




## MIDI I/O and Visual Custom Controls

When you add a custom control to your project, its icon is displayed in the Toolbox. You can select the custom control by clicking on its icon.

The toolbox icons for the custom controls in this package are listed below:

	<u>Horizontal Slider</u>		<u>Knob</u>
	<u>Vertical Slider</u>		<u>MIDI Input</u>
	<u>Vertical Indicator</u>		<u>MIDI Output</u>
	<u>Horizontal Indicator</u>		<u>MIDI File</u>



## Help for Horizontal Indicator VBX

[Properties](#)

[Events](#)

### Description

Put description here.

### File Name

HINDIC.VBX

### Object Type

HIndicator

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>BackColor</u>	<u>*ItemBackColor</u>	<u>*Max</u>
<u>*BevelInner</u>	<u>*ItemCount1</u>	<u>*Min</u>
<u>*BevelOuter</u>	<u>*ItemCount2</u>	<u>Name</u>
<u>*BevelWidth</u>	<u>*ItemCount3</u>	<u>Parent</u>
<u>*Border</u>	<u>*ItemForeColor1</u>	<u>Tag</u>
<u>*BorderWidth</u>	<u>*ItemForeColor2</u>	<u>*ThreeD</u>
<u>Enabled</u>	<u>*ItemForeColor3</u>	<u>Top</u>
<u>Height</u>	<u>Left</u>	<u>*Value</u>
<u>hWnd</u>	<u>*LinkControl</u>	<u>Visible</u>
<u>Index</u>	<u>*LinkProperty</u>	<u>Width</u>

Value is the default value for the control.

## 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 (\*).

Click

DbClick



## Help for Horizontal Slider VBX

[Properties](#)

[Events](#)

### Description

Put description here.

### File Name

HSLIDE.VBX

### Object Type

HSlider

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>BackColor</u>	<u>Left</u>	<u>*TickColor</u>
<u>*BevelInner</u>	<u>*LinkControl</u>	<u>*TickCount</u>
<u>*BevelOuter</u>	<u>*LinkProperty</u>	<u>*TickLength</u>
<u>*BevelWidth</u>	<u>*Max</u>	<u>*TickMarks</u>
<u>*BorderWidth</u>	<u>*Min</u>	<u>*TickWidth</u>
<u>Enabled</u>	<u>Name</u>	<u>Top</u>
<u>*Gap</u>	<u>Parent</u>	<u>*TrackBevel</u>
<u>Height</u>	<u>Tag</u>	<u>*TrackWidth</u>
<u>hWnd</u>	<u>*ThumbHeight</u>	<u>*Value</u>
<u>Index</u>	<u>*ThumbStyle</u>	<u>Visible</u>
<u>*LargeChange</u>	<u>*ThumbWidth</u>	<u>Width</u>

Value is the default value for the control.

## 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 (\*).

\*Change

GotFocus

LostFocus

MouseDown

MouseMove

MouseUp

\*Scroll



## Help for Knob VBX

[Properties](#)

[Events](#)

### **Description**

This control displays a knob (round) that behaves like a slider or scroll bar.

### **File Name**

KNOB.VBX

### **Object Type**

Knob

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---



## 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 (\*).

<u>BackColor</u>	<u>Height</u>	<u>*LinkProperty</u>	<u>*TickCount</u>
<u>*BevelWidth</u>	<u>hWnd</u>	<u>*Max</u>	<u>*TickGap</u>
<u>*BorderWidth</u>	<u>Index</u>	<u>*Min</u>	<u>*TickLength</u>
<u>Enabled</u>	<u>*Indicator</u>	<u>Name</u>	<u>*TickWidth</u>
<u>FontBold</u>	<u>*IndicatorColor</u>	<u>Parent</u>	<u>Top</u>
<u>FontItalic</u>	<u>*IndicatorWidth</u>	<u>*Radius</u>	<u>*Value</u>
<u>FontName</u>	<u>*KnobColor</u>	<u>Tag</u>	<u>Visible</u>
<u>FontSize</u>	<u>*KnobStyle</u>	<u>*TickCaption</u>	<u>Width</u>
<u>FontStrikethru</u>	<u>Left</u>	<u>*TickCaptionColor</u>	
<u>FontUnderline</u>	<u>*LinkControl</u>	<u>*TickColor</u>	

Value is the default value for the control.

## 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 (\*).

\*Change

GotFocus

LostFocus

MouseDown

MouseMove

MouseUp

\*Scroll



## Help for MIDI File VBX

[Properties](#)

[Events](#)

### Description

The MIDIFILE VBX 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, and you can insert, delete and modify tracks and messages at anytime.

### File Name

MIDIFILE.VBX

### Object Type

MIDIFile

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

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

**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



## Help for MIDI Input VBX

[Properties](#)

[Events](#)

### Description

The MIDIIN VBX is used to receive MIDI messages from external MIDI devices. Messages can be retrieved using Events or polling, and are time-stamped with millisecond accuracy. The MIDIIN VBX has an internal queuing mechanism so if messages arrive faster than your application can handle them they will not be lost.

### File Name

MIDIIN.VBX

### Object Type

MIDIInput

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>*Action</u>	<u>Enabled</u>	<u>Name</u>
<u>Align</u>	<u>*HMidiDevice</u>	<u>*ProductID</u>
<u>*Buffer</u>	<u>Index</u>	<u>*ProductName</u>
<u>*Data1</u>	<u>Left</u>	<u>*State</u>
<u>*Data2</u>	<u>*Message</u>	<u>Tag</u>
<u>*DeviceCount</u>	<u>*MessageCount</u>	<u>*Time</u>
<u>*DeviceID</u>	<u>*MessageEventEnable</u>	
<u>*DriverVersion</u>	<u>*ManufacturerID</u>	<u>Top</u>

## 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

\*Message





## Help for MIDI Output VBX

[Properties](#)

[Events](#)

### Description

The MIDIOUT VBX gives you complete control over the contents and timing of MIDI messages sent to either internal or external MIDI devices. You can queue as many messages as you like (within the constraints of available memory) before starting output, or you can queue one or more messages prior to starting output and then add more as the output proceeds. Messages are scheduled for transmission at a time you specify relative to the time that output is started. As with the MIDIIN control timing has millisecond resolution, giving you the ability to precisely control the timing of sent MIDI messages.

### File Name

MIDIOUT.VBX

### Object Type

MIDIOutput

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>*Action</u>	<u>Enabled</u>	<u>*ProductID</u>
<u>Align</u>	<u>*HasLRVolume</u>	<u>*ProductName</u>
<u>*Buffer</u>	<u>*HasVolume</u>	<u>*State</u>
<u>*CanCache</u>	<u>*HMidiDevice</u>	<u>Tag</u>
<u>*Channels</u>	<u>Index</u>	<u>*Time</u>
<u>*Data1</u>	<u>Left</u>	<u>Top</u>
<u>*Data2</u>	<u>*ManufacturerID</u>	<u>*Voices</u>
<u>*DeviceCount</u>	<u>*Message</u>	<u>*VolumeLeft</u>
<u>*DeviceID</u>	<u>*MessageTag</u>	<u>*VolumeRight</u>
<u>*DeviceType</u>	<u>Name</u>	
<u>*DriverVersion</u>	<u>*Notes</u>	

## 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

\*MessageSent

\*QueueEmpty

\*Timer



## Help for Vertical Indicator VBX

[Properties](#)

[Events](#)

### Description

Put description here.

### File Name

VINDIC.VBX

### Object Type

VIndicator

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>BackColor</u>	<u>*ItemBackColor</u>	<u>*Max</u>
<u>*BevelInner</u>	<u>*ItemCount1</u>	<u>*Min</u>
<u>*BevelOuter</u>	<u>*ItemCount2</u>	<u>Name</u>
<u>*BevelWidth</u>	<u>*ItemCount3</u>	<u>Parent</u>
<u>*Border</u>	<u>*ItemForeColor1</u>	<u>Tag</u>
<u>*BorderWidth</u>	<u>*ItemForeColor2</u>	<u>*ThreeD</u>
<u>Enabled</u>	<u>*ItemForeColor3</u>	<u>Top</u>
<u>Height</u>	<u>Left</u>	<u>*Value</u>
<u>hWnd</u>	<u>*LinkControl</u>	<u>Visible</u>
<u>Index</u>	<u>*LinkProperty</u>	<u>Width</u>

Value is the default value for the control.

## 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 (\*).

Click

DbClick



## Help for Vertical Slider VBX

[Properties](#)

[Events](#)

### Description

Put description here.

### File Name

VSLIDE.VBX

### Object Type

VSlider

**Distribution Note** When you develop and distribute an application that uses this control, you should install the VBX 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.

