

**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](mailto:tegosoft@msn.com)

# Lesson 13 - Input Boxes for User Interface

In this lesson you'll learn to implement simple input user interface (a mechanism that lets the user input data).

## What is an Input User Interface?

Many of the programs that you'll design and implement will require user interface. That is, during the execution of the program, the user will be asked to enter data which will be used by the program. Likewise, your programs will occasionally need to display messages to the user during the execution of the programs. The process of taking data from the user and the process of displaying data to the user is called user interface.

## The Input Box


Because taking user input is so common in Windows programs, the designers of Visual Basic implemented a special way for implementing the task of taking user's input. The designers of Visual Basic incorporated a special function into the Visual Basic language that lets you implement a mechanism for taking user input very quickly. You'll now learn how to use the **InputBox()** function, a statement that lets you implement a mechanism for taking user input very quickly.


## Creating the Directory and Project Files

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


=====

 Create the **C:\VBMyProg\Lesson13** directory. You'll save the files of this lesson into this directory.

 Make sure Form1 is the selected window, and then select **Save File As** from the **File** menu of Visual Basic. Then save the form as **MyInput.FRM** inside the **C:\VBMyProg\Lesson13** directory.

 Select **Save Project As** from the **File** menu of Visual Basic, and save the project as **MyInput.VBP** inside the **C:\VBMyProg\Lesson13** directory.

## Always Use the Option Explicit Statement


 Type the **Option Explicit** statement inside the general declarations section of Form1:

**Option Explicit**

## Placing the Pick a Number Button


You'll now place a CommandButton inside Form1.

 Place a CommandButton inside Form1.

 Set the properties of the Command Button as follows:

**Name:** cmdPickANumber

**Caption:** &Pick a Number

 Arrange the size of Form1 and the location of the **Pick a Number** button so that Form1 will look as shown in Figure 13.1.

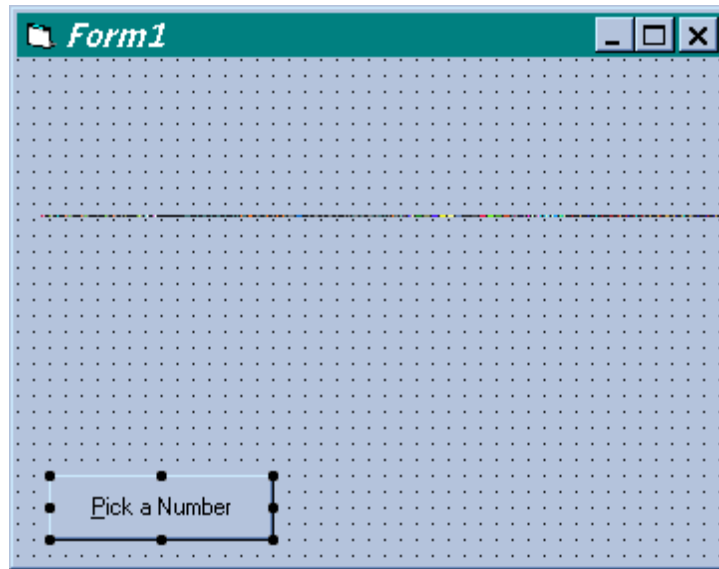


Figure 13.1. Form1 with the Pick a Number button in it.

## Attaching Code to the Click Event of the Pick a Number Button

You'll now attach code to the **Click** event of the **Pick a Number** button.

 Double click the **Pick a Number** button, and then type the following code inside the **cmdPickANumber\_Click()** procedure:

```
Private Sub cmdPickANumber_Click()  
Dim Message  
Dim Title  
Dim DefaultValue  
Dim UserValue  
  
Title = "The MyInput Program"  
Message = "Please type a number between 1 and 9"  
DefaultValue = 3  
  
UserValue = InputBox(Message, Title, DefaultValue)
```

End Sub

## Executing the MyInput Program

Before going over the code of the MyInput program, let's execute the program:

