Midi Mapper for Sound Effects

-
Alyce
APIs for Liberty BASIC

The Midi Mapper

You can play midi notes directly from the midi mapper without the need for files on disk. You can play midi music, but you can also play sound effects. There are several midi voices that produce sound effects like a gunshot, telephone ring, or bird tweet.

midiOutOpen

You must first open the midi device with midiOutOpen. The first argument is the address of the handle to the midi device. This must be passed by reference so that the function can assign the handle to this variable. It is not possible to pass a numeric variable by reference into an API call in Liberty BASIC. It can be done with the use of a struct, since structs are passed by reference. Create a simple struct with one member that will be used to retrieve the handle of the midi device.

```
struct m, a$ As ptr
```

The midiOutOpen function looks like this. A return of 0 indicates success. An error code is returned in the function is not successful.

```
CallDLL #winmm, "midiOutOpen",_
m As struct,_ 'address of midiOut handle
-1 As ulong,_ 'ID of MIDI output device
0 As ulong,_ 'callback, not used
0 As ulong,_ 'callback instance, not used
0 As ulong,_ 'callback event flag, not used
ret As ulong '0=success (MMSYSERR_NOERROR = 0)
```

After the function returns, you can retrieve the handle to the midi device and assign its value to a numeric variable to be used in subsequent calls to the device.

```
hMidiOut = m.a$.struct 'handle to midi device
```

Bytes and Words

Messages sent to the midi device are often composed of multiple values that are combined into a dword. A dword consists of two word values and each word value consists of two bytes, a high byte and a low byte. To create a word from these two byte values, the high byte value is multiplied by 256 and added to the low byte value.

hibyte = value2 *256
lobyte = value1
lowword = hibyte + lowbyte

A dword consists of two word values. The high word value is multipled by (256*256) and added to the low word value to form a dword.

hiword = value3 * 256 * 256
dwMsg = hiword + lowword

Setting Midi Volume

The message midiOutSetVolume sets the volume for the left and right speakers. It requires the handle to the midi device from the midiOutOpen function, as well as a dword value specifying the volume. The left channel's volume is passed in the low word and the right channel's volume is passed in the high word. If a device does not support both left and right volume control, the low word specifies the mono volume level, and the high word is ignored. The values for each may be in the range of 0 to 65535, which is the same as hexadecimal FFFF.

Create the dword and set the volume as in this example that specifies a medium volume from both speakers.

```
lowVol = hexdec("FF") 'left channel medium volume
hiVol = hexdec("FF") 'right channel medium volume
'dword = (hiword value * 256) + lowword value
midiVol = (hiVol * 256) + lowVol
calldll #winmm, "midiOutSetVolume",_ 'set volume for midi playback
hMidiOut as ulong,_ 'handle to midi device
midiVol as ulong,_ 'volume
```

```
ret As ulong '0=success (MMSYSERR_NOERROR = 0)
```

midiOutShortMsg

The function midiOutShortMsg is used to send messages to the opened midi device. It requires the handle to the opened midi device and the message is a dword, as described above.

```
CallDLL #winmm, "midiOutShortMsg",_
hMidiOut As ulong,_ 'handle to opened device
dwMsg As ulong,_ 'message
ret As ulong
```

Selecting a Voice

There are 128 voices available in the midi mapper. They are indexed 0 – 127. A list of voices can be found at the end of this topic. Voices range from traditional musical instruments like piano, violin and flute to sound effects such as a telephone and gunshot. The midiOutShortMsg function extracts the message for the desired event from the low byte from the low word of the dword message. The event value to signify a change in voice is 192. The high byte of the low word specifies the voice, which may be in the range 0-127. The high word contains the velocity, which must be in the range of 0-127.

```
event=192 'event 192 = change
voice=19 'values 0-127, 19=church organ
velocity=127
low=(voice*256)+event
hi=velocity*256*256
dwMsg=low+hi
CallDLL #winmm, "midiOutShortMsg",_
hMidiOut As ulong,_ 'handle to opened device
dwMsg As ulong,_ 'message
ret As ulong
```

Playing a Note

Use midOutShortMsg to cause a note to be played. It will be played using the currently set voice. The low word - low byte event value for playing a note on channel 1 is 144. The pitch of the desired note is contained in the high byte value of the low word. Use values from 0 to 127. The note C has a value of 48, C# is 49, etc. To stop the note from sounding, set velocity (the high word) to zero.

```
note = 48 'play C note
event=144 'event 144 = play on channel 1
low=(note*256)+event
velocity=127
hi=velocity*256*256
dwMsg=low+hi
CallDLL #winmm, "midiOutShortMsg",_
hMidiOut As ulong,_ 'handle to opened device
dwMsg As ulong,_ 'message
ret As ulong
```

Stop All Notes from Sounding

Use midiOutShortMsg to stop all notes from sounding. The event value is 128.

```
vent=128 'event 128 = stop play
dwMsg=event
CallDLL #winmm, "midiOutShortMsg",_
hMidiOut As ulong,_ 'handle to opened device
dwMsg As ulong,_ 'message
ret As ulong
```

Close the Midi Device

When it is no longer needed, the midi device is closed with midiOutClose.

```
CallDLL #winmm, "midiOutClose",_
hMidiOut As ulong,_ 'handle to opened device
ret As ulong '0=success (MMSYSERR_NOERROR = 0)
```