---

## 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 (\*).

<u>BackColor</u>	<u>Left</u>	<u>*TickColor</u>
<u>*BevelInner</u>	<u>*LinkControl</u>	<u>*TickCount</u>
<u>*BevelOuter</u>	<u>*LinkProperty</u>	<u>*TickLength</u>
<u>*BevelWidth</u>	<u>*Max</u>	<u>*TickMarks</u>
<u>*BorderWidth</u>	<u>*Min</u>	<u>*TickWidth</u>
<u>Enabled</u>	<u>Name</u>	<u>Top</u>
<u>*Gap</u>	<u>Parent</u>	<u>*TrackBevel</u>
<u>Height</u>	<u>Tag</u>	<u>*TrackWidth</u>
<u>hWnd</u>	<u>*ThumbHeight</u>	<u>*Value</u>
<u>Index</u>	<u>*ThumbStyle</u>	<u>Visible</u>
<u>*LargeChange</u>	<u>*ThumbWidth</u>	<u>Width</u>

Value is the default value for the control.



## 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 (\*).

\*Change

GotFocus

LostFocus

MouseDown

MouseMove

MouseUp

\*Scroll

## Action Property, MIDI File Control

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#)

### Description

Action to take using current [DeviceID](#).

### Usage

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

### Remarks

Setting this property causes an action to occur using current [DeviceID](#). The actions are:

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

### Data Type

Integer

**See Also**

Properties:

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## Action Property, MIDI Input Control

[See Also](#)

[Example](#)

### Applies To

[MIDI input](#)

### Description

Action to take using current [DeviceID](#).

### Usage

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

### Remarks

Setting this property causes an action to occur using current [DeviceID](#). The actions are:

<b>Value</b>	<b>Meaning</b>
0	No action
1	Open device
2	Close device
3	Reset MIDI device.
4	Start MIDI input
5	Stop MIDI input
6	Remove current MIDI message from queue

### Data Type

Integer

**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Output\)](#)

## Action Property, MIDI Output Control

[See Also](#)

[Example](#)

### Applies To

[MIDI Output](#)

### Description

Action to take using current [DeviceID](#).

### Usage

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

### Remarks

Setting this property causes an action to occur using current [DeviceID](#). The actions are:

<b>Value</b>	<b>Meaning</b>
0	No action
1	Open device
2	Close device
3	Reset MIDI device.
4	Start MIDI output
5	Stop MIDI output
6	Queue message given by <a href="#">Message</a> , <a href="#">Data1</a> , and <a href="#">Data2</a> will be queued for playing at <a href="#">Time</a> milliseconds after output is started (Action = 4)
7	Immediate. Sends the message given by <a href="#">Message</a> , <a href="#">Data1</a> , and <a href="#">Data2</a> immediately if output is started (Action = 5)
8	Timer. Fires a <a href="#">Timer</a> event when <a href="#">Time</a> milliseconds have elapsed. This provides a high-resolution timer for you to use. When the <a href="#">Timer</a> event is fired, it will pass back to you the contents of the <a href="#">MessageTag</a> property in effect at the time that action was set to <a href="#">Time</a> .
9	Pauses the sending of queued message and stops the queue timer clock.

### Data Type

Integer

**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

## BevelInner Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Determines the 3-D style of the border immediately surrounding the control.

### Usage

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

### Remarks

The value of this property determines the style of the inner border. This property may be one of four values:

<b>Value</b>	<b>Description</b>
0	Normal frame
1	Raised frame (3-D)
2	Inset frame (3-D)
3	Lowered frame (3-D)

### Data Type

Integer (enumerated)



**See Also**

Properties:

[BevelOuter](#)

[BevelWidth](#)

[BorderWidth](#)

## BevelOuter Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Determines the 3-D style of the border (if any) surrounding the control.

### Usage

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

### Remarks

The value of this property determines the style of the control's border. This property may be one of four values:

<b>Value</b>	<b>Description</b>
0	Normal frame
1	Raised frame (3-D)
2	Inset frame (3-D)
3	Lowered frame (3-D)

### Data Type

Integer (enumerated)

**See Also**

Properties:

[BevelInner](#)

[BevelWidth](#)

[BorderWidth](#)

## BevelWidth Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Knob](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Determines the width of the inner and outer borders (bevels).

### Usage

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

### Remarks

The value of this property determines the width of the inner border (if any, see [BevelInner](#)) and the outer border (if any, see [Border](#) and [BevelOuter](#)). This is always measured in pixels.

In the case of the [Knob](#) control, this determines the width of the bevel that surrounds the knob.

### Data Type

Integer

**See Also**

Properties:

[BevelInner](#)

[BevelOuter](#)

[Border](#)

[BorderWidth](#)

## Border Property

[See Also](#)

### Applies To

[Horizontal Indicator](#), [Vertical Indicator](#)

### Description

Determines if a border is used.

### Usage

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

### Remarks

The value of this property determines the style of the border. If this property is set to None, no border (inner or outer) is used. This property may be one of the following values:

<b>Value</b>	<b>Description</b>
0	None
1	Single width

### Data Type

Integer (enumerated)

**See Also**

Properties:

[BevelInner](#)

[BevelOuter](#)

[BevelWidth](#)

[BorderWidth](#)

## BorderWidth Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Knob](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Determines the distance between the inner border and the outer border.

### Usage

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

### Remarks

The value of this property determines the distance between the outer border (if any, see [Border](#) and [BevelOuter](#)) and the inner border (if any, see [BevelInner](#)). This is always measured in pixels.

With the [Knob](#) control, this property determines the distance between the bevel on the knob, and the outside edge of the [indicator](#) .

### Data Type

Integer



**See Also**

Properties:

[BevelInner](#)

[BevelOuter](#)

## Buffer Property

[Example](#)

### Applies To

[MIDI file](#), [MIDI input](#), [MIDI output](#)

### 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. For instance:

```
Dim sysexMsg as string
sysexMsg = &HF0 + GetRestOfSysexMessage() + &HF7
```

and 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

## CanCache Property

### Applies To

MIDI output

### Description

Specifies whether or not the current device supports patch caching.

### Usage

[*form.*][*control.*]**CanCache**

### Remarks

This property is read-only.

### Data Type

Integer (boolean)

## Channels Property

### Applies To

MIDI output

### Description

Specifies channels device supports.

### Usage

[*form.*][*control.*]**Channels**( *ChannelIndex* )

### Remarks

Elements in this array are True for each channel (specified by *ChannelIndex*) this device will respond to.

This property is read-only.

### Data Type

Integer

## Clocks Property

### Applies To

MIDI file

### 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](#)

[Example](#)

### Applies To

[MIDI file](#), [MIDI input](#), [MIDI output](#)

### 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.

### Data Type

Integer (0-255)

**See Also**

Properties:

[Message](#)

## Denominator Property

[See Also](#)

### Applies To

[MIDI file](#)

### Description

Denominator represents the denominator of a time signature as it would be notated.

### 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](#)

## DeviceCount Property

Example

### Applies To

MIDI input, MIDI output

### Description

Determines the number of MIDI devices.

### Usage

[*form.*][*control.*]**DeviceCount**

### Remarks

This property determines the number of MIDI devices available. Note that the number of input devices may not be the same as the number of output devices.

This property is read-only.

### Data Type

Integer

## DeviceID Property

[See Also](#)

[Example](#)

### Applies To

[MIDI input](#), [MIDI output](#)

### Description

Determines the device to use.

### Usage

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

### Remarks

In the MIDI output control this property ranges from -1 through DeviceCount - 1, a value of -1 represents the MIDI mapper and all other values represent a MIDI device.

In the MIDI input control this property ranges from zero through DeviceCount - 1, with all values representing MIDI devices.

### Data Type

Integer

**See Also**

Properties:

[DeviceCount](#)

## DeviceType Property

[See Also](#)

### Applies To

[MIDI output](#)

### Description

Type of device currently selected.

### Usage

[*form.*][*control.*]**Voices**

### Remarks

Specifies type of device selected by [DeviceID](#). Values are:

<b>Value</b>	<b>Meaning</b>
0	MIDI hardware port
1	Square wave synthesizer
2	FM synthesizer
3	MIDI mapper

This property is read-only.

### Data Type

Integer

**See Also**

Properties:

[DeviceID](#)

## DriverVersion Property

[See Also](#)

### Applies To

[MIDI input](#), [MIDI output](#)

### Description

Driver version of [DeviceID](#).

### Usage

[*form.*][*control.*]**DriverVersion**

### Remarks

This property returns the driver version number for the device specified by [DeviceID](#). The high-byte contains the major version number and the low-byte contains the minor version number.

