This is the source code for a utility made in LB, that makes it easy to associate .BAS files to Liberty BASIC, no matter the version.

It automatically find LB installs, assuming they're in default locations, and gives the option to associate any of them to the .BAS file extension.

It also has a manual selection option, to allow it to associate any EXE(or any copy of LB in a non-standard location) to the BAS file extension.

```
call InitRegistry
gosub [findLBexes]

dim LBExeList$(numLBexes)
For x = 1 to numLBexes
    LBExeList$(x) = word$(LBExes$, x, "|")
Next x

'Form created with the help of Freeform 3 v07-15-08
'Generated on Jan 15, 2016 at 22:13:48


[setup.m.Window]

    nomainwin

    '-----Begin code for #m

    WindowWidth = 520
    WindowHeight = 245
    UpperLeftX=int((DisplayWidth-WindowWidth)/2)
    UpperLeftY=int((DisplayHeight-WindowHeight)/2)


    '-----Begin GUI objects code

    button #m.btnFindExe,"Locate custom EXE",[findEXE], UL,  40,
172, 140,  25
    TextboxColor$ = "white"
    textbox #m.tbExePath,  40, 132, 415,  25
    button #m.btnSetAssociation,"Set Association",[
setAssociation], UL, 350, 172, 103,  25
    ListboxColor$ = "white"
    listbox #m.lbExeList, LBExeList$(), [
selectListEntry],   40,  17, 415, 100
```

```
    '-----End GUI objects code

    open "Set BAS file association" for window as #m
    print #m, "font ms_sans_serif 10"
    #m.btnSetAssociation, "!disable"
    #m.lbExeList, "singleclickselect"
    #m, "trapclose [quit.m]"


[m.inputLoop]   'wait here for input event
    wait

[selectListEntry]
    #m.lbExeList, "selection? LBPath$"
    #m.tbExePath, LBPath$
    #m.btnSetAssociation, "!enable"
    wait

[findEXE]   'Perform action for the button named 'btnFindExe'

    'Insert your own code here

    filedialog "Locate LB exe...", "*.exe", LBPath$
    if LBPath$ = "" then
        #m.tbExePath, "<no EXE file selected>"
        #m.btnSetAssociation, "!disable"
    else
        #m.btnSetAssociation, "!enable"
    end if

    #m.tbExePath, LBPath$

    wait


[setAssociation]
'Perform action for the button named 'btnSetAssociation'

    'Insert your own code here
    assocPath$ = chr$(34) + LBPath$ + chr$(34) + " " + chr$(34) +
"%1" + chr$(34)
    a = RegOpenKeyEx(_HKEY_CLASSES_ROOT, ".bas", 0, _KEY_READ, hBas)
    If a <> 0 then
        goto [skipBackup]
    end if
```

```
    bufSize = 0
[bufferLoop]
    buf$ = space$(bufSize)

    a = RegQueryValueEx(hBas, "", buf$, bufSize)
    if a = ERROR.MORE.DATA then [bufferLoop]

    if a <> 0 then
        a = RegCloseKey(hBas)
        goto [skipBackup]
        wait
    end if

    originalBasAssocation$ = trim$(buf$)

[skipBackup]
    a = RegCreateKeyEx(_HKEY_CURRENT_USER,
"Software\Classes\LibertyBASIC.BasFile\shell\open\command",_
        0, _KEY_ALL_ACCESS, hCommand)

    If a <> 0 then
        ret = FormatSystemErrorMessage(a, formattedMessage$)
        errMsg$ = "Registry error" + chr$(13) + _

"Unable to open HKCU\Software\Classes\LibertyBASIC.BasFile\shell\open\
command for writing."
        errMsg$ = errMsg$ + chr$(13) + chr$(13) +
"RegCreateKeyEx() returned ";a;":"
        errMsg$ = errMsg$ + chr$(13) + formattedMessage$
        notice errMsg$
        wait
    end if

    a = RegSetValueEx(hCommand, "", assocPath$)
    If a <> 0 then
        ret = FormatSystemErrorMessage(a, formattedMessage$)
        errMsg$ = "Registry error" + chr$(13) + _

"Unable to write new association to HKCU\Software\Classes\LibertyBASIC
.BasFile\shell\open\command\(default)"
        errMsg$ = errMsg$ + chr$(13) + chr$(13) +
"RegSetValueEx() returned ";a;":"
        errMsg$ = errMsg$ + chr$(13) + formattedMessage$
        notice errMsg$
```

```
        a = RegCloseKey(hCommand)
        wait
    end if

    a = RegCloseKey(hCommand)

    a = RegCreateKeyEx(_HKEY_CURRENT_USER,
"Software\Classes\.bas", 0, _KEY_ALL_ACCESS, hBas)
    If a <> 0 then
        ret = FormatSystemErrorMessage(a, formattedMessage$)
        errMsg$ = "Registry error" + chr$(13) +
"Unable to open HKCU\Software\Classes\.bas for writing."
