

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 31-

Interrupting Processes


In this lesson you'll learn to write code that lets the user interrupt long processes. That is, sometimes your program may be involved in a very long computation process, and you may want your user to be able to interrupt the process in an honorable manner (e.g., **not** by pressing Alt+Ctrl+Delete).


Creating the Directory of the Project and Saving the Files of the Project


As usual, you'll start by creating the directory of the project, and saving the files of the project.

 Create the **C:\VBMyProg\Lesson31** directory.


You'll save the files of this lesson to this directory.

 Select **New Project** from the **File** menu of Visual Basic.

 Make the window of Form1 the selected window, select **Save File As** from the **File** menu of Visual Basic, and save the file as **StopIt.FRM** inside the **C:\VBMyProg\Lesson31** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **StopIt.VBP** inside the **C:\VBMyProg\Lesson31** directory.

Always Use Option Explicit

 Inside the general declarations section of Form1 type the following code:

```
' Force variable declarations  
Option Explicit
```

The **Option Explicit** statement forces you to declare variables before using the variables.

Designing the Window of the StopIt Program

You'll design the window of the StopIt program so that it will look as shown in Figure 31.1.

 Set the properties of Form1 as follows:

Name: Form1

Caption: The StopIt Program

BackColor: White

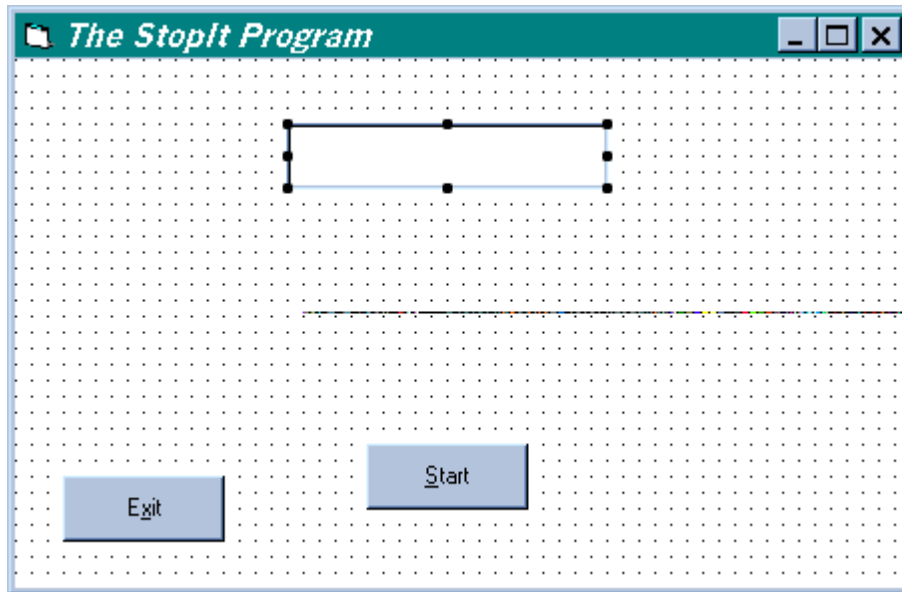



Figure 31.1. Form1 of the StopIt program

Implementing the Exit Button

You'll now implement the Exit button:

 Place a CommandButton inside Form1, then set the properties of the CommandButton as follows:

Name: cmdExit

Caption: E&xit

Attaching Code to the Click Event of the Exit Button

You'll now attach code to the **Click** event of the Exit button:

 Add the following code inside the **cmdExit_Click()** procedure:

```
Private Sub cmdExit_Click()
```


End

End Sub

So whenever the user clicks the Exit button, the StopIt program terminates itself.

Adding the Start Button

You'll now add a CommandButton to Form1. This CommandButton will serve to start a counting process.


 Place a CommandButton inside Form1. Then set the properties of this CommandButton as follows:

Name: cmdStart

Caption: &Start

Adding the Progress Label

You'll now add a Label to Form1. This label will serve to display the progress of the counting process.

 Place a Label control inside Form1. Then set the properties of this label as follows:

Name: lblProgress

Caption: Make it empty

BackStyle: 1-Fixed Single

BackColor: White

Alignment: 2-Center

Font:


Font: MS Sans Serif

Font Size: 12

Font Bold: Yes

Attaching Code to the Click Event of the Start Button

You'll now attach code to the **Click** event of the **cmdStart** button.