This property is read-only.

### Data Type

Integer

**See Also**

Properties:

[DeviceCount](#)



## Filename Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#)

### 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

## Format Property

### Applies To

MIDI file

### Description

Determines the format of the current MIDI file.

### Usage

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

### Remarks

Determines the format of the current MIDI file.

<b>Value</b>	<b>Meaning</b>
0	Single track
1	One or more simultaneous tracks

### Data Type

Integer

## Frame and FractionalFrames Properties

[See Also](#)

### Applies To

[MIDI file](#)

### Description

Determines the offset of a message

### Usage

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

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

### Remarks

These properties specify 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](#)

### Applies To

[MIDI file](#)

### 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](#)

## Gap Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Vertical Slider](#)

### Description

Determines the distance between the inside of the border and the tick marks.

### Usage

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

### Remarks

The value of this property determines the distance between the inner border and the tick marks. This property is measured in pixels.

### Data Type

Integer



**See Also**

Properties:

[BevelInner](#)

[BevelOuter](#)

[BevelWidth](#)

[BorderWidth](#)

## HasLRVolume Property

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

Specifies whether or not the current device supports separate left and right volume control.

### Usage

[*form.*][*control.*]**HasLRVolume**

### Remarks

Specifies whether or not the current device ([DeviceID](#)) supports separate left and right volume control.

This property is read-only.

### Data Type

Integer (boolean)

**See Also**

Properties:

[HasVolume](#)

## HasVolume Property

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

Specifies whether or not the current device supports volume.

### Usage

[*form.*][*control.*]**HasVolume**

### Remarks

Specifies whether or not the current device ([DeviceID](#)) supports volume.

This property is read-only.

### Data Type

Integer (boolean)

**See Also**

Properties:

[HasLRVolume](#)

## **HMidiDevice Property**

[See Also](#)

### **Applies To**

[MIDI input](#), [MIDI output](#)

### **Description**

Handle of MIDI device.

### **Usage**

[*form.*][*control.*]**HMidiDevice**

### **Remarks**

Device handle of MIDI device specified by [DeviceID](#). Only valid while device is open.

### **Data Type**

Integer

**See Also**

Properties:

[Action \(MIDI input\)](#)

[Action \(MIDI output\)](#)

## Hour, Minute, and Second Properties

### Applies To

MIDI file

### Description

Determines the time offset of a message

### Usage

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

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

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

### Remarks

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

### Data Type

Integer (0-255)



## Indicator Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines what style of indicator to use for the knob.

### Usage

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

### Remarks

The value of this property determines what kind of indicator to use for the knob.

<b>Value</b>	<b>Description</b>
0	Spot
1	Line

### Data Type

Integer (enumerated)

**See Also**

Properties:

[IndicatorColor](#)

[IndicatorWidth](#)

[Value](#)

## IndicatorColor Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines what color the indicator will be.

### Usage

[*form.*][*control.*]**IndicatorColor**[ = *color* ]

### Remarks

This property determines the color of the indicator on the knob.

### Data Type

Color

**See Also**

Properties:

[Indicator](#)

[IndicatorWidth](#)

## IndicatorWidth Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines what width the indicator will be.

### Usage

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

### Remarks

This property determines the width of the indicator on the knob.

### Data Type

Integer

**See Also**

Properties:

[Indicator](#)

[IndicatorWidth](#)

## ItemBackColor Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Vertical Indicator](#)

### Description

Determines the color of the background of the items.

### Usage

[*form.*][*control.*]**ItemBackColor**[ = *color* ]

### Remarks

This property specifies the color of the item backgrounds. The items are filled with this color when not "on" (i.e. filled with one of the ItemForeColor).

### Data Type

Color

**See Also**

Properties:

[ItemForeColor1](#)

[ItemForeColor2](#)

[ItemForeColor3](#)



## ItemCount1, ItemCount2, and ItemCount3 Properties

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Vertical Indicator](#)

### Description

Determines the number of items in the indicator.

### Usage

[*form.*][*control.*]**ItemCount1**[ = *integer* ]

[*form.*][*control.*]**ItemCount2**[ = *integer* ]

[*form.*][*control.*]**ItemCount3**[ = *integer* ]

### Remarks

This property specifies the number of the items in the control. These properties must be greater than or equal to zero. If all three are zero, no items are displayed.

The first ItemCount1 items are painted with [ItemForeColor1](#). The next ItemCount2 items are painted with [ItemForeColor2](#). And, the remaining ItemCount3 items are painted with [ItemForeColor3](#).

### Data Type

Integer

## See Also

Properties:

[ItemBackColor](#)

[ItemForeColor1](#)

[ItemForeColor2](#)

[ItemForeColor3](#)

[Max](#)

[Min](#)

[Value](#)

## ItemForeColor1, ItemForeColor2, and ItemForeColor3 Properties

[See Also](#)

[Example](#)

### Applies To

[Horizontal Indicator](#), [Vertical Indicator](#)

### Description

Determines the color of the selected items.

### Usage

[*form.*][*control.*]**ItemForeColor1**[ = *color* ]

[*form.*][*control.*]**ItemForeColor2**[ = *color* ]

[*form.*][*control.*]**ItemForeColor3**[ = *color* ]

### Remarks

This property specifies the color of the items when they are selected (this is dependent upon the [Min](#), [Max](#), [Value](#), and [ItemCount](#) properties).

The first [ItemCount1](#) items are painted with ItemForeColor1. The next [ItemCount2](#) items are painted with ItemForeColor2. And, the remaining [ItemCount3](#) items are painted with ItemForeColor3.

### Data Type

Color

## See Also

Properties:

[ItemBackColor](#)

[ItemCount1](#)

[ItemCount2](#)

[ItemCount3](#)

[Max](#)

[Min](#)

[Value](#)

## **KnobColor Property**

[See Also](#)

[Example](#)

### **Applies To**

[Knob](#)

### **Description**

Determines the knob's color.

### **Usage**

[*form.*][*control.*]**KnobColor**[ = *color* ]

### **Remarks**

This property determines the color of the knob's face.

### **Data Type**

Color

**See Also**

Properties:

[KnobStyle](#)

## KnobStyle Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines the knob's style.

### Usage

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

### Remarks

This property determines the style of the knob. Valid values are:

<b>Value</b>	<b>Description</b>
0	Normal
1	Raised
2	Lowered
3	Textured

### Data Type

Integer

**See Also**

Properties:

[KnobColor](#)



## LargeChange Property

### Applies To

Horizontal Slider, Vertical Slider

### Description

Determines the how far the slider moves when clicked outside the thumb..

### Usage

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

### Remarks

The value of this property determines how far the thumb moves when the control is clicked outside the thumb and near the track.

### Data Type

Integer

## VolumeLeft Property

Example

### Applies To

MIDI output

### Description

Sets left side volume

### Usage

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

### Remarks

Sets the volume for the left channel of DeviceID. This value must range from 0 to 32767. If HasLRVolume is False, setting this property sets both VolumeLeft and VolumeRight.

You should save the VolumeRight and VolumeLeft properties when you open a MIDI device that supports volume control, and restore the properties just before you close the device. If you do not restore the properties the default volume for the MIDI device will be changed system-wide.

### Data Type

Integer (0-32767)

## LinkControl and LinkProperty Properties

### Applies To

Horizontal Indicator, Horizontal Slider, Knob, Vertical Indicator, Vertical Slider

### Description

Sets up link to another control.

### Usage

[*form.*][*control.*]**LinkControl**

[*form.*][*control.*]**LinkProperty**

### Remarks

These properties set up a link with another control. When the Value property changes, the control sends the new value to the control and property specified by these properties. If the other control is one of the controls in this package (i.e., Horizontal Indicator, Horizontal Slider, Knob, MIDI File, MIDI Input, MIDI Output, Vertical Indicator, or Vertical Slider), the current control's Value property will be updated when the other control's property changes.

At design-time, be sure to set the LinkControl property first. The LinkProperty combo box will display all of the valid properties for that control.

These properties are changable at design-time, and read-only at run-time.

### Data Type

String

## **ManufacturerID Property**

[See Also](#)

### **Applies To**

[MIDI input](#), [MIDI output](#)

### **Description**

Manufacturer's ID for [DeviceID](#).

### **Usage**

[*form.*][*control.*]**ManufacturerID**

### **Remarks**

This property returns the manufacturer's ID number for the device specified by [DeviceID](#).

This property is read-only.

