A True LB API Popup Menu.

By : Duncan Bushaw
Published on : March 22, 2007.

Table of Contents

Appendix C ; The CFC API Popup Menu Tree.

In this article I will be explaining how I used the following 12 API functions; CheckMenuItem, CheckMenuRadioItem, CreatePopupMenu, DestroyMenu, EnableMenuItem, HiliteMenuItem, lnsertMenuItem, SetMenuDefaultItem, SetMenuInfo, SetMenuItemBitmaps, SetMenuItemlnfo, TrackPopupMenuEx, and 3 API structures; MENUINFO, MENUITEMINFO, and TPMPARAMS, and how they work together in the example
"Classic Face Clock.bas".

CFC.cab

- Details
- Download
- 83 KB

You can also go to my domain at Duncan B to download a full working package.

editor's note: Some handles in the example program are passed as type long. All handles in Liberty BASIC should be passed as type ulong. Carl Gundel, author of Liberty BASIC says, "ULONG is recommended for things like handles to windows and other resources."

author's note : Add to cab file 5 versions of the 'Classic Face Clock.bas' where I played around with Long and Ulong, And found that they are not completly interchangable.

1. *'Classic Face Clock Long.bas' - Changed all Ulong to Long this will cause an error when starting the program which affects the settings file 'Classic Face Clock.txt' which then causes an error when starting the other bas files. To recover just delete 'Classic Face Clock.txt'*
2. *'Classic Face Clock Ulong.bas' - Changed all Long to Ulong this will cause an error when starting the program which affects the settings file 'Classic Face Clock.txt' which then causes an error when starting the other bas files. To recover just delete 'Classic Face Clock.txt'*
3. *'Classic Face Clock window handles.bas' - Changed all Window handles to Ulong. There was no noticible change the program still works fine.*
4. *'Classic Face Clock window-menu handles.bas' - Changed all Window and menu handles to Ulong. There was no noticible change the program still works fine.*
5. *'Classic Face Clock window-menu-image handles.bas' - Changed all Window, menu, and image handles to Ulong. There was no noticible change the program still works fine. This version I believe follows the recommendation concerning Ulong correctly.*

The program consists of a graphicbox in a popup window with an API popupmenu called with the event

handler of the graphicbox. I will also be describing some of the neat things that can be done with a popupmenu that I discovered while I was writing the example program.

When you first start the example program it will begin with the default setting for all options. The size of the clock will be medium with a buttonface background, in this mode and larger modes the date and AM/PM indicator will be shown, in the small mode just numbers and hands are shown.

When the example program runs it first creates an unnamed window that is 0x0 in size. The purpose of the window is;
1) to be able to load image files into memory using Alyce Watson's jpeg.dll before creating the main program window.
2) to pass this window's handle to the HiliteMenuItem function call, because if the main program window handle is used some of the menuitems that are hilighted by the dll call get painted to the main program window. Using the dummy window prevents that from happening.

After creating the dummy window the example program loads the images into memory using either Alyce Watson's jpeg.dll or the LB Loadbmp command. If the menuitem [Options > Load .bmp Only (resets menu)] is checked then the example program loads images using the LB command.

The example program then creates the popupmenu, whose handle is kept in the variable MPM. Every menuitem in the popupmenu should have a different ID number assigned to it at this time which is then returned with the TrackPopupMenuEx dll call. This ID will be needed to disable or enable items later.

Not all menuitems in the example program have different ID numbers. All separators were assigned the number zero as they cannot be selected. All other menuitems are assigned a unique number between 1 and 65535.

The program is designed to make it as easy as possible to move the menuitems around so as to change the layout of the popup menu when it is called, without having to rewrite a lot of other code to make it work. This also applies to moving menuitems from one assigned menu to another, which is accomplished by moving these two lines only ;

```
z=z+1 ' Increment counter.

null=InsertMenuItem(y,z,0,0,46,SMNMI,0,0,"0 - Enabled",0) ' Insert menuitem.
```

the arguments for the InsertMenuItem function are explained below:
[y] hMenu, is the handle to the menu to put the menuitem in.
[z] uItem, is the position counter for inserting the menuitem and is used to ensure that the next menuitem is inserted at the bottom of the menu.
[0] fType, The type of menuitem.
[0] fState, The state of the menuitem.
[46] wID, is the ID number to assign.
[SMNMI] hSubMenu, the handle of a sub-menu.
[0] hbmpChecked, the handle to the 12x12 bitmap image to use if the menuitem is checked.
[0] hbmpUnchecked, the handle to the 12x12 bitmap image to use if the menuitem is unchecked.

["0 - Enabled"] dwTypeData$, The text to put into the menuitem.
[0] hbmpItem, the handle to the image to use.

Because all function calls other than HiliteMenuItem(hwnd,hmenu,uItemHilite,uHilite) and SetMenuDefaultItem(hMenu,uItem) do not require the assigned menu handle to work correctly, but instead are called with the MPM handle, they do not need to be changed if changing the menu layout. HiliteMenuItem(hwnd,hmenu,uItemHilite,uHilite) and SetMenuDefaultItem(hMenu,uItem) must be used with the menu handle that actually contains the menuitem.

When inserting menuitems into a submenu they do not need to be in the order of their ID numbers. To modify a menuitem in the example program it is accessed by the ID number and not by the menu position, except for SetMenuDefaultItem. This makes rearranging menuitems in a popupmenu easy to do without having to change a lot of code, even if it is from one submenu to another. More than one menuitem in a submenu may be hilighted or set as a default item.

Menuitems that are hilighted with the dll call remain hilighted until the mouse moves over them. Only the first default menuitem in a submenu is valid. If the default menuitems have been marked then double clicking on the ' Preferences ' menuitem will cause the ' Preferences > Top Most ' menuitem to be either checked or unchecked as if that were that menuitem that had been clicked by the mouse.

Holding the right button down and moving it from above the clock to another portion of the screen and then releasing the button will cause the menu to be centered to mouse.

Right click on the clock to call the menu, which will then be displayed in two ways depending on if the ' Preferences > Top Most ' menuitem is checked or not.

