Help for MIDI File Control

Properties Events

Registration Information

Order Form

Getting Custom Controls Written

Licensing Information

Description

The MIDIFile control provides the Visual Basic programmer with an easy way to read and write MIDI files, both formats 0 (single track) and 1 (multiple-tracks) are supported.

Using the MIDIFile control you can modify existing MIDI files or create entirely new ones from scratch. You have complete control over and access to every type of MIDI message. You can insert, delete and modify tracks and messages at any time.

This control is a storage structure for MIDI data. It has no recording or playback functions. It was designed using the Standard MIDI File 1.0 specification. What you do with the MIDI data that is loaded into this control is completely up to your programming.

The Standard MIDI File custom control allows you to read any Standard MIDI file into a data structure with full access to the MIDI data. Open MIDI files, close MIDI files, create new MIDI files, save changes, insert new MIDI messages, modify MIDI messages and delete MIDI messages. Supports MIDI file types 0 & 1.

- Provides the Visual Basic programmer with a way to read and write Standard MIDI files.
- Both formats 0 (single track) and 1 (multiple-tracks) are supported.
- Modify existing MIDI files or create entirely new ones from scratch.
- Complete control over and access to every type of MIDI message.
- Insert, delete and modify tracks and messages at anytime.
- Action property to open existing filename, close current file, create new file and save the data to the current file.
- Action property to insert messages, change the current message and delete the current message.
- Insert new tracks or delete tracks.
- Complete support for system exclusive messages.

File Name

MIDIFILE.VBX, MIDIFL16.OCX, MIDIFL32.OCX

Object Type

MIDIFile

VBX Compatibility

VB 2.0 and up

Remarks

We strongly advise that you purchase a copy of one of more of the MIDI specifications. To purchase these specifications, you should contact the MIDI Manufacturers Association.

MIDI Manufacturers Association
Post Office Box 3173
La Habra, CA 90632

Phone: 310-947-8689
Fax: 310-947-4569

You can also purchase MIDI books from the Mix Bookshelf. They sell a wide variety of books about

MIDI and music.

Mix Bookshelf
6400 Hollis Street
Suite 12
Emeryville, CA   94608

Phone: 800-233-9604 or 510-653-3307
Fax: 510-653-5142

Microsoft sells three books that are specifically aimed at multimedia development on Windows.   We have found these books to be quite valuable.   The books are:

Microsoft Windows Multimedia Programmer's Reference
Microsoft Windows Multimedia Programmer's Workbook
Microsoft Windows Multimedia Authoring and Tools Guide

United States: 800-MSPRESS
Canada: 416-293-8141
Other Locations: Contact your local Microsoft office

Distribution Note   When you develop and distribute an application that uses this control, you should install the control into the users Windows SYSTEM directory.   This control has version information built into it.   So, during installation, you should ensure that you are not overwriting a newer version.

Registration Information

Credits

The MIDI Pack was written by James Shields and Zane Thomas. The MIDI File control was written by Zane Thomas.

Inquiries, tech support, comments should be sent to Mabry Software. Our address is 71231,2066 on CompuServe, or mabry@mabry.com on Internet. You can call us at 2066341443 or fax us at 206632-0272. If you need to send something via U.S. Mail, the address is:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Registration

You can register this program by sending $40 ($45 for international orders) and your address. You can register MIDI File **and** its C++ source code by sending $120 ($125 for international orders). With your order, you will receive a copy of our manual documenting all of the MIDI Pack controls.

Add $5 per order for shipping and handling.

For your convenience, an order form has been provided that you can print out directly from this help file.

Prices are subject to change without notice.

E-mail Discount

You may take a $5 discount for e-mail delivery of this package (CompuServe or Internet). If you choose this option, please note: a printed manual is not included. Be sure to include your full mailing address with your order. Sometimes (on the Internet) the package cannot be e-mailed. So, we are forced to send it through the normal mails.

CompuServe members may also take the $5 e-mail discount by registering this package in the software registration forum (GO SWREG). MIDI Files SWREG ID number is 10284. The source code version's ID number is 10285.