### **Data Type**

Integer

**See Also**

Properties:

ProductID

## Message Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#), [MIDI input](#), [MIDI output](#)

### 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](#)

### Applies To

[MIDI file](#), [MIDI input](#)

### 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)



## MessageEventEnable Property

### Applies To

MIDI input

### Description

Enables Message event.

### Usage

[*form.*][*control.*]**MessageEventEnable**[ = *boolean* ]

### Remarks

When this property is set to True, the Message event will be fired whenever messages are available. When this property is set to False, the Message event will not be fired.

### Data Type

Integer (boolean)

## MessageNumber Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#)

### 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](#)

## MessageTag Property

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

The MessageTag property allows you to associate a long integer value with each particular MIDI message. When a MIDI message with a non-zero MessageTag is sent the MessageSent event will be fired.

### Usage

*[form.]***MessageTag**[ = *long* ]

### Remarks

Using the MessageTag property and MessageSent event you can synchronize your program with MIDI events of your choosing.

### Data Type

Integer (long)

**See Also**

Events:

[MessageSent](#)

## **Mi Property**

[See Also](#)

### **Applies To**

[MIDI file](#)

### **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

## Min and Max Properties

[See Also](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Knob](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Determines the range of values for this control.

### Usage

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

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

### Remarks

These properties determine the range of values for the control in question. If Max is set to less than Min, then the range of values is swapped.

### Data Type

Integer



**See Also**

Properties:

Value

## **MsgText Property**

Example

### **Applies To**

MIDI file

### **Description**

String representing meta-event.

### **Usage**

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

### **Remarks**

Specifies the name of the meta event.

<b>Value</b>	<b>Meaning</b>
1	Non-specific text string
2	Copyright notice
3	Sequence/track name
4	Instrument name
5	Lyric
6	Marker
7	Cue point
8-15	Undefined text string

This property is read-only.

### **Data Type**

Integer

## **Notated32nds Property**

[See Also](#)

### **Applies To**

[MIDI file](#)

### **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](#)

## Notes Property

### Applies To

MIDI output

### Description

Number of simultaneous notes the device may play.

### Usage

[*form.*][*control.*]**Notes**

### Remarks

Number of simultaneous notes (polyphony) that may be played by internal DeviceID. Always zero for MIDI ports.

This property is read-only.

### Data Type

Integer

## NumberOfTracks Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#)

### 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](#)

### **Applies To**

[MIDI file](#)

### **Description**

The numerator of the time signature as it would be notated.

### **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](#)

## **ProductID Property**

[See Also](#)

### **Applies To**

[MIDI input](#), [MIDI output](#)

### **Description**

Product ID for [DeviceID](#).

### **Usage**

[*form.*][*control.*]**ProductID**

### **Remarks**

This property returns the product ID number for the device specified by [DeviceID](#).

This property is read-only.

### **Data Type**

Integer

**See Also**

Properties:

ManufacturerID

## ProductName Property

### Applies To

MIDI input, MIDI output

### Description

Product name for DeviceID.

### Usage

[*form.*][*control.*]**ProductName**

### Remarks

This property returns the product name for the device specified by DeviceID.

This property is read-only.

### Data Type

String

**See Also**

Properties:

[DeviceID](#)

## Radius Property

Example

### Applies To

Knob

### Description

Determines what size of the knob.

### Usage

[*form.*][*control.*]**Radius**[ = *radius* ]

### Remarks

This property determines the size of the knob. When the knob is sized at design-time, this property is automatically scaled.

### Data Type

Real

## VolumeRight Property

Example

### Applies To

MIDI output

### Description

Sets right side volume

### Usage

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

### Remarks

Sets the volume for the left channel of DeviceID. This value must range from 0 to 32767. If HasLRVolume is False, setting this property does nothing.

You should save the VolumeRight and VolumeLeft properties when you open a MIDI device that supports volume control, and restore the properties just before you close the device. If you do not restore the properties the default volume for the MIDI device will be changed system-wide.

### Data Type

Integer (0-32767)

## Sequence Property

### Applies To

MIDI file

### Description

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



## Sequence Property

### Applies To

MIDI file

### Description

When reading/writing meta-event 0, this property contains the sequence number.

### Usage

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

### Remarks

When reading/writing meta-event 0, this property contains the sequence number.

### Data Type

Integer

## **Sf Property**

[See Also](#)

### **Applies To**

[MIDI file](#)

### **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 specifies 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

## State Property

[SeeAlso](#)

[Example](#)

### Applies To

[MIDI input](#), [MIDI output](#)

### Description

Current state of [DeviceID](#).

### Usage

[*form.*][*control.*]**State**

### Remarks

Setting this property returns the state of [DeviceID](#). The states are:

<b>Value</b>	<b>Meaning</b>
0	Closed
1	Open
2	Started
3	Stopped
4	Paused

This property is read-only.

### Data Type

Integer

**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## **Tempo Property**

Example

### **Applies To**

MIDI file

### **Description**

Sets the tempo.

### **Usage**

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

### **Remarks**

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

### **Data Type**

Integer (long)

## ThreeD Property

[See Also](#)

### Applies To

[Horizontal Indicator](#), [Vertical Indicator](#)

### Description

Determines whether or not 3-D styles are used.

### Usage

[*form.*][*control.*]**ThreeD**[ = *boolean* ]

### Remarks

If this property is set to False, no 3-D style bevels are used. If this property is set to True, any bevel can be used.

### Data Type

Integer (boolean)

**See Also**

Properties:

[BevelInner](#)

[BevelOuter](#)



## ThumbHeight and ThumbWidth Properties

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Vertical Slider](#)

### Description

Determines the size of the thumb.

### Usage

[*form.*][*control.*]**ThumbHeight**[ = *height* ]

[*form.*][*control.*]**ThumbWidth**[ = *width* ]

### Remarks

The value of these properties determine the size of the thumb. These properties are measured in twips.

### Data Type

Real

**See Also**

Properties:

[ThumbStyle](#)

## ThumbStyle Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Vertical Slider](#)

### Description

Determines the style of the thumb.

### Usage

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

### Remarks

The value of this property determines the style of the control's border. This property may be one of four values:

<b>Value</b>	<b>Description</b>
0	Normal
1	Pointer up/left
2	Pointed down/right
3	Lined

### Data Type

Integer (enumerated)

**See Also**

Properties:

[ThumbHeight](#)

[ThumbWidth](#)

## TickCaption Property

[See Also](#)

### Applies To

[Knob](#)

### Description

Determines what captions will be on the tick marks.

### Usage

[*form.*][*control.*]**TickCaption**( *TickIndex* )[ = *string* ]

### Remarks

This property array specifies the text that's associated with each tick mark. *TickIndex* is numbered from 0 to (TickCount - 1), starting at the left-bottom of the knob and moving around clock-wise.

You can set this property at design-time by selecting this property, and then pressing the ellipsis button. The dialog box that pops up lets you enter and edit captions.

### Data Type

Integer

**See Also**

Properties:

[TickCaptionColor](#)

[TickColor](#)

[TickCount](#)

[TickGap](#)

[TickLength](#)

[TickWidth](#)

## TickCaptionColor Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines what color the tick caption text.

### Usage

[*form.*][*control.*]**TickCaptionColor** [ = *color* ]

### Remarks

This property sets the color of the tick captions.

### Data Type

Color

## See Also

Properties:

[TickCaption](#)

[TickColor](#)

[TickCount](#)

[TickGap](#)

[TickLength](#)

[TickWidth](#)



## TickColor Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Determines what color the ticks will be.

### Usage

[*form.*][*control.*]**TickColor**[ = *color* ]

### Remarks

This property specifies the color of the tick marks.

### Data Type

Color

## See Also

Properties:

[TickCaption](#)

[TickCaptionColor](#)

[TickCount](#)

[TickGap](#)

[TickLength](#)

[TickWidth](#)

## TickCount Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Determines how many tick marks there will be.

### Usage

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

### Remarks

This property determines how many tick marks there will be.

### Data Type

Integer

## See Also

Properties:

[TickCaption](#)

[TickCaptionColor](#)

[TickColor](#)

[TickGap](#)

[TickLength](#)

[TickWidth](#)

## TickGap Property

[See Also](#)

[Example](#)

### Applies To

[Knob](#)

### Description

Determines the distance between the tick marks and the knob.

### Usage

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

### Remarks

This property specifies the distance between the inside edge of the tick marks and the outside edge of the knob. This property is measured in pixels.

### Data Type

Integer

## See Also

Properties:

[TickCaption](#)

[TickCaptionColor](#)

[TickColor](#)

[TickCount](#)

[TickLength](#)

[TickWidth](#)

## TickLength Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Determines the length of the tick marks.

### Usage

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

### Remarks

This property specifies the length, in pixels, of the tick marks.