If the menuitem ' Preferences > Top Most ' setting is checked then the clock is displayed above all other windows and the menu will not appear above or over the clock, if possible. This is done by the sending a structure containing the coordinates of a rectangle to be excluded in the TrackPopupEx dLL call which tells the system not to put the menu in that rectangle. If the menuitem ' Preferences > Top Most ' setting is not checked then the clock is not displayed above all other windows and the menu will appear over the clock so the menu is centered to mouse.

I need to talk about the TrackPopupMenuEx dll call. If you compare my translation of the C syntax you will notice that in C the return value is BOOL, but my translation of it is Long and not Boolean. The reason for this is that the boolean type of LB is only 2 byte long, whereas the BOOL type of C is 4 byte long.

When you change the clock size, the clock window will close and then reopen centered to the mouse. This was done so that the clock will always have a portion shown on the monitor after resizing.

Mouse Event Handlers. Only 4 graphicbox event handlers are used in the program.

- when leftButtonDown [Down] - The program will hide the mouse and then get the current position of the mouse and save it in a temporary variable that is used to help reposition the clock window.

- when leftButtonMove [Move] - The program will get the current position of the mouse and using the temporary variable repositions the clock window.
- when leftButtonUp [Up] - The program will save the program settings so the new position of the clock window is stored and then show the mouse.
- when rightButtonUp [Menu] - The program will highlight all the menuitems set for highlighting. There are 2 menuitems on the main popup menu that are highlighted. Both of these are in a disabled state but are not grayed. The other menuitems that are highlighted are the current clock size and color settings.

The following is a description of the menuitems in the Options sub-menu.

- Load .bmp only (resets menu) - When checked makes the program only load image files with the .bmp extention using the LB command LOADBMP, When it is unchecked the program will load all images using Alyce Watson's jpeg.dll. When selected the program destroys the menu and deletes the images from memory before rebuilding the menu. All unsaved options will be reset. This is a saved option.
- MNS_CHECKORBMP - When checked the menuitem images and the checkmark images will appear in the same column. If a menuitem has both uncheckmark bmp handle assigned and a menuitem image handle assigned only the uncheckmark image will be seen and the height of the menuitem will be that of the menuitem image. This is a saved option.
- Disable/Enable Examples Menu - When selected will either disable or enable the Examples sub-menu. This is an unsaved option.
- Unmark/Mark Default Menuitems - When selected will either Unmark or Mark the default menuitems. This is an unsaved option.
- Remove/Install Checkmark BMP's - When selected will either remove or install the checkmark and uncheckmark images. This is an unsaved option.
- Remove/Install Menuitem BMP's - When selected will either remove or install the menuitem images. This is an unsaved option.
- View "Classic Face Clock.txt" File. - When selected the program will open, for viewwing, the settings file using the systems default text editor.

This is just a brief description of what I did. Open the bas file and you will find that it is full of rem statements describing each step thru the program. I could go on for pages describing things that I did in the program.

What follows in the article are 3 appendices:
Appendix A ; The Windows API DLL calls.
Appendix B ; The Windows API Structures.
Appendix C ; The CFC API Popup Menu Tree.

Table of Contents

Appendix A ; The Windows API DLL calls.

Contents :

CheckMenuItem :

This function either checks or unchecks the menuitem specified.
LB Syntax :
```
FUNCTION CheckMenuItem(hmenu, uIDCheckItem, uCheck)
CALLDLL #user32, "CheckMenuItem",_
hmenu As uLong,_
ulDCheckItem As Ulong,_
uCheck As Ulong,_
CheckMenuItem As Long
END FUNCTION
```
Parameters :
hmenu [in] ; The handle of the menu containing the menuitem to check or uncheck.
ulDCheckItem [in] ; The menuitem to check or uncheck, as determined by the uCheck parameter.
uCheck [in] ; Flags that control interpretation of the ulDCheckItem parameter, and either check or uncheck the menuitem. This value must be a combination of (MF_BYCOMMAND or MF_BYPOSITION) and (MF_CHECKED or MF_UNCHECKED).
MF_BYCOMMAND = 0. Uses the identifier of a menuitem.
MF_BYPOSITION = 1024. Uses zero based relative position of a menuitem in a menu.
MF_UNCHECKED = 0. Indicates that the menuitem is unchecked.
MF_CHECKED =8. Indicates that the menitem is checked.
Valid values for this parameter are 0, 8, 1024, and 1032.
CheckMenuItem [out] ; Return value. If the menuitem doesn't exist, this value is -1.
This value is the previous state of the menuitem (0 or 8).
The example program pays no attention to this parameter.
Remarks :
In the example program uCheck is either 0 or 8 and hmenu is always MPM, and uIDCheckItem is the ID number of the menuitem. For a menuitem that is a submenu, uIDCheckItem must point to the zero based position of the menuitem, and hmenu must be the menu containing the menuitem, and uCheck must be either 1024 or 1032. The example program does not use this method.

CheckMenuRadioItem :

This function checks a menuitem and makes it a radiomenuitem, and at the same time, it unchecks all other menuitems in the associated group and clears the radiomenuitem type flag for those items.
LB Syntax :
FUNCTION CheckMenuRadioItem(hmenu, idFirst, idLast, idCheck, uFlags)
CALLDLL #user32, "CheckMenuRadioItem",_
hmenu As uLong,_
idFirst As Ulong,_
idLast As Ulong,_
idCheck As Ulong,_
uFlags As Ulong,_
CheckMenuRadioItem As Long
END FUNCTION
Parameters :
hmenu [in] ; The handle of the menu that contains the menuitem.
idFirst [in] ; The first menuitem in a group of menuitems.
idLast [in] ; The last menuitem in a group of menuitems.
idCheck [in] ; The menuitem to check, as determined by the uFlags parameter.
uFlags [in] ; Flags that control interpretation of the idFirst, the idLast, and the idCheck parameters.
MF_BYCOMMAND = 0. Uses the identifier of a menuitem.
MF_BYPOSITION = 1024. Uses zero based relative position of a menuitem in a menu.
Valid values for this parameter are 0 and 1024.
CheckMenuRadioItem [out] ; Return value. An error is indicated with a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
The example program pays no attention to this parameter.
Remarks :
This function sets the MFT_RADIOCHECK and the MFS_CHECKED flag for the menuitem specified by idCheck and clears both flags for all other menuitems in a group.
The selected menuitem is shown with a bullet bitmap and not a checkmark bitmap.

