

TegoSoft Inc.

Copyright © 1994-1996 TegoSoft Inc. ® All Rights Reserved

P.O.Box 389, Bellmore, NY 11710

Web Site: <http://www.tegosoft.com>

 e-mail: tegosoft@msn.com


Lesson 8 - Timer


During the course of developing Windows programs, you may need to write programs that utilize **Timers**. In this lesson you'll learn how to write programs that utilize timers.


Create the Directory of the MyTimer Program, Start Visual Basic 4, and Save the Form and the Project Files

In this lesson you are going to create several files. So first of all, let's create a directory where the files that you'll create during the course of this lesson are saved.


 Create the **C:\VBMyProg\Lesson08** directory.

 Start Visual Basic, and then select **New Project** from the **File** menu of Visual Basic.

 Select **Project** from the **View** menu of Visual Basic to display the Project window. Click the **View Form** button of the Project window to make **Form1** the selected form.


 While Form1 is the selected window, select **Save File As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save File As** dialog box.

 Save the Form file as **MyTimer.frm** inside the **C:\VBMyProg\Lesson08** directory.


 Select **Save Project As** from the **File** menu of Visual Basic.

Visual Basic responds by displaying the **Save Project As** dialog box.

 Save the project file as **MyTimer.vbp** inside the **C:\VBMyProg\Lesson08** directory.

Set the Properties of Form1

You'll now set the properties of **Form1**.

 Make the **Form1** window the selected window, and then press the **F4** key on your keyboard to display the **Properties** window of Form1.


 Set the properties of **Form1** as follows:


Property	Setting
Name	frmMyTimer
Caption	The MyTimer Program
Height	3570
Width	3885
BackColor	White

 Select **Save Project** from the **File** menu of Visual Basic to save your work.


Implementing the Exit button

You'll now place the **Exit** button inside the **frmMyTimer** form.

 Make sure that the frmMyTimer form is the selected window, and then double-click the CommandButton icon inside the Toolbox window.

 Make the CommandButton that you placed inside the form is the selected object, press F4 on your keyboard to display the Properties window, and then set the properties of the CommandButton as follows:

Property	Setting
Name	cmdExit
Caption	E&xit
Height	495
Width	1215
Top	2520
Left	1200

 Double click the **cmdExit** button to display the **Code** window, set the **Object** list box of the Code window to **cmdExit**, set the **Proc** list box of the Code window to **Click**, and then type the following code inside the **cmdExit_Click()** procedure:

```
Private Sub cmdExit_Click()
```

```
End
```

```
End Sub
```

 Select **Save project** from the **File** menu of Visual Basic to save your work.

The code that you typed is executed automatically whenever the user clicks the **Exit** button. This code causes the program to terminate itself.

=====

Placing the Timer Control

You'll now place the **Timer** control inside the frmMyTimer form.

The Timer control inside the Toolbox is shown in Figure 8.1. In Figure 8.1 the Timer control is shown on the left column as the fifth icon from the bottom. But in your Toolbox, the Timer control may be located in a different location.



Figure 8.1. *The Timer Control inside the Toolbox.*



NOTE

When you place the mouse cursor (without clicking) on the Timer control, a yellow rectangle pops up with the text **Timer** in it.



Make sure that Form1 is the selected window, and then double click the **Timer** control inside the Toolbox window.

Visual Basic responds by placing the Timer control inside the form. Your form should now look as shown in Figure 8.2.

