,h%,addrof(ret%))

Where n is a value from 1 to 9.

156. w32LBoxResetContentH(control%)

157. w32ListGetRect(lis%,item%,subitem%,addrof(left%),addrof(top%),addrof(right%),addrof(bottom%),addrof(ret%))

158. w32ListSetColumn(list%,column%,title\$,addrof(ret%))

159. w32ListSetFullBkColor(lis%,r%,g%,b%)

160. w32MAPISendMail(subject\$,content\$,destination_ address\$,name\$,number_of_files%,file\$,addrof(ret%))

161. w32CreateMenu(addrof(menu%))

Creates a reference to a context menu.

162. w32MenuInsertItem(menu%, id%, type%, text\$, addrof(ret%))

Adds an element at the end of a context menu

Menu%, (Value returned by w32CreateMenu)

ID%, ID associated to the item

Type%,

0, Normal Item

1, Subitem

2, SEPARATOR

163. w32MenuItemBitmap(menu%,ID%,image\$,addrof(image%))

164. w32MenuPopup(menu%, control%, left%, top%, addrof(sel%))

Shows a context menu associated to the control specified as the second parameter.

165. w32MenuDestroy(menu%,addrof(ret%))

Frees memory.

Sample W32 BASIC Program:

```
$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart
.
.
.
call w32CreateMenu(addrof(submenu%))
call w32MenuInsertItem(submenu%,6,0,"Opción 1",addrof(ret%))
call w32MenuInsertItem(submenu%,7,0,"Opción 2",addrof(ret%))

call w32CreateMenu(addrof(menu%))
call w32MenuInsertItem(menu%,1,0,"Insertar",addrof(ret%))
call w32MenuInsertItem(menu%,2,2," ",addrof(ret%))
call w32MenuInsertItem(menu%,3,0,"Eliminar",addrof(ret%))
call w32MenuInsertItem(menu%,submenu%,1,"Más",addrof(ret%))

Rem ancho 13, alto 13
call w32MenuItemBitmap(menu%,3,".\eliminar.bmp",addrof(ret%))
```

```

call w32MenuItemBitmap(menu%,6, ".\insertar.bmp",addrof(ret%))
.
.
.
call
w32ListGetEvents2(padre%,lis%,250,addrof(evento%),addrof(l_item%),add
rof(l_sub%))

REM@# iniend
$EXIT

$SUB dlg1BtnClick
REM      Button Click save area
REM@# btnstart
SELECT CMDID%

CASE 250
    Rem Notificaciones para la lista lis%
    Rem evento%->Numero de evento
    Rem l_item%->item de la lista (row)
    Rem l_sub%->Sub-item de la lista (column)
SELECT evento%

CASE 4
Rem Click con el botón de derecho en un campo
call w32GetWindowRectEx(lis%, addrof(izq%), addrof(alto%),
addrof(der%), addrof(fondo%),addrof(ret%))
desx%=izq%
desy%=alto%
Rem Coordinadas relativas a la lista, por eso se aplica
desplazamiento
call w32ListGetRect(lis%, l_item%, l_sub%, addrof(izq%),
addrof(alto%), addrof(der%), addrof(fondo%),addrof(ret%))
donde%=fondo%+desy%
call w32MenuPopup(menu%,lis%,izq%+desx%,donde%,addrof(sel%))
    SELECT sel%
    CASE 3
        Call w32ListDeleteItem(lis%,l_item%,addrof(ret%))
        .
        .
        .
    CEND

CEND
CEND
REM@# btnend
GOSUB dlg1.BTNEND
$EXIT

```

<input type="checkbox"/>		c6	Rest. Don Quijote	6
<input type="checkbox"/>		c7	Rest. Don Quijote	7
<input type="checkbox"/>		c8	Inserir	8
<input type="checkbox"/>		c9	Eliminar	9
<input type="checkbox"/>		c10	Más	10
<input type="checkbox"/>		c11	*Opción 1	11
<input type="checkbox"/>		c12	Opción 2	12

166. w32SetWindowOrder(control%,control_ref%,addrof(ret%))

Set that the control specified as the first parameter will follow, in tabulation order, to the control specified as the second parameter.