CreatePopupMenu :

This function creates a drop down menu, submenu, or shortcut menu.
This menu is empty until menuitems are added.
LB Syntax :
FUNCTION CreatePopupMenu()
CALLDLL #user32, "CreatePopupMenu",_
CreatePopupMenu AS uLong
END FUNCTION
Parameters :
CreatePopupMenu [out] ; Return value. An error is indicated with a value of 0.
Any other value is the handle of the new menu.

The GetLastError dll call must be used to get the extended error information.
Remarks :
A program can insert the new menu into any menu or it can display a popupmenu
by using the TrackPopupMenu or TrackPopupMenuEx dll calls. Resources used with a menu that is assigned to a window are freed automatically. If the menu is not assigned to a window, a program must free resources used with the menu before closing. A program frees these resources by calling the DestroyMenu function. The example program uses the DestroyMenu function. If using Windows 95/98/Me a maximum of 16,364 menu handles can be assigned.
No limit is given in the MSDN online database for other versions of Windows.

DestroyMenu :

This function destroys a menu and frees memory that it occupies.
LB Syntax :
```
FUNCTION DestroyMenu(hMenu)
CALLDLL #user32, "DestroyMenu",_
hMenu As uLong,_
DestroyMenu As Long
END FUNCTION
```
Parameters :
hMenu [in] ; The handle to a menu to be destroyed.
DestroyMenu [out] ; Return value. An error is indicated with a value of 0. All other values mean it worked. The GetLastError dll call must be used to get the extended error information.
Remarks :
Before a program closes it must use the DestroyMenu dll call to destroy a menu not assigned to a window. Any menu that is assigned to a window is destroyed automatically when the program closes. DestroyMenu will destroy the menu and all its submenus, that is to say it is a recursive function.

EnableMenuItem :

This function enables or disables the specified menuitem.
LB Syntax :
```
FUNCTION EnableMenuItem(hMenu, uIDEnableItem, uEnable)
CALLDLL #user32, "EnableMenuItem",_
hMenu As uLong,_
uIDEnableItem As Ulong,_
uEnable As Ulong,_
EnableMenuItem As Long
END FUNCTION
```
Parameters :
hMenu [in] ; The handle to a menu.
uIDEnableItem [in] ; The menuitem to be enabled or disabled, as determined by the uEnable parameter.
uEnable [in] ; Flags that control interpretation of the uIDEnableItem parameter and sets the state of the menuitem.
This value must be (MF_BYCOMMAND or MF_BYPOSITION) and (MF_ENABLED or

MF_DISABLED or MF_GRAYED).
MF_BYCOMMAND = 0. Uses the identifier of a menuitem.
MF_BYPOSITION = 1024. Uses zero-based relative position of a menuitem in a menu.
MF_ENABLED = 0. Indicates that the menuitem is enabled and not grayed.
MF_GRAYED =1. Indicates that the menuitem is disabled and grayed.
MF_DISABLED = 2. Indicates that the menuitem is disabled, but not grayed.
Valid values for this parameter are 0, 1, 2, 1024, 1025, and 1026.
EnableMenuItem [out] ; Return Value. If the menuitem doesn't exist the value is -1.
This returns the previous state of the menuitem (0, 1, or 2). The example program ignores this parameter.
Remarks :
A program must use the MF_BYPOSITION flag to specify the correct menu handle.
If the menu handle to the menu bar is specified, the top-level menu item (an item in the menu bar) is affected. To set the state of an item in a drop down menu or submenu by position, a program must specify a handle to the drop-down menu or submenu. When a program specifies the MF_BYCOMMAND flag, the system checks all items that open submenus in the menu identified by the specified menu handle. Unless duplicate menuitems are present, specifying the menu handle to the menu bar is sufficient.

HiliteMenuItem :

This function highlights or unhighlights a menuitem in a menu.
LB Syntax :

```
FUNCTION HiliteMenuItem(hwnd, hmenu, uItemHilite, uHilite)
CALLDLL #user32, "HiliteMenuItem",_
hwnd As uLong,_
hmenu As uLong,_
uItemHilite As Ulong,_
uHilite As Ulong,_
HiliteMenuItem As Long
END FUNCTION
```

Parameters :
hwnd [in] ; The handle to the window that contains the menu.
hmenu [in] ; The handle to the menu bar that contains the menuitem.
uItemHilite [in] ; The menuitem to be highlighted or unhighlighted, as determined by the uHilite parameter.
uHilite [in] ; Controls the interpretation of the uItemHilite parameter and whether the menuitem is highlighted.
This value must be (MF_BYCOMMAND or MF_BYPOSITION) and MF_HILITE or MF_UNHILITE).
MF_BYCOMMAND = 0. Uses the identifier of a menuitem.
MF_BYPOSITION = 1024. The zero-based relative position of a menuitem in a menu.
MF_UNHILITE = 0. Removes highlighting from the menu item.
MF_HILITE = 128. Highlights the menu item.
HiliteMenuItem [out] ; Return Value. An error is indicated by a value of 0.
All other values mean the function worked.
There is no the extended error information for this dll call.
Remarks :

The MF_HILITE and MF_UNHILITE flags can only be used with the HiliteMenuItem dll call; they do not work with the ModifyMenu dll call.

