Schleifen und Verzweigungen

Einen ersten Schleifentyp haben Sie im Kapitel über die Portbefehle schon kennen gelernt, die BEGIN-UNTIL-Schleife. Sie hat folgende Form:

```
begin Bef1 Bef2 Bef3 ... until
```

Durch diese Konstruktion wird die Befehlsfolge zwischen `begin` und `until` solange ausgeführt, bis das Wort `until` auf dem TOS eine 1 vorfindet. 1 wird allgemein als Wahrheitswert TRUE interpretiert, 0 als FALSE. Die Schleife wird also solange ausgeführt, bis auf dem Stack der Wahrheitswert TRUE vorliegt. Zu beachten ist, dass `until` den Wert auch aus dem TOS holt; es muss also bei jedem Schleifendurchlauf dafür gesorgt werden, dass für das Wort `until` ein passender Wahrheitswert auf den Stack gelegt wird.

Da im Kapitel über die Portbefehle schon einige praktische Beispiele für die BEGIN-UNTIL-Schleife vorgestellt worden sind, wollen wir uns gleich dem nächsten Schleifentyp zuwenden, der Zählschleife. In FORTH sieht sie folgendermaßen aus:

```
ew sw do Bef1 Bef2 Bef3 ... loop
```

Die Bezeichner `ew` (Endwert) und `sw` (Startwert) stehen hier für den Wert des Schleifenindex beim letzten bzw. ersten Schleifendurchlauf. Innerhalb der Zählschleife, also zwischen den Wörtern `do` und `loop`, kann man auf den Schleifenindex mithilfe des Wortes `I` zurückgreifen: `I` legt den aktuellen Schleifenindex auf den Stapel. Schauen wir uns ein einfaches Beispiel dazu an:

```
: zählen 25 10 do I . 100 waitms loop ;
```

Bei diesem Wort startet die Zählschleife mit dem Index 10. Diese Zahl wird zunächst durch das Wort `I` auf den Stapel gelegt und mit dem Wort . am Port B ausgegeben. Nach 100 Millisekunden Wartezeit wird der Schleifenindex automatisch erhöht und die Schleife ein weiters mal durchlaufen. Die Schleife wird ein letztes Mal durchlaufen, wenn der Schleifenindex den Wert 25 hat. Unser Programm zählt also im Zehntelsekunden-Rhythmus von 10 bis 25 und hört dann auf.

Man beachte bei der Angabe der Werte für den Schleifenindex die Reihenfolge: Zuerst wird der Endwert und dann der Startwert angegeben.

Als weiteres Beispiel schauen wir uns die FORTH-Definition der Multiplikation an:

```
: * 0 swap 1 do swap dup rot + loop ;
```

Lautet das Hauptprogramm z. B.

```
: main 12 7 * . ;
```

so wird die Zahl 12 insgesamt 7 mal zur 0 addiert; die Multiplikation wird also auf eine Mehrfachaddition zurückgeführt. Wie das im Detail abläuft, sollte der Leser einmal selbst überlegen, indem er für jeden einzelnen Schritt den Inhalt des Stacks notiert.

Unser MikroForth besitzt nur einen einzigen Verzweigungstyp, die `skipIf`-Anweisung. Dieses Wort wertet zunächst den TOS aus; liegt auf dem TOS der Wert 1 (TRUE), wird die nächste Anweisung übersprungen. Liegt auf dem TOS der Wert 0 (FALSE), wird einfach mit dem nächsten Befehl (Wort) weitergearbeitet.

`1 skipIf Bef1 Bef2 Bef3 ...`	Hier wird nach dem Wort `skipIf` das Wort `Bef1` übersprungen und sofort mit dem Wort `Bef2` weitergearbeitet. Es folgt das Wort `Bef3` usw.
`0 skipIf Bef1 Bef2 Bef3 ...`	Hier wird nach dem Wort `skipIf` mit dem Wort `Bef1` weitergearbeitet. Es folgen die Worte `Bef2` und `Bef3` usw.

Häufig ergeben sich dabei die Wahrheitswerte 0 und 1 als Ergebnisse von Vergleichen. Hier kommen Vergleichsoperatoren zum Einsatz. Ähnlich wie die Rechenoperatoren +, *, - und / werden sie bei FORTH auch in der Postfix-Schreibweise benutzt. Durch

```
7  2  >
```

wird also überprüft, ob 7 > 2 gilt. Da das in diesem Fall wahr ist, wird als Ergebnis dieser Vergleichsoperation der Wert 1 (TRUE) auf den Stapel gelegt. Weitere Vergleichsoperatoren sind < und =.