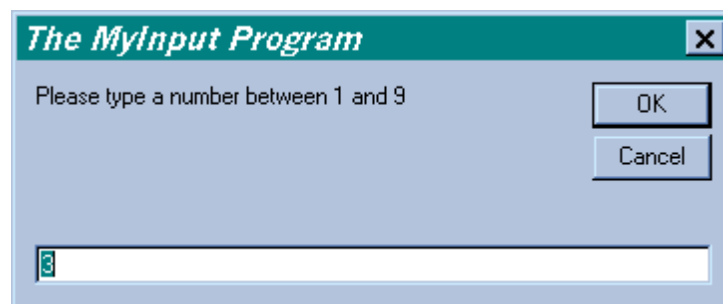
 Select **Save Project** from the **File** menu of Visual basic to save your work.

 Select **Start** from the **Run** menu of Visual Basic.

Visual Basic responds by executing the MyInput program.


 Click the **Pick a Number** button.

The MyInput program responds by displaying the Input dialog box shown in Figure 13.2.



**Figure 13.2.** *The Input dialog box that is displayed after you click the Pick a Number button.*

As you can see from Figure 13.2, the Input dialog box asks you to type a number between 1 and 9.

 Click the OK button of the Input dialog box, and then terminate the MyInput program.

## The Code of the Pick a Number Button

Let's go over the code that you typed inside the `cmdPickANumber_Click()` procedure.

The code that you typed start by declaring a few variables:

```
Dim Message
Dim Title
Dim DefaultValue
Dim UserValue
```

You then assigned values to the variables:

```
Title = "The MyInput Program"
Message = "Please type a number between 1 and 9"
DefaultValue = 3
```

Finally, you execute the `InputBox()` function as follows:

```
UserValue = InputBox(Message, Title, DefaultValue)
```

You assigned the string **Please type a number between 1 and 9** to the **Message** variable:

```
Message = "Please type a number between 1 and 9"
```

You supplied **Message** as the first parameter of the `InputBox()` function:

```
UserValue = InputBox(Message, Title, DefaultValue)
```

Now take a look at Figure 13.2. As you can see, the string **Please type a number between 1 and 9** appears inside the Input dialog box. This is so,

=====

because this is the string that you supplied to the first parameter of the **InputBox()** function.

You set the value of the variable **Title** as follows:

```
Title = "The MyInput Program"
```

You then supplied **Title** as the second parameter of the **InputBox()** function:

```
UserValue = InputBox(Message, Title, DefaultValue)
```

So when the Input dialog box is displayed, the title of the Input dialog box is **The MyInput Program** (see Figure 13.2).

You assigned the value **3** to the **DefaultValue** variable as follows:

```
DefaultValue = 3
```

You supplied **DefaultValue** as the third parameter of the **InputBox()** function as follows:

```
UserValue = InputBox(Message, Title, DefaultValue)
```

So when the Input dialog box is displayed, the default value is **3** (see Figure 13.2).



## Summary

The **InputBox()** function lets you display a dialog box where the user is asked to type data.

- The first parameter of the **InputBox()** function is the text that will appear inside the Input dialog box.

=====


• The second parameter of the **InputBox()** dialog box is the text that will appear as the title of the Input dialog box.

• The third parameter of the **InputBox()** dialog box is the default value that will appear inside the Input dialog box.

## Doing Something With the User Input

So far, you implemented an Input dialog box that lets the user type a number. But the program does not do anything with the number that the user typed. You'll now add code to the MyInput program that demonstrates how your program can do something with the data that the user typed.


 Place a Label control inside Form1.