InsertMenuItem :

This function inserts a new menuitem in to a menu at the specified position.
LB Syntax :
FUNCTION InsertMenuItem(hMenu, uItem, fType, fState, wID, hSubMenu, hbmpChecked, hbmpUnchecked, dwTypeData$, hbmpItem)
lpmii.fMask.struct = 463 ' This value allows all members of the MENUITEMINFO structure to be set.
lpmii.fType.struct = fType
lpmii.fState.struct = fState
lpmii.wID.struct = wID
lpmii.hSubMenu.struct = hSubMenu
lpmii.hbmpChecked.struct = hbmpChecked
lpmii.hbmpUnchecked.struct = hbmpUnchecked
lpmii.dwTypeData$.struct = dwTypeData$
lpmii.cch.struct = LEN(lpmii.dwTypeData$.struct)
lpmii.hbmpItem.struct = hbmpItem
CALLDLL #user32, "InsertMenuItemA",_
hMenu AS uLong,_
uItem AS Ulong,_
fByPosition AS Long,_
lpmii AS Struct,_
InsertMenuItem AS Long
END FUNCTION
Parameters :
hMenu [in] ; The handle to the menu in which the new menu item is inserted.
uItem [in] ; The identifier or the position of the menuitem before which to insert the new menuitem. The meaning of this parameter depends on the value of fByPosition.
fByPosition [in] ; If this parameter is FALSE = 0, uItem is a menuitem identifier.
If this parameter is TRUE = 1, uItem is a menuitem position.
lpmii [in] ; The pointer to the MENUITEMINFO structure that holds the information about the new menuitem.
InsertMenuItem [out] ; Return Value. An error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks :
The program must call the DrawMenuBar dll call whenever a menu that is attached to a window changes, whether or not the menu is showing. In order for keyboard accelerators to work with bitmap or owner-drawn menuitems, the owner of the menu must process the WM_MENUCHAR message.

SetMenuDefaultItem :

This function sets the default menu item for the specified menu.

LB Syntax :
FUNCTION SetMenuDefaultItem(hMenu, uItem)
CALLDLL #user32, "SetMenuDefaultItem",_
hMenu AS uLong,_
uItem AS Ulong,_
fByPos AS Ulong,_
SetMenuDefaultItem AS Long
END FUNCTION
Parameters :
hMenu [in] ; The handle to the menu to set the default item for.
uItem [in] ; The identifier or the position of the menuitem to set or -1 for no default item. This parameter depends on the value of fByPos.
fByPos [in] ; If this parameter is FALSE = 0, uItem is a menuitem identifier.
If this parameter is TRUE = 1, uItem is a menuitem position.
SetMenuDefaultItem [out] ; Return Value. An error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks :
This function acts like CheckMenuRadioItem in that when it sets a default menuitem it clears the default flag in all the other menuitems in a menu .

SetMenuInfo :

This function sets information for a menu and/or its submenus.
LB Syntax :
FUNCTION SetMenuInfo(hmenu, fMask, dwStyle)
lpcmi.fMask.struct = fMask
lpcmi.dwStyle.struct = dwStyle
CALLDLL #user32, "SetMenuInfo",_
hmenu AS uLong,_
lpcmi AS Struct,_
SetMenuInfo AS Long
END FUNCTION
Parameters :
hmenu [in] ; The handle to a menu.
lpcmi [in] ; The pointer to a MENUINFO structure for a menu.
SetMenuInfo [out] ; Return Value. An error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks :

SetMenuItemBitmaps :

This function associates the specified bitmap with a menuitem.
Whether the menuitem is selected or clear, the system displays the appropriate bitmap next to the menuitem.

LB Syntax :
FUNCTION SetMenuItemBitmaps(hMenu, uPosition, hBitmapUnchecked, hBitmapChecked)
CALLDLL #user32, "SetMenuItemBitmaps",_
hMenu As uLong,_
uPosition As Ulong,_
uFlags As Ulong,_
hBitmapUnchecked As uLong,_
hBitmapChecked As uLong,_
SetMenuItemBitmaps As Long
END FUNCTION
Parameters :
hMenu [in] ; The handle to the menu containing the item to receive new check-mark bitmaps.
uPosition [in] ; The identifier or the position of the menuitem, as determined by the uFlags parameter.
uFlags [in] ; This specifies how the uPosition parameter is interpreted.
MF_BYCOMMAND = 0. Uses the identifier of a menuitem.
MF_BYPOSITION = 1024. Uses the zero based relative position of a menuitem in a menu.
hBitmapUnchecked [in] ; The handle to the bitmap displayed when the menuitem is not selected.
hBitmapChecked [in] ; The handle to the bitmap displayed when the menuitem is selected.
SetMenuItemBitmaps [out] ; Return Value. An error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks
If either the hBitmapUnchecked or hBitmapChecked parameter is NULL, the system displays nothing next to the menu item for the corresponding check state. If both parameters are NULL, the system displays the default check-mark bitmap when the item is selected, and removes the bitmap when the item is not selected. When the menu is destroyed, these bitmaps are not destroyed; it is up to the program to destroy them. The selected and clear bitmaps should be monochrome. The system uses the Boolean AND operator to combine bitmaps with the menu so that the white part becomes transparent and the black part becomes the menuitem color. If you use color bitmaps, the results may be undesirable. Use the GetSystemMetrics function with the CXMENUCHECK and CYMENUCHECK values to retrieve the bitmap dimensions.

SetMenuItemInfo :

This function changes information about a menuitem.
LB Syntax :
FUNCTION SetMenuItemInfo(hMenu, fMask, fType, fState, wID, hSubMenu, hbmpChecked, hbmpUnchecked, dwTypeData$, hbmpItem)
lpmii.fMask.struct = fMask
lpmii.fType.struct = fType
lpmii.fState.struct = fState
lpmii.wID.struct = wID
lpmii.hSubMenu.struct = hSubMenu
lpmii.hbmpChecked.struct = hbmpChecked
lpmii.hbmpUnchecked.struct = hbmpUnchecked
lpmii.dwTypeData$.struct = dwTypeData$

```
lpmii.cch.struct = LEN(lpmii.dwTypeData$.struct)
lpmii.hbmpItem.struct = hbmpItem
CALLDLL #user32, "SetMenuItemInfoA",_
hMenu As uLong,_
wID As Ulong,_
0 As Long,_
lpmii As Struct,_
SetMenuItemInfo As Long
END FUNCTION
```

Parameters :
hMenu [in] ; The handle to a menu that contains a menuitem.
uItem [in] ; The identifier or the position of the menuitem to change, as determined by the fByPosition parameter.
fByPosition [in] ; If this parameter is FALSE = 0, uItem is a menuitem identifier.
If this parameter is TRUE = 1, uItem is a menuitem position.
lpmii [in] ; The pointer to a MENUITEMINFO structure that holds information about a menuitem.
SetMenuItemInfo [out] ; Return Value. An error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks :
The program must call the DrawMenuBar dll call whenever a menu that is attached to a window changes, whether or not the menu is showing. In order for keyboard accelerators to work with bitmap or owner-drawn menuitems, the owner of the menu must process the WM_MENUCHAR message.