### Data Type

Integer

## See Also

Properties:

[TickCaption](#)

[TickCaptionColor](#)

[TickColor](#)

[TickCount](#)

[TickGap](#)

[TickWidth](#)



## TickMarks Property

[See Also](#)

### Applies To

[Horizontal Slider](#), [Vertical Slider](#)

### Description

Determines where the ticks will appear.

### Usage

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

### Remarks

This property where the tick marks will be. The legitimate values are:

<b>Value</b>	<b>Meaning</b>
0	No tick marks
1	Top for HSlider, Left for VSlider
2	Bottom for HSlider, Right for VSlider
3	Both

### Data Type

Integer

**See Also**

Properties:

[TickColor](#)

[TickCount](#)

[TickLength](#)

[TickWidth](#)

## TicksPerFrame Property

### Applies To

MIDI file

### 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

### Applies To

MIDI file

### 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

## TickWidth Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Determines the width of the tick marks.

### Usage

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

### Remarks

This property determines the width of the tick marks. This property is measured in pixels.

### Data Type

Integer

## See Also

Properties:

[TickCaption](#)

[TickCaptionColor](#)

[TickColor](#)

[TickCount](#)

[TickGap](#)

[TickLength](#)

## Time Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#), [MIDI input](#), [MIDI output](#)

### 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:

$$\text{Beats Per Minute} = 60,000,000 / \text{Tempo}$$

The number of Milliseconds Per Tick is:

$$\text{Milliseconds Per Tick} = (\text{Tempo} / 1000) / \text{TicksPerQuarterNote}$$

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:

$$\text{Millisecond Delay} = \text{Ticks between events} * \text{Milliseconds Per Tick}$$

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

$$\text{Ticks Per Milliseconds} = (\text{MIDIFile1.TicksPerQuarterNote} / \text{MIDIFile1.Tempo}) * 1000$$

Then convert elapsed milliseconds to ticks like this:

$$\text{Ticks between events} = \text{Milliseconds between events} * \text{Ticks Per Milliseconds}$$

### Data Type

Integer (long)

**See Also**

Properties:

[TicksPerQuarterNote](#)

[Tempo](#)



## TimeFormat Property

[See Also](#)

### Applies To

[MIDI file](#)

### Description

Determines the method of time-keeping used.

### Usage

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

### Remarks

Determines the method of time-keeping used.

<b>Value</b>	<b>Meaning</b>
0	Ticks per quarter note (see <a href="#">TicksPerQuarterNote</a> )
1	SMPTE/MIDI (see <a href="#">FrameRate</a> and <a href="#">TicksPerFrame</a> )

### Data Type

Integer

**See Also**

Properties:

[Time](#)

## TrackBevel Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Vertical Slider](#)

### Description

Determines the 3-D style of the track.

### Usage

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

### Remarks

The value of this property determines the style of the control's border. This property may be one of four values:

<b>Value</b>	<b>Description</b>
0	Normal track
1	Raised track (3-D)
2	Inset track (3-D)
3	Lowered track (3-D)

### Data Type

Integer (enumerated)

**See Also**

Properties:

[TrackWidth](#)

## TrackNumber Property

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#)

### 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](#)

## TrackWidth Property

[See Also](#)

[Example](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Determines the width of the track.

### Usage

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

### Remarks

The value of this property determines the width of the track. This property is measured in pixels.

### Data Type

Integer

**See Also**

Properties:

[TrackBevel](#)



## Value Property

[See Also](#)

### Applies To

[Horizontal Indicator](#), [Horizontal Slider](#), [Knob](#), [Vertical Indicator](#), [Vertical Slider](#)

### Description

Specifies the current position of the control.

### Usage

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

### Remarks

This property determines the current value of the control. This is the default property of these controls.

### Data Type

Integer

## See Also

Events:

[Change](#)

[Scroll](#)

Properties:

[LinkControl](#)

[LinkProperty](#)

[Max](#)

[Min](#)

## Voices Property

### Applies To

MIDI output

### Description

Number of voices supported by selected device.

### Usage

[*form.*][*control.*]**Voices**

### Remarks

Number of voices supported by internal MIDI (DeviceID). Always zero for MIDI ports.

This property is read-only.

### Data Type

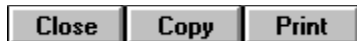
Integer



## Action Property Example, MIDI File Control

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

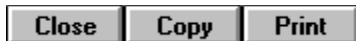
```
Sub MidiFileFun ()
    '
    ' Delete the current track
    '
    MIDIFile1.Action = MIDIFILE_DELETE_TRACK
    '
    ' Create a new track
    '
    MIDIFile1.Action = MIDIFILE_INSERT_TRACK
    '
    ' Add a note-on message (Ch. 3, C3, forte, time 0) to the new track
    '
    MIDIFile1.Message = &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
```



## Action Property Example, MIDI Input Control

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.

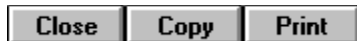
```
Sub MIDIInput1_Message()  
    Dim Message As Integer  
    Dim Data1 As Integer  
    Dim Data2 As Integer  
  
    Do While (MIDIInput1.MessageCount > 0 )  
        '  
        'This is the incoming MIDI data  
        '  
        Message = MIDIInput1.Message  
        Data1 = MIDIInput1.Data1  
        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
```



## Action Property Example, MIDI Output Control

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
```



## Bevel Properties Example

In this example, the program shows what happens when you vary the bevels on the controls. To try this example, paste the code into the Declarations section of a form that contains a knob, a horizontal indicator, and a horizontal slider control. Press F5. Play with the knob.

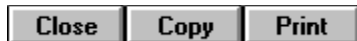
```
Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Knob1.Width = 3000
    Knob1.Height = 2000
    Knob1.Radius = 500
    Knob1.TickCount = 4
    Knob1.Min = 0
    Knob1.Max = 3
    Knob1.Value = 0
    Knob1.FontSize = 7
    Knob1.FontBold = False
    Knob1.FontName = "Arial"
    Knob1.FontSize = 7
    Knob1.TickCaption(0) = "None"
    Knob1.TickCaption(1) = "Raised"
    Knob1.TickCaption(2) = "Inset"
    Knob1.TickCaption(3) = "Lowered"

    HIndicator1.BackColor = &HC0C0C0

    HSlider1.TrackBevel = 0
    HSlider1.TrackWidth = 5
    HSlider1.BorderWidth = 4
End Sub

Sub Knob1_Scroll ()
    HSlider1.BevelInner = Knob1.Value
    HSlider1.BevelOuter = Knob1.Value
    HIndicator1.BevelInner = Knob1.Value
    HIndicator1.BevelOuter = Knob1.Value
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.





## **CanCache Property Example**



## **Channels Property Example**

Close Copy Print

## Clocks Property Example

Close

Copy

Print

## Data1 and Data2 Properties 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

## DeviceCount Property Example

This example shows how to load combo-boxes with lists of input devices and output devices.

```
Sub Form_Load ()
    Dim i As Integer

    '
    ' Fill output device combo box
    '
    For i = -1 To MIDIOutput1.DeviceCount - 1
        MIDIOutput1.DeviceID = i
        OutputDevCombo.AddItem MIDIOutput1.ProductName
    Next
    '
    ' Select first in list
    '
    MIDIOutput1.DeviceID = -1
    OutputDevCombo.ListIndex = 0
    '
    ' Fill input device combo box
    '
    For i = 0 To MIDIInput1.DeviceCount - 1
        MIDIInput1.DeviceID = i
        InputDevCombo.AddItem MIDIInput1.ProductName
    Next
    '
    ' Select first in list
    '
    MIDIInput1.DeviceID = -1
    InputDevCombo.ListIndex = 0
End Sub
```

Close

Copy

Print

## DeviceID Property Example

This example shows how to load combo-boxes with lists of input devices and output devices.

```
Sub Form_Load ()
  Dim i As Integer

  '
  ' Fill output device combo box
  '
  For i = -1 To MIDIOutput1.DeviceCount - 1
    MIDIOutput1.DeviceID = i
    OutputDevCombo.AddItem MIDIOutput1.ProductName
  Next
  '
  ' Select first in list
  '
  MIDIOutput1.DeviceID = -1
  OutputDevCombo.ListIndex = 0
  '
  ' Fill input device combo box
  '
  For i = 0 To MIDIInput1.DeviceCount - 1
    MIDIInput1.DeviceID = i
    InputDevCombo.AddItem MIDIInput1.ProductName
  Next
  '
  ' Select first in list
  '
  MIDIInput1.DeviceID = -1
  InputDevCombo.ListIndex = 0
End Sub
```



## **DeviceType Property Example**



## **DriverVersion Property Example**





## 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
```



## **Format Property Example**



## **Frame Property Example**

Close Copy Print

## FrameRate Property Example



## Gap Property Example

In this example, the program shows what happens when you vary the gap. To try this example, paste the code into the Declarations section of a form that contains a horizontal scroll bar, a label, and a horizontal slider control. Press F5. Play with the horizontal scroll bar.