 Type the following code inside the **cmdStart_Click()** procedure:

```
Private Sub cmdStart_Click()  
  
Dim Counter  
  
For Counter = 1 To 1000  
    DoEvents  
    lblProgress.Caption = Str(Counter)  
    Form1.Refresh  
Next  
  
End Sub
```

The **cmdStart_Click()** procedure is executed when the user clicks the **Start** button.

A **For()** loop is executed:

```
For Counter = 1 To 1000  
    lblProgress.Caption = Str(Counter)  
    Form1.Refresh  
Next
```

The **For()** loop converts the **Counter** number to a string, and the string is assigned to the **Caption** property of the Label:

```
lblProgress.Caption = Str(Counter)
```


Then the **Refresh** method is executed:

```
Form1.Refresh
```

Putting it all together, the **For()** loop displays the values of **Counter** inside the label.

Let's see your code in action:


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

 Execute the StopIt program.

 Click the **Start** button.

The StopIt program responds by starting to count from 1 to 1000.

If you click the Exit button during the execution of the counting, the program will **not** terminate itself. Rather, the counting will continue, and at the end of the counting, the pending event of clicking the Exit button will be executed. This will cause the execution of the **cmdExit_Click()** procedure, and the StopIt program will be terminated.

 Experiment with the StopIt program. Then click the Exit button to terminate the program.

Executing Pending Event Immediately

In the previous section you saw that clicking the Exit button does **not** cause the immediate termination of the program. Rather, the **Click** event of the Exit button is treated as a pending event. When the **cmdStart_Click()** procedure is terminated, the program will process the pending **Click** event of the Exit button, and the program will terminate itself.

Is there a way to execute pending events immediately? That is, can you cause the **cmdExit_Click()** procedure to be executed as soon as the user clicks Exit button and thus to interrupt the process of the counting? The answer is yes.

 Modify the code inside the **cmdStart_Click()** procedure so that it will look as follows:

```
Private Sub cmdStart_Click()  
  
Dim Counter  
  
For Counter = 1 To 1000  
    DoEvents  
    lblProgress.Caption = Str(Counter)  
    Form1.Refresh  
Next  
  
End Sub
```

You added **DoEvents** statement:

```
DoEvents
```


DoEvents tells the program to go and look if there are any pending events. If for example the user clicks the Exit button during the execution of **For()** loop, when the **DoEvents** statement is encountered, the program will go and look if there are any pending events. Because the **Click** event of the Exit button exists as a


=====

pending event, the program will execute the **cmdExit_Click()** procedure (and this will terminate the program).


Let's see your code in action:

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


 Execute the StopIt program.

 Click the **Start** button.

The StopIt program responds by starting to count.

 While the program is counting, click the Exit button.

The StopIt program responds by immediately terminating itself. That is, the **DoEvents** statement inside the **For()** loop is executed, and because the **Click** event of the Exit button is pending, the **cmdExit_Click()** procedure is executed.

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

Disabling The Start Button

Because you execute the **DoEvents** statement from inside the **For()** loop of the **cmdStart_Click()** procedure, you have to consider what will happen if the user will click the **Start** button during the counting. If you execute the program, start the counting, and while the counting is in progress click the **Start** button again, you'll observe the following:

The **DoEvents** will cause the program to look if there are pending events. Because the **Start** button was clicked while counting is in progress, there

=====

is a pending event. This pending event causes the execution of the **cmdStart_Click()** procedure. So now the counting will start all over again.

You can see from the preceding discussion that this could cause a great confusion to your user. The solution is to disable the **Start** button during the counting:

 Modify the **cmdStart_Click()** procedure so that it will look as follows:

```
Private Sub cmdStart_Click()  
  
Dim Counter  
  
cmdStart.Enabled = False  
  
For Counter = 1 To 1000  
    DoEvents  
    lblProgress.Caption = Str(Counter)  
    Form1.Refresh  
Next  
  
cmdStart.Enabled = True  
  
End Sub
```

Before the **For()** loop starts, you disable the **Start** button:


```
cmdStart.Enabled = False
```


After the **For()** loop is terminated, you enable the **Start** button:

```
cmdStart.Enabled = True
```

Let's see your code in action:

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


 Execute the StopIt program.

 Click the **Start** button.

The StopIt program responds by starting to count. In addition, the **Start** button becomes dimmed (disabled) during the counting.


When the counting is completed, the **Start** button becomes available again.

The thing to note now is that during the counting, you cannot create a pending event for the **Click** event of the **Start** button (because during the counting, the **Start** button is dimmed).

 Click the Exit button to terminate the StopIt program.