Demo

```
'get midi mapper handle
hMidi=midiOutOpen()
print "Midi handle: ";hMidi

'set volume to medium
call midiOutSetVolume hMidi, hexdec("FF"),hexdec("FF")

'signal a change in instrument
```

```
    'if event=192(change), voice=instrument
    event=192  'event 192 = change
    voice=127  'gunshot
    velocity=127
call midiOutShortMsg hMidi,voice,event,velocity

print "Press key to hear gunshot."
input a$

'now play designated note
    'if event=144(play), voice=note
    event=144   'event 144 = play on channel 1
    voice=48    '48=note C
    velocity=127
call midiOutShortMsg hMidi,voice,event,velocity

print "Press key to hear phone ring."
input a$

'signal a change in instrument
    'if event=192(change), voice=instrument
    event=192  'event 192 = change
    voice=124  'phone ring
    velocity=127
call midiOutShortMsg hMidi,voice,event,velocity

'now play designated note
    'if event=144(play), voice=note
    event=144   'event 144 = play on channel 1
    voice=48    '48=note C
    velocity=127
call midiOutShortMsg hMidi,voice,event,velocity

print "Press key to stop sound."
input a$

'stop current note from playing
    event=144     'play on channel 1
    velocity=0    'stop note from playing
call midiOutShortMsg hMidi,voice,event,velocity

'stop all notes from playing
    'event 128 = stop all notes
call midiOutShortMsg hMidi,voice,event,velocity

'close midi mapper
```

```
call midiOutClose hMidi

print "Goodbye"
end

Sub midiOutSetVolume hMidiOut, leftVol, rightVol
    'volume range=0-65535 or hexadecimal FFFF
    midiVol = (rightVol * 256) + leftVol
    calldll #winmm, "midiOutSetVolume",_ 'set volume for midi playback
    hMidiOut as ulong,_ 'handle to midi device
    midiVol as ulong,_ 'volume
    ret As ulong '0=success (MMSYSERR_NOERROR = 0)
    end sub

Function midiOutOpen()
    struct m, a$ As ptr
    CallDLL #winmm, "midiOutOpen",_
    m As struct,_ 'address of midiOut handle
    -1 As ulong,_ 'ID of MIDI output device
    0 As ulong,_ 'callback, not used
    0 As ulong,_ 'callback instance, not used
    0 As ulong,_ 'callback event flag, not used
    ret As ulong '0=success (MMSYSERR_NOERROR = 0)
    midiOutOpen = m.a$.struct 'handle to midi device
    end function

Sub midiOutClose hMidiOut
    CallDLL #winmm, "midiOutClose",_
    hMidiOut As ulong,_ 'handle to opened device
    ret As ulong '0=success (MMSYSERR_NOERROR = 0)
    end sub

Sub midiOutShortMsg hMidiOut,voice,event,velocity
    'voice 0-127
    'velocity 0-127, 0 stops note from playing
    'event 192 = change voice
    'event 144 = play on channel 1
    'event 128 = stop all notes
    'notes: 48=C, 49=C#, etc.
    low=(voice*256)+event
    hi=velocity*256*256
    dwMsg=low+hi
    CallDLL #winmm, "midiOutShortMsg",_
    hMidiOut As ulong,_ 'handle to opened device
    dwMsg As ulong,_ 'message
    ret As ulong
```

```
    end sub
```

List of Voices

```
'list of 128 voices, in order of their MIDI indexes
'VOICE 0 = GRAND PIANO
Data "Grand Piano","Bright Grand","Electric Grand","Honky Tonk"
Data "Rhodes","Chorus Piano","Harpsichord","Clavinet"
Data "Celesta","Glockenspiel","Music Box","Vibraphone"
Data "Marimba","Xylophone","Tubular Bells","Dulcimer"
Data "Hammond Organ","Percussion Organ","Rock Organ"
Data "Church Organ","Reed Organ","Accordian","Harmonica"
Data "Tango Accordian","Accoustic Nylon Guitar"
Data "Accoustic Steel Guitar","Electric Jazz Guitar"
Data "Electric Clean Guitar","Electric Mute Guitar"
Data "Overdrive Guitar","Distorted Guitar","Guitar Harmonic"
Data "Accoustic Bass","Electric Bass Finger","Electric Bass Pick"
Data "Fretless Bass","Slap Bass One","Slap Bass Two"
Data "Synth Bass One","Synth Bass Two","Violin","Viola","Cello"
Data "Contrabass","Tremolo Strings","Pizzicato Strings"
Data "Orchestra Harp","Timpani","String Ensemble One"
Data "String Ensemble Two","Synth Strings One","Synth Strings Two"
Data "Choir Ahhs","Voice Oohs","Synth Voice","Orchestra Hit"
Data "Trumpet","Trombone","Tuba","Mute Trumpet","French Horn"
Data "Brass Section","Synth Brass One","Synth Brass Two"
Data "Soprano Sax","Alto Sax","Tenor Sax","Bari Sax","Oboe"
Data "English Horn","Bassoon","Clarinet","Piccolo","Flute"
Data "Recorder","Pan Flute","Bottle Blow","Shakuhachi","Whistle"
Data "Ocarina","Square Wave","Sawtooth","Caliope","Chiff Lead"
Data "Charang","Solo Synth VX","Brite Saw","Brass and Lead"
Data "Fantasia Pad","Warm Pad","Poly Synth Pad","Space Vox Pad"
Data "Bowd Glas Pad","Metal Pad","Halo Pad","Sweep Pad"
Data "Ice Rain","Sound Track","Crystal","Atmosphere","Brightness"
Data "Goblin","Echo Drops","Star Theme","Sitar","Banjo","Shamisen"
Data "Koto","Kalimba","Bagpipe","Fiddle","Shanai"
Data "Tinkle Bell","Agogo","Steel Drums","Wood Block","Taiko Drum"
Data "Melodic Tom","Synth Drum","Rev Cymbal"
Data "Guitar Fret Noise","Breath Noise","Sea Shore","Bird Tweet"
Data "Phone Ring","Helicopter","Applause","Gunshot"
'VOICE 127 = GUNSHOT
```