Ein Beispiel soll erläutern, wie Vergleichsoperatoren und Verzweigungen sinnvoll eingesetzt werden können. Ein Messprozess möge bereits zwei Messwerte (z. B. Temperaturwerte) auf den Stapel gelegt haben. Das Wort `unterschied` soll - wie der Name schon sagt - den Unterschied der beiden Zahlen bestimmen; es könnte z. B. erforderlich sein, vom Mikrocontroller bestimmte Maßnahmen einleiten zu lassen, wenn dieser Unterschied zu groß ist.

Kümmern wir uns zunächst um Berechnung und Ausgabe des Unterschieds. Auf den ersten Blick scheint dieses Problem recht einfach zu lösen zu sein:

```
: unterschied  - . ;
```

Zu Testzwecken geben wir bei unserem Forth-Compiler ein:

```
: main 7 2 unterschied ;
```

Nach dem Interpretieren, Compilieren und Übertragen zeigt unser Mikrocontroller den Wert 5 an - wie erwartet! Nun geben wir die Messwerte aber einmal in umgekehrter Reihenfolge ein:

```
: main 2 7 unterschied ;
```

Nun zeigt der Mikrocontroller am Port B den Wert 251(!) an. Wie lässt sich dieses offensichtlich unsinnige Ergebnis erklären, und - mindestens genau so wichtig - wie lässt sich unser Programm verbessern?

Zunächst zur Erklärung: Bei der Subtraktion 2 - 7 gelangt der Mikrocontroller in den Bereich unter 0. Er arbeitet dabei wie ein Kilometerzähler: Wenn man ausgehend vom Kilometerstand 0002 nun 7 km rückwärts fährt, kommt man beim Stand von 9995 aus. Der Kilometerstand springt nämlich beim Rückwärtszählen von 0000 auf 9999. Ganz ähnlich arbeitet der Mikrocontroller: Er springt beim Rückwärtszählen von 000 auf 255.

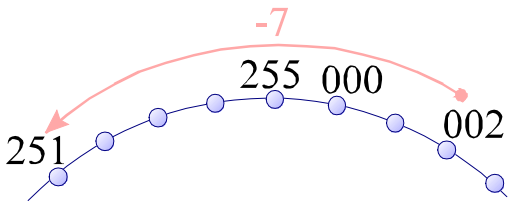

Abbildung 1

Das Problem liegt also offensichtlich in der Reihenfolge der beiden Messwerte. Um das Programm zu verbessern, müssen wir dafür sorgen, dass die Messwerte ausgetauscht werden, wenn der erste Messwert kleiner als der zweite ist. Hier kann unser `skipIf`-Wort zum Einsatz kommen; für den zugehörigen Vergleich müssen die Messwerte allerdings vorher noch kopiert werden.

```
: unterschied over over > skipIf swap - . ;
```

Das folgende Stapelbild macht deutlich, was bei der Ausführung von `unterschied` geschieht.

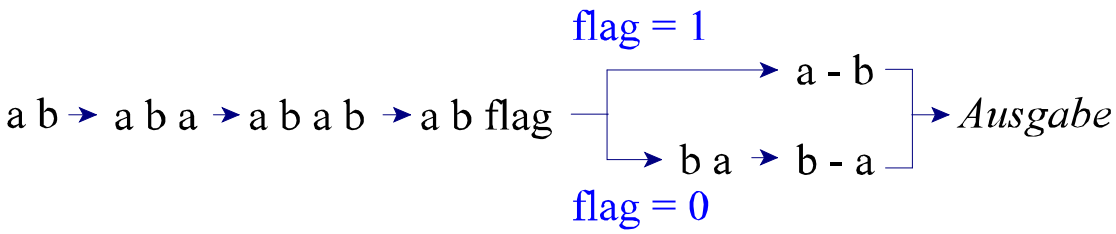

Abbildung 2

Aufgabe 1

Wenn der Unterschied der beiden Messwerte auf dem Stapel größer als 3 ist, dann soll ein Warnton über D.6 und den Beeper ausgegeben werden.

Aufgabe 2

Schreiben Sie eine Definition für das FORTH-Wort “<=”.

Aufgabe 3

Wie lautet die FORTH-Definition für das Wort `not` ?