

# Error Trapping



During run-time there are parts of a program where errors are unlikely, such as reading or writing a value to an internal variable, but there are other sections where the likelyhood of errors occuring is much higher. For example, anywhere in a program which tries to open a file could run into problems. The file might not be there, it could have been deleted a few days ago in the File Manager! Instead of the program grinding to a halt when it hits one of these errors the programmer can make some educated guesses where errors will most likely occur and then trap for them.

The **On Error** command is used to tell VB what to do if an error occurs. If one does not happen and the program runs smoothly then any error trapping/handling code is ignored. The full syntax of this command is shown below:

**On Error** {**Goto** *line* | **Resume Next** | **Goto 0**}

The On Error command does the error *trapping* so it is important to place this **before** and lines of code which could produce an error. The most common way to use this command is to set a line to goto when an error occurs:

**On Error GoTo ErrHandling**

Now that the error trapping code has been written we can put in the code which will *handle* the error. To do this we need to place the line label where VB will go in the even of an error. The above On Error specified the line ErrHandling, but this could be any user defined name. Below this line label can be placed any lines of code which are specifically written to deal with any error which occurs. After this we have to add Resume Next so that VB goes back to the line after where the error occured. The only other bit of code we need is an Exit Sub just before the error line label so that the error handling code is never run when there are no errors. The resulting code should look like:

**Private Sub Form\_Load ()  
 On Error GoTo ErrHandling ' Trap errors with this line  
 Open "C:\not\_here.txt" For Input As #1  
 Close #1  
  
 Exit Sub  
  
 ErrHandling:  
 Rem Error handling section  
 Debug.Print "Error Number = "; Err  
 Debug.Print "Error Message = "; Error$(Err)  
 Resume Next ' Carry on program after the error occurred  
 End Sub**

Although the above program does not do much, there is a blank form displayed with no controls, it does, however, print two lines on the debug window:

**Error Number = 53   
 Error Message = File not found**

These two lines are printed because the file “NOT\_HERE.TXT” cannot be found on Drive ‘C’. Thus, when the error happens the program jumps down to the line label ErrHandle prints the two messages to the debug window, and then returns back up to the line after the error occurred. Try changing the name of the file to one that does exist and run the program again to see what happens.

 [Menu](http://docs.google.com/dvbhome.html)