Implementing the Stop Button

You'll now implement the **Stop** button as shown in Figure 31.2.

 Place a CommandButton inside Form1, then set the properties of the CommandButton as follows:

Name: cmdStop

Caption: Sto&p

Enabled: False

Your Form1 should now look as shown in Figure 31.2.

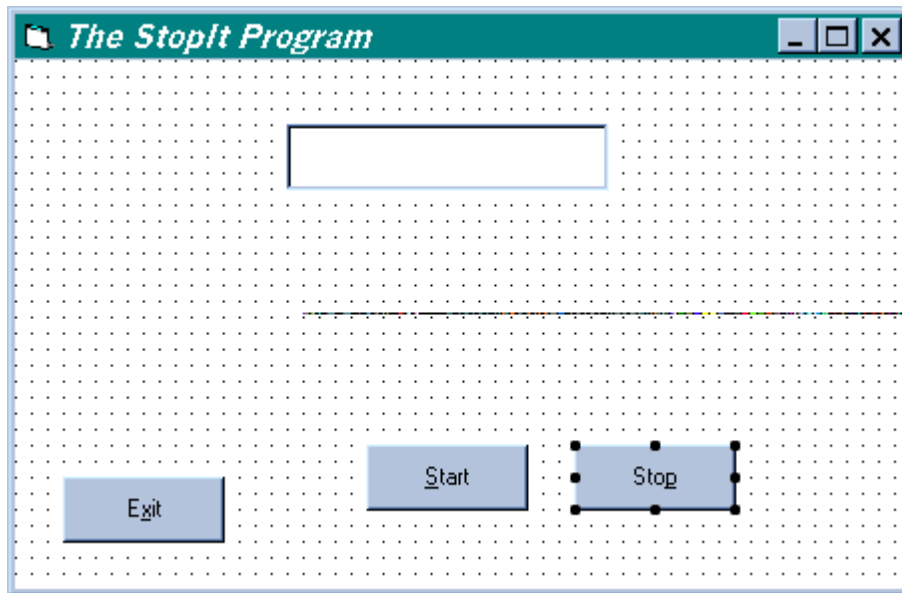



Figure 31.2. Form1 with the Stop button in it.

Declaring the gStopIt General Variable

You'll now declare a variable inside the general declarations section of Form1.

 Double click Form1 to display the Code window, set the **Object** list box of the Code window to **(General)**, and set the **Proc** list box of the Code window to **(declarations)**. You add the declaration of a variable called **gStopIt**. After adding the declaration, the general declarations section of Form1 should look as follows:


```
Option Explicit
```

```
Dim gStopIt As Boolean
```

Note that you declared the variable **gStopIt** as a **Boolean**. This means that you can set the value of **gStopIt** to either **True** or **False**. Also, because you declared this variable inside the general declarations section of Form1, this variable is accessible from any procedure of Form1.

Attaching Code to the Click Event of the cmdStop Button

You'll now attach code to the **Click** event of the **cmdStop** button.

 Type the following code inside the **cmdStop_Click()** procedure:

```
Private Sub cmdStop_Click()
```

```
gStopIt = True
```

```
cmdStop.Enabled = False
```

```
End Sub
```

You set the value of the **gStopIt** variable to **True**:

```
gStopIt = True
```

The **gStopIt** variable will serve as a flag that indicates whether the counting should be stopped. The **cmdStop_Click()** procedure is executed because the user clicked the **cmdStop** button to stop the counting, hence, you set the value of the **gStopIt** variable to **True**.

You also disabled the **cmdStop** button:

```
cmdStop.Enabled = False
```

So now the user cannot click the **Stop** button again.

The Code of the cmdStart_Click() Procedure

You'll now modify the code of the **cmdStart_Click()** procedure.

=====

 Modify the code of the **cmdStart_Click()** procedure so that it will look as follows:

```
Private Sub cmdStart_Click()  
  
Dim Counter  
  
cmdStop.Enabled = True  
  
cmdStart.Enabled = False  
  
For Counter = 1 To 1000  
    DoEvents  
    lblProgress.Caption = Str(Counter)  
    Form1.Refresh  
  
    If gStopIt = True Then  
        gStopIt = False  
        Exit For  
    End If  
Next  
  
cmdStart.Enabled = True  
  
cmdStop.Enabled = False  
  
End Sub
```

You declare the **Counter** local variable:

```
Dim Counter
```

Because the counting begun (the user clicked the **Start** button), you enable the **Stop** button (so that the user will be able to stop the counting):

```
cmdStop.Enabled = True
```

You disable the **Start** button (so that the user will not be able to click the **Start** button again during the counting):

```
cmdStart.Enabled = False
```

Then you execute the **For()** loop:

```
For Counter = 1 To 1000
    DoEvents
    lblProgress.Caption = Str(Counter)
    Form1.Refresh

    If gStopIt = True Then
        gStopIt = False
        Exit For
    End If
Next
```

The **DoEvents** statement inside the **For()** loop checks to see if there are pending events. For example, if the user clicked the **Stop** button during the counting, the **DoEvents** will cause the program to immediately execute the **cmdStop_Click()** procedure.

