# Inhaltsverzeichnis

**Makrobeispiel: Alter berechnen**

Schreiben Sie ein Makro, das aus einem per Dialog angeforderten Geburtsdatum das Alter berechnet. Da dieses Makro keine ausführbaren Aktionen enthält, müssen Sie die Codezeilen ohne Makrorekorder erstellen.

Aktivieren Sie den Visual Basic Editor. Legen Sie ein neues Modul an und schreiben Sie diesen Code:

**Makro Nr. 13**

```
Sub AlterBerechnen()
 Dim GDatum, GDatumA, varAlter
D_Eingabe:
 GDatum = InputBox("Bitte geben Sie Ihr Geburtsdatum ein:")
 If GDatum = "" Then Exit Sub
 If Not IsDate(GDatum) Then
 MsgBox "Bitte geben Sie ein gültiges Datum ein", vbCritical
  GoTo D_Eingabe
 End If
 GDatumA = DateSerial(Year(Date), Month(GDatum), Day(GDatum))
 If GDatumA <= Date Then
   alter = Year(Date) - Year(GDatum)
 Else
   alter = Year(Date) - Year(GDatum) - 1
 End If
 MsgBox "Geburtstag: " _
  & Format(GDatum, "dddd, d. MMMM YYYY") & vbCr _
 & "Alter: " & alter
End Sub
```

## 5.3 Schrittweise testen

Mit dem Schritt-Test wird ein Makro Zeile für Zeile durchgetestet. Sie können Makros schon beim Aufruf im Excel-Fenster schrittweise ablaufen lassen. Markieren Sie das Makro in der Makroliste und klicken Sie auf die Schaltfläche *Schritt*.

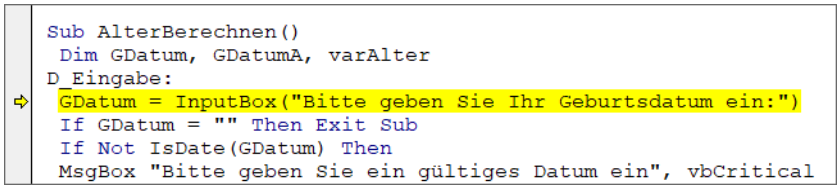


*Abbildung 5.2: Der Schrittmodus: Die gelbe Zeile ist die nächste*

Der Visual Basic Editor wird aktiviert, die erste Zeile des Makros erhält einen gelben Hintergrund. Drücken Sie die Taste F8, um Zeile für Zeile durch den Code zu wandern. In der Markierungsleiste am linken Rand zeigt ein kleiner Pfeil auf die jeweils nächste Zeile, die zur Ausführung ansteht.

Die Symbolleiste *Debuggen* bietet auch Symbole für das schrittweise Austesten an, der *Einzelschritt* entspricht der Taste F8, mit dem *Prozedurschritt* können Sie so testen, dass Unterprogramme nicht einzeln, sondern als Block durchgerechnet werden.

Mit der Taste F5 oder einem Klick auf das *Fortsetzen*-Symbol wechseln Sie vom Einzelschritt in die normale Ausführung.

Wenn Sie das Makro ab einer Stelle weiter unten oder oben im Code testen wollen, setzen Sie einfach den Zeiger um. Klicken Sie in die Zeile und drücken Sie Strg+F9 oder ziehen Sie das Zeigersymbol am linken Rand mit gedrückter Maustaste an die neue Position.

Ziehen Sie den gelben Pfeil am Rand mit dem Mauszeiger an eine neue Position, läuft das Makro ab dieser Zeile weiter. Sie können auch den Cursor in eine andere Zeile setzen und Strg+F9 drücken.

## 5.4 Schrittmodus per Stop-Anweisung

Sie können ein Makro so präparieren, dass es an einer bestimmten Codezeile automatisch in den Schrittmodus wechselt. Geben Sie die Anweisung *Stop* ein.

Das Makro durchläuft einen Bereich in der aktiven Tabelle und wechselt in den Schrittmodus, wenn es auf eine Zelle stößt, der das EUR-Zahlenformat zugewiesen ist.

**Makro Nr. 14**

```
Sub StopTest()
 MsgBox "Makro stoppt in der ersten EUR-Zelle"
 For Each varzelle In Range("F1:F20")
   varzelle.Select
   If InStr(varzelle.NumberFormatLocal, "EUR")  0 Then
     Stop
   End If
 Next varzelle
End Sub
```

## 5.5 Unterprogramme testen

Mit der Anweisung *Call* wird ein Unterprogramm aufgerufen. In der Praxis ist es sehr zeitraubend, bereits getestete Routinen im Einzelschritt abzuarbeiten. Verwenden Sie das Symbol *Prozedurschritt* oder diese beiden Tastenkombinationen:

⇧+F8 überspringt das Unterprogramm, es wird als eine Codezeile ausgeführt.