```
Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Label1.BackColor = &HC0C0C0
    Label1.Top = 240
    Label1.Left = 2840
    Label1.Height = 255

    HSlider1.Height = 1000
    HSlider1.Width = 2000

    HScroll1.Top = 240
    HScroll1.Left = 720
    HScroll1.Width = 2000
    HScroll1.Min = 0
    HScroll1.Max = 20
    HScroll1.Value = 2

    HSlider1.BevelOuter = 1
    HSlider1.BevelInner = 3
    HSlider1.TickMarks = 3
    HSlider1.TickCount = 11
    HSlider1.Height = 1000
    HSlider1.Width = 2000
    HSlider1.ThumbHeight = 360
    HSlider1.ThumbWidth = 120
    HSlider1.Gap = HScroll1.Value
    HSlider1.Value = 50
End Sub

Sub HScroll1_Change ()
    Call HScroll1_Scroll
End Sub

Sub HScroll1_Scroll ()
    HSlider1.Gap = HScroll1.Value
    Label1.Caption = "Gap: " & HScroll1.Value
End Sub
```



## HasLRVolume Property Example

```
Sub CloseOutputDevice ()
    '
    ' Restore volume before closing
    '
    If MIDIOutput1.State >= MIDISTATE_OPEN Then
        If (MIDIOutput1.HasLRVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
            MIDIOutput1.VolumeRight = rVolume
        ElseIf (MIDIOutput1.HasVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
        End If
        '
        ' Close
        '
        MIDIOutput1.Action = MIDIOUT_CLOSE
    End If
End Sub
```



## HasVolume Property Example

```
Sub CloseOutputDevice ()
    '
    ' Restore volume before closing
    '
    If MIDIOutput1.State >= MIDISTATE_OPEN Then
        If (MIDIOutput1.HasLRVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
            MIDIOutput1.VolumeRight = rVolume
        ElseIf (MIDIOutput1.HasVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
        End If
        '
        ' Close
        '
        MIDIOutput1.Action = MIDIOUT_CLOSE
    End If
End Sub
```



## **HMidiDevice Property Example**





## **Hour Property Example**



## Indicator Properties Example

In this example, the program shows what happens when you change the look of the knob's indicator. To try this example, paste the code into the Declarations section of a form that contains a horizontal scroll bar, a label, two command buttons, a common dialog control, and a knob. Press F5. Play with the scroll bar and the command buttons.

```
Sub Command1_Click ()
    Knob1.Indicator = 1 - Knob1.Indicator
End Sub

Sub Command2_Click ()
    CMDialog1.Color = Knob1.IndicatorColor
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    Knob1.IndicatorColor = CMDialog1.Color
End Sub

Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Command1.Caption = "Change Style"
    Command1.Top = 720
    Command1.Left = 240
    Command1.Width = 1800
    Command1.Height = 360

    Command2.Caption = "Change Color"
    Command2.Top = 1200
    Command2.Left = 240
    Command2.Width = 1800
    Command2.Height = 360

    Label1.BackColor = &HC0C0C0
    Label1.Top = 240
    Label1.Left = 2160
    Label1.Height = 255
    Label1.Width = 4000

    HScroll1.Top = 240
    HScroll1.Left = 240
    HScroll1.Width = 1800
    HScroll1.Min = 0
    HScroll1.Max = 20
    HScroll1.Value = 2

    Knob1.Top = 1680
    Knob1.Left = 240
    Knob1.Width = 1800
    Knob1.Height = 1800
    Knob1.Radius = 600
End Sub
```

```
Sub HScroll11_Change ()  
    Call HScroll11_Scroll  
End Sub
```

```
Sub HScroll11_Scroll ()  
    Knob1.IndicatorWidth = HScroll11.Value  
    Label1.Caption = "IndicatorWidth: " & HScroll11.Value  
End Sub
```



## ItemBackColor Property Example

In this example, the program shows what happens when you vary the gap. To try this example, paste the code into the Declarations section of a form that contains a horizontal scroll bar, a label, and a horizontal slider control. Press F5. Play with the horizontal scroll bar.

```
Sub Command1_Click ()
    CMDialog1.Color = HIndicator1.ItemBackColor
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    HIndicator1.ItemBackColor = CMDialog1.Color
End Sub

Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Command1.Top = 240
    Command1.Left = 240
    Command1.Width = 1800
    Command1.Height = 360
    Command1.Caption = "Change Color"

    HIndicator1.Top = 720
    HIndicator1.Left = 240
    HIndicator1.Width = 3600
    HIndicator1.Height = 900
    HIndicator1.BevelInner = 3
    HIndicator1.BevelOuter = 1
    HIndicator1.BackColor = &HC0C0C0

    HIndicator1.ItemBackColor = &HC0C0C0
End Sub
```



## ItemForeColor and ItemCount Properties Example

In this example, the program shows what happens when you vary the color and count of the items in an indicator. To try this example, paste the code into the Declarations section of a form that contains three horizontal scroll bars, three labels, three command buttons, a common dialog box control, and a horizontal indicator control. Press F5. Play with the command buttons and the scroll bars.

```
Sub Command1_Click ()
    CMDialog1.Color = HIndicator1.ItemForeColor1
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    HIndicator1.ItemForeColor1 = CMDialog1.Color
End Sub

Sub Command2_Click ()
    CMDialog1.Color = HIndicator1.ItemForeColor2
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    HIndicator1.ItemForeColor2 = CMDialog1.Color
End Sub

Sub Command3_Click ()
    CMDialog1.Color = HIndicator1.ItemForeColor3
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    HIndicator1.ItemForeColor3 = CMDialog1.Color
End Sub

Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    HIndicator1.Top = 1680
    HIndicator1.Left = 240
    HIndicator1.Width = 6000
    HIndicator1.Height = 600
    HIndicator1.BevelInner = 3
    HIndicator1.BevelOuter = 1
    HIndicator1.Value = 100
    HIndicator1.BackColor = &HC0C0C0
    HIndicator1.ItemBackColor = &HC0C0C0

    Command1.Top = 240
    Command1.Left = 240
    Command1.Width = 1800
    Command1.Height = 360
    Command1.Caption = "Change Color 1"

    HScroll11.Top = 240
    HScroll11.Left = 2160
    HScroll11.Width = 900
```

```

HScroll11.Min = 0
HScroll11.Max = 20
HScroll11.Value = HIndicator1.ItemCount1

Label1.Top = 240
Label1.Left = 3180
Label1.Width = 2000
Label1.BackColor = &HC0C0C0

Command2.Top = 720
Command2.Left = 240
Command2.Width = 1800
Command2.Height = 360
Command2.Caption = "Change Color 2"

HScroll12.Top = 720
HScroll12.Left = 2160
HScroll12.Width = 900
HScroll12.Min = 0
HScroll12.Max = 20
HScroll12.Value = HIndicator1.ItemCount2

Label2.Top = 720
Label2.Left = 3180
Label2.Width = 2000
Label2.BackColor = &HC0C0C0

Command3.Top = 1200
Command3.Left = 240
Command3.Width = 1800
Command3.Height = 360
Command3.Caption = "Change Color 3"

HScroll13.Top = 1200
HScroll13.Left = 2160
HScroll13.Width = 900
HScroll13.Min = 0
HScroll13.Max = 20
HScroll13.Value = HIndicator1.ItemCount3

Label3.Top = 1200
Label3.Left = 3180
Label3.Width = 2000
Label3.BackColor = &HC0C0C0
End Sub

Sub HScroll11_Change ()
HIndicator1.ItemCount1 = HScroll11.Value
Label1.Caption = "ItemCount1: " & HScroll11.Value
End Sub

Sub HScroll11_Scroll ()
Call HScroll11_Change
End Sub

Sub HScroll12_Change ()
HIndicator1.ItemCount2 = HScroll12.Value

```

```
Label2.Caption = "ItemCount2: " & HScroll2.Value  
End Sub
```

```
Sub HScroll2_Scroll ()  
Call HScroll2_Change  
End Sub
```

```
Sub HScroll3_Change ()  
HIndicator1.ItemCount3 = HScroll3.Value  
Label3.Caption = "ItemCount3: " & HScroll3.Value  
End Sub
```

```
Sub HScroll3_Scroll ()  
Call HScroll3_Change  
End Sub
```



## Knob Style Properties Example

