The ABCs of APIs Lesson 7
Passing Numbers by Reference

Table of Contents

In Lesson 5 we learned how to pass a string variable "by reference" so that the API function can modify the contents. This is done by passing a memory pointer to the variable. Liberty BASIc has no type that allows us to pass a numeric variable "by reference."

Passing a Number by Reference

We can pass a numeric variable "by reference" by including it in a STRUCT and passing the STRUCT into the API call. A STRUCT, like a PTR is a pointer to a memory address.

Important Note

This article and the demonstration program are meant to explain passing numeric variables "by reference." The MCI API is used in the demonstration. MCI stands for Media Control Interface, a high-level API for controlling multimedia devices, such as CD-ROM players and audio controllers. This area of programming is complex and is not addressed in this simple tutorial. To learn more, visit the Microsoft Developers Network topic on MCI, here:
http://msdn2.microsoft.com/en-us/library/ms704979.aspx

STRUCT with Numeric Member

We learned about STRUCTS in Lesson 6. We'll create a STRUCT with one numeric member. Since this member will contain a handle, it is type "ulong." See Lesson 3 for more information on handle types in API calls.

```
    struct LPHMIDIOUT, handle as ulong
```

Passing a STRUCT Argument

We don't need to assign a value to the handle member of the struct. It will be filled in the API call to **midiOutOpen** as shown below. The comments give a brief explanation of the API call.

```
    struct LPHMIDIOUT, handle as ulong

    CallDLL #winmm, "midiOutOpen",_
        LPHMIDIOUT As struct,_  'Pointer to HMIDIOUT handle
        -1 As ulong,_           'device ID of MIDI_MAPPER
        0 As ulong,_            'callback address - not used
        0 As ulong,_            'callback instance - not used
        0 As ulong,_            'callback flags - not used
        ret As long             'returns 0 if successful
```

Retrieving a Number from a STRUCT

When the **midiOutOpen** function returns, the **handle** member of the **LPHMIDIOUT** STRUCT contains the handle of the MIDI device. We retrieve it like this:

```
    hMidiOut = LPHMIDIOUT.handle.struct    'handle to MIDI device
```

We've now assigned the value of the handle to the variable called **hMidiOut**. We'll use that variable any time we need to pass the MIDI device handle into an API call.

Demonstration Program

The following program retrieves the handle of the MIDI device from a STRUCT. It and uses this handle to play a note. Be sure the sound is not muted on your computer, so that you can hear the note.

```
    struct LPHMIDIOUT, handle as ulong

    CallDLL #winmm, "midiOutOpen",_
        LPHMIDIOUT As struct,_  'Pointer to HMIDIOUT handle
        -1 As ulong,_           'device ID of MIDI_MAPPER
        0 As ulong,_            'callback address - not used
        0 As ulong,_            'callback instance - not used
        0 As ulong,_            'callback flags - not used
        ret As long             'returns 0 if successful

    hMidiOut = LPHMIDIOUT.handle.struct    'handle to MIDI device

PRINT "Handle of MIDI device is ";hMidiOut

    'default instrument (voice) is Grand Piano

    'set up to play a MIDI note
    note = 60       'for C. C# = 61, etc.
    event = 144     'event 144 = play on channel 1
    loWord = (note*256)+event
    velocity = 127
    hiWord = velocity*256*256
    dwMsg = loWord + hiWord

    'send message to the opened MIDI device
    CallDLL #winmm, "midiOutShortMsg",_
        hMidiOut As ulong,_ 'handle to MIDI device
        dwMsg As ulong,_    'message to play a note
        ret As ulong

    'a three second timer
    'allows note to play for 3 seconds
    TIMER 3000, [stopPlay]
    wait

[stopPlay]
    'close MIDI device
    CallDLL #winmm, "midiOutClose",_    'close opened device
        hMidiOut As ulong,_             'handle of opened device
        ret As ulong

print "Done"
```

```
end
```

More?

The piano6.bas program that comes with Liberty BASIC has a simple interface for playing MIDI notes. You can use it as a starting point to learn more about using the MIDI mapper. The Microsoft Developers Network (MSDN) has all of the information you'll need. Start with midiOutOpen, here:
http://msdn2.microsoft.com/en-us/library/ms711632.aspx

What's Next?

Lesson 8 will discuss the native Windows API features of Liberty BASIC.

Written by Alyce Watson. For more on APIs, see:
APIs for Liberty BASIC