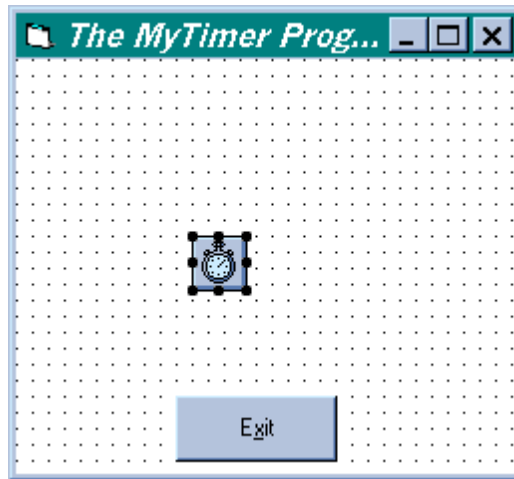



Figure 8.2. *The form after placing the Timer control in it.*

Invisible Controls

If you take a look at the Properties window of the Exit button, you'll see that one of the properties is the **Visible** property. The Visible property can be either True or False. The default value that visual Basic set for the Visible property of the Exit button is True. Hence, the Exit button is visible during runtime (during the execution of the program). Had you set the Visible property to False, you would not see the Exit button during the execution of the program.

Some controls are always invisible during runtime. These controls do not have the Visible property, because these control are not displayed during runtime. The Timer control that you placed inside the form is an example of a control that is always invisible. Lets see this in action:

 Select Start from the Run menu of Visual Basic to start the program.

Visual Basic responds by displaying the window of the MyTimer program as shown in Figure 8.3. As you can see, the Timer control does not appear in the window.

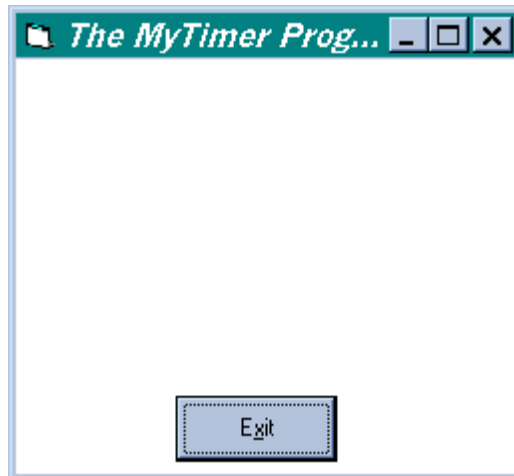



Figure 8.3. *The Timer control is invisible during runtime.*

 Click the Exit button to terminate the program.

NOTE


Because the Timer control is invisible during runtime, it does not matter where you place this control inside the form.

The Enabled Property

Almost all controls have the **Enabled** property. The Enabled property can be either **True** or **False**.

As implies by the name of the Enabled property, when the Enabled property is set to True, the control is enabled. When the Enabled property is set to False, the control is disable. Let's see the Enabled property in action:

 Set the Enabled property of the Exit button to False.

 Select Start from the Run menu of Visual Basic to start the program.

The window of the MyTimer program appears as shown in Figure 8.4. As you can see, the Exit button is disabled. You can click it all you want, but the button will not react to your clicking.

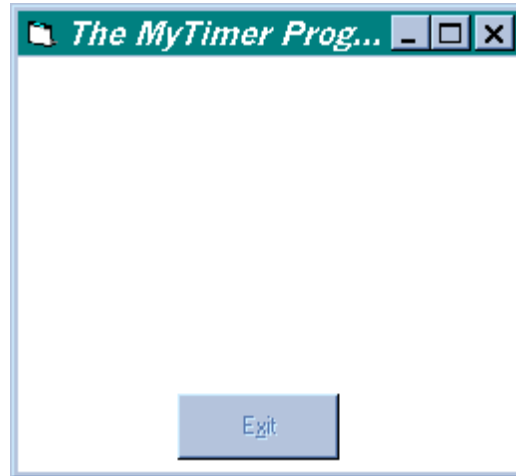





Figure 8.4. *The Exit button is disabled.*

Note that when the Exit button is disabled (see Figure 8.4), the button is grayed, thus giving the user an indication that the button is not operational.

 Select End from the Run menu of Visual Basic to terminate the program.

 Set the Enabled property of the cmdExit button back to True.

In the preceding experiment you saw the effect of making a button disabled. The Timer control also has an Enabled property. However, setting the Enabled property of the Timer control to True/False has a different effect than setting the Enabled property of a button. When you set the Enabled property of the Timer control to True, the Timer is operational. When you set the Enabled property of the Timer control to False, the Timer is not operational, which means that it is as if you never placed the Timer control inside the Form.

