VISUAL BASIC MIDI

by Josep M. Mainat

Files needed:

VBRUN300.DLL
CMDIALOG.VBX
THREED.VBX

The VB_MIDI.ZIP package includes the following files:

VB-MIDI.BAS / MIDIHOOK.FRM / MSGHOOK.VBX (All purpose VB MIDI modules)

These modules can be the heart of any Visual Basic MIDI application. If you include a copy of them in your project, most of the MIDI low level programming tasks are already done. The code is fully commented so that you can use the modules as they are or customize them depending on your needs. The comments include plenty of suggestions on how to customize the procedures.

1)VB_MIDI.BAS

This module declares several global variables and API functions and includes all the necessary procedures to detect your available MIDI Devices, open and close them, record and playback MIDI data, sync to external MIDI Time Code (MTC) at any rate, write MTC out at any rate, manage MTC timing calculations, save your standard MIDI configuration into a private INI file, recall it later when you come back to your application, and much more. The most useful procedures are:

Sub Midi_Populate_Lists (cboMidiIn As Control, cboMidiOut As Control)

Your application must include two combo boxes style 2 (Dropdown List). The subroutine fills the first combo with all available MIDI In Devices in your system and the second with all available MIDI Out Devices.

cboMidiIn and **cboMidiOut** are the names of the combos

After this procedure call, the combos will also contain a "Device not enabled" entry to allow the user to disable and close devices if needed. The routine sets the global variables **nInDevices** and **nOutDevices** telling your application how many available "hardware" MIDI devices are present in your system. The **cboMidiIn_Click** and **cboMidiOut_Click** procedures of your application must include code to open or close the appropriate selected devices. (See **VB_MIDI** examples).

Sub MidiIn_Open (nDevice)

This procedure opens a MIDI In Device and sets **Midihook** control to start receiving MIDI In message events.

nDevice is the number of the MIDI In Device to be open.

In windows, MIDI In device numbers start at 0 and finish at **nInDevices - 1** (in the same order as in the cboMidiIn combo).

Sub MidiIn_Close ()

This procedure closes the currently open MIDI In device and cancels **Midihook** control activity.

Sub MidiOut_Open (nDevice)

This procedure opens a MIDI Out Device

nDevice is the number of the MIDI Out Device to be open.

MIDI Out device numbers start at -1 (the **Microsoft MIDI Mapper** "software" device) and finish at **nOutDevices -1** (in the same order as in the cboMidiOut combo). Thus, MIDI In device number 0 and MIDI Out device number 0 are the first "hardware" MIDI devices detected in your system.

Sub MidiOut_Close
This procedure closes the currently open MIDI Out device.

Function MidiOut_NoteOn (nChannel As Integer, nNoteNumber As Integer, nKeyvel As Integer)
This function sends a MIDI Note On message to te currently open MIDI Out device.
nChannel is the MIDI channel assigned to the note (0...15)
nNoteNumber is the MIDI note number (0...127 / Middle C = 60)
nKeyvel is the Key velocity (0...127 / 0 = Note off)
The function returns **True** if the operation is successful and **False** if not.

Function MidiOut_NoteOff (nChannel As Integer, nNoteNumber As Integer, nKeyvel As Integer)
This function sends a MIDI Note Off message to the currently open MIDI Out device.
nChannel is the MIDI channel assigned to the note (0...15)
nNoteNumber is the MIDI note number (0...127 / Middle C = 60)
nKeyvel is the Key velocity (0...127)
The function returns **True** if the operation is successful and **False** if not.

Function MidiOut_ProgramChange (nChannel As Integer, nProgramNumber As Integer)
This function sends a MIDI program change to te currently open MIDI Out device.
nChannel is the MIDI channel assigned to the program change (0...15)
nProgramNumber is the program number (0...127)
The function returns **True** if the operation is successful and **False** if not.