167. w32TabStopWindow(control%,yes_no%)

Set if a control will be reachable pressing tab key.

168. w32TipUpdateTextH(tip%, control%, new_text\$, addrof(retorno%))

169. w32TreeGetRect(tree%, item%, addrof(left%), addrof(top%), addrof(right%), addrof(bottom%),addrof(ret%))

170. w32WinHelp(parent%,help_file\$,operation%,value\$,addrof(ret%))

Samples:

Rem 11 Show the search index

```
call w32WinHelp(padre%,"videos31.hlp",11,"",addrof(ret%))
```

Rem 4 Help about how to use help

```
call w32WinHelp(padre%,"videos31.hlp",4,"",addrof(ret%))
```

Rem 257 Search for a key word

```
call w32WinHelp(padre%,"videos31.hlp",257,"CINTES VIDEO",addrof(ret%))
```

171. w32ListGetColumnsOrder(lis%,number_of_cols%,addrof(columns_order\$),addrof(ret%))

172. w32CreateGrid(parent%,title\$,ID%,x%,y%,w%,h%,addrof(grid%))

Requires file CETgrid.dll and versión 9.32m or higher

Notes:

- Up to 20 simultaneous grid in the same executable
- Up to 32000 rows
- Up to 256 columns

- Up to 150 rows

Líneas de la Factura 08541
(Versión Beta 1.0)

	Código Referencia	Descripción	Unid.	Precio	Total	Revisado
1	9687250	VARIO CASE P/SISTEMA VAS C/TAP	1	56.78	56.78	<input checked="" type="checkbox"/>
2	9314132	DESTORNILLADOR EXAGONAL" T"3.0	1	0	0	<input type="checkbox"/>
3	9388380	VAINA SUJECION DESTORNILLADOR	1	78.90	78.9	<input checked="" type="checkbox"/>
4	9388532	AVELLANADOR PARA USS VAS	1	123.45	123.45	<input checked="" type="checkbox"/>
5	9388357	EXTRACTOR PARA USS VAS	1	67.89	67.89	<input type="checkbox"/>
6	9497191	Insertar AS 6.2X30 T.CPTO	2	12.34	24.68	<input checked="" type="checkbox"/>
7	9497192	Eliminar AS 6.2X35 TAN	4	567.89	2271.56	<input type="checkbox"/>
8	9497193	Más *Color de Celda...	4	1234.56	4938.24	<input checked="" type="checkbox"/>
9	9497194	Opción 2 TORNILLO USS	4	34.56	138.24	<input checked="" type="checkbox"/>

Seleccione el menú grid para testear las opciones

400 - Evento: 25 (Cambio de fila)	P1: 7	P2: 2
400 - Evento: 24 (Click en Encabezado)	P1: 5	P2: 0
400 - Evento: 13 (Inicio de edición: EXTRACTOR PARA USS VAS)		
400 - Evento: 29 (Click con el boton derecho)	P1: 5	P2: 2

Ocultar Deshabilitar
Aceptar Cancelar

Sample W32 BASIC Program: *Grid.prj*

173. **w32GridDeleteAllItems(grid%)**

174. **w32GridDeleteColumn(grid%, col%)**

175. **w32GridDeleteItem(grid%, temp%)**

176. **w32GridEditCurCell(grid%)**

177. **w32GridEditing (grid%, addrof(editting%))**

178. **w32GridGetCellProtection(grid%, row%, col%, addrof(protected%))**

179. **w32GridGetCol(grid%, addrof(current_col%))**

180. **w32GridGetColCount(grid%, addrof(num_of_cols%))**

181. **w32GridGetEditString(grid%, addrof(txt\$))**

182. **w32GridGetEvent(grid%, addrof (event%))**

Returns the number of the event generated by the grid. Functions *w32GridGetEventP1* and *w32GridEventP2* should be called to obtain the properties for the event.