 Make sure that the Enabled property of the Timer control is set to True.

The Interval Property of the Timer Control

If you take a look at the Properties window of the Timer control, you see that the **Name** property of the Timer control is **Timer1**. (If you have only one timer in the program, it is ok to leave the **Name** property as **Timer1** instead of naming the control as **tmrMyTimer** for example).

Another property that you see in the Properties window of the Timer control is the **Interval** property.

 Set the **Interval** property of Timer1 to 500.

In the next section you'll learn about the effect of setting the Interval property.

Save your work:

 Select Save Project from the File menu of Visual Basic to save your work.

Attaching Code to the Timer Event

You'll now attach code to the **Timer** event.

When you click the **cmdExit** button, the **Click** event occurs, and as a result, the **cmdExit_Click()** procedure is executed automatically.


In the same manner, when the Enabled property of the Timer is set to True, the **Timer** event occurs, and as a result, the **Timer1_Timer()** procedure is executed automatically.

In the preceding section you set the **Interval** property of the Timer1 control to **500**. This means that the **Timer1_Timer()** procedure is executed automatically every **500** milliseconds (every 0.5 seconds).

=====

Had you set the Interval property to 250, the Timer1_Timer() procedure would executed automatically every 250 milliseconds (every 0.25 seconds), and so on.

Let's attach code to the Timer1_Timer() procedure so that we can prove ourselves that indeed the Timer1_Timer() procedure is executed automatically every 500 milliseconds:

 Double click the Timer1 control, set the **Object** list to **Timer1**, and set the **Proc** list to **Timer**.

Visual Basic responds by displaying the **Timer1_Timer()** procedure ready to be edited as follows:

Private Sub Timer1_Timer()


End Sub

 Type the following code inside the Timer1_Timer() procedure:

Private Sub Timer1_Timer()


Beep

End Sub


 Select Save Project from the File menu of Visual Basic to save your work.

The code that you typed causes the PC to beep. So when you'll execute the MyTimer program, the PC will execute the Timer1_Timer() procedure every 500 milliseconds, which means that you'll hear the PC beep every 500 milliseconds.

=====

 Select Start from the Run menu of Visual Basic to start the MyTimer program.

As you can hear, the PC beeps every 500 milliseconds.

 Click the Exit button to terminate the program.

What You Accomplished in This Lesson



You completed Lesson 8 of the Self Study Visual Basic tutorial.

The MyTimer program that you implemented in this lesson demonstrates how to use the Timer control.

Frequently Asked Questions



Q1. With the MyTimer program, I caused the PC to beep every 500 milliseconds. How else can I use the Timer control?

A1. The Timer control is probably one of the most used controls! Make sure that you understand how the MyTimer program works, as this program illustrates the usefulness of the Timer control.

The Timer control is used for example for performing animation. Instead of typing the Beep statement inside the Time1_Timer() procedure as you did in the MyTimer program, you can type code that displays pictures.

The first time the Timer1_Timer() procedure is executed, a certain picture is displayed (let's say picture #1 is displayed). After 500 milliseconds, another picture is displayed (let's say picture #2 is displayed). And so on.

=====

The user will therefore see the following:

Picture #1 is displayed for a period of 500 milliseconds.

Then the picture is replaced by picture #2. Picture #2 will now be displayed for a period of 500 milliseconds.

Then picture #3 will be displayed for a period of 500 milliseconds, and so on.

The process of displaying one picture after the other is called **animation**.

Exam



1. The Timer control is used for:
 - (a) Telling the time and date
 - (b) Keeping track of how long it takes you to develop your project
 - (c) Performing a certain task periodically.

