Binäre Operationen

Binäre Operationen

AND, OR, XOR und NOT

220px-George_Boole_color.jpg
George Boole Quelle: Wikipedia

Oftmals besteht der Wunsch einzelne Bits in einer Bytevariablen logisch zu manipulieren. Logisch heißt in diesem Fall, dass der Wert im Programm nicht einfach auf 0 oder 1 gesetzt wird, sondern das die Änderung in Abhängigkeit des aktuell gültigen Wertes erfolgt.

Um das System zu verstehen muss man sich einmal die Mühe machen, die Veränderungen im Detail nachzuvollziehen. Was hier zur Anwendung kommt, ist die sogenannte boolesche Algebra, benannt nach dem englischen Mathematiker George Boole. Hierbei werden zwei Bytewerte(8 Bit) binär verrechnet, der Inhalt des ersten Ausdrucks also mit dem Inhalt des zweiten Ausdrucks auf eine vorbestimmte Art ausgewertet.
Im Einzelnen stehen hier die folgenden Operationen zur Verfügung:

AND OR XOR NOT

Diese werden wir hier nun Schritt für Schritt bearbeiten und am Ende sehen, dass die Arbeit mit den binären Operationen im Grunde genommen nicht schwer ist.

AND (Und)

Als Erstes wenden wir uns der AND- Operation zu. Wie oben geschrieben haben wir zunächst unseren Ausgangswert, den wir durch die AND- Operation verändern möchten:
In unserem Beispiel ist der Ausgangswert folgende 8- Bit- Folge:

&B00001110

Um uns die ganze Situation bildlich vorzustellen, nehmen wir an, die Bitfolge steht für 8 Leuchtdioden, welche wir an den Mikrokontroller angeschlossen haben. Eine „0“ steht für LED aus, eine „1“steht für LED an. Bildlich sieht das ganze dann so aus:

AND1.png

Jetzt kommt unsere AND- Operation ins Spiel. Um eine LED einfach ein- oder auszuschalten können wir diese ja direkt im Programm ändern. Was aber, wenn beispielsweise LED 2 nur eingeschaltet bleiben soll, wenn diese bereits eingeschaltet ist?

Hier hilft die AND- Verknüpfung. Die LED´s, die eingeschaltet bleiben sollen, wenn sie bereits leuchten, bekommen in unserem zweiten 8 Bit- Wert eine 1, LED´s die leuchten, aber ausgeschaltet werden bekommen eine 0.
Und was ist mit den LED´s die zum Abfragezeitpunkt dunkel sind? Diese bleiben auf jeden Fall dunkel, da hier die AND- Bedingung nicht zutrifft.

Eine Tabelle soll diesen Sachverhalt noch einmal verdeutlichen:

AND-Tabelle.png

Merke: Bei der AND- Verknüpfung erscheint im Ergebnis nur eine 1, wenn sowohl der Ausgangswert, als auch der Vergleichswert 1 sind. Sind beide Werte 0 oder unterschiedlich, so ist auch das entsprechende Bit im Ergebnis 0!

Werfen wir noch einen Blick auf unsere LED´s nach der AND- Operation:

AND3.png

Lediglich LED 2 leuchtet noch, alle anderen LED´s sind jetzt dunkel.
Und so sieht der Codeschnipsel in BASCOM hierzu aus:

```
Dim Ausgangswert As Byte
Dim Ergebnis As Byte
Ausgangswert = &B00001110
Ergebnis = Ausgangswert AND &B00000010
Print Ergebnis
```

OR (Oder)

Analog hierzu wird die OR- Operation verwendet. Hier ist das entsprechende Ergebnisbit nur 1, wenn mindestens ein Bit der beiden Vergleichsbits gesetzt ist.
Auch hier wollen wir uns das Ganze noch einmal Schritt für Schritt anschauen. Um bei unserem Beispiel zu bleiben ist unser Ausgangswert wieder:

&B00001110

Zur Erinnerung hier unser LED- Display:

AND1.png

Hier kommt nun unser Vergleichswert für die OR- Operation:

&B10100110

In unserer Tabelle sieht das folgendermaßen aus:

OR- Tabelle.png

Merke: Bei der OR- Verknüpfung erscheint immer eine 1 im Ergebnisbit, außer beide Vergleichswerte sind 0

Nun sollte die Arbeitsweise der booleschen Algebra in Bezug auf die binären Operationen bereits verständlich geworden sein. Zur Vervollständigung unserer Beispiele folgen nun noch unser verändertes LED- Display, sowie der BASCOM-Beispielcodeschnipsel.

OR2.png

```
Dim Ausgangswert As Byte
Dim Ergebnis As Byte
Ausgangswert = &B00001110
Ergebnis = Ausgangswert OR &B10100110
Print Ergebnis
```

XOR (Exklusiv Oder)

Bleibt die XOR Operation, welche in gleicher Weise erklärt werden soll.
Zunächst wieder der bekannte Ausgangszustand:

&B00001110

AND1.png

Verwenden wir hier einmal den gleichen Vergleichswert wie für die OR- Operation, um den Unterschied zu sehen:

&B10100110

Die XOR- Operation verhält sich genau wie die OR Verknüpfung, jedoch mit dem entscheidenden Unterschied, dass das Ergebnis 0 und nicht 1 ist, wenn sowohl der Ausgangswert, als auch der Vergleichswert 1 ist. Am besten lässt sich dass wieder anhand der Tabelle erkennen:

XOR- Tabelle.png

Unser abweichendes LED- Display (LED 1 und 2 weichen von der OR- Verknüpfung ab):

LED4.png

Merke: Bei der XOR- Verknüpfung erscheint im Ergebnis nur eine 1, wenn entweder der Ausgangswert oder der Vergleichswert 1 sind. Sind beide Werte gleich, so ist das entsprechende Bit im Ergebnis 0!

Der BASCOM- Codeschnipsel sieht analog so aus:

```
Dim Ausgangswert As Byte
```

```
Dim Ergebnis As Byte
Ausgangswert = &B00001110
Ergebnis = Ausgangswert XOR &B10100110
Print Ergebnis
```

NOT (Nicht)

Auch wenn er ein bisschen aus der Reihe fällt, gehört der NOT- Befehl auch in diese Gruppe. Daher sei er hier der Vollständigkeit halber erklärt. Die NOT- Operation macht nichts anderes, als jedes einzelne Bit unseres Bytes zu invertieren; was 0 war wird 1 und was 1 war wird 0.

Zu Anfang wieder unser LED- Display mit einer neuen Ausgangsvariable:

&B10101010:

NOT1.png

So wenden wir den Befehl in BASCOM an:

```
Dim Ausgangswert As Byte
Dim Ergebnis As Byte
Ergebnis = &B10101010
Do
Waitms 500
Ergebnis = Not Ausgangswert
Waitms 500
Ausgangswert = Ergebnis
Loop
```

Nach der ersten NOT- Operation sieht die LED- Ausgabe so aus:

NOT2.png

Nach einer weiteren NOT- Operation erhalten wir wieder den Ausgangswert. Eine Darstellung in einer Tabelle ist hier überflüssig, denn das Verhalten lässt sich ganz einfach beschreiben:

Merke: Bei der NOT- Verknüpfung wird jedes einzelne Bit einer Bytevariable invertiert, also umgekehrt. War der Wert zu Anfang 0, ist er nach der Ausführung 1; