



Sending Messages to ServoCenterMIDI Programmatically in Windows

1. Overview

Yost Engineering's ServoCenterMIDI is an embedded controller board that allows traditional musical instrument digital interface (MIDI) continuous controller (CC) messages to be translated into physical movements of RC-style servo motors.

Many people use traditional musical instruments, MIDI controllers, and/or a computer with MIDI sequencing software to control the ServoCenterMIDI, but sometimes these systems don't allow a sufficient level of control and make real-time control difficult. Thus, a method is sometimes needed that allows real-time control messages to be sent directly to the ServoCenterMIDI.

This document covers the basic concept of controlling Yost Engineering's ServoCenterMIDI embedded controller programmatically using only MS Windows API calls. Example program code is in VisualBasic6, but these techniques can be applied to any programming language/environment that has access to the MS Windows API and supports the use of DLL function calls.

2. Windows MIDI support

Windows API is a set of application programming interfaces that are available in Microsoft Windows operating systems. This API is largely implemented as a library of callable functions that perform different tasks within the Windows environment. The functions are usually grouped according to function into different dynamic link libraries (DLLs) that are part of every Windows operating system.

One of the Windows API DLLs allows access to and control of multi-media devices, including MIDI devices. This DLL is named "winmm.dll" for 32-bit applications and "mmsystem.dll" for 16-bit windows applications. The two DLLs contain essentially the same functions. If you're using a 32-bit compiler, then winmm.dll should be used; if you're using a 16-bit compiler, then mmsystem.dll should be used.

Since the basic MIDI communication is built-in to the Windows API, there is no need to install or use 3rd party libraries, modules, or DLLs, so long as your language and programming environment support the use of DLL function calls.

3. Using the Windows API with ServoCenterMIDI

To effectively use the Yost Engineering ServoCenterMIDI with the Windows Multimedia API, only a few API functions must be used. These functions are:

- midiOutGetNumDevs – Returns the number of available MIDI output devices that are currently connected.
- midiOutGetDevCaps – Returns information about a particular MIDI output device.
- midiOutOpen – Opens a MIDI output device.
- midiOutClose – Closes a MIDI output device.
- midiOutShortMsg – Sends a MIDI channel message to an opened MIDI output device.

It is also useful to define a user-defined-type to hold the MIDI Output capabilities information that is returned from the midiOutGetDevCaps function.

In order to use the API, the following program flow sequence is usually implemented:

1. Call `midiOutGetNumDevs` to get the number of MIDI output devices that are currently available.
2. Devices can be identified by calling `midiOutGetDevCaps` for each available device and referencing the `szPname` string within the `MIDIOUTCAPS` structure that is filled in when the function returns.
3. If there are one or more MIDI output devices available then choose which one is to be opened.
4. Open the selected device by calling the `midiOutOpen` function.
5. Once the device has been opened, send data by calling the `midiOutShortMsg` function.
6. Once the program is done with the MIDI output device it should be closed by calling the `midiOutClose` function.

A few notes, however:

- MIDI devices must be opened before data can be sent using `MIDIOutShortMesg`.
- MIDI devices can only be opened and accessed in a mutually exclusive manner (one application at time), so you should ensure that other programs are not currently using a MIDI device that you are trying to open. Also, be sure to close the device when you are done with it.
- The `midiOutSortMessage` function sends raw MIDI data that is packed into a 4-byte long integer.
- Device indexes can change as additional MIDI devices are added so care should be taken that the correct device is being opened.
- Do not try to open device indexes that are outside of the range of devices that are currently identified as available by the `midiOutGetNumDevs` function.
- The `szPname` element of the `MIDIOUTCAPS` structure contains a null-terminated string that is stored within a fixed length buffer of 32 bytes. Character values after the null in the buffer are undefined and may contain unpredictable byte values.

4. Code Example

The code in this section illustrates the use of these API functions to send MIDI continuous controller messages to control a servo motor connected to a `ServoCenterMIDI` board. The code was written for Visual Basic 6, but these techniques can be applied to any programming language/environment that has access to the MS Windows API and supports the use of DLL function calls.

The program defines a code module that is used to perform all of the API interfacing and defines a `MoveServo` function that can be used to issue MIDI messages that will move a servo that is connected to a `ServoCenterMIDI` controller board. The program also includes a form module that consists of a simple form with a vertical scroll bar. As the vertical scroll bar is moved up and down on the screen, the servo is commanded to track the position accordingly.