TrackPopupMenuEx :

This function displays a menu at the specified location and tracks the selection of items on the menu. This menu can be displayed anywhere on the screen.
LB Syntax :

```
CALLDLL #user32, "TrackPopupMenuEx",_
hmenu As Long,_
fuFlags As Ulong,_
x As Long,_
y As Long,_
hwnd As Long,_
lptpm AS Struct,_
r As Long
or
CALLDLL #user32, "TrackPopupMenuEx",_
hmenu As uLong,_
fuFlags As Ulong,_
x As Long,_
y As Long,_
hwnd As uLong,_
0 AS Long,_
```

r As Long
Parameters :
hmenu [in] ; The handle to the shortcut menu to be displayed.
This handle can be obtained by using the CreatePopupMenu dll call to create a new menu
or by using the GetSubMenu dll call to retrieve a handle to a menu associated with an existing menuitem.
fuFlags [in] ; Specifies function options.
Use one of the following flags to specify how the function positions the shortcut menu horizontally.
TPM_LEFTALIGN = 0. This positions the menu so that its left side is aligned with the coordinate set by the x parameter.
TPM_CENTERALIGN = 4. This centers the menu horizontally relative to the coordinate set by the x parameter.
TPM_RIGHTALIGN = 8. This positions the menu so that its right side is aligned with the coordinate set by the x parameter.
Use one of the following flags to specify how the function positions the shortcut menu vertically.
TPM_TOPALIGN = 0. This positions the menu so that its top side is aligned with the coordinate set by the y parameter.
TPM_VCENTERALIGN = 16. This centers the menu vertically relative to the coordinate set by the y parameter.
TPM_BOTTOMALIGN = 32. This positions the menu so that its bottom side is aligned with the coordinate set by the y parameter.
Use the following flags to determine the user selection without having to set up a parent window for the menu.
TPM_NONOTIFY = 128. This does not send notification messages when the user clicks on a menu item.
TPM_RETURNCMD = 256. This returns the menuitem identifier of the user's selection in the return value.
Use one of the following flags to specify which mouse button the shortcut menu tracks.
TPM_LEFTBUTTON = 0. The user can only select menuitems with the left mouse button.
TPM_RIGHTBUTTON = 2. The user can select menuitems with both the left and right mouse buttons.
Use any reasonable combination of the following flags to modify the animation of a menu.
TPM_HORNEGANIMATION. This animates the menu from right to left.
TPM_HORPOSANIMATION. This animates the menu from left to right.
TPM_NOANIMATION. This displays menu without animation.
TPM_VERNEGANIMATION. This animates the menu from bottom to top.
` TPM_VERPOSANIMATION. This animates the menu from top to bottom.
For any animation to occur, the SystemParametersInfo function must set SPI_SETMENUANIMATION.
Also, all the TPM_*ANIMATION flags, except TPM_NOANIMATION, are ignored if menu fade animation is on,
see the SPI_GETMENUFADE flag in SystemParametersInfo.
Use the TPM_RECURSE flag to display a menu when another menu is already displayed.
This is intended to support context menus within a menu.
Use one of the following flags to specify whether to accommodate horizontal or vertical alignment with the excluded rectangle.
The excluded rectangle is a portion of the screen that the menu should not overlap; it is set by lptpm.
TPM_HORIZONTAL = 0. The system tries to accommodate the horizontal alignment before the vertical alignment.
TPM_VERTICAL = 64. The system tries to accommodate the vertical alignment before the horizontal

alignment.
Windows XP: To have text layout from right-to-left, use TPM_LAYOUTRTL. By default, the text layout is left-to-right.
x [in] ; The horizontal screen coordinates of the menu.
y [in] ; The vertical screen coordinates of the menu.
hwnd [in] ; The handle to the window that owns the shortcut menu. This window receives all messages from the menu.
The window does not receive a WM_COMMAND message from the menu until the function returns.
If you specify TPM_NONOTIFY in the fuFlags parameter, the function does not send messages to the window identified by hwnd. However, you must still pass a window handle in hwnd. It can be any window handle from your program. lptpm [in] ; The pointer to a TPMPARAMS structure that specifies an area of the screen the menu should not overlap. This parameter can be 0 AS Long.
SetMenuItemInfo [out] ; Return Value. If the user cancels the menu without making a selection, or if an error occurs, then the value is zero.
If TPM_RETURNCMD is specified in the fuFlags parameter, then the value is the menuitem identifier of the menuitem that was selected.
If no menuitem was selected then the value is 0.
If TPM_RETURNCMDis not specified in the fuFlags parameter, an error is indicated by a value of 0. All other values mean it worked.
The GetLastError dll call must be used to get the extended error information.
Remarks :
Call GetSystemMetrics(SM_MENUDROPALIGNMENT) to determine the correct horizontal alignment flag (TPM_LEFTALIGN or TPM_RIGHTALIGN) and/or horizontal animation direction flag (TPM_HORPOSANIMATION or TPM_HORNEGANIMATION) to pass to TrackPopupMenu or TrackPopupMenuEx.
To display a context menu for a notification icon, the current window must be the foreground window before the program calls TrackPopupMenu or TrackPopupMenuEx. Otherwise, the menu will not disappear when the user clicks outside of the menu or the window that created the menu .