2. The Visible property can be set to: _____

3. Setting the Visible property of a button control to True causes the control to be _____ during runtime.

4. The Enabled property can be set to: _____

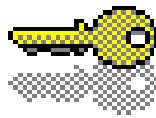
5. Setting the Enabled property of a control to True cause the control to be:

=====

- (a) Visible
- (b) Enabled

- 6. Making the Invisible property of the Timer control to True makes the Timer control _____.
- 7. The Interval property of the Timer control determines the _____

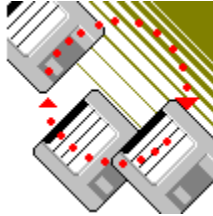
Answers to Exam




- 1. (c) Performing a certain task periodically.
- 2. The Visible property can be set to: True or False.
- 3. Setting the Visible property of a button control to True causes the control to be invisible during runtime.
- 4. The Enabled property can be set to: True or False
- 5. (b) Setting the Enabled property of a control to True cause the control to be Enabled.
- 6. (This is a trick question) The Timer control does not have a Visible property.
- 7. The Interval property of the Timer control determines the frequency at which the Timer event occurs. For example, if you set the Interval property of the Timer control to 50, the Timer1_Timer() procedure will be executed automatically every 50 milliseconds.



Project



Modify the MyTimer program so that a scroll bar determines the frequency at which the beeping occurs.

 Place a vertical scroll bar inside the form, and set the properties of the scroll bar as follows:

Name: vsbFrequency

Min: 0

Max: 3000


Value: 500

Height: 1455

Width: 255

Top: 600

Left: 1680

 Select Save Project from the File menu of Visual Basic to save your work.

You form should now look as shown in Figure 8.5.

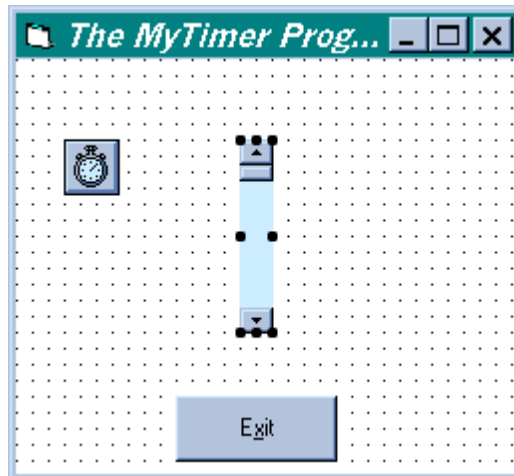



Figure 8.5. Adding the vertical scroll bar to the form.

 Type the following code inside the `vsbFrequency_Change()` procedure:

Private Sub vsbFrequency_Change()

 Timer1.Interval = vsbFrequency.Value

End Sub


 Select Save Project from the File menu of Visual Basic to save your work.

The code that you typed is executed automatically whenever the user changes the scroll bar position. This code sets the **Interval** property of the **Timer1** control to the **Value** property of the scroll bar. For example, if the scrollbar position is at the extreme top position, the **Value** property of the scrollbar is **0**, and thus the **Interval** property of the **Timer1** control is set to **0**. This means that the `Timer1_Timer()` procedure will not be executed.

When the user sets the scroll bar to the 500 point, the `Timer1_Timer()` procedure will be executed every 500 milliseconds.

=====


When the user sets the scroll bar position to the bottom point of the scroll bar (which is 3000), the Timer1_Timer() procedure will be executed every 3000 milliseconds.

 Select Start from the Run menu of Visual Basic to start the MyTimer program.

Currently, the scrollbar position is at 500 (because you set the **Value** property of the scroll bar to 500), so you hear the PC beeps every 500 milliseconds.


 Drag the square of the scroll bar to its extreme top position.

The scrollbar position is now at **Value=0**, so the **Interval** property of the Timer1 control is set to **0**. This means that the Timer1_Timer() procedure is not executed, and hence you do not hear the PC beeps.

 Drag the square of the scroll bar to its extreme bottom position.

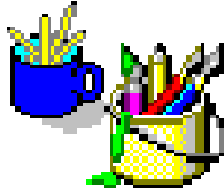
Now the **Value** property of the scroll bar is set to 3000, which means that the **Interval** property of Timer1 is set to 3000. Thus, you now hear the PC beeps every 3 seconds. (3000 milliseconds equals 3 seconds).

Note that if you set the scroll bar position to a value of approximately 10 for example, you will not hear the PC beep every 10 milliseconds, because each Beep duration is more than 10 milliseconds. You will however hear the beeping one after the other.