Credit Card Orders

We accept VISA, Mastercard and American Express. If you e-mail your order to us, please be sure to include your card number, expiration date, complete mailing address, and your phone number (in case we have any questions about your order).

MIDI File Order Form

Use the Print Topic.. command from the File menu to print this order form.

Mail this form to:	Mabry Software, Inc. Post Office Box 31926 Seattle, WA 98103-1926
	Phone: 206-634-1443 Fax: 206-632-0272 CompuServe: 71231,2066 Internet: mabry@mabry.com Web: www.mabry.com

Where did you get this copy of MIDI File?

____

Ship to: ____

____

____

____

____

Phone: ____

Fax: ____

E-Mail: ____

MC/VISA/AMEX: ____________________________ exp. ____________

P.O. # (if any): ____

qty ordered	____	REGISTRATION $40 each ($45 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.
qty ordered	____	SOURCE CODE AND REGISTRATION $120 each ($125 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

MIDI File Properties

All of the properties that apply to this control are in this table. Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***Action** Property
Align Property
***Buffer** Property
***Clocks** Property
***Data1** Property
***Data2** Property
***Denominator** Property
Enabled Property
***Filename** Property
***Filter** Property
***Format** Property
***FractionalFrames** Property
***Frame** Property
***FrameRate** Property
***Hour** Property
Index Property
Left Property
***Message** Property
***MessageCount** Property
***MessageNumber** Property
***Minute** Property
***Mi** Property
***MsgText** Property
Name Property
***Notated32nds** Property
***NumberOfTracks** Property
***Numerator** Property
***Second** Property
***Sequence** Property
***Sf** Property
Tag Property
***Tempo** Property
***TicksPerFrame** Property
***TicksPerQuarterNote** Property
***Time** Property
***TimeFormat** Property
Top Property
***TrackNumber** Property

MIDI File Events

All of the events that apply to this control are in this table.   Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***Error** Event

Action Property (MIDI File Control)

Example

Description

Action to take.

Usage

[*form.*][*control.*]**Action**[= *integer*]

Remarks

Setting this property causes an action to occur using current data. The actions are:

Constant	Value	Meaning
msMFANone	0	None. No action
msMFAOpen	1	Open. Open existing filename
msMFAClose	2	Close. Closes current file. File contents are not changed by this action. See Save Changes.
msMFANew	3	New. Creates new file specified by Filename. An error will occur if the file already exists.
msMFASave	4	Save Changes. Saves the data to the current file, but does not close it.
msMFAClear	5	Clear Data. The current MIDI file contents (if any) are discarded.
msMFAInsert	6	Insert Message. Insert the message specified by Time, Message, Data1, and Data2 immediately after the message given by MessageNumber. MessageNumber is incremented by one.
msMFAModify	7	Modify Message. Changes the current message using the values of the Time, Message, Data1, and Data2 properties.
msMFADelete	8	Delete Mesage. Deletes the current message and loads the properties from the next message. Do not delete the last messag. MessageCount should always be greater than zero.
msMFAInsertTrack	9	Insert Track. Creates a new track and inserts it immediately after the track given by TrackNumber. TrackNumber is then incremented by one.
msMFADeleteTrack	10	Delete Track. The current track is deleted and the next track becomes the current track. Do not delete the last track. NumberOfTracks should always be greater than zero
msMFASaveAs	11	Save As. Saves the current MIDIFile control contents into the file given by Filename. IMPORTANT NOTE: if Filename already exists it will be overwritten.

Data Type

Integer

Buffer Property

Example

Description

Holding area for system exclusive messages.

Usage

[*form.*][*control.*]**Buffer**[= *string*]

Remarks

When sending or receiving a System Exclusive (Sysex) message the buffer property is used to transfer the contents of the Sysex message.   The contents of Sysex messages is determined solely by the MIDI device sending or receiving the sysex message.

It is important to note that there is a subtle difference between the way the Buffer property is used in the MIDI File control and the MIDI In and Out controls.

When you transmit a Sysex message to a midi device using the MIDI Out control you will need to supply the sysex start and end bytes (&HF0 and &HF7) as message delimiters.