        errMsg$ = errMsg$ + chr$(13) + chr$(13) +
"RegCreateKeyEx() returned ";a;":"
        errMsg$ = errMsg$ + chr$(13) + formattedMessage$
        notice errMsg$
    End If

    a = RegSetValueEx(hBas, "", "LibertyBASIC.BasFile")
    If a <> 0 then
        ret = FormatSystemErrorMessage(a, formattedMessage$)
        errMsg$ = "Registry error" + chr$(13) + _

"Unable to write new association identifier to HKCU\Software\Classes\.
bas\(default)."
        errMsg$ = errMsg$ + chr$(13) + chr$(13) +
"RegSetValueEx() returned ";a;":"
        errMsg$ = errMsg$ + chr$(13) + formattedMessage$
        notice errMsg$

        a = RegCloseKey(hBas)
        wait
    End If

    a = RegCloseKey(hBas)

    Call SHNotifyAssocChange

    Notice "New association set!"
    wait

[quit.m] 'End the program
    call EndRegistry
    close #m
    end
```

```
'================================================
'          SUBS/FUNCTIONS BELOW
'================================================

[findLBexes]
CSIDL.PROGRAMFILES = 38
programFilesName$ =  GetFileName$(GetSpecialFolder$(
CSIDL.PROGRAMFILES))

'EXE names to search for
searchExes$ = "liberty.exe lbpro.exe lbworkshop.exe jbasic.exe"
driveNum = 1
driveLetter$ = word$(Drives$, driveNum) + "\"
dim info$(10, 10)

[nextDrive]
files driveLetter$, info$()

numFiles = val(info$(0,0))
numFolders = val(info$(0,1))

searchPath$ = ""

'Confirm that the folder <driveLetter>\<programFilesName> exists
if numFolders = 0 then [skipSearchFolder]
for x = numFiles+1 to (numFiles+numFolders)
    folderName$ = info$(x, 1)
    if folderName$ = programFilesName$ then
        searchPath$ = driveLetter$ + folderName$
    end if
next x

print "searchPath$ = ";searchPath$

'Search through <programFilesName> for LB-related program folders

LBFolderList$ = ""
numLBFolders = 0
if searchPath$ <> "" then
    files searchPath$, info$()
    numFiles = val(info$(0,0))
    numFolders = val(info$(0,1))

    if numFolders = 0 then [skipSearchFolder]
```

```
    for x = numFiles+1 to (numFiles+numFolders)
        folderName$ = info$(x, 1)

        foundLBfolder = 0
        if left$(folderName$, 13) = "Liberty BASIC" then
 foundLBfolder = 1
        if left$(folderName$, 10) = "Just BASIC" then
 foundLBfolder = 1
        if folderName$ = "LB Workshop" then foundLBfolder = 1

        if foundLBfolder = 1 then
            LBFolderList$ = LBFolderList$ + searchPath$ + "\" +
 folderName$ + "|"
            numLBFolders = numLBFolders + 1
        end if
    next x
end if

print LBFolderList$


'For each LB-related program folder, find the EXE name
if numLBFolders = 0 then [skipSearchFolder]
for x = 1 to numLBFolders
    searchPath$ = word$(LBFolderList$, x, "|")

    files searchPath$, info$()
    numFiles = val(info$(0, 0))

    if numFiles = 0 then [doNextFolder]
    For y = 1 to numFiles
        if instr(searchExes$, info$(y, 0)) > 0 then
            LBExes$ = LBExes$ + searchPath$ + "\" + info$(y, 0) + "|"
            numLBexes = numLBexes + 1
        end if
    next y

    [doNextFolder]
next x
[skipSearchFolder]

driveNum = driveNum + 1
driveLetter$ = word$(Drives$, driveNum) + "\"
if driveLetter$ <> "\" then [nextDrive]

return
```

```
[theEnd]
end

Function GetFileName$(fullPath$)
    lenFullPath = len(fullPath$)

    For x = lenFullPath to 1 step -1
        if mid$(fullPath$, x, 1) = "\" then
            GetFileName$ = mid$(fullPath$, x+1)
            goto [skip]
        end if
    next x
    [skip]
End Function

Function GetSpecialFolder$(CSIDL)
    struct IDL, _
        cb   As uLong, _
        abID As short
    calldll #shell32, "SHGetSpecialFolderLocation",_
        0     as ulong, _
        CSIDL as ulong, _
        IDL   as struct,_
        ret   as ulong
    if ret=0 then
        Path$ = Space$(_MAX_PATH)