Event	Description	Property 1	Property 2
1	<i>F1</i>	Current row	Current column
2	<i>F2</i> (Reserved for edition)		
3	<i>F3</i>	Current row	Current column
4	<i>F4</i>	Current row	Current column
5	<i>F5</i>	Current row	Current column
6	<i>F6</i>	Current row	Current column
7	<i>F7</i>	Current row	Current column
8	<i>F8</i>	Current row	Current column
9	<i>F9</i>	Current row	Current column
10	<i>F10</i> (Reserved)		
11	<i>F11</i>	Current row	Current column
12	<i>F12</i>	Current row	Current column
13	Edit Begin	Row	Column
14	Edit End	Row	Column
15	Delete Key	Current row	Current column
16	<i>Insert Key</i>	Current row	Current column
17	Home Key	Current row	Current column
18	End key	Current row	Current column
19	Arrow up.	1	Current column
20	Arrow down.	Last Row	Current column
21	<i>Prior Pag key.</i>	1	Current column
22	<i>Next Pag key</i>	Last Row	Current column
23	(Reserved)		
24	Clic beyond cells area .	Corresponding row, -1 otherwise	Corresponding column, -1 otherwise
25	Row change	Previous row	Previous column
26	Column change	Previous row	Previous column
27	The grid has obtained the focus	ID of the previously focused control or 0 if the previous control belongs to other dialog or application	-1
28	The grid has lost the focus	ID of the control that has just obtained the focus or 0 if the control belongs to other dialog or application	-1
29	Right button click.	Corresponding row, -1 otherwise	Corresponding column, -1 otherwise

30	Double-Click beyond cells area	Corresponding row, -1 otherwise	Corresponding column, -1 otherwise
31	ENTER pressed	Current row	Current column
32	ESC pressed	Current row	Column

Sample W32 BASIC Program:

```

$SUB grid1Event
SELECT CET_EVTYPE%
REM      Event notification save area
REM@# evtstart
CASE 15
    Rem 15 → Special events....
    SELECT CMDID%
    Rem 400 → ID associated by w32CreateGrid
    CASE 400
        call w32GridGetEvent(grid%,addrof(ev%))
        SELECT ev%
        CASE 9
            call w32GridGetEventP1(grid%,addrof(p1%))
            call w32GridGetEventP2(grid%,addrof(p2%))
            call w32GridGetText(grid%,p1%,p2%,addrof(txt$))
            texto$=" (F9 pressed: "&txt$&" ) "

        CEND

    CEND

CEND

REM@# evtend
CEND
$EXIT

REM
REM      The dialog is being initialized
REM      Set the initial contents and state of the list box(s)
REM      and other controls
REM

$SUB grid1Init
REM      Initialize dialog save area
REM@# inistart
di%=grid1HDLG%
call cSetFocus(di%,200)
call w32GetParent(addrof(padre%))
call w32CreateGrid(padre%,"Líneas de la Factura
08541"&chr$(13)&"(Versión Beta 1.0)",400,10,15,625,285,addrof(grid%))

```

183. w32GridGetEventP1(grid%, addrof (p1%))

184. w32GridGetEventP2(grid%, addrof (p2%))

185. w32GridGetItemCount(grid%, addrof (rows%))

**186. call w32GridGetRect (grid%, row%, col%, addrof(left%), addrof(top%),
addrof(right%), addrof(bottom%))**

187. w32GridGetRow(grid%, addrof (row%))

Return the current active row.

188. **w32GridGetText(grid%, row%, col%, addrof(text\$))**
189. **w32GridInsertColumn(grid%, position%)**
190. **w32GridInsertItem(grid%, position%)**
191. **w32GridIsHighlighted(grid%, addrof(yes_no%))**
192. **w32GridProtectCell(grid%, row%, col%, protect%)**
193. **w32GridSetAllowColResize(grid%, redim%)**
194. **w32GridSetBkColor(grid%, r%, g%, b%)**
195. **w32GridSetCellBkColor(grid%,row%,col%,r%,g%,v%)**
196. **w32GridSetColMaxChars(grid%,column%,max_char%)**
197. **w32GridSetColsNumbered(grid%, auto_numbered%)**
198. **w32GridSetColWidth(grid%, col%,width%)**
199. **w32GridSetCursorColor(grid%,r%,g%,b%)**
200. **w32GridSetCursorPos(grid%,row%, column%)**
201. **w32GridSetDim(grid%,num_rows%, num_columns%)**
202. **w32GridSetEditable(grid%, editable%)**
203. **w32GridSetEnterMode (grid%, mode%)**
204. **w32GridSetSelColor (grid%, r%, g%,b%)**
205. **w32GridSetSelTextColor (grid%, r%, g%, b%)**
206. **w32GridSetText(grid%, row%, column%, text\$)**
207. **w32GridSetTitleFont(grid%, font\$, points%, orientation%, bold%, italic%, inderline%,strike_through%)**
208. **w32GridSetTitleHeight(grid%,height%)**
209. **w32GridShowHilighht(grid%,yes_no%)**
210. **w32EditENTEREx(index%,hand%,ID%,addrof(key%))**