In this example, a Sysex message is sent which resets the Roland SoundCanvas SC-88 to General MIDI mode:

```
MidiOutput1.Message = &HF0

MidiOutput1.Buffer = Chr$(&HF0) + Chr$(&H7E) + Chr$(&H7F) + Chr$(9) +
     Chr$(1) + Chr$(&HF7)

MidiOutput1.Action = MIDIOUT_SEND
```

In this example the first and last bytes (&HF0 and &HF7) signal the beginning and end of a Sysex message.   The middle bytes are the Sysex messages contents.

When you receive a sysex message using the MIDI In control the start and end bytes will be the first and last bytes in the string contained by the Buffer property.

However when you read a sysex message from the MIDI File control the start and end bytes will NOT be in the string contained by Buffer.   So to transmit a sysex message retrieved from the MIDI File control you should use something like:

```
sysexMsg = &HF0 + MIDIFile1.Buffer + &HF7
```

Data Type

String

Clocks Property

Description

Number of MIDI clocks in a metronome click.

Usage

[*form.*][*control.*]**Clocks**[= *integer*]

Remarks

Valid only after a Time Signature meta-event (&H58) becomes the current message. Once the values are loaded from a Time Signature meta-event they remain valid until another Time Signature meta-event is encountered.

Data Type

Integer (0 - 255)

Data1 and Data2 Properties

See Also

Description

MIDI message data bytes.

Usage

[*form.*][*control.*]**Data1**[= *integer*]
[*form.*][*control.*]**Data2**[= *integer*]

Remarks

The contents of Data1 and Data2 depend on the type of MIDI message being sent/received.

If the Message property is 255 (for a meta event), MsgText is loaded with a string. Data1 determines what the string represents. The following table lists the possible values:

Constant	Value	Meaning
msMFMTText	1	Non-specific text string
msMFMTCopyright	2	Copyright notice
msMFMTTrack	3	Sequence/track name
msMFMTInstrument	4	Instrument name
msMFMTLyric	5	Lyric
msMFMTMarker	6	Marker
msMFMTCuePoint	7	Cue point
	8-15	Undefined text string

Data Type

Integer (0-255)

See Also

Properties:

Message

Denominator Property

See Also

Description

Denominator represents the denominator of a time signature as it would be notated in accordance with the Standard MIDI File specification.

Usage

[*form.*][*control.*]**Denominator**[= *integer*]

Remarks

Valid only when the current messages is a Time Signature meta-message (&H58).

Data Type

Integer (0 - 255)

See Also

Properties:

Numerator

Error Event

See Also Example

Description

Fires when an error occurs.

Syntax

Sub *ctlname*_**Error (***Error* **As Integer,** *ErrorMessage* **As String)**

Remarks

This event is fired whenever an error occurs.   Both an error code and a textual description of the error are passed as arguments.

The argument *Error* holds the error number.

The argument *ErrorMessage* gives the error in string form.

See Also

Properties:

Action (MIDI File)

Filename Property

See Also Example

Description

Filename to open or create.

Usage

[*form.*][*control.*]**Filename**[= *string*]

Remarks

Filename to open or create. See the Action property.

Data Type

String

See Also

Properties:

Action

Filter Property

Description

Allows for filtering and discarding of unwanted MIDI messages.

Usage

[*form.*][*control.*]**Filter(***arrayindex***)**[= *boolean*]

Remarks

This property allows you to automatically filter (remove) incoming MIDI messages.

To filter out the MIDI note off message, set

```
MIDIInput.Filter(&H80) = True
```

To filter MIDI clock messages:

```
MIDIInput.Filter(&HF8) = True
```

Data Type

Boolean array

Format Property

Description

Determines the format of the current MIDI file.

Usage

[*form.*][*control.*]**Format**[= *integer*]

Remarks

Determines the format of the current MIDI file.

Constant	Value	Meaning
msMFFSingle	0	Single track
msMFFMulti	1	One or more simultaneous tracks

Data Type

Integer

Frame and FractionalFrames Properties

See Also

Description

Determines the offset of a message

Usage

[*form.*][*control.*]**Frame**[= *integer*]
[*form.*][*control.*]**FractionalFrames** = *integer*]

Remarks

These properties specifiy the offset. They become valid when a SMPTE Offset meta-message (&H54) becomes the current message and remain valid until either another SMPTE Offset meta-message is received or until changed by your program.