 Click the Exit button to terminate the program.

Cosmetic Considerations

=====





Note that you set the **Min** property of the scroll bar to **0**, and you set the **Max** property of the scroll bar to **3000**. As you know, you can change the scrollbar position by dragging the square of the scroll bar. Alternatively, you can change the scroll bar position by clicking the up arrow icon or the down arrow icon that are on the extreme points of the scroll bar. Because the **Min** property of the scroll bar is set to **0** and the **Max** property of the scroll bar is set to **3000**, it means that if you are changing the scroll bar position from the top position to the bottom position by clicking the arrow icons, you have to click 3001 times! Why? Because currently, each click on the arrow icons produces an advance of **1** unit.

Advancing in steps of 1 unit at a time is fine if you want to set the scroll bar position with great accuracy.


Currently, when you click inside the scrollbar in-between the square and the arrow icon, the scroll bar position advances in steps of **1** unit at a time.

You'll now modify the steps in which the scroll bar position is advancing when you click the arrow icons of the scroll bar:

 Set the **SmallChange** property of the vsbFrequency scroll bar to **10**.

 Set the **LargeChange** property of the vsbFrequency scroll bar to **100**.


 Select Save Project from the File menu of Visual Basic to save your work.

 Select Start from the Run menu of Visual Basic to start the MyTimer program.

You can change the scroll bar position to any desired position by dragging the square of the scroll bar.

You can change the scroll bar position by clicking the arrow icons of the scroll bar. Each click now produces an advance of **10** units (because you set the **SmallChange** property of the scroll bar to **10**). You can keep the arrow icon of the scroll bar pressed down, and this will produce a continuous changes of **10** units at a time.

Alternatively, you can change the scroll bar position by clicking in the area between the square and the arrow icon of the scroll bar. Each click produces an advance of **100** units (because you set the **LargeChange** property of the scroll bar to **100**).

 Experiment with the MyTimer program, then click the Exit button to terminate the program.

How To Contact TegoSoft



You can contact TegoSoft Inc. by any one of the following methods:

- Use TegoSoft Internet Web site:

<http://www.tegosoft.com>

- Send TegoSoft an e-mail:

tegosoft@msn.com

- Send TegoSoft a letter:

TegoSoft Inc.

P.O.Box 389

Bellmore, NY 11710


USA

Technical Support



If you have a technical question, you can post the question to the TegoSoft Technical Support staff, and they will try to answer your question.

The **best** way to post a technical question is by sending TegoSoft an e-mail.

 The e-mail of TegoSoft is:

tegosoft@msn.com

When sending TegoSoft an e-mail with a technical question, please follow the following format:

Date: _____

Your name: _____

Company (if applicable): _____

Your phone number: _____

Your e-mail: _____

Country (if not USA): _____

State (if inside USA): _____

Operating System used: _____

Programming language and version : _____

My technical question is:

Copyright © and Notices

Copyright © 1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

=====

TegoSoft Self Study Tutorials & Software

Copyright ©1994, 1995, 1996 by TegoSoft Inc. ® All Rights Reserved

Although precaution was taken in the preparation of this document, TegoSoft assumes no responsibility for errors or omissions. TegoSoft is not liable for damages resulting from the use of the information contained in this document. Use this document at your own risk. This document is copyright protected. This means that you should treat this document like any other copyright material. No part of this document should be copied in any way. You are not allowed to use this document or any part of this document for any purpose other than read it. You are not allowed to publish this document or any part of this document in any way. You are not allowed to sell this document or any part of this document, you are not allowed to incorporate this document or any part of this document in any book, magazine, Web Site, Electronic forums, disks, CDs, or any other media. The only thing that you are allowed to do with this document is read it for the sole purpose of studying the material that is presented in this document. If this document includes software, the software must be treated in the exact same way that this document is treated. You are not allowed to distribute the software in any way. The sole purpose of supplying the accompanying software is to enable you to experience with it. No part of this document should be modified. If accompanying software is included with this document, you are not allowed to modify the software.

Rev. 031796-1