        id = IDL.cb.struct
        calldll #shell32, "SHGetPathFromIDListA",_
            id    as ulong, _
            Path$ as ptr, _
            ret   as ulong
        GetSpecialFolder$ = trim$(Path$)
    end if
    if GetSpecialFolder$ = "" then
 GetSpecialFolder$ = "Not Applicable"
End Function

Sub SHNotifyAssocChange
    SHCNE.ASSOCCHANGED = hexdec("08000000")
    SHCNF.IDLIST = 0

    CallDLL #shell32, "SHChangeNotify",_
    SHCNE.ASSOCCHANGED as long,_
    SHCNF.IDLIST as long,_
    0 as long,_
```

```
    0 as long,_
    ret as void
End Sub

Function FormatSystemErrorMessage(code, byref buffer$)
    bufLen = (1024 * 64) - 1
    buffer$ = space$(bufLen)

    CallDLL #kernel32, "FormatMessageA",_
    _FORMAT_MESSAGE_FROM_SYSTEM as long,_
    0 as long,_
    code as long,_
    0 as long,_
    buffer$ as ptr,_
    bufLen as long,_
    0 as long,_
    FormatSystemErrorMessage as long

    buffer$ = trim$(buffer$)
End Function

Function GetLastError()
    CallDLL #kernel32, "GetLastError",_
    GetLastError as long
End Function

Sub InitRegistry
    Open "advapi32" for DLL as #advapi32
    Global ERROR.MORE.DATA : ERROR.MORE.DATA = 234
End Sub

Sub EndRegistry
    close #advapi32
End Sub

Function RegCreateKeyEx(hKey, subKey$, dwOptions,
 samDesired, byref phkResult)
    struct res, a as ulong

    CallDLL #advapi32, "RegCreateKeyExA",_
    hKey as ulong,_
    subKey$ as ptr,_
    0 as long,_             'Reserved, must be 0.
    0 as ulong,_            'User-defined class type of key.
    _                       'Very unlikely to be used, so 0.
    dwOptions as long,_
```

```
    samDesired as long,_
    0 as ulong,_
'lpSecurityAttributes, used for setting permissions on
    _
'the key, among other things. Unlikely to be used.
    res as struct,_
    0 as ulong,_
'lpDisposition, tells us if the key was opened or created.
    _                       'Again, unlikely to be used, so 0.
    RegCreateKeyEx as long

    phkResult = res.a.struct
End Function

'For ease of function use, all registry keys will be strings.
Function RegSetValueEx(hKey, valueName$, data$)
    cbSize = len(data$)
    CallDLL #advapi32, "RegSetValueExA",_
    hKey as ulong,_
    valueName$ as ptr,_
    0 as long,_             'Reserved.
    _REG_SZ as long,_       'Always string.
    data$ as ptr,_
    cbSize as long,_
    RegSetValueEx as long
End Function

Function RegOpenKeyEx(hKey, subKey$, dwOptions, samDesired, byref
 phkResult)
    struct res, a as ulong

    CallDLL #advapi32, "RegOpenKeyExA",_
    hKey as ulong,_
    subKey$ as ptr,_
    0 as long,_             'Reserved, must be 0.
    0 as ulong,_            'User-defined class type of key.
    _                       'Very unlikely to be used, so 0.
    dwOptions as long,_
    samDesired as long,_
    RegCreateKeyEx as long

    phkResult = res.a.struct
End Function

Function RegQueryValueEx(hKey, valueName$, byref data$, byref bufSize)
    struct a, size as long
```

```
    a.size.struct = bufSize

    CallDLL #advapi32, "RegQueryValueExA",_
    hKey as ulong,_
    valueName$ as ptr,_
    0 as long,_             'Reserved.
    0 as ulong,_
'Datatype.  Not used, this function only uses REG_SZ.
    data$ as ptr,_
    a as struct,_
    RegQueryValueEx as long

    bufSize = a.size.struct
End Function

Function RegDeleteValue(hKey, valueName$)
    CallDLL #advapi32, "RegDeleteValueA",_
    hKey as ulong,_
    valueName$ as ptr,_
    RegDeleteValue as long
End Function

Function RegDeleteKey(hKey, keyName$)
    CallDLL #advapi32, "RegDeleteKeyA",_
    hKey as ulong,_
    keyName$ as ptr,_
    RegDeleteKey as long
End Function

Function RegCloseKey(hKey)
    CallDLL #advapi32, "RegCloseKey",_
    hKey as ulong,_
    RegCloseKey as long
End Function
```