Table of Contents

Appendix B ; The Windows API Structures.

Contents :
- MENUINFO
- MENUITEMINFO
- TPMPARAMS

MENUINFO.

This structure holds information about a menu.
LB Syntax :

```
STRUCT MENUINFO,_
cbSize As Ulong,_
fMask As Ulong,_
dwStyle As Ulong,_
cyMax As Ulong,_
hbrBack As Long,_
dwContextHelpID As Ulong,_
```

dwMenuData As Ulong
Members :
cbSize [in] ; The size of the structure, in bytes. This must be MENUINFO.cbSize.struct = LEN(MENUINFO.struct).
fMask [in] ; The members to set or get. This can be one or more of the following values.
MIM_APPLYTOSUBMENUS = 2147483648. The settings apply to the menu and all of its submenus. This is only valid with the SetMenuInfo dll call.
MIM_BACKGROUND = 2. This sets or gets the hbrBack member.
MIM_HELPID = 4. This sets or gets the dwContextHelpID member.
MIM_MAXHEIGHT = 1. This sets or gets the cyMax member.
MIM_MENUDATA = 8. This sets or gets the dwMenuData member.
MIM_STYLE = 16. This sets or gets the dwStyle member.
dwStyle [in/out] ; Style of the menu. This can be one or more of the following values.
MNS_AUTODISMISS = 268435456. Menu automatically ends when mouse is outside the menu for approximately 10 seconds.
MNS_CHECKORBMP = 67108864. The same space is reserved for the check mark and the bitmap. If the menuitem is checked the checkmark is drawn and the bitmap is not.
MNS_DRAGDROP = 536870912. Menu items are OLE drop targets or drag sources. Menu owner receives WM_MENUDRAG and WM_MENUGETOBJECT messages.
MNS_MODELESS = 1073741824. Menu is modeless; that is, there is no menu modal message loop while the menu is active.
MNS_NOCHECK = 2147483648. There will be no space reserved to the left of a menuitem for the checkmark. The menuitem can still be selected, but the checkmark will not show.
MNS_NOTIFYBYPOS = 134217728. Menu owner receives a WM_MENUCOMMAND message instead of a WM_COMMAND message when the user makes a selection.
cyMax [in/out] ; Maximum height of the menu in pixels. When the menuitems exceed the space available, scroll bars will show. The default (0) is the screen height.
hbrBack [in/out] ; Brush to use for the menu's background.
dwContextHelpID [in/out] ; The context help identifier. This is the same value used in GetMenuContextHelpId and SetMenuContextHelpId.
dwMenuData [in/out] ; An application-defined value.
Remarks :
This will allow you to do many things to many in the first parameter is set the size of the structure the second parameter must set MIM_STYLE to set the third parameter MIM_APPLYTOSUBMENUS and MIM_MAXHEIGHT seemed to work OK the others have not been tested thoroughly

---

MENUITEMINFO.

This structure holds information about a menuitem.
Syntax :
STRUCT MENUITEMINFO,_
cbSize As Ulong,_
fMask As Ulong,_

fType As Ulong,_
fState As Ulong,_
wID As Ulong,_
hSubMenu As uLong,_
hbmpChecked As uLong,_
hbmpUnchecked As uLong,_
dwItemData As Ulong,_
dwTypeData$ As Ptr,_
cch As Ulong,_
hbmpItem As Long
Members :
cbSize [in] ; The size of structure, in bytes. This must be MENUITEMINFO.cbSize.struct = LEN(MENUITEMINFO.struct).
fMask [in] ; Members to retrieve or set. This can be one or more of these values. [463] and [64 or 128 or 256] or 99,163, or 291
MIIM_BITMAP = 128. This sets or gets the hbmpItem member. Windows 98/Me, Windows 2000/XP only.
MIIM_CHECKMARKS = 8. This sets or gets the hbmpChecked and hbmpUnchecked members.
MIIM_DATA = 32. This sets or gets the dwItemData member.
MIIM_FTYPE = 256. This sets or gets the fType member. Windows 98/Windows Me, Windows 2000/Windows XP only.
MIIM_ID = 2. This sets or gets the wID member.
MIIM_STATE = 1. Retrieves or sets the fState member.
MIIM_STRING = 64. This sets or gets the dwTypeData member. Windows 98/Windows Me, Windows 2000/Windows XP only.
MIIM_SUBMENU = 4. This sets or gets the hSubMenu member.
MIIM_TYPE = 16. This sets or gets the fType and dwTypeData members.
Windows 98/Me, Windows 2000/XP: MIIM_TYPE is replaced by MIIM_BITMAP, MIIM_FTYPE, and MIIM_STRING.
fType [in/out] ; Menu item type. This can be one or more of the following values.
The MFT_BITMAP, MFT_SEPARATOR, and MFT_STRING values cannot be combined with one another.
Set fMask to MIIM_TYPE to use fType.Windows 98/Me and Windows 2000/XP, fType is used only if fMask has a value of MIIM_FTYPE.
MFT_BITMAP = 4. Displays the menu item using a bitmap. The low-order word of the dwTypeData member is the bitmap handle, and the cch member is ignored. Windows 98/Me, Windows 2000/XP:Â MFT_BITMAP is replaced by MIIM_BITMAP and hbmpItem.
MFT_MENUBARBREAK = 32. Places the menuitem on a new line (for a menu bar) or in a new column (for a drop-down menu, submenu, or shortcut menu). For a drop-down menu, submenu, or shortcut menu, a vertical line separates the new column from the old.
MFT_MENUBREAK = 64. Places the menuitem on a new line (for a menu bar) or in a new column (for a dropdown menu, submenu, or shortcut menu). For a drop-down menu, submenu, or shortcut menu, the columns are not separated by a vertical line.
MFT_OWNERDRAW = 256. Assigns responsibility for drawing the menu item to the window that owns the menu. The window receives a WM_MEASUREITEM message before the menu is displayed for the first time, and a WM_DRAWITEM message whenever the appearance of the menu item must be updated.