Data Type

Integer (0-255)

See Also

Properties:

FrameRate

FrameRate Property

See Also

Description

SMPTE frames per second.

Usage

[*form.*][*control.*]**FrameRate**[= *integer*]

Remarks

Determines the speed of frames.   Valid only when TimeFormat = 1 (SMPTE/MIDI).

Data Type

Integer

See Also

Properties:

- Fractional Frames
- Frame
- Time
- TimeFormat

Hour, Minute, and Second Properties

Description

Determines the time offset of a message

Usage

[*form.*][*control.*]**Hour**[= *integer*]
[*form.*][*control.*]**Minute**[= *integer*]
[*form.*][*control.*]**Second**[= *integer*]

Remarks

These properties specifiy the current time offset.   They are valid only when the current message is a SMPTE Offset meta-message (&H54).

Data Type

Integer (0-255)

Message Property

See Also Example

Description

Message byte.

Usage

[*form.*][*control.*]**Message**[= *integer*]

Remarks

Part of the data sent/received.

Data Type

Integer (0-255)

See Also

Properties:

Data1 and Data2

MessageCount Property

Example

Description

Number of messages available.

Usage

[*form.*][*control.*]**MessageCount**[= *integer*]

Remarks

As messages arrive at the MIDI Input control they are queued by the control. Your program can determine how many messages the MIDI Input control has queued by examining this property.

There is (or at least should be) an End of Track message at the end of each MIDI track. When you create a new track using the MIDI File control an End of Track message is placed in the track. The MessageCount property is actually one less than the number of messages since the End of Track message is not counted, cannot be accessed, and cannot be deleted.

Data Type

Integer (long)

MessageNumber Property

See Also Example

Description

Specifies current message.

Usage

[*form.*][*control.*]**MessageNumber**[= *long*]

Remarks

Specifies the current message. This must range from 1 to MessageCount.

Data Type

Integer (long)

See Also

Properties:

MessageCount

Mi Property

See Also

Description

When Mi is set to 1 the current track is in a minor key, when set to 0 the current track is in a major key.

Usage

[*form.*][*control.*]**Mi**[= *integer*]

Remarks

Valid when the current message is a Key Signature meta-message (&H59).

Data Type

Integer (0 - 255)

See Also

Properties:

Sf

MsgText Property

Example

Description

String representing meta-event.

Usage

[*form.*][*control.*]**MsgText**

Remarks

Specifies the name of the meta event.

This property is read-only.

Data Type

Integer

Notated32nds Property

See Also

Description

The number of notated 32nd notes in a MIDI quarter-note (24 MIDI clocks).

Usage

[*form.*][*control.*]**Notated32nds**[= *integer*]

Remarks

Valid when the current message is a Time Signature meta-message (&H58).

Data Type

Integer (0 - 255)

See Also

Properties:

Clocks

NumberOfTracks Property

See Also Example

Description

Number of tracks available.

Usage

[*form.*][*control.*]**NumberOfTracks**[= *integer*]

Remarks

Current number of tracks available, this number will change as you insert and/or delete tracks.

Data Type

Integer

See Also

Properties:

TrackNumber

Numerator Property

See Also

Description

The numerator of the time signature as it would be notated in accordance with the Standard MIDI File specification.

Usage

[*form.*][*control.*]**Numerator**[= *integer*]

Remarks

Valid when the current message is a Time Signature meta-message (&H58).

Data Type

Integer (0 - 255)

See Also

Properties:

Denominator

Sequence Property

Description

MIDI files may contain a Sequence Number meta-event at the beginning of a track and before any non-zero delta-time events, and before any transmittable MIDI events.   The Sequence Property is set to the value of the Seqence Number whenever the Sequence Number meta-event is encountered.

Usage

[*form.*][*control.*]**Sequence**[= *long*]

Remarks

Sequence number is generally not useful in format 0 or 1 MIDI files.

Data Type

Integer (long)

Sf Property

See Also

Description

Sharps/Flats, number of sharps or flats in the current key. Values between 1 and 127 specify 1 or more sharps, values between 128 and 255 specify one or more flats, and 0 specficies the key of C.