Strg+⇧+F8 drücken Sie im Unterprogramm, wenn Sie alle weiteren Zeilen ausführen und zum Einzelschrittmodus im aufrufenden Programm zurückkehren möchten.

## 5.6 Haltepunkte

Haltepunkte sind besonders in längeren Makrocodes von großem Nutzen, wenn die schrittweise Ausführung eines bereits getesteten Codes nicht ausreicht oder nicht mehr nötig ist. Setzen Sie bei den Makroanweisungen, ab denen Excel das Makro schrittweise testen soll, Haltepunkte:

Klicken Sie im Modul auf die Anweisung, bei der Sie den Unterbrechungsmodus einschalten wollen. Setzen Sie mit *Debuggen/Haltepunkt ein/aus* einen Haltepunkt. Die Zeile wird rot unterlegt. Starten Sie das Makro per Klick auf das Symbol *Sub/UserForm ausführen*. Das Makro läuft bis zur Zeile mit dem Haltepunkt und schaltet dort in den Schrittmodus.

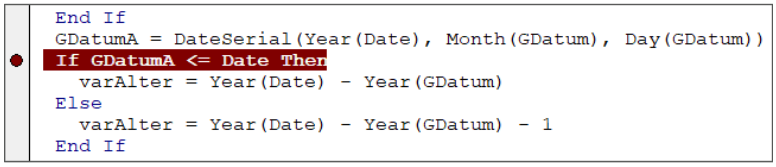


*Abbildung 5.3: Ein Haltepunkt*

Mit der Taste F5 springen Sie im Makro von Haltepunkt zu Haltepunkt, setzen Sie diese jeweils an die letzte Zeile eines ausgetesteten Blocks.

Ein Haltepunkt wird genauso gelöscht, wie er gesetzt wurde: Markieren Sie die Anweisung und wählen Sie *Debuggen/Haltepunkt ein/aus*.

**Ein schneller Haltepunktschalter**

Mit F9 oder einem Klick auf die Markierungsleiste links wird ein Haltepunkt gesetzt und auch wieder entfernt. *Alle Haltepunkte löschen* aus dem Menü *Debuggen* macht, was es ausdrückt: Die Option entfernt alle gesetzten Haltepunkte in einem Makro.

## 5.7 Das Direktfenster

Das wichtigste Testwerkzeug ist gerade für Einsteiger das Direktfenster. Es bietet die Möglichkeit, vor oder während des Makrolaufs Objekte, Methoden und Eigenschaften abzurufen, Variablen oder Ausdrücke zu testen oder einfach Zellinhalte, Datei- und Tabellennamen abzufragen.

Öffnen Sie das Fenster mit Strg+G oder *Ansicht/Direktfenster*.

Geben Sie ein Fragezeichen (?) ein. Das ist die Print-Anweisung für dieses Fenster, der Rest der Zeile wird »in das Fenster gedruckt«. Geben Sie eine Anweisung oder einen Teilcode ein und drücken Sie ↵. Das Ergebnis wird unter der Fragezeichenzeile angezeigt, Sie können eine neue Zeile eingeben oder die erste Zeile überschreiben. Fragen Sie im Direktfenster Informationen aus Windows oder Excel ab. Das sind Informationen, die Excel (oder Windows) zur Verfügung stellt und die im Makrocode über Anweisungen, Konstanten oder Objekteigenschaften abgerufen werden:

| Systeminfo | Ergebnis |
|---|---|
| `? Date` | Tagesdatum |
| `? Time` | Aktuelle Uhrzeit |
| `? Application.Username` | Name des Excel-Anwenders |
| `? Activesheet.Name` | Name der aktiven Tabelle |
| `? Application.Version` | Versionsnummer von Excel |