If this value is specified, the dwTypeData member contains an application-defined value.
MFT_RADIOCHECK = 512. Displays selected menuitems using a radiobutton instead of a checkmark if the hbmpChecked member is NULL.
MFT_RIGHTJUSTIFY = 16384. Right-justifies the menu item and any subsequent items. This value is valid only if the menu item is in a menu bar.
MFT_RIGHTORDER = 8192. This specifies that menus cascade right-to-left (the default is left-to-right). This is used to support right-to-left languages, such as Arabic and Hebrew.
MFT_SEPARATOR = 2048. Specifies that the menu item is a separator.
A menuitem separator appears as a horizontal dividing line. The dwTypeData and cch members are ignored. This value is valid only in a drop-down menu, submenu, or shortcut menu.
MFT_STRING = 0. Displays the menu item using a text string.
The dwTypeData member is the pointer to a null-terminated string, and the cch member is the length of the string. Windows 98/Me, Windows 2000/XP:Â MFT_STRING is replaced by MIIM_STRING.
fState [in/out] ; Menu item state. This can be one or more of these values.
MFS_CHECKED = 8. Checks the menu item.
MFS_DEFAULT = 4096. Specifies that the menu item is the default. A default menuitem is displayed in bold.
MFS_ENABLED = 0. Enables the menu item so that it can be selected.
MFS_GRAYED = 1. Disables the menu item and grays it so that it cannot be selected.
MFS_DISABLED = 2. Disables the menu item and does not gray it so that it cannot be selected.
MFS_HILITE = 128. Highlights the menu item.
MFS_UNCHECKED = 0. Unchecks the menu item. For more information about clear menu items, see the hbmpChecked member.
MFS_UNHILITE = 0. Removes the highlight from the menu item. This is the default state.
wID [in/out] ; Application-defined 16-bit value that identifies the menu item.
hSubMenu [in/out] ; The handle to a drop down menu or a submenu associated with the menuitem. If the menu item is not an item that opens a drop-down menu or submenu, this member is NULL.
hbmpChecked [in/out] ; The handle to a bitmap to display next to the item if it is selected. If this is NULL, a default bitmap is used. If the MFT_RADIOCHECK type value is specified, the default bitmap is a bullet. Otherwise, it is a checkmark.
hbmpUnchecked [in/out] ; The handle to a bitmap to display next to the item if it is not selected. If this is NULL, no bitmap is used.
dwItemData [in/out] ; A program defined value associated with the menuitem.
dwTypeData [in/out] ; The content of the menuitem.
The meaning of this member depends on the value of fType and is used only if the MIIM_TYPE flag is set in the fMask member. To retrieve a menuitem of type MFT_STRING, first find the size of the string by setting the dwTypeData member of MENUITEMINFO to NULL and then calling GetMenuItemInfo. The value of cch+1 is the size needed. Then allocate a buffer of this size, place the pointer to the buffer in dwTypeData, increment cch, and call GetMenuItemInfo once again to fill the buffer with the string. If the retrieved menuitem is of some other type, then GetMenuItemInfo sets the dwTypeData member to a value whose type is specified by the fType member. When using with the SetMenuItemInfo function, this member should contain a value whose type is specified by the fType member. Windows 98/Me and Windows 2000/XP: dwTypeData is used only if the MIIM_STRING flag is set in the fMask member.
cch [in/out] ; The length of the menuitem text, in bytes, when information is received about a menu item of the MFT_STRING type. However, cch is used only if the MIIM_TYPE flag is set in the fMask member and is zero otherwise. Also, cch is ignored when the content of a menuitem is set by calling

SetMenuItemInfo.
Note that, before calling GetMenuItemInfo, the application must set cch to the length of the buffer pointed to by the dwTypeData member. If the retrieved menuitem is of type MFT_STRING (as indicated by the fType member), then GetMenuItemInfo changes cch to the length
of the menuitem text. If the retrieved menuitem is of some other type, GetMenuItemInfo sets the cch field to zero. Windows 98/Me, Windows 2000/XP, The cch member is used when the MIIM_STRING flag is set in the fMask member.
hbmpItem [in/out] ; Windows 98/Me, Windows 2000/XP only, The handle to the bitmap to be displayed, or it can be one of the values in the following table.
HBMMENU_CALLBACK = -1. A bitmap that is drawn by the window that owns the menu.
The program must process the (WM_MEASUREITEM=44) and (WM_DRAWITEM=43) messages.
HBMMENU_MBAR_CLOSE = 5. Close button for the menu bar.
HBMMENU_MBAR_CLOSE_D = 6. Disabled close button for the menu bar.
HBMMENU_MBAR_MINIMIZE = 3. Minimize button for the menu bar.
HBMMENU_MBAR_MINIMIZE_D = 7. Disabled minimize button for the menu bar.
HBMMENU_MBAR_RESTORE = 2. Restore button for the menu bar.
HBMMENU_POPUP_CLOSE = 8. Close button for the submenu.
HBMMENU_POPUP_MAXIMIZE = 10. Maximize button for the submenu.
HBMMENU_POPUP_MINIMIZE = 11. Minimize button for the submenu.
HBMMENU_POPUP_RESTORE = 9. Restore button for the submenu.
HBMMENU_SYSTEM = 1. Windows icon or the icon of the window specified in dwItemData.
Remarks :
The MENUITEMINFO structure is used with the GetMenuItemInfo, InsertMenuItem, and SetMenuItemInfo functions.
This Structure is the best way to access menu items and alter the them the first parameter is set the size and structure and only needs to be dumb ones done wants wants wants to me one in the program settings and her those in Herman the second parameter to foreigners 63 463 was set or get all other parameters

---

TPMPARAMS.