```
Sub Command1_Click ()
    Knob1.KnobStyle = (Knob1.KnobStyle + 1) Mod 4
End Sub

Sub Command2_Click ()
    CMDialog1.Color = Knob1.KnobColor
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    Knob1.KnobColor = CMDialog1.Color
End Sub

Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Command1.Caption = "Change Style"
    Command1.Top = 720
    Command1.Left = 240
    Command1.Width = 1800
    Command1.Height = 360

    Command2.Caption = "Change Color"
    Command2.Top = 1200
    Command2.Left = 240
    Command2.Width = 1800
    Command2.Height = 360

    Knob1.Top = 1680
    Knob1.Left = 240
    Knob1.Width = 1800
    Knob1.Height = 1800
    Knob1.Radius = 600
End Sub
```





## VolumeLeft Property Example

```
Sub CloseOutputDevice ()
    '
    ' Restore volume before closing
    '
    If MIDIOutput1.State >= MIDISTATE_OPEN Then
        If (MIDIOutput1.HasLRVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
            MIDIOutput1.VolumeRight = rVolume
        ElseIf (MIDIOutput1.HasVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
        End If
        '
        ' Close
        '
        MIDIOutput1.Action = MIDIOUT_CLOSE
    End If
End Sub
```



## **ManufacturerID Property Example**



## 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
```



## 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
```



## **MessageEventEnable Property Example**



## MessageNumber Property Example

The following searches through 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
```



## MessageTag Property Example

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



## **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
```



## **Notated32nds Property Example**



## **Notes Property Example**



## 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
```



## **Numerator Property Example**



## **ProductID Property Example**



## ProductName Property Example

This example shows how to load combo-boxes with lists of input devices and output devices.

```
Sub Form_Load ()
    Dim i As Integer

    '
    ' Fill output device combo box
    '
    For i = -1 To MIDIOutput1.DeviceCount - 1
        MIDIOutput1.DeviceID = i
        OutputDevCombo.AddItem MIDIOutput1.ProductName
    Next
    '
    ' Select first in list
    '
    MIDIOutput1.DeviceID = -1
    OutputDevCombo.ListIndex = 0
    '
    ' Fill input device combo box
    '
    For i = 0 To MIDIInput1.DeviceCount - 1
        MIDIInput1.DeviceID = i
        InputDevCombo.AddItem MIDIInput1.ProductName
    Next
    '
    ' Select first in list
    '
    MIDIInput1.DeviceID = -1
    InputDevCombo.ListIndex = 0
End Sub
```



## Radius Property Example

In this example, the program shows what happens when you vary the radius of a knob. To try this example, paste the code into the Declarations section of a form that contains a knob, a horizontal scroll bar, and a label control. Press F5. Play with the scroll bar.

```
Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    HScroll1.Min = 100
    HScroll1.Max = 950
    HScroll1.Value = 200

    Knob1.Width = 2000
    Knob1.Height = 2000
    Knob1.Radius = HScroll1.Value

    Label1.Caption = Knob1.Radius
    Label1.BackColor = &HC0C0C0
End Sub

Sub HScroll1_Scroll ()
    Knob1.Radius = HScroll1.Value
    Label1.Caption = HScroll1.Value
End Sub
```





## VolumeRight Property Example

```
Sub CloseOutputDevice ()
    '
    ' Restore volume before closing
    '
    If MIDIOutput1.State >= MIDISTATE_OPEN Then
        If (MIDIOutput1.HasLRVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
            MIDIOutput1.VolumeRight = rVolume
        ElseIf (MIDIOutput1.HasVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
        End If
        '
        ' Close
        '
        MIDIOutput1.Action = MIDIOUT_CLOSE
    End If
End Sub
```



## **Sequence Property Example**



## Sequence Property Example



## **Sf Property Example**



## State Property Example

This example checks the MIDIOutput State property to see if the output device is open before trying to close it.

```
Sub CloseOutputDevice ()
    '
    ' Restore volume before closing
    '
    If MIDIOutput1.State >= MIDISTATE_OPEN Then
        If (MIDIOutput1.HasLRVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
            MIDIOutput1.VolumeRight = rVolume
        ElseIf (MIDIOutput1.HasVolume) Then
            MIDIOutput1.VolumeLeft = lVolume
        End If
        '
        ' Close
        '
        MIDIOutput1.Action = MIDIOUT_CLOSE
    End If
End Sub
```



## 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
```



## ThumbHeight, ThumbStyle, and ThumbWidth Properties Example

In this example, the program shows what happens when you vary the size of the thumb. To try this example, paste the code into the Declarations section of a form that contains a horizontal slider, a horizontal scroll bar, a vertical scroll bar, a knob, and two label controls. Press F5. Play with the scroll bars and the knob.

```
Sub Form_Load ()
    Form1.BackColor = &HC0C0C0
    Form1.Height = 4880
    Form1.Width = 4000

    Knob1.Left = 204
    Knob1.Top = 2400
    Knob1.Width = 3400
    Knob1.Height = 2000
    Knob1.Radius = 500
    Knob1.Min = 0
    Knob1.Max = 3
    Knob1.TickCount = 4
    Knob1.TickCaption(0) = "Normal"
    Knob1.TickCaption(1) = "Pointed Up"
    Knob1.TickCaption(2) = "Pointed Down"
    Knob1.TickCaption(3) = "Lined"

    Label1.BackColor = &HC0C0C0
    Label1.Top = 240
    Label1.Left = 2840
    Label1.Height = 255

    Label2.BackColor = &HC0C0C0
    Label2.Top = 1840
    Label2.Left = 240
    Label2.Height = 255

    HSlider1.Height = 1000
    HSlider1.Width = 2000

    HScroll1.Top = 240
    HScroll1.Left = 720
    HScroll1.Width = 2000
    HScroll1.Min = 90
    HScroll1.Max = 500
    HScroll1.Value = 120

    VScroll1.Top = 720
    VScroll1.Left = 240
    VScroll1.Height = 1000
    VScroll1.Min = 90
    VScroll1.Max = 500
    VScroll1.Value = 240

    HSlider1.Height = 1000
    HSlider1.Width = 2000
```

```
        HSlider1.ThumbHeight = VScroll1.Value
        HSlider1.ThumbWidth = HScroll1.Value
        HSlider1.Value = 50
End Sub

Sub HScroll1_Change ()
    Call HScroll1_Scroll
End Sub

Sub HScroll1_Scroll ()
    HSlider1.ThumbWidth = HScroll1.Value
    Label1.Caption = HScroll1.Value
End Sub

Sub Knob1_Change ()
    Call Knob1_Scroll
End Sub

Sub Knob1_Scroll ()
    HSlider1.ThumbStyle = Knob1.Value
End Sub

Sub VScroll1_Change ()
    Call VScroll1_Scroll
End Sub

Sub VScroll1_Scroll ()
    HSlider1.ThumbHeight = VScroll1.Value
    Label2.Caption = VScroll1.Value
End Sub
```





## Tick Properties Example

In this example, the program shows what happens when you change the look of the tick marks. To try this example, paste the code into the Declarations section of a form that contains a horizontal slider, a knob, two command buttons, four horizontal scroll bars, four labels, and a common dialog control. Press F5. Play with the scroll bars and the command buttons.

```
Sub Command1_Click ()
    CMDialog1.Color = HSlider1.TickColor
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    HSlider1.TickColor = CMDialog1.Color
    Knob1.TickColor = CMDialog1.Color
End Sub

Sub Command2_Click ()
    CMDialog1.Color = Knob1.TickCaptionColor
    CMDialog1.Flags = 1
    CMDialog1.Action = 3

    Knob1.TickCaptionColor = CMDialog1.Color
End Sub

Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    HSlider1.Top = 1680
    HSlider1.Left = 240
    HSlider1.Width = 6000
    HSlider1.Height = 600
    HSlider1.Value = 100
    HSlider1.BackColor = &HC0C0C0

    Knob1.Top = 2400
    Knob1.Left = 240
    Knob1.Width = 1800
    Knob1.Height = 1800
    Knob1.Radius = 400
    Knob1.TickCount = 5

    Command1.Top = 240
    Command1.Left = 240
    Command1.Width = 1800
    Command1.Height = 360
    Command1.Caption = "Change TickColor"

    Command2.Top = 720
    Command2.Left = 240
    Command2.Width = 1800
    Command2.Height = 360
    Command2.Caption = "Change TickCaptionColor"
```

```

HScroll11.Top = 240
HScroll11.Left = 2160
HScroll11.Width = 900
HScroll11.Min = 0
HScroll11.Max = 20
HScroll11.Value = Knob1.TickCount