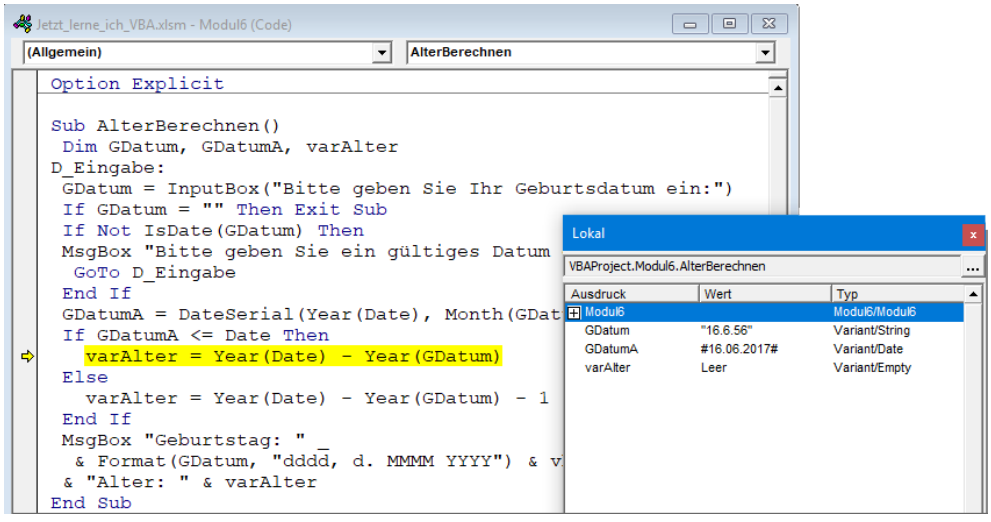


*Abbildung 5.4: Das Direktfenster*

## 5.8 Ausgabe in das Direktfenster per Makro

Das Direktfenster trägt den Objektnamen *Debug*, und um es von einer Prozedur aus anzusteuern, verwenden Sie diese Anweisung:

```
Debug.Print xx
```

Sie können alle Abfragen, die Sie eingetippt hatten, auch zu einem Makro zusammenfassen. Schreiben Sie dieses Makro in ein Modul:

**Makro Nr. 15**

```
Sub XLInfos ()
 Debug.Print Date
 Debug.Print Time
 Debug.Print "Benutzer: " & Application.UserName
 Debug.Print "Programm: " & Application.Name & " " & Application.Version
 Debug.Print "Aktuelle Tabelle: " & ActiveSheet.Name
 Debug.Print "Zellzeiger steht in Zelle: " _
             & ActiveCell.Address
 Debug.Print ActiveWorkbook.Name
End Sub
```

Starten Sie das Makro mit geöffnetem Fenster, sehen Sie, wie die Informationen Zeile für Zeile in das Direktfenster gedruckt werden.

### Makrobeispiel: Makro nach Bedingung anhalten

Sie können das Debug-Objekt auch dazu benutzen, das Makro anzuhalten, wenn eine bestimmte Bedingung nicht erfüllt ist. Verwenden Sie die Methode *Assert*, die das Makro anhält. Hier ein Beispiel: Das Makro durchläuft in einer Schleife die ersten 50 Zellen der Spalte A, stoppt bei der ersten Zelle, die leer ist, und wechselt in den Einzelschritt.

**Makro Nr. 16**

```
Sub DebugAssertTest()
 For i = 1 To 50
   Cells(i, 1).Select
   Debug.Assert (ActiveCell.Value  "")
 Next i
End Sub
```

## 5.9 Das Lokal-Fenster

Im Lokal-Fenster finden Sie alle deklarierten Variablen der aktiven Prozedur oder Funktion, wenn Sie im Einzelschrittmodus arbeiten. Aktivieren Sie dieses Fenster über *Ansicht/Lokal-Fenster*.