This structure specifies an area of the screen the menu should not overlap.
LB Syntax :
STRUCT TPMPARAMS,_
cbSize As Ulong,_
rcExclude.Left As Long,_
rcExclude.Top As Long,_
rcExclude.Right As Long,_
rcExclude.Bottom As Long
Members :
cbSize [in] ; The size of structure, in bytes. This must be TPMPARAMS.cbSize.struct = LEN(TPMPARAMS.struct).
rcExclude.Left [in] ; This specifies the left edge of the excluded area of the screen.
rcExclude.Top [in] ; This specifies the top edge of the excluded area of the screen.
rcExclude.Right [in] ; This specifies the right edge of the excluded area of the screen.
rcExclude.Bottom [in] ; This specifies the bottom edge of the excluded area of the screen.

Remarks:

Table of Contents

Appendix C ; The CFC API Popup Menu Tree.

menuitem = wID

Preferences > = 28

Size > = 6
Huge = 21
Large = 22
Medium = 23
Small = 24

Colors > = 7
Current Date > = 81
black = 101
blue = 102
brown = 103
buttonface = 104
cyan = 105
darkblue = 106
darkcyan = 107
darkgray = 108
darkgreen = 109
darkpink = 110
darkred = 111
green = 112
lightgray = 113
pink = 114
red = 115
white = 116
yellow = 117

Face B.G. > = 82
black = 121
blue = 122
brown = 123
buttonface = 124
cyan = 125
darkblue = 126
darkcyan = 127
darkgray = 128
darkgreen = 129
darkpink = 130
darkred = 131
green = 132

lightgray = 133
pink = 134
red = 135
white = 136
yellow = 137

Hour Hand > = 83
black = 141
blue = 142
brown = 143
buttonface = 144
cyan = 145
darkblue = 146
darkcyan = 147
darkgray = 148
darkgreen = 149
darkpink = 150
darkred = 151
green = 152
lightgray = 153
pink = 154
red = 155
white = 156
yellow = 157

Minute Hand > = 84
black = 161
blue = 162
brown = 163
buttonface = 164
cyan = 165
darkblue = 166
darkcyan = 167
darkgray = 168
darkgreen = 169
darkpink = 170
darkred = 171
green = 172
lightgray = 173
pink = 174
red = 175
white = 176
yellow = 177

Second Hand > = 85
black = 181

blue = 182
brown = 183
buttonface = 184
cyan = 185
darkblue = 186
darkcyan = 187
darkgray = 188
darkgreen = 189
darkpink = 190
darkred = 191
green = 192
lightgray = 193
pink = 194
red = 195
white = 196
yellow = 197

Centre Edge > = 86
black = 201
blue = 202
brown = 203
buttonface = 204
cyan = 205
darkblue = 206
darkcyan = 207
darkgray = 208
darkgreen = 209
darkpink = 210
darkred = 211
green = 212
lightgray = 213
pink = 214
red = 215
white = 216
yellow = 217
Centre Fill > = 87
black = 221
blue = 222
brown = 223
buttonface = 224
cyan = 225
darkblue = 226
darkcyan = 227
darkgray = 228
darkgreen = 229
darkpink = 230

darkred = 231
green = 232
lightgray = 233
pink = 234
red = 235
white = 236
yellow = 237

AM / PM > = 88
black = 241
blue = 242
brown = 243
buttonface = 244
cyan = 245
darkblue = 246
darkcyan = 247
darkgray = 248
darkgreen = 249
darkpink = 250
darkred = 251
green = 252
lightgray = 253
pink = 254
red = 255
white = 256
yellow = 257

Window B.G. > = 89
black = 261
blue = 262
brown = 263
buttonface = 264
cyan = 265
darkblue = 266
darkcyan = 267
darkgray = 268
darkgreen = 269
darkpink = 270
darkred = 271
green = 272
lightgray = 273
pink = 274
red = 275
white = 276
yellow = 277

Dial B.G. > = 90
black = 281
blue = 282
brown = 283
buttonface = 284
cyan = 285
darkblue = 286
darkcyan = 287
darkgray = 288
darkgreen = 289
darkpink = 290
darkred = 291
green = 292
lightgray = 293
pink = 294
red = 295
white = 296
yellow = 297

Dial O.D. > = 91
black = 301
blue = 302
brown = 303
buttonface = 304
cyan = 305
darkblue = 306
darkcyan = 307
darkgray = 308
darkgreen = 309
darkpink = 310
darkred = 311
green = 312
lightgray = 313
pink = 314
red = 315
white = 316
yellow = 317

5m Increments > = 92
black = 321
blue = 322
brown = 323
buttonface = 324
cyan = 325
darkblue = 326
darkcyan = 327

darkgray = 328
darkgreen = 329
darkpink = 330
darkred = 331
green = 332
lightgray = 333
pink = 334
red = 335
white = 336
yellow = 337

Dial Numbers > = 93
black = 341
blue = 342
brown = 343
buttonface = 344
cyan = 345
darkblue = 346
darkcyan = 347
darkgray = 348
darkgreen = 349
darkpink = 350
darkred = 351
green = 352
lightgray = 353
pink = 354
red = 355
white = 356
yellow = 357

1m Increments > = 94
black = 361
blue = 362
brown = 363
buttonface = 364
cyan = 365
darkblue = 366
darkcyan = 367
darkgray = 368
darkgreen = 369
darkpink = 370
darkred = 371
green = 372
lightgray = 373
pink = 374
red = 375

white = 376
yellow = 377

Dial I.D. > = 95
black = 381
blue = 382
brown = 383
buttonface = 384
cyan = 385
darkblue = 386
darkcyan = 387
darkgray = 388
darkgreen = 389
darkpink = 390
darkred = 391
green = 392
lightgray = 393
pink = 394
red = 395
white = 396
yellow = 397

Options > = 29

Examples > = 13
0, Enabled = 41
1, Disabled = 42
2, Disabled = 43

0, Enabled > = 46
{EMPTY} = 19

1, Disabled > = 47
{EMPTY} = 19

2, Disabled > = 48
{EMPTY} = 19

Load .bmp Only (resets menu) = 9
MNS_CHECKORBMP = 10
Disable Examples Menu = 60
Unmark Default Items = 61
Remove Checkmark BMP's = 62
Remove MenuItem BMP's = 63
Start Registry Editor = 95

Help = 99
About = 14
Close = 17