Changing the Icon

by -
Alyce

Table of Contents

What is an icon?

A program's icon is the small picture that appears in the upper left corner of the titlebar on a window. It also appears in the taskbar as a slightly larger picture. The icon appears in your file listings if you are using one of the icon views, too.

If you have a registered copy of Liberty BASIC, you can use the runtime engine to distribute executable versions of your programs. Liberty BASIC has an icon editor that allows you to edit, create and save icons to disk. It also allows you to embed them in the runtime engine.

Why change the icon at runtime?

There are a couple of reasons that you might want to change the program's icon at runtime. The Liberty BASIC runtime engine can only accept a 16-color icon. If you attempt to embed a 256-color icon, it won't work properly. If you'd like to use a 256-color icon with your program, you can change its icon at runtime using a file on disk that you distribute along with your program.

You might also want to change the icon at runtime to reflect differences in your program. Perhaps your program contains more than one window and you'd like each to have its own icon. By default all windows in a program use the icon from the runtime engine. You can set the icon in each window to be different if you'd like.

Getting the Instance Handle

The first step in using the API to change an icon is to obtain the instance handle of the window whose icon is to be changed. This argument can be passed as a null value in the function that loads the icon from disk, but we'll document the proper way to get a window's instance handle for the sake of completeness. We get the instance handle with GetWindowLongA . There are several long values associated with windows. The first argument is the handle of the window and the next argument is the flag _GWL_HINSTANCE to tell the function which long value that we want to retrieve. Do it like this:

```
'get window handle:
 hWin = hwnd(#win)

 CallDLL #user32, "GetWindowLongA",_
 hWin As ulong,_            'window handle
 _GWL_HINSTANCE As long,_   'flag for type
 hInstance As ulong         'returns instance handle
```

Loading an Icon from Disk

The next step is to load the icon image into memory from a disk file. You'll need to pass the disk filename to the LoadImageA function. If the icon file is in the program's DefaultDir$ you don't need to include any path information. You must also specify the type of image you are loading. Use the type _IMAGE_ICON. If you pass arguments of 0 for the width and height of the image the actual size of the image is used. If you pass numbers for these arguments you can change the size of an image, but don't do that for this method! The function returns the handle of the icon.

//see also: [[Load+Bitmap+to+Size|Load a Bitmap to Desired Size]]

```
flag=_LR_LOADFROMFILE or _LR_DEFAULTSIZE
file$ = "your_icon_name.ico"
calldll #user32, "LoadImageA",_
hInstance as ulong,_   'instance handle
file$ as ptr,_         'filename
_IMAGE_ICON as long,_  'type of image
```

```
    0 as long,_             'desired width
    0 as long,_             'desired height
    flag as long,_          'load option
    hIcon as ulong          'icon handle
```

Setting the Icon of the Window

The function SendMessageA is used to set the icon on the window. The function needs to know the handle of the window and the message to send to the window, which is _WM_SETICON. It also needs to know which icon to be set; the big icon, the small icon, or both. The handle of the icon is also passed into the function.

The small icon in the titlebar can be changed by setting the bit for ICON_SMALL. Since Liberty BASIC doesn't recognize this Windows constant, we create it ourselves in the code as ICON.SMALL. The value is 0. If we also want to set the big icon, we OR the value for the small icon together with the value for the big icon, which is 1.

```
    ICON.SMALL=0
    ICON.BIG=1
    wParam = ICON.SMALL OR ICON.BIG

    CallDLL #user32, "SendMessageA",_
    hWnd As ulong,_          'window handle
    _WM_SETICON As long,_     'message
    wParam As long,_        'flag for which icon(s) to set
    hIcon As ulong,_         'icon handle
    re As long
```

Destroying the Icon

When the program closes, we remove the icon from memory with a call to DestroyIcon.

```
    calldll #user32, "DestroyIcon",_
    hIcon as ulong,_ 'handle of icon
    res as long
```

DEMO

This demo is from The Liberty BASIC 4 Companion - used by permission. Please do not republish this demo, but you may use the methods in your own code without credit to the author.

The demo uses a filedialog to allow the user to select an icon on disk. You can hard code the name of the icon instead. The demo wraps the API calls into Liberty BASIC functions.

```
'load an icon and use
'as small icon in titlebar
'and large icon in task list

nomainwin
Open "Change an Icon" for window_nf as #1
#1 "trapclose [quit]"

filedialog "Open","*.ico",ifile$
if ifile$="" then [quit]

h=hwnd(#1)  'window handle
ICON.SMALL=0
ICON.BIG=1

hIconWin=LoadIcon(h,ifile$)
call SendMessageLong h, _WM_SETICON,ICON.SMALL or ICON.BIG,hIconWin

wait

[quit]
ret=DestroyIcon(hIconWin)
close #1:end

Function LoadIcon(hWin,file$)
    hInst=GetWindowLong(hWin,_GWL_HINSTANCE)

    flag=_LR_LOADFROMFILE or _LR_DEFAULTSIZE
    calldll #user32, "LoadImageA",_
    hInst as ulong,_    'instance handle
    file$ as ptr,_      'filename
    _IMAGE_ICON as long,_   'type of image
    0 as long,_         'desired width
    0 as long,_         'desired height
    flag as long,_      'load option
    LoadIcon as ulong   'icon handle
  End Function

Function DestroyIcon(hIcon)
```

```
    calldll #user32, "DestroyIcon",_
    hIcon as ulong,_ 'handle of icon
    DestroyIcon as long
    End Function

Function GetWindowLong(hW, type)
    CallDLL #user32, "GetWindowLongA",_
    hW As ulong,_    'window handle
    type As long,_   'flag for type
    GetWindowLong As ulong
    End Function

Sub SendMessageLong hWnd,msg,w,l
    CallDLL #user32, "SendMessageA",_
    hWnd As ulong,_  'window handle
    msg As long,_    'message
    w As long,_      'wParam
    l As long,_      'lParam
    re As long
    End Sub
```