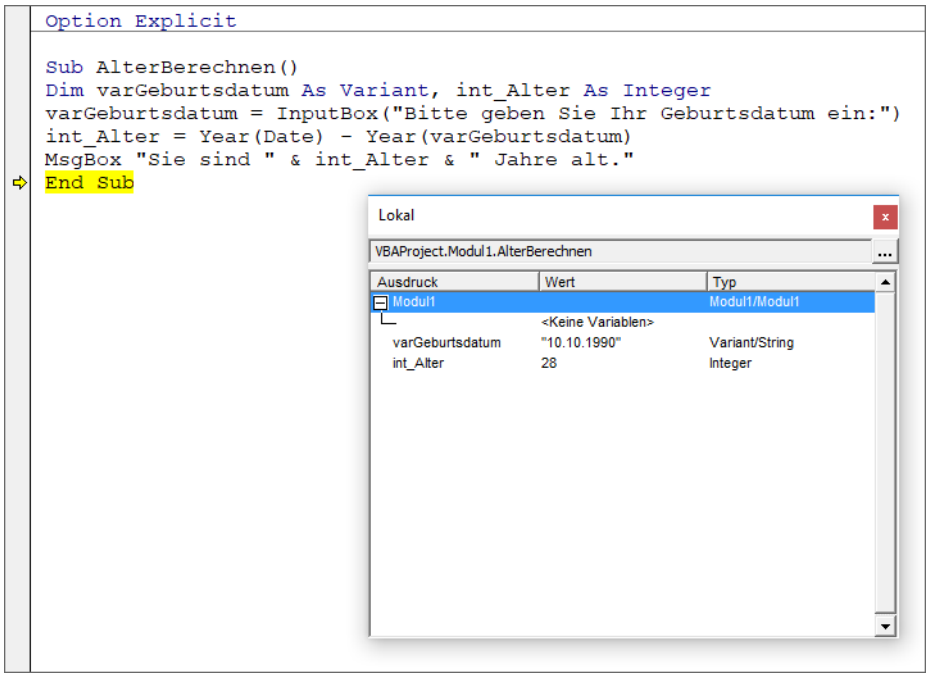


*Abbildung 5.5: Im Lokalfenster sehen Sie alle Variablen*

Sie können den Variableninhalt in der zweiten Spalte ändern, setzen Sie einfach den Cursor in den Eintrag und geben Sie einen neuen Wert ein. Das Symbol mit den drei Punkten rechts oben in der Ecke bietet alle Funktionen und Prozeduren des aktiven Moduls an.

## 5.10 Das Überwachungsfenster

Um die Änderungen in Variablen und anderen beweglichen Teilen des Makros gezielt kontrollieren zu können, schalten Sie das Überwachungsfenster ein. Sie können das Fenster einfach einblenden und Teile des Makrocodes darin einfügen:

Wählen Sie *Ansicht/Überwachungsfenster*. Markieren Sie einen Teil des Makrocodes, eine Variable, ein Objekt oder einen Ausdruck. Wählen Sie *Debuggen/Überwachung hinzufügen*, geben Sie die Überwachungsart an und bestätigen sie, um die Über-

wachung einzufügen. Für eine schnelle Überwachung ziehen Sie den markierten Codeteil einfach mit gedrückter Maustaste in das Fenster. Die Überwachungsart kontrollieren Sie mit *Überwachung bearbeiten* aus dem Kontextmenü der rechten Maustaste.

Testen Sie das Überwachungsfenster mit einem Makro, das Zufallszahlen generiert:

**Makro Nr. 17**

```
Sub Zufallszahl()
 Dim intZähler
 For intZähler = 1 To 10
   Randomize
   TestWert = (Int(10 * Rnd) + 1)
 Next intZähler
End Sub
```

Ziehen Sie die Variablen *intZähler* und *TestWert* in das Überwachungsfenster und testen Sie das Makro mit F8. Mit jedem Durchlauf der Schleife erhalten die beiden Variablen auf der Überwachungskarte andere Werte. Der Schleifenzähler zählt sequenziell von 1 bis 10, die Zufallszahl erhält Zufallswerte zwischen 1 und 10.

Eine weitere Variante der Überwachung ist die Unterbrechung: Sie können eine Überwachung so definieren, dass sie die Ausführung eines Makros unterbricht und in das Textfenster schaltet, sobald eine Variable einen bestimmten Wert angenommen oder sich verändert hat. Fügen Sie für unser Beispiel eine Überwachung ein, die das Makro unterbricht, wenn eine Zufallszahl erzeugt wird, die größer als 90 ist.

Wählen Sie *Debuggen/Überwachung hinzufügen*. Geben Sie diesen Ausdruck ein:

```
TestWert 5
Unterbrechen, wenn der Wert True ist
```

Wählen Sie die Überwachungsart. Starten Sie das Makro mit F5. Der Editor stoppt die Ausführung und blendet das Testfenster ein, sobald eine Zufallszahl größer als fünf erzeugt wurde.

*Art der Überwachung* legt fest, wie VBA auf den Überwachungsausdruck reagiert. Wählen Sie *Überwachungsausdruck*, wird dieser zusammen mit seinem aktuellen Wert im Überwachungsfenster angezeigt. Im Haltemodus wird der Wert automatisch aktualisiert. Wählen Sie *Unterbrechen, wenn der Wert True ist*, wechselt das Makro automatisch in den Haltemodus, wenn der Ausdruck als wahr oder als beliebiger Wert ungleich null ausgewertet wird (nicht in Zeichenfolgenausdrücken). Mit *Unterbrechen, wenn Wert geändert wurde* wechselt das Makro automatisch in den Haltemodus, wenn sich der Wert des Ausdrucks innerhalb des angegebenen Kontextes ändert.

Um eine Überwachung wieder zu löschen, markieren Sie sie im Überwachungsfenster und drücken die Taste Entf. Ein Doppelklick auf die Zeile öffnet die Dialogbox zur Bearbeitung der Überwachung (auch unter *Extras/Überwachung bearbeiten*).