Label1.Top = 240
Label1.Left = 3180
Label1.Width = 2000
Label1.BackColor = &HC0C0C0

HScroll12.Top = 600
HScroll12.Left = 2160
HScroll12.Width = 900
HScroll12.Min = 0
HScroll12.Max = 20
HScroll12.Value = Knob1.TickGap

Label2.Top = 600
Label2.Left = 3180
Label2.Width = 2000
Label2.BackColor = &HC0C0C0

HScroll13.Top = 960
HScroll13.Left = 2160
HScroll13.Width = 900
HScroll13.Min = 0
HScroll13.Max = 20
HScroll13.Value = Knob1.TickLength

Label3.Top = 960
Label3.Left = 3180
Label3.Width = 2000
Label3.BackColor = &HC0C0C0

HScroll14.Top = 1320
HScroll14.Left = 2160
HScroll14.Width = 900
HScroll14.Min = 0
HScroll14.Max = 20
HScroll14.Value = Knob1.TickWidth

Label4.Top = 1320
Label4.Left = 3180
Label4.Width = 2000
Label4.BackColor = &HC0C0C0
End Sub

Sub HScroll11_Change ()
Dim I As Integer

HSlider1.TickCount = HScroll11.Value
Knob1.TickCount = HScroll11.Value
Label1.Caption = "TickCount: " & HScroll11.Value

For I = 0 To HScroll11.Value - 1

```

```
        Knob1.TickCaption(I) = Chr$(I + 65)
    Next I
End Sub

Sub HScroll11_Scroll ()
    Call HScroll11_Change
End Sub

Sub HScroll12_Change ()
    HSlider1.Gap = HScroll12.Value
    Knob1.TickGap = HScroll12.Value
    Label2.Caption = "TickGap: " & HScroll12.Value
End Sub

Sub HScroll12_Scroll ()
    Call HScroll12_Change
End Sub

Sub HScroll13_Change ()
    HSlider1.TickLength = HScroll13.Value
    Knob1.TickLength = HScroll13.Value
    Label3.Caption = "TickLength: " & HScroll13.Value
End Sub

Sub HScroll13_Scroll ()
    Call HScroll13_Change
End Sub

Sub HScroll14_Change ()
    HSlider1.TickWidth = HScroll14.Value
    Knob1.TickWidth = HScroll14.Value
    Label4.Caption = "TickWidth: " & HScroll14.Value
End Sub

Sub HScroll14_Scroll ()
    Call HScroll14_Change
End Sub
```



## **TicksPerFrame Property Example**



## 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
```



## **TimeFormat Property Example**



## TrackBevel Property Example

In this example, the program shows what happens when you vary the track bevel. To try this example, paste the code into the Declarations section of a form that contains a knob, and a horizontal slider control. Press F5. Play with the knob.

```
Sub Form_Load ()
    Form1.BackColor = &HC0C0C0

    Knob1.Width = 3000
    Knob1.Height = 2000
    Knob1.Radius = 500
    Knob1.TickCount = 4
    Knob1.Min = 0
    Knob1.Max = 3
    Knob1.Value = 0
    Knob1.FontSize = 7
    Knob1.FontBold = False
    Knob1.FontName = "Arial"
    Knob1.FontSize = 7
    Knob1.TickCaption(0) = "Normal"
    Knob1.TickCaption(1) = "Raised"
    Knob1.TickCaption(2) = "Inset"
    Knob1.TickCaption(3) = "Lowered"

    HSlider1.TrackBevel = 0
    HSlider1.TrackWidth = 5
End Sub

Sub Knob1_Scroll ()
    HSlider1.TrackBevel = Knob1.Value
End Sub
```





## 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
```



## TrackWidth Property Example

In this example, the program shows what happens when you vary the track width. To try this example, paste the code into the Declarations section of a form that contains a label, a vertical scroll bar, and a horizontal slider control. Press F5. Play with the scroll bar.

```
Sub Form_Load ()
    Label1.Caption = "0"

    HSlider1.TrackBevel = 3

    VScroll1.Min = 0
    VScroll1.Max = 20
End Sub

Sub VScroll1_Scroll ()
    Label1.Caption = VScroll1.Value
    HSlider1.TrackWidth = VScroll1.Value
End Sub
```



## **Voices Property Example**

## Change Event

[See Also](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Occurs when the value has changed.

### Syntax

**Sub** *ctlname*\_Change ( )

### Remarks

This event occurs when the value of the control has changed (usually through user interaction). When this event occurs, the control also updates the control specified by the link properties.

## See Also

Events:

[Scroll](#)

Properties:

[LinkControl](#)

[LinkProperty](#)

[Value](#)

## Error Event

[See Also](#)

[Example](#)

### Applies To

[MIDI file](#), [MIDI input](#), [MIDI output](#)

### 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 \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## Message Event

[See Also](#)

[Example](#)

### Applies To

[MIDI input](#)

### Description

Fires when a message is received.

### Syntax

**Sub** *ctlname*\_Message ( )

### Remarks

This event is fired whenever MIDI messages are available and [MessageEventEnable](#) is set to True.



**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## MessageSent Event

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

Fires when a message is sent.

### Syntax

**Sub** *ctlname*\_**MessageSent** ( *MessageTag* **As Long** )

### Remarks

This event is fired what a tagged message has been sent to the MIDI channel. *MessageTag* identifies the message sent.

**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## QueueEmpty Event

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

Fires when the output queue becomes empty.

### Syntax

**Sub** *ctlname*\_QueueEmpty ( )

### Remarks

Fires when the output queue becomes empty.

**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)

## Scroll Event

[See Also](#)

### Applies To

[Horizontal Slider](#), [Knob](#), [Vertical Slider](#)

### Description

Occurs while a user changes the value.

### Syntax

**Sub** *ctlname*\_**Scroll** ( )

### Remarks

You can use this event to perform calculations or to manipulate controls that must be coordinated with changes in these controls. Use the [Change](#) event when you want an update to occur after the change is complete.

**See Also**

Events:  
[Change](#)

Properties:  
[Value](#)

## Timer Event

[See Also](#)

[Example](#)

### Applies To

[MIDI output](#)

### Description

Fires when a timer expires.

### Syntax

**Sub** *ctlname\_Timer* ( )

### Remarks

Fires when a timer expires.



**See Also**

Properties:

[Action \(MIDI File\)](#)

[Action \(MIDI Input\)](#)

[Action \(MIDI Output\)](#)



## **Error Event Example**

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



## **Error Event Example**

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



## Message Event Example

```
Sub MIDIInput1_Message ()
    Dim InMessage As Integer
    Dim InData1 As Integer
    Dim InData2 As Integer
    Dim Y As Integer

    If (fGotFirst = False) Then
        PreviousTime = MIDIInput1.Time
        fGotFirst = True
        fRecording = True
    End If
    '
    'This do while loop allows you to take all the messages that are
    'waiting in the message queue.
    '
    Do While MIDIInput1.MessageCount > 0
        '
        'This is the incoming MIDI data
        '
        InMessage = MIDIInput1.Message
        InData1 = MIDIInput1.Data1
        InData2 = MIDIInput1.Data2
        '
        ' Copy input to output?
        '
        If (MidiThruCheck.Value) Then
            '
            'Tell MIDIOutput1 to send the MIDI data
            '
            MIDIOutput1.Message = InMessage
            MIDIOutput1.Data1 = InData1
            MIDIOutput1.Data2 = InData2
            MIDIOutput1.Action = MIDIOUT_SEND
        End If

        If (InsertRecordingCheck.Value) Then
            '
            ' Copy message parameters
            '
            MIDIFile1.Message = MIDIOutput1.Message
            MIDIFile1.Data1 = MIDIOutput1.Data1
            MIDIFile1.Data2 = MIDIOutput1.Data2
            '
            ' Calculate time in ticks
            '
            CurrentTime = MIDIInput1.Time
            MIDIFile1.Time = Int(CurrentTime - PreviousTime) * ticksPerMs
            PreviousTime = CurrentTime
            '
            ' insert message into MIDI file
            '
            MIDIFile1.Action = MIDIFILE_INSERT_MESSAGE
        End If
    End Do
End Sub
```

```
End If
'
'Remove the MIDI data from the MIDI IN queue
'
MIDIInput1.Action = MIDIIN_REMOVE
Loop
End Sub
```



## MessageSent Event Example

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



## Timer Event Example

```
Sub MIDIOutput1_Timer (TimerTag As Long)
  If (TimerTag = 1) Then
    Shape1.Visible = True
  Else
    Shape1.Visible = False
  End If
End Sub
```