Function MidiOut_ControlChange (nChannel As Integer, nControlNumber As Integer, nControlValue As Integer)
This function sends a MIDI control change to te currently open MIDI Out device.
nChannel is the MIDI channel assigned to the control change (0...15)
nControlNumber is the controller number (0...127)
nControlValue is the new control value (0...127)
The function returns **True** if the operation is successful and **False** if not.
Constants for controller numbers are declared in **VB_MIDI.BAS** module.
For instance, to send a Volume change you must call the procedure as follows
vntRet = MidiOut_ControlChange (nChannel, MAIN_VOLUME, nNewVolume)
Where nChannel and nNewVolume are the desired MIDI channel and Volume values, and MAIN_VOLUME is a constant equal to the volume controller number (7)

Function MidiOut_Msg (lMsg As Long)
This function sends any kind of MIDI message to the currently open MIDI Out device.
lMsg is the MIDI message to be sent (long integer).
A MIDI message consists of a Status byte followed by one or two Data bytes. You must pack them into a long integer, thus the lMsg variable passed to the function must be set as follows:
lMsg = (nStatus + nChannel) + (&H100& * nMidiData1) + (&H10000 * nMidiData2)
- Status from &HF0 upwards are channel independent (nChannel must not be added to the Status integer.)
- Some messages don't need nMidiData2. Some real time messages contain only the status byte.
The function returns **True** if the operation is successful and **False** if not.

Function Mtc_SetMode (nMode As Integer)

This function sets the MTC mode to the appropriate frames per second rate.

nMode is the MTC frame mode as follows:

0 if 24 f/s
1 if 25 f/s
2 if 30 f/s drop frame
3 if 30 f/s no drop.

The function also sets the following global variables needed for MTC timing calculations:

nMtcMode (0, 1, 2 or 3 as nMode)
fMsPerQF (milliseconds per quarter frame. Must be float)
fMsPerFrame (milliseconds per frame. Must be float)
nFramesPerSecond (24, 25 or 30 depending of nMtcMode)

The Function returns a string message that can be used in displaying the current MTC mode in your application (i.e.: "25 f/s")

Sub Mtc_Adjust (nHours As Integer, nMinutes As Integer, nSeconds As Integer, nFrames As Integer)

This procedure adjusts MTC values and sets them to a valid MTC standard format.

nHours is the MTC hours counter.
nMinutes is the MTC minutescounter.
nSeconds is the MTC seconds counter.
nFrames is the MTC frames counter.

The subroutine assigns the new values to the passed variables. For instance if:

nHours = 1
nMinutes = 73 (max. is 59)
nSeconds = 10
nFrames = 22

the 73 minutes will be converted to 1 hour and 13 minutes so the new values returned in the passed variables will be:

nHours = 2 (1 + 1)
nMinutes = 13 (73 - 60)
nSeconds = 10
nFrames=22

The procedure also adjusts negative values:

00:05:(-2):00 will be converted to 00:04:58:00

This allows you to do any calculations with MTC values (add, subtract, increment, decrement, etc.) without taking care of overflows or negative values. Then, you call this procedure to convert it to a valid MTC.

Function MidiOut_Mtc (nQfID As Integer, nHours As Integer, nMinutes As Integer, nSeconds As Integer, nFrames As Integer)

This function sends MTC to the currently open MIDI Out device.

nQfID is the quarter frame ID number and ranges from 0 to 7.
nHours is the MTC hours counter.
nMinutes is the MTC minutes counter.
nSeconds is the MTC seconds counter.
nFrames is the MTC frames counter.

MTC messages must be sent every quarter frame, thus the procedure must be called at the appropriate milliseconds rate calculated in the previous call to **Mtc_SetMode** and automatically saved in the global variable **fMsPerQF** (milliseconds per quarter frame)
A whole MTC message takes 8 quarter frames to be completed. The quarter frame ID number must start with 0 and must be incremented after each call, wrapping around from 7 to 0. When the whole MTC message is completelly sent, the frame counter must be incremented by two (8 quarter frames = 2 frames). So, after each call to te procedure you must include the following lines of code:

```
nQfID = nQfID + 1
If nQfID = 8 Then
    nQfID = 0
    nFrames = nFrames + 2
    Call Mtc_Adjust (nHours, nMinutes, nSeconds, nFrames)
End If
```

The function returns **True** if the operation is successful and **False** if not.
(See **VB_MIDI** examples **Play_Internal** and **Rec_Internal** procedures to study how to use it in a real situation.)

Sub Ini_Write (sIniName As String, sSection As String, sParamName As String, sParamValue As String)

This procedure writes a parameter into an INI file. It allows your application to "remember" your standard MIDI configuration.

sIniName is the name of the INI file (i.e. "VB_MIDI.INI")
sSection is the name of the section where you want the parameter to be stored (i.e. "MIDI DEVICES")
sParamName is the parameter name (i.e.: "In")
sParamValue is the parameter value in string format (i.e. "2")

If you use the above examples you'll get an INI file called VB_MIDI.INI with the following lines:

```
[MIDI DEVICES]
In=2
```

You may call the procedure as many times as you need to add new sections and parameters to store your default MIDI In and Out devices, your default MTC frame mode, your prefered options, your last open files, etc....
If the INI file is not found, a new INI file is created in the WINDOWS directory. If the INI file already exists the procedure will only change the existing INI file contents.