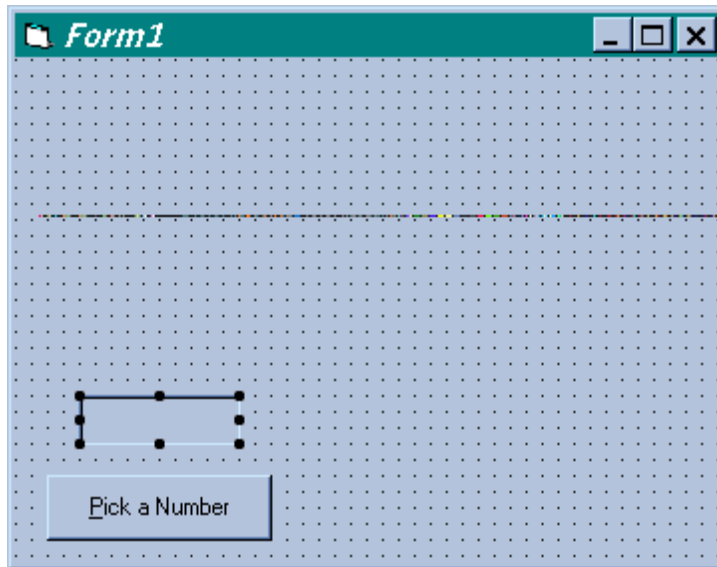
 Set the properties of the Label as follows:

**Name:** lblUserNumber

**Caption:** Make it empty

**BorderStyle:** 1-Fixed Single

 Drag the Label so that Form1 will look as shown in Figure 13.3.



**Figure 13.3.** *Form1 with the Label control in it.*

## Updating the Label with the User Input

You'll now write code that displays the number that the user selected inside the **lbl userInput** label.

Take a look at the statement that displays the Input dialog box:

```
UserValue = InputBox(Message, Title, DefaultValue)
```

The **InputBox()** function returns a value, and the returned value is assigned to the **UserValue** variable. For example, if the user typed **This is my data**, and then clicked the OK button of the Input dialog box, then the **UserValue** variable is updated with the text **This is my data**. You'll now set the **Caption** property of the **lbl userInput** label with the value of the **UserValue** variable:

 Add code to the **cmdPickANumber\_Click()** procedure so that it will look as follows:

```
Private Sub cmdPickANumber_Click()
```

```
...
```

```
...
```

```
=====
```


```
...  
lblUserInput.Caption = UserValue  
End Sub
```

You added the following statement to the end of the **cmdPickANumber\_Click()** procedure:

```
lblUserInput.Caption = UserValue
```


So the **Caption** property of the **lblUserInput** label is updated with the returned value of the **InputBox()** function. Let's see this in action:

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


 Select **Start** from the **Run** menu of Visual Basic to execute the MyInput program.

 Click the **Pick a Number** button.


The MyInput program responds by displaying the Input dialog box.

 Type something inside the Input dialog box, and then click the OK button of the Input dialog box.

The MyInput program responds by closing the Input dialog box, and inside the label control you can see the data that you typed inside the Input dialog box.

 Experiment with the MyInput program by clicking the **Pick a Number** button, type something inside the Input dialog box, and then click the OK button of the Input dialog box.

=====


 Experiment with the MyInput program by clicking the **Pick a Number** button, type something inside the Input dialog box, and then click the **Cancel** button of the Input dialog box. Note that if you click the **Cancel** button of the Input dialog box, the **lblUserInput** label does not display any text. Why? Because when you click the **Cancel** button of the Input dialog box, the **InputBox()** function returns null (nothing). Then the following statement is executed:

```
lblUserInput.Caption = UserValue
```

Because after you click the **Cancel** button the **UserValue** variable is equal to null, it is as if the following statement is executed after you click the **Cancel** button:

```
lblUserInput.Caption = ""
```

This explains why the **lblUserInput** label is empty after you click the **Cancel** button of the Input dialog box.

 Experiment with the MyInput program and then terminate the program.

## Checking the Validity of the User Data

So far, you displayed inside the **lblUserInput** label whenever the user typed inside the Input dialog box. You did not make any efforts to check the validity of the user input. The user is instructed to type a number between 1 and 9. And yet, nothing bad happens if the user type any text, or a number that is not inside the range of numbers between 1 and 9.

You'll now add code that checks the validity of the user input. That is, your code will check that the user typed a number between a and 9, and if this is not the case, your program will display an error message to the user, telling the user that incorrect data was entered.