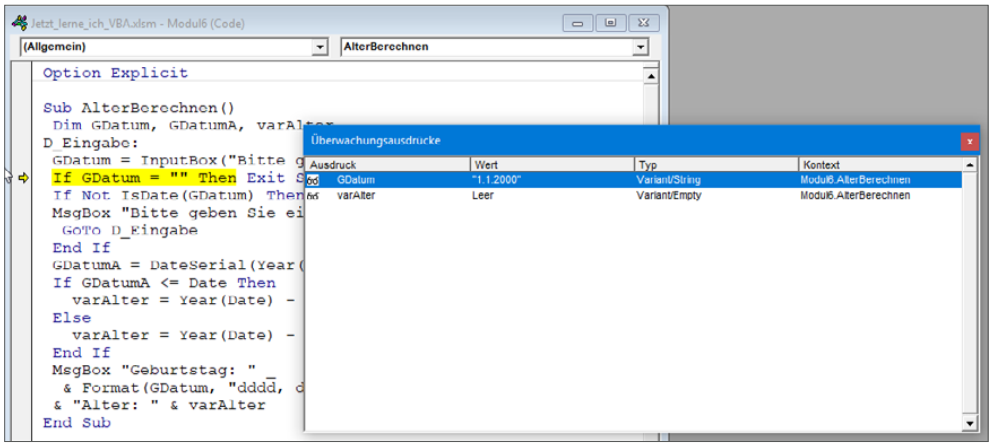


*Abbildung 5.6: Das Überwachungsfenster*

## 5.11 Makros kompilieren

Mit der Kompilierung eines Makros werden alle Makros noch mal gründlich auf Fehler getestet:

1. Markieren Sie das Projekt, das Sie kompilieren wollen.
2. Wählen Sie *Debuggen/Kompilieren von Projektxx*.

Im Unterschied zur Kompilierung in selbstständigen Programmiersprachen wie Visual Basic, C++ oder Java findet keine Umsetzung in Maschinensprache statt (nur intern), es wird auch keine ausführbare Datei erzeugt. Der Compiler endet auch ohne weitere Kommentare, wenn keine Fehler mehr im Code entdeckt wurden, und der Befehl im *Debuggen*-Menü ist inaktiv, solange das Projekt nach der Kompilierung nicht verändert wurde.

Mit der *bedingten Kompilierung* haben Sie die Möglichkeit, zwei Versionen von Makrocodes zu schreiben, eine *Testversion* und eine *Auslieferversion*. Wenn Sie Anweisungen wie *Stop*, *Debug.Assert* oder *Debug.Print* oder Meldungen im Makro haben, die nur beim Austesten zur Laufzeit auftauchen sollen und nicht in der Endversion, die an Kunden ausgeliefert wird, verpacken Sie diese in bedingte Kompilierungsblöcke. Sie brauchen zunächst eine Compilervariable, geben Sie diese mit einer *#CONST*-Anwei-

sung am Kopf des Moduls ein. Hier heißt die Variable *myTest*, sie wird auf den Boolean-Wert *True* gesetzt:

```
#Const myTest = True
```

Schreiben Sie ein Makro, das eine Reihe von Zahlen in einem Zellbereich abgreift und in eine Variable aufsummiert.

**Makro Nr. 18**

```
Sub ZahlenCheck()
 Dim varZelle, lngSumme
 Range("Testzahlen").Select
 For Each varZelle In Selection
  lngSumme = lngSumme + varZelle
 Next varZelle
```

Die Meldung der Summe erfolgt über eine bedingte Kompilierung, dazu wird vor *If* und *End If* jeweils ein #-Zeichen gesetzt.

```
 #If myTest = True Then
   MsgBox "Summe: " & lngSumme
 #End If
End Sub
```

Dieser Block wird nur ausgeführt, wenn die Compilervariable auf *True* gesetzt ist. Sie können jetzt in allen Makros Anweisungsblöcke schreiben, die vom Wert der Compilervariablen abhängig sind. Ist das Makro fertig getestet, setzen Sie die Variable auf *False* und die Blöcke werden nicht mehr ausgeführt. Schneller geht's mit einem Unterprogramm. Verpacken Sie die kompilierten Bedingungen in eine Subroutine und rufen Sie diese auf:

```
Call Check
Sub Check
#If . . .
#End If
End Sub
```

## 5.12 Makro abbrechen

Auch das wird Ihnen passieren: Eine Schleife läuft endlos, weil das Schleifenende nicht zu finden ist, oder ein Datenzugriff dauert ewig, weil Excel auf den Server wartet. Um ein laufendes Makro abzubrechen, drücken Sie einfach die Tastenkombination Strg+Pause.