Usage

[*form.*][*control.*]**Sf**[= *integer*]

Remarks

Valid when the current message is a Key Signature meta-message (&H59).

Data Type

Integer (0 - 255)

See Also

Properties:

Mi

Tempo Property

Example

Description

Sets the tempo.

Usage

[*form.*][*control.*]**Tempo**[= *long*]

Remarks

Valid whenever the current message is a Tempo meta-event (&H51).

According to the Standard MIDI File specification, the Tempo value gives the number of microseconds per MIDI quarter note.

To calculate the Beats per Minute (BPM) of a song, use this formula:

BPM = 60,000,000 / MidiFile.Tempo

Data Type

Integer (long)

TicksPerFrame Property

Description

Determines the number of ticks in each frame.

Usage

[*form.*][*control.*]**TicksPerFrame**[= *integer*]

Remarks

Determines the number of ticks in each frame. Valid only when TimeFormat = 1 (SMPTE/MIDI).

Data Type

Integer

TicksPerQuarterNote Property

Example

Description

Determines the number of ticks in each quarter note.

Usage

[*form.*][*control.*]**TicksPerQuarterNote**[= *integer*]

Remarks

Determines the number of ticks in each quarter note. Valid only when TimeFormat = 0 (ticks per quarter note).

Data Type

Integer

Time Property

See Also Example

Description

Time of message in ticks or milliseconds (see TimeFormat).

Usage

[*form.*][*control.*]**Time**[= *integer*]

Remarks

Time of message in ticks. It is important to note that Time has a different meaning in the MIDI input and output controls than it does in the MIDI file control. MIDI input and output times are always milliseconds elapsed time since the start of either recording or playback, while the MIDI file control always sets Time to the number of Ticks which elapse between events.

For the MIDI input and MIDI output controls Time is always in milliseconds.

With the MIDI file control the meaning of Time is defined by the contents of the MIDI header values TicksPerQuarterNote and the Tempo meta-event value Tempo when TimeFormat is 0 (Ticks per quarter note) or by FrameRate and TicksPerFrame when TimeFormat is 1 (SMPTE).

When using TimeFormat 0 files you may need to convert between MIDI ticks and milliseconds. Since Tempo gives the number of microseconds per MIDI quarter note the number of beats per minute is given by:

Beats Per Minute = 60,000,000 / Tempo

The number of Milliseconds Per Tick is:

Milliseconds Per Tick = (Tempo / 1000) / TicksPerQuarterNote

When reading a MIDI file and playing it using the MIDI output control you can use the Milliseconds Per Tick value to calculate the number of milliseconds between one event and the next by using the following equation:

Millisecond Delay = Ticks between events * Milliseconds Per Tick

When reading MIDI messages from the MIDI input control you need to convert from milliseconds to ticks, you can use the following equation:

Ticks Per Milliseconds = (MIDIFile1.TicksPerQuarterNote / MIDIFile1.Tempo) * 1000

Then convert elapsed milliseconds to ticks like this:

Ticks between events = Milliseconds between events * Ticks Per Milliseconds

Data Type

Integer (long)

See Also

Properties:

- TicksPerQuarterNote
- Tempo

TimeFormat Property

See Also

Description

Determines the method ot time-keeping used.

Usage

[*form.*][*control.*]**TimeFormat**[= *integer*]

Remarks

Determines the method ot time-keeping used.

Constant	Value	Meaning
tfTicks	0	Ticks per quarter note (see TicksPerQuarterNote)
tfSMPTE	1	SMPTE/MIDI (see FrameRate and TicksPerFrame)

Data Type

Integer

See Also

Properties:

Time

TrackNumber Property

See Also Example

Description

Currentl selected track.

Usage

[*form.*][*control.*]**TrackNumber**[= *integer*]

Remarks

Currently selected track. Trakcs can be accessed at random by using this property. Tracks are numbered from 1 to NumberOfTracks.