Function Ini_Read (sIniName As String, sSection As String, sParamName As String)

This procedure reads a parameter stored in an INI file.

sIniName is the name of the INI file
sSection is the name of the section where the parameter is stored
sParamName is the name of the requested parameter.

The function returns the parameter value in string format. If the INI file doesn't exist or the requested parameter is not found, the function returns an empty string ("")

2)**MSGHOOK.VBX**

A free control included in the book: *"VISUAL BASIC HOW-TO"* by Zane Thomas. The control can intercept any Windows message sent to a form. A **MSGHOOK.VBX** control has been added to the **MIDIHOOK.FRM** and instructed to "hook" the MIDI In messages sent by Windows to the application.

3) **MIDIHOOK.FRM**
A form with a **MSGHOOK.VBX** control named **Midihook**. The form must be loaded but never showed as it controls the MIDI In data flow in the background. If a MIDI In Device is currently open, every time that Windows sends a message to the form indicating your application that some MIDI data has just arrived at the corresponding MIDI port, the **Midihook_Message** event is automatically triggered and the incoming MIDI message can be interpreted or saved or whatever is needed. The **Midihook_Message** event procedure actually does the following tasks:

- If the global flags **bDataThru** and/or **bMtcThru** are set to True by the application, the incoming MIDI data and/or MTC are sent automatically to the currently open MIDI Out device.

- If the global variable **nSyncMode** is set to the global constant **SYNC_EXTERNAL** (declared in **VB_MIDI.BAS**) the **Midihook_Message** event procedure decodes received MTC and sets the following global flags and variables:

bInSync : True while MTC is correctly interpreted
False if discontinuous or incorrect MTC is received.
Polling this flag indicates your application if you are in sync or out of sync.

bMtcModeError : True if received MTC is not in the expected mode.
(i.e.: 25 f/s are expected and received MTC is in 24 f/s mode)
You will poll this flag only if you are not sure about received MTC mode.

lMtcTime : A long integer with the current received MTC time converted to milliseconds. It allows your application to sync to external MTC.
It is updated every quarter frame.

nMtcTotalFrames : A total frame counter with the current MTC received time converted to frames. It allows your application to display a clock with the current MTC time. It is updated every frame.

nNewMtc : Incremented every time that the application is in out of sync state and a new correct MTC time is received.
It allows your application to resync after an out of sync state
(i.e. when MTC changes discontinuously after a Rewind or Forward operation)

To start reading and interpreting MTC correctly you must first set the following global variables and flags declared in **VB_MIDI.BAS**:

bInSync = False (you start in out of sync state)
bMtcModeError = False (no error)
nNewMtc = 0 (new MTC time not yet received)
nQfIdExpected = 0 (Identifier of first expected quarter frame MTC message)

You may also set **lMtcTime** and **nMtcTotalFrames** to -1 to indicate your application that those values are not yet available. As soon as the **Midihook_Message** event procedure starts reading correct MTC, those variables are automatically updated.
(See **VB_MIDI** examples **Play_External** and **Rec_External** procedures to study how to use it in a real situation.)

VB-SEQ (A FULLY FUNCTIONAL MIDI SEQUENCER)

This utility includes all source code to compile VB_SEQ.EXE file. It serves as an example of how to use the all purpose VB MIDI modules in a real situation.
The VB_SEQ utility is a record and playback sequencer with the ability to sync to internal time or MTC.
It can also generate MTC at any standard rate (24, 25 or 30 f/s).
Recorded files can be saved and open.
Program changes can be sent to any MIDI channel.
The user interface has a 3D look with flashing leds to show the flow of MTC and MIDI data In and Out.
It also includes a display clock, a set of tracking buttons (PLAY / REC / STOP / REWIND / FORWARD), two combo boxes to open or close the available MIDI devices, and menus to set MTC mode, enable or disable the flashing leds and select all available options (MIDI thru, MTC thru, MTC out,...).
On exit, your last used configuration is saved to an INI file.

BUTTONS

PLAY button

Internal Sync Mode: starts playing the previously open or recorded file at the point indicated by the clock time.

