10.5.3 Stop

Die Anweisung STOP stoppt ein Programm und weckt den Debugger, als ob sich ein Haltepunkt in der aktuellen Zeile befinden würde. Dieser Befehl unterbricht das Programm – mehr passiert nicht.

10.5.3.1 Stop vs. Breakpoint

Relevant ist STOP eigentlich nur im Debug-Modus beim Programm-Test. Normalerweise entwickelt man Programme in der IDE und kann dort Breakpoints setzen. Aber es existiert ein kleiner, feiner Unterschied: STOP können Sie an eine Bedingung knüpfen, während ein Breakpoint fest an eine Zeile geknüpft ist. Wenn zum Beispiel in einer Loop-Kontrollstruktur nur der letzte Durchlauf interessiert, dann können Sie im Schleifen-Körper schreiben:

```
If iCurrentIndex = iLastIndex Then Stop
```

Das Programm wird *höchstens einmal* unterbrochen, wenn *iCurrentIndex* in der Schleife einen ganz speziellen Wert annimmt. Würden Sie dagegen einen zeilengebundenen Breakpoint setzen, hält das Programm in jeder Iteration an, was nervig werden kann.

10.5.3.2 Stop Event

Die komplexe Anweisung STOP EVENT ist eine eigenständige Anweisung, hat keinen Bezug zu dem o.a. Stop und muss in einem *Event-Handler* verwendet werden. Sie teilt dem Interpreter mit, dass das (native) *Ereignis abgebrochen wird*, welches den Event-Handler aufgerufen hatte.

Die folgenden drei Beispiele demonstrieren die Verwendung von *Stop Event*:

Beispiel 1

```
PUBLIC SUB TextBox1_KeyPress()
  If Key.Text Not Like "[0-9]" Then STOP EVENT
END ' TextBox1_KeyPress()
```

```
PUBLIC SUB TextBox1_KeyPress()
 IF Instr("0123456789", Key.Text) = 0 THEN STOP EVENT
END ' TextBox1_KeyPress()
```

Diese oft angegebenen zwei Prozeduren zur sicheren Eingabe von Ziffern aus dem Intervall [0-9] haben den generellen Nachteil, dass Sie keine Möglichkeiten haben, fehlerhafte Eingaben direkt über die Tastatur zu löschen respektive zu korrigieren oder in der TextBox zu navigieren! Es bleibt jedoch die Möglichkeit, fehlerhafte Ziffern oder Ziffernblöcke *mit der Maus* zu markieren und mit den korrekten Ziffern zu überschreiben.

Eine echte Alternative – von Charles Guerin und Benoît Minisini – akzeptiert nicht nur die Ziffern 0..9 als sichere Eingabe, sondern auch die sechs Tasten, die Sie für notwendige Korrekturen und zur Navigation in der TextBox benötigen:

```
Public Sub CheckInput(sAllowed As String)
  Dim iAllow As Integer = 0

  If Key.Code = Key.BackSpace Then iAllow = 1 ' 2xLöschen
  If Key.Code = Key.Delete Then iAllow = 1
  If Key.Code = Key.Left Then iAllow = 1 ' 4xNavigation
  If Key.Code = Key.Right Then iAllow = 1
  If Key.Code = Key.End Then iAllow = 1 ' Ende
  If Key.Code = Key.Home Then iAllow = 1 ' Pos1

  If Key.Text And (InStr(sAllowed, Key.Text) > 0) Then iAllow = 1

' Wird eine nicht erlaubte Taste gedrückt, dann wird das _KeyPress-Ereignis abgebrochen
  If iAllow = 0 Then STOP EVENT

End ' CheckInput(sAllowed As String)

PUBLIC SUB TextBox1_KeyPress()
  CheckInput("0123456789")
END ' TextBox1_KeyPress()
```

Als Erweiterung können Sie nach '*If Key.Code = Key.Home …*' noch die folgenden Zeilen in die Prozedur *CheckInput(..)* einfügen. Mit der Enter- oder Return-Taste übernehmen Sie den Inhalt der TextBox – wenn dort mindestens ein Zeichen vorhanden ist – und speichern ihn in der Variablen sInput:

```
If TextBox1.Text And (Key.Code = Key.Enter Or Key.Code = Key.Return) Then
   iAllow = 1
   sInput = TextBox1.Text ' Print TextBox1.Text
Endif
```

Beispiel 2

```
PUBLIC SUB MyObserver_Show()
  Message.Info("Das Hauptfenster hat das Licht des Monitors bereits erblickt!")
  STOP EVENT  ' Das Ereignis MyObserver_Show wird abgebrochen
END ' MyObserver_Show()
```

Beispiel 3

```
PUBLIC SUB Form_Close()
  IF NOT ModulGlobal.AllowClose THEN
     STOP EVENT ' Das Fenster bleibt offen … Das Ereignis Form_Close wird abgebrochen
  ELSE
     ME.Close
  ENDIF
END ' Form_Close()
```