Data Type

Integer

See Also

Properties:

NumberOfTracks

Action Property Example, MIDI File Control

This subroutine shows how to perform a number of common tasks using the MIDIFile controls Action property.

```
Sub MidiFileFun ()
  '
  ' Delete the current track
  '
  MIDIFile1.Action = MIDIFILE_DELETE_TRACK
  '
  ' Create a new track
  '
  MIDIFile1.Action = MIDIFILE_INSERT_TRACK
  '
  ' Add a note-on message (Ch. 3, C3, forte, time 0) to the new track
  '
  MIDIFile1.Message = &H92
  MIDIFile1.Data1 = &H60
  MIDIFile1.Data2 = &H96
  MIDIFile1.Time = 0
  MIDIFile1.Action = MIDIFILE_INSERT_MESSAGE
  '
  ' Add a note-off message (Ch. 3, C3, standard, 50 ticks later)
  '
  MIDIFile1.Message = &H82
  MIDIFile1.Data1 = &H60
  MIDIFile1.Data2 = &H64
  MIDIFile1.Time = 50
  MIDIFile1.Action = MIDIFILE_INSERT_MESSAGE
  '
  ' Backup to first message and change its start time (moving to a message
  ' reloads the message so we only need to modify the time property)
  '
  MIDIFile1.MessageNumber = 1
  MIDIFile1.Time = 25
  MIDIFile1.Action = MIDIFILE_MODIFY_MESSAGE
  '
  ' Save the file using a new name
  '
  MIDIFile1.Filename = newname.mid
  MIDIFile1.Action = MIDIFILE_SAVE_AS
  '
  ' Close the file
  '
  MIDIFile1.Action = MIDIFILE_CLOSE
End Sub
```

Buffer Property Example

In this example, a Sysex message is sent which resets the Roland SoundCanvas SC-88 to General Midi mode.

```
Sub SetGMMode_Click ()
        Midioutput1.Buffer = Chr$(&HF0) + Chr$(&H7E) + Chr$(&H7F) + Chr$(9) + Chr$(1) + Chr$
(&HF7)
        Midioutput1.Message = &HF0
        Midioutput1.Action = MIDIOUT_SEND
End Sub
```

In this example the first and last bytes (&HF0 and &HF7) signal the beginning and end of a Sysex message.   The middle bytes are the Sysex messages contents.

Clocks Property Example

Close Copy Print

Error Event Example

```
Sub MIDIOutput1_Error (ErrorCode As Integer, ErrorMessage As String)
      MsgBox ErrorMessage
End Sub
```

Filename Property Example

This example shows how to open a midi file.   First the CMDialog control is used for its FileOpen Dialog capability, then the user-selected filename is put into the MIDI File control, and finally the file is opened using the MIDI File controls Action property.

```
Sub FileOpen_Click ()
  On Error Resume Next
  CMDialog1.DialogTitle = "Open MIDI File"
  CMDialog1.Flags = &H1000&
  CMDialog1.Action = 1
  If (Err) Then
     Exit Sub
  End If
  MIDIFile1.Filename = CMDialog1.Filename
  MIDIFile1.Action = MIDIFILE_OPEN
End Sub
```

Close Copy Print

Format Property Example

Close Copy Print

Frame Property Example

Close Copy Print

FrameRate Property Example

Close Copy Print

Hour Property Example

Close Copy Print

Message Property Example

The following subroutine shows a sample MIDIInput_Message event handler. All of the available messages are read and output using the MIDI output control, this provides a MIDI-thru capability.

```
Sub MIDIInput1_Message()
 Dim Message As Integer
 Dim Data1   As Integer
 Dim Data2   As Integer

 Do While (MIDIInput1.MessageCount > 0 )
    '
    'This is the incoming MIDI data
    '
    Message = MIDIInput1.Message
    Data1 = MIDIInput1.Data1
    Data2 = MIDIInput1.Data2
    '
    ' Tell MIDIOutput1 to send the MIDI data
    '
    MIDIOutput1.Message = Message
    MIDIOutput1.Data1 = Data1
    MIDIOutput1.Data2 = Data2
    MIDIOutput1.Action = MIDIOUT_SEND
    '
    ' Remove the input message
    '
    MIDIInput1.Action = MIDIIN_REMOVE
 Loop
End Sub
```