 Add code to the **cmdPickANumber\_Click()** procedure so that it will look as follows:

```
Private Sub cmdPickANumber_Click()  
  
Dim Message  
Dim Title  
Dim DefaultValue  
Dim UserValue  
  
Title = "The MyInput Program"  
Message = "Please type a number between 1 and 9"  
DefaultValue = 3  
  
UserValue = InputBox(Message, Title, DefaultValue)  
  
UserValue = Val(UserValue)  
  
If UserValue < 1 Or UserValue > 9 Then  
    MsgBox "Number must be between 1 and 9"  
Else  
    lblUserInput.Caption = Str(UserValue)  
End If  
  
End Sub
```

Let's go over the code that you added:

You added the following statement:

```
UserValue = Val(UserValue)
```

=====

The **Val()** function converts a string to a number. For example, if you want to convert the string "123" to a the number 122, you have to type "123" as the parameter of the **Val()** function.

For example, if you supply the string "33" as the parameter of the **Val()** function, the returned value is the number 33.

If for example you supply the string "This is my data" as the parameter of the **Val()** function, the **Val()** function will return 0. (i.e., when the string that you supply as the parameter is not something that can be converted to a number, the **Val()** function will convert the string to 0).

You then use the following statements:

```
If UserValue < 1 Or UserValue > 9 Then
    MsgBox "Number must be between 1 and 9"
Else
    lblUserInput.Caption = Str(UserValue)
End If
```

The preceding statements are **If...Else...End If** statements that examines the value of the **UserValue** variable.

The **If** statement is as follows:

```
If UserValue < 1 Or UserValue > 9 Then
```

So if the value of the variable **UserValue** is less than 1 or greater than 9, the code under the **If** statement is executed. The code under the **If** statement displays a message box to the user:

```
MsgBox "Number must be between 1 and 9"
```


If the **UserValue** variable is a number between 1 and 9, the **If** condition is **not** satisfied. In this case, the code under the **If** statement is **not** executed. Because the **If** condition is not satisfied, the code under the **Else** is executed:


```
lblUserInput.Caption = Str(UserValue)
```

The preceding statement uses the **Str()** function to convert the number **UserValue** to a string, and then the string is assigned to the **Caption** property of the **lblUserInput** label.


Let's see you code in action:

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


 Select **Start** from the **Run** menu of Visual Basic to execute the MyInput program.

 Click the **Pick a Number** button, and then type **4** inside the Input dialog box. Then click the OK button of the Input dialog box.


The MyInput program responds by displaying **4** inside the **lblUserInput** label control (because you typed **4** and this is a number between 1 and 9).

 Click the **Pick a Number** button, and then type **-4**. Then click the OK button of the Input dialog box.

The MyInput program responds by displaying a message box, telling you that the number has to be in the range between 1 and 9.

 Click the **Pick a Number** button, and then type **My Number**. Then click the OK button of the Input dialog box.

The MyInput program responds by displaying a message box, telling you that the number has to be in the range between 1 and 9.

 Experiment with the MyInput program, and then terminate the program.

## Modal Dialog Boxes

The Input dialog box that you implemented is called a **modal** dialog box. A modal dialog box forces the user to close the dialog box before continuing with the execution of the program. To see this in action, perform the following experiment:


 Execute the MyInput program.

 Click the **Pick A Number** button.

The MyInput program responds by displaying the Input dialog box.

 Click inside the window of the MyInput program.

Windows refuses to switch to the window of the MyInput program. Why? Because the Input dialog box is a modal dialog box, and you will be able to switch back to the window of the MyInput program only after you close the Input dialog box.

 Experiment with the MyInput program, and then terminate the program.

## What You Accomplished in This Lesson



You completed Lesson 13 of the Self Study Visual Basic tutorial. In this lesson you learned about the **InputBox()** function, a function that lets you take user input with great ease.

## Frequently Asked Questions



**Q1.** Should I use the **InputBox()** function to take user input, or should I use the Edit box control to take user input?

**A1.** As you saw in this lesson, the **InputBox()** function lets you implement a user input mechanism with great ease. So use the **InputBox()** function whenever possible. Windows user are used to this type of user interface, any by using conventional standard user interface mechanism, you are making your program easy to use.