SCMIDI.bas Code Module:

```
Const MAXPNAMELEN = 32
Type MIDIOUTCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As String * MAXPNAMELEN
    wTechnology As Integer
    wVoices As Integer
    wNotes As Integer
    wChannelMask As Integer
    dwSupport As Long
End Type
```

```

Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "midiOutGetDevCapsA" _
    (ByVal uDeviceID As Long, lpCaps As MIDIOUTCAPS, ByVal uSize As Long) As Long
Declare Function midiOutGetNumDevs Lib "winmm.dll" () As Integer
Declare Function midiOutClose Lib "winmm.dll" (ByVal hMidiOut As Long) As Long
Declare Function midiOutOpen Lib "winmm.dll" (lphMidiOut As Long, ByVal uDeviceID As Long, _
    ByVal dwCallback As Long, ByVal dwInstance As Long, ByVal dwFlags As Long) As Long
Declare Function midiOutShortMsg Lib "winmm.dll" (ByVal hMidiOut As Long, ByVal dwMsg As Long) _
    As Long

Dim hMidiOut As Long

'Returns a string that contains the Ids and names of all available MIDI out devices
Function GetDeviceList() As String
    Dim MidiCaps As MIDIOUTCAPS
    Dim Cnt As Long
    Dim retval As String
    'Get the number of installed MIDI devices
    numDevs = midiOutGetNumDevs()
    For Cnt = 0 To numDevs - 1
        'Get the device name and capabilities
        Call midiOutGetDevCaps(Cnt, MidiCaps, Len(MidiCaps))
        Call ClearAfterNull(MidiCaps.szPname)
        retval = retval + (Str$(Cnt) + "=" + MidiCaps.szPname) + vbCrLf
    Next Cnt
    GetDeviceList = retval
End Function

'Returns a string that has the null character and the characters
'after the null character replaced with space.
Sub ClearAfterNull(st As String)
    flag = 0
    newst = ""
    For i = 1 To Len(st) - 1
        ch = Mid$(st, i, 1)
        If Asc(ch) = 0 Then flag = 1
        If flag = 1 Then
            ch = " "
        End If
        newst = newst + ch
    Next i
    st = newst
End Sub

'Open a MIDI out device as referenced by the devNumber parameter.
Sub MidiOpen(devNum As Integer)
    Call midiOutOpen(hMidiOut, devNum, 0, 0, 0)
End Sub

'Close the currently opened MIDI out device.
Sub MidiClose()
    Call midiOutClose(hMidiOut)
End Sub

'Send a MIDI message.
Sub MidiSendMessage(midi_status As Integer, mididatal As Integer, mididata2 As Integer)
    Dim midivalue As Long
    Dim lowint As Long
    Dim highint As Long
    midivalue = mididata2
    midivalue = midivalue * 256 + mididatal
    midivalue = midivalue * 256 + midi_status
    Call midiOutShortMsg(hMidiOut, midivalue)
End Sub

'Send a MIDI continuous controller message to update the specified motor's position.
Sub MoveServo(midiChannel As Integer, svNum As Integer, svPos As Integer)
    'send coarse CC value ( 7 bits )
    Call MidiSendMessage(&HB0 + midiChannel - 1, svNum, Int(svPos / 128))
    'send fine CC value ( 7 bits )
    Call MidiSendMessage(&HB0 + midiChannel - 1, svNum + 32, svPos Mod 128)
End Sub

```

SCMIDI.frm Form Module:

```
Private Sub Form_Load()  
    VScroll11.Max = 16383  
    VScroll11.Min = 0  
    Call MidiOpen(1)  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    Call MidiClose  
End Sub  
  
Private Sub VScroll11_Change()  
    Call MoveServo(1, 1, VScroll11.Value)  
End Sub  
  
Private Sub VScroll11_Scroll()  
    Dim midichannel As Integer  
    Dim servonumber As Integer  
    Dim servoposition As Integer  
    midichannel = 1  
    servonumber = 1  
    servoposition = VScroll11.Value  
    Call MoveServo(midichannel, servonumber, servoposition)  
End Sub
```

5. Additional Information

Additional information on the Windows API is available from the following sites:

http://en.wikipedia.org/wiki/Windows_API – General description of the Windows API.

<http://msdn2.microsoft.com/en-us/library/ms712636.aspx> – Microsoft developer network documentation describing the Windows multi-media API.

<http://www.YostEngineering.com/ServoCenter> – Yost Engineering web page which contains information and documentation for the ServoCenterMIDI and other members of the ServoCenter family of controllers.

6. Contact Information

Postal Mail:

Yost Engineering, Inc.
630 Second Street
Portsmouth Ohio 45662

Web:

<http://www.YostEngineering.com>

Email:

Support: support@YostEngineering.com
Sales: sales@YostEngineering.com

Phone:

Voice: 1-740-355-9029
Fax: 1-740-354-1170