Sample W32 BASIC Program:

```

$SUB dlg1Event
SELECT CET_EVTYPE%
REM      Event notification save area
REM@# evtstart
CASE 15

```

```

Rem Special events processing
SELECT CMDID%
CASE 100
    SELECT tecla100%
    CASE 112
        Rem F1 pressed

    CEND

CASE 101
    SELECT tecla101%
    CASE 33
        Rem NEXT
    CASE 34
        Rem PRIOR
    CEND

CEND
REM@# evtend
CEND
$EXIT

```

```

$SUB dlg1Init
REM    Initialize dialog save area
REM@# inistart
Call cSetFocus(dlg1HDLG%,200)

call w32GetParent(addrOf(padre%))
...

Call w32GetWindow(padre%,100,addrOf(ed100%))
Call w32GetWindow(padre%,101,addrOf(ed101%))

...

call w32EditENTEREx(1,ed100%,100,addrOf(tecla100%))
call w32EditENTEREx(2,ed101%,101,addrOf(tecla101%))

REM@# iniend

```

211. w32ListGetEventsExn(parent%,lis%,ID%,addrOf(event%),addrOf(param1%),addrOf(param2%))

Sample W32 BASIC Program:

```

$SUB dlg1Event
SELECT CET_EVTYPE%
REM    Event notification save area

```

```

REM@# evtstart
CASE 15
SELECT CMDID%
...
CASE 400
    Rem Notifications for list control lis%
    Rem evento% -> Event number
    Rem l_item% -> Property 1
    Rem l_sub% -> Property 2
    SELECT evento%
    CASE 1
        Rem Click
        rem MsgBox "Click: "&str$(l_item%)&","&str$(l_sub%)

    CASE 2
        Rem Selection changed
        rem MsgBox "Selection Changed: "&str$(l_item%)
        call w32ListGetSelItem(lis%,addrof(item%))
        if item%<>-1
        then
        call
w32ListGetText(lis%,item%,0,addrof(text$),addrof(ret%))
        call cSetCtrlValue(dlg1HDLG%,100,text$)
        ifend

    CASE 3
        Rem Double-Click
        MsgBox "Double-Click: "&str$(l_item%)&","&str$(l_sub%)

    CASE 4
        Rem Right Button
        MsgBox "Right Button: "&str$(l_item%)&","&str$(l_sub%)

    CASE 5
        Rem Key pressed:
        MsgBox "Tecla: "&str$(l_item%)

    CASE 6

        Rem Column clicked
        MsgBox "Click en column: "&str$(l_sub%)
        call w32ListSortItems(lis%,l_sub%,0,1,addrof(ret%))

    CEND

...
CEND
REM@# evtend
CEND
$EXIT

```

```

$SUB dlg1Init
REM      Initialize dialog save area
REM@# inistart

call w32GetParent(addrOf(padre%))
...
call w32CreateListH(padre%,10,195,475,180,addrOf(lis%))
...
call
w32ListGetEventsEx1(padre%,lis%,400,addrOf(evento%),addrOf(l_item%),a
ddrof(l_sub%))
...
REM@# iniend

```

212. w32TreeGetEventsExn(parent%,tree%,ID%,addrOf(etree%),addrOf(ptree%))

213. w32TreeSetImage(tree%,ref%,image_index%,addrOf(ret%))