Close Copy Print

MessageCount Property Example

The following subroutine shows a sample MIDIInput_Message event handler.   All of the available messages are read and output using the MIDI output control,   this provides a MIDI-thru capability.

```
Sub MIDIInput1_Message()
  Dim Message As Integer
  Dim Data1   As Integer
  Dim Data2   As Integer

  Do While (MIDIInput1.MessageCount > 0 )
      '
      'This is the incoming MIDI data
      '
      Message = MIDIInput1.Message
      Data1 = MIDIInput1.Data1
      Data2 = MIDIInput1.Data2
      '
      ' Tell MIDIOutput1 to send the MIDI data
      '
      MIDIOutput1.Message = Message
      MIDIOutput1.Data1 = Data1
      MIDIOutput1.Data2 = Data2
      MIDIOutput1.Action = MIDIOUT_SEND
      '
      ' Remove the input message
      '
      MIDIInput1.Action = MIDIIN_REMOVE
  Loop
End Sub
```

MessageNumber Property Example

The following searches throught the messages in a track looking for a track name event.

```
Function GetTrackName (Track As Integer) As String
  Dim i As Integer

  MIDIFile1.TrackNumber = Track

  For i = 1 To MIDIFile1.MessageCount
      MIDIFile1.MessageNumber = i
      '
      'Meta Event
      '
      If (MIDIFile1.Message = 255) And MIDIFile1.Data1 = 3 Then
          If (MIDIFile1.MsgText = "") Then
              GetTrackName = "Track" & Str(Track) & " (null)"
          Else
              GetTrackName = MIDIFile1.MsgText
          End If
          Exit Function
      End If
  Next
  GetTrackName = "Track" & Str(Track)
End Function
```

Close Copy Print

MessageTag Property Example

```
Sub MIDIOutput1_MessageSent (MessageTag As Long)
  If (MessageTag = 1) Then
     Shape1.Visible = True
  Else
     Shape1.Visible = False
  End If
End Sub
```

Close Copy Print

Mi Property Example

MsgText Property Example

This example shows how to change the MsgText for the current message.

```
Sub CmdModifyMessage_Click ()
 MIDIFile1.MsgText = MsgTextEdit.Text
 MIDIFile1.Action = MIDIFILE_MODIFY_MESSAGE
End Sub
```

Close Copy Print

Notated32nds Property Example

Close Copy Print

Notes Property Example

Close Copy Print

NumberOfTracks Property Example

This example shows how to load track names into a list box.

```
Sub DisplayTrackList ()
  Dim m As Integer
  Dim t As Integer

  TrackList.Clear
  For t = 1 To MIDIFile1.NumberOfTracks
     TrackList.AddItem GetTrackName(t)
     If (t = 1) Then
        msPerTick = ((MIDIFile1.Tempo) / 1000) /
MIDIFile1.TicksPerQuarterNote
        ticksPerMs = (MIDIFile1.TicksPerQuarterNote / MIDIFile1.Tempo) * 1000
     End If
  Next
End Sub
```

Close Copy Print

Numerator Property Example

Close Copy Print

Sf Property Example

Tempo Property Example

This example shows how to locate a Tempo sysex event in a track and how to calculate MillisecondsPerTick and TicksPerMillisecond..

```
Sub CalculateTimingValues( Track As Integer )
  Dim m As Integer

  MIDIFile1.TrackNumber = Track
  For m = 1 To MIDIFile1.MessageCount
      MIDIFile1.Message = m
      If ((MIDIFile1.Message = &HFF) And (MIDIFile1.Message = &H51)) Then
         msPerTick = ((MIDIFile1.Tempo) / 1000) /
MIDIFile1.TicksPerQuarterNote
         ticksPerMs = (MIDIFile1.TicksPerQuarterNote / MIDIFile1.Tempo) * 1000
      End If
  Next
End Sub
```