**Q2.** Why did the designers of Visual Basic designed the Input dialog box as a modal dialog box?

**A2.** It makes sense to have the Input dialog box as a modal dialog box. You use the Input dialog box to take user input, and then your program does something with the input. By having the Input dialog box as a modal dialog box, the user is forced to finish the business of entering the data. The user can click the **Cancel** button of the Input dialog box, the user can close the Input dialog box by clicking the close icon that the Input dialog box has, and the user can close the input dialog box by clicking the **OK** button. No matter how the user choose to close the Input dialog box, the process of entering the data (whether valid data or not) is completed when the user closed the Input dialog box.

**Q3.** Suppose that the Input dialog box asks the user to input a certain number. The user has to switch to another program (such as Excel, or Word for Windows), and take the requested number form another program. Is it possible to do so?

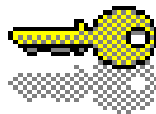
**A3.** Yes, it is possible to do so. When the Input dialog box is displayed, the user cannot switch back to the window of the MyInput program, because the Input dialog box (a modal dialog box) was executed from the MyInput program. But the user can switch to other programs.

## Exam




1. The Input dialog box is an example of modal dialog box. Give an example of another modal dialog box that is used in the MyInput program.


## Answers to Exam




1. If the user types a number that is not inside the range of numbers between 1 and 9, the MyInput dialog box displays an error message.

Let's see this in action:

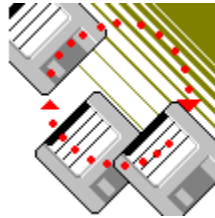
 Execute the MyInput program, and then click the **Pick a number** button.

 Type the number **10** inside the Input dialog box, and then click the OK button of the Input dialog box.

The MyInput program responds by displaying a message box, telling you to type a number between 1 and 9. This message box is a modal dialog box. While the message box is displayed click inside the window of the MyInput program, and verify that you cannot switch to the window of the MyInput program unless you first close the message box.

 Experiment with the MyInput program, and then terminate the program.

## Project



Currently, Windows decides where on the screen the Input dialog box will be displayed. In this project you'll write code that will determine the location where the Input dialog box will be displayed.

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

```
Private Sub cmdPickANumber_Click()  
...  
...  
...  
  
UserValue = InputBox(Message, Title, DefaultValue, 10, 10)  
  
...  
...  
...  
End Sub
```

=====


You added two more parameters to the **InputBox()** function:

```
UserValue = InputBox(Message, Title, DefaultValue, 10, 10)
```


You supplied 10 as the fourth parameter of the **InputBox()** function, and you supplied 10 as the fifth parameter of the **InputBox()** function.

The fourth parameter specifies the distance between the left edge of the Input dialog box and the left edge of the screen. The fifth parameter specifies the distance between the top edge of the Input dialog box and the top edge of the screen.

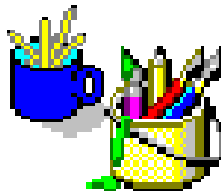
The fourth and fifth parameters of the **InputBox()** function are specified in units of **twips**. What is a twip? A twip is a unit of distance. There are 1440 twips in one inch.

 Execute the MyInput program. Click the **Pick a Number** button to display the Input dialog box.

As you can see, the Input dialog box is displayed 10 twips from the left edge of the screen, and 10 twips from the top edge of the screen.

 Close the Input dialog box, and then terminate the MyInput dialog box.

## Cosmetic Considerations



You learned in the Project section of this lesson that you can supply fourth and fifth parameters to the **InputBox()** function, and the fourth and fifth parameters will place the Input dialog box at the location that you specified.

If you do not specify the fourth and fifth parameters of the **InputBox()** function, the default values will be used.

The default values for the fourth parameter of the **InputBox()** function is such that the Input dialog box will be displayed horizontally centered.

The default values for the fifth parameter of the **InputBox()** function is such that the Input dialog box will be displayed approximately one-third of the way down the screen.

Cosmetically speaking, the default location of the input dialog box is the perfect place. So in your programs, do not supply any values for the fourth and fifth parameters of the **InputBox()** function.

## 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](mailto: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**

=====

**TegoSoft Visual Basic 4 Self Study Tutorial - Lesson 13**

**Page 20 of 21**

**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