After the **cmdStart_Click()** procedure is executed, the program will continue with the execution of the statement that follows the **DoEvents** statement.

Inside the **For()** loop you execute the following **If** statement:

```
If gStopIt = True Then
```

```
=====
```

```
gStopIt = False
Exit For
End If
```

If the **cmdStop_Click()** procedure was executed, **gStopIt** is equal to **True**. Thus, the code under the **If** statement will be executed, and this code uses the **Exit For** statement. The **Exit For** statement immediately terminates the **For()** loop.


The last two statements that you typed are:

```
cmdStart.Enabled = True
cmdStop.Enabled = False
```

The preceding two statements make the **Start** button enabled, and the **Stop** button disable.

Putting it all together, if the user clicks the **Stop** button, the counting is terminated. Let's see this in action:


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

 Execute the StopIt program.

 Click the **Start** button to start the counting.

 While counting is in progress, click the **Stop** button.

The StopIt program responds by stopping the counting.

 Experiment with the StopIt program. Then click the Exit button to terminate the program.

=====

What You Accomplished in This Lesson



You completed Lesson 31 of the Self Study Visual Basic tutorial. In this lesson you learned to interrupt processes using **DoEvents**. The StopIt program illustrates how the **For()** loop process can be interrupted. The user can stop the counting by clicking the **Stop** button, and the user can click the Exit button during the counting process to terminate the program.

Frequently Asked Questions



Q1. In this lesson I learned to interrupt a **For()** loop with the **DoEvents** statement. Can you give me an example of a practical usage of this technique?

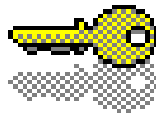
A1. In your future projects you may perform a complex process that takes a long time to perform. For example, a complex mathematical computation may take a long time, and during the process of the computation, the user may decide to abort the operation. Naturally, you want to let the user interrupt the process by clicking a button, rather than telling the user that the only way to stop the process is by turning the PC off.

Exam



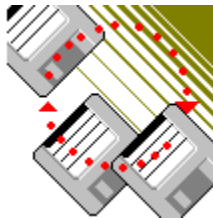
1. What does the **DoEvents** statement do?

Answers to Exam



1. The **DoEvents** statement causes the program to see if a pending event occurred, and if so, the procedure that corresponds to the event is immediately executed.

Project



Modify the code inside the **cmdStart_Click()** procedure so that it will use the **Do While** loop instead of the **For()** loop.

 Modify the **cmdStart_Click()** procedure so that it will look as follows:

```
Private Sub cmdStart_Click()
```

```
=====
```

```

Dim Counter

cmdStop.Enabled = True

cmdStart.Enabled = False

Counter = 1
Do While Counter <= 1000
    DoEvents
    lblProgress.Caption = Str(Counter)
    Form1.Refresh
    Counter = Counter + 1

    If gStopIt = True Then
        gStopIt = False
        Exit Do
    End If
Loop

cmdStart.Enabled = True

cmdStop.Enabled = False

End Sub

```


Note that under the **If** statement inside the **Do While** loop, you cause the **Do While** loop to terminate with the **Exit Do** statement:

```


If gStopIt = True Then
    gStopIt = False
    Exit Do
End If


```

=====

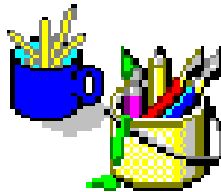
 Execute the StopIt program.

 Click the **Start** button to start the counting.

 While counting is in progress, click the **Stop** button, and verify that the counting is stopped.

 Experiment with the StopIt program and then click the Exit button to terminate the program.

Cosmetic Considerations



Always implement a mechanism that lets your user interrupt long processes (as was demonstrated in the StopIt program). One of the most annoying things that could happen to a user is when the user is forced to press Alt+Ctrl+Delete or the user is forced to turn the PC off in order to stop a process.

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