Close Copy Print

TicksPerFrame Property Example

Close Copy Print

TicksPerQuarterNote Property Example

This example shows how to locate a Tempo sysex event in a track and how to use TicksPerQuarterNote to calculate MillisecondsPerTick and TicksPerMillisecond..

```
Sub CalculateTimingValues( Track As Integer )
  Dim m As Integer

  MIDIFile1.TrackNumber = Track
  For m = 1 To MIDIFile1.MessageCount
     MIDIFile1.Message = m
     If ((MIDIFile1.Message = &HFF) And (MIDIFile1.Message = &H51)) Then
        msPerTick = ((MIDIFile1.Tempo) / 1000) /
MIDIFile1.TicksPerQuarterNote
        ticksPerMs = (MIDIFile1.TicksPerQuarterNote / MIDIFile1.Tempo) * 1000
     End If
  Next
End Sub
```

Time Property Example

This example shows how to change time for the current message.

```
Sub CmdModifyMessageTime_Click ()
 MIDIFile1.Time = Val(TimeEdit.Text)
 MIDIFile1.Action = MIDIFILE_MODIFY_MESSAGE
End Sub
```

Close Copy Print

TimeFormat Property Example

TrackNumber Property Example

This example shows how to load track names into a list box.

```
Sub DisplayTrackList ()
  Dim m As Integer
  Dim t As Integer

  TrackList.Clear
  For t = 1 To MIDIFile1.NumberOfTracks
      TrackList.AddItem GetTrackName(t)
      If (t = 1) Then
          msPerTick = ((MIDIFile1.Tempo) / 1000) /
MIDIFile1.TicksPerQuarterNote
          ticksPerMs = (MIDIFile1.TicksPerQuarterNote / MIDIFile1.Tempo) * 1000
      End If
  Next
End Sub
```

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact us at the following:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA   98103-1926

Phone: 206-634-1443
Fax: 206-632-0272

CompuServe: 71231,2066
Internet: mabry@mabry.com

You can also contact Zane Thomas.   He can be reached at:

Zane Thomas
Post Office Box 121
Indianola, WA   98342

Internet: zane@mabry.com

Licensing Information

Legalese Version

Mabry Software grants a license to use the enclosed software to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited, and adherence to this requirement is the sole responsibility of the purchaser.

Customer written executable applications containing embedded Mabry products may be freely distributed, without royalty payments to Mabry Software, provided that such distributed Mabry product is bound into these applications in such a way so as to prohibit separate use in design mode, and that such Mabry product is distributed only in conjunction with the customers own software product. The Mabry Software product may not be distributed by itself in any form.

Neither source code for Mabry Software products nor modified source code for Mabry Software products may be distributed under any circumstances, nor may you distribute .OBJ, .LIB, etc. files that contain our routines. This control may be used as a constituent control only if the compound control thus created is distributed with and as an integral part of an application. Permission to use this control as a constituent control does not grant a right to distribute the license (LIC) file or any other file other than the control executable itself.This license may be transferred to a third party only if all existing copies of the software and its documentation are also transferred.

This product is licensed for use by only one developer at a time. Mabry Software expressly prohibits installing this product on more than one computer if there is any chance that both copies will be used simultaneously. This restriction also extends to installation on a network server, if more than one workstation will be accessing the product. All developers working on a project which includes a Mabry Software product, even though not working directly with the Mabry product, are required to purchase a license for that Mabry product.

This software is provided as is. Mabry Software makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

MABRY SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Mabry Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.

To be eligible for free technical support by telephone, the Internet, CompuServe, etc. and to ensure that you are notified of any future updates, please complete the enclosed registration card and return it to Mabry Software.

English Version

We require that you purchase one copy of a control per developer on a project. If this is met, you may distribute the control with your application royalty free. You may never distribute the LIC file. You may not change the product in any way that removes or changes the requirement of a license file.

We encourage the use of our controls as constituent controls when the compound controls you create are an integral part of your application. But we don't allow distribution of our controls as constituents of other controls when the compound control is not part of an application. The reason we need to have this restriction is that without it someone might decide to use our control as a constituent, add some trivial (or even non-trivial) enhancements and then sell the compound control. Obviously there would be little difference between that and just plain reselling our control.

If you have purchased the source code, you may not re-distribute the source code either (nor may you copy it into your own project). Mabry Software retains the copyright to the source code.

Your license is transferable. The original purchaser of the product must make the transfer request. Contact us for further information.