External Sync Mode: Waits for external MTC to arrive at the currently open MIDI In device. As soon as MTC is received, the application starts playing the previously open or recorded file at the point indicated by the current MTC time.

REC button.

Internal Sync Mode: Clears previously open or recorded file and starts recording MIDI in messages arriving at the currently open MIDI In device. The messages are timestamped with current clock time.

External Sync Mode: Clears previously open or recorded file and waits for external MTC to arrive at the currently open MIDI In device. As soon as MTC is received, the application starts recording MIDI in messages arriving at the currently open MIDI In device. The messages are timestamped with current MTC time.

STOP button

Stops Record or Playback activity.

FORWARD and REWIND buttons

Those buttons increment and decrement the following clock values:

- Frames if not any key pressed
- Seconds if [ALT] key pressed
- Minutes if [CONTROL] key pressed
- Hours if [SHIFT] key pressed

SYNC button

Switches between Internal and External sync modes.

CHANNEL Increment and Decrement buttons

Those buttons change the current MIDI channel and recall the Program number assigned to that channel.

PROGRAM Increment and Decrement buttons

Those buttons change the Program number of the current MIDI channel. The Program number assigned

to each channel is remembered by the application.

MIDI In combo

Selecting a MIDI In device will open it and enable MIDI in activity.
Selecting the "Device not enabled" entry closes all MIDI In devices and stops MIDI In activity.

MIDI Out combo

Selecting a MIDI Out device will open it and enable MIDI Out activity.
Selecting the "Device not enabled" entry closes all MIDI Out devices and stops MIDI Out activity.

MENUS

File Menu

About...

Opens the About dialog. Here are some acknowledges and my Compuserve ID number. Any mailed comments, questions or suggested improvements are welcome.

New

Clears buffer of currently recorded MIDI messages.

Open...

Opens the file dialog to choose an existing file to be open. Filename must have **.SNG** suffix

Save...

Opens the file dialog to save the currently recorded MIDI messages to a file. The **.SNG** suffix is automatically added.

Exit

Close currently open MIDI devices and exits.

Options Menu

Midi Data Thru (Always)

If checked, MIDI data received at the currently open MIDI In device is automatically sent to the currently open MIDI Out device. (only MIDI data, not MTC)

MTC Thru (Always)

If checked, Midi Time Code received at the currently open MIDI In device, is automatically sent to the currently open MIDI Out device.

MTC Out (Internal Sync)

If checked, while playing or recording in internal sync mode, Midi Time Code is generated and sent to the currently open MIDI Out device. If both MTC Thru and MTC Out are activated, avoid receiving external MTC while playing or recording in internal sync mode because received MTC and internally generated MTC would be mixed and sent to the MIDI Out device and the resulting sequence of MTC messages would have no sense.

Frame Mode...

This menu opens a submenu with Frame Mode standard rates (24 f/s, 25 f/s, 30 f/s drop frame and 30 f/s no drop). Check the desired option.

Visualize Menu

The purpose of this menu is to deactivate clock and/or flashing leds in order to achieve maximum timing accuracy. Normally, you may have the visualize options enabled as the application will never be out of sync in standard situations but, if you're managing a massive flow of MIDI data in and out and you feel that MIDI devices are overloaded, disabling some visualitazion can help to increase timing accuracy.

Clock
If checked, the application clock displays current time while in internal or external sync mode.

Data Flow
If checked, the Midi data leds are activated.

Mtc Flow
If checked the MTC leds are activated.

All
Activates all visualize options

None
Deactivates all visualize options

KEYBOARD SHORTCUTS

[ENTER] and [RETURN] keys activate playback.

Numpad multiply key [*] activates recording.

[SPACEBAR] stops playback or recording.

[ESCAPE] key resets clock to 00:00:00:00

NUMBER keys [0]...[9] write their corresponding values in the last digit of the clock while the previous digits are shift to the left. This allows to set a new clock value just by typing the complete time code number (8 digits)

TIP: If the new time code you want to set contains only a few digits at the right side (i.e. 00:00:05:22), you may press [ESCAPE] to reset the clock to 00:00:00:00 and then type only the necessary digits (i.e.5, 2, 2).

Note: An example file called **RAGTIME.SNG** is also included.

Josep M. Mainat
Gestmusic s.a.(R&D)
Barcelona
SPAIN
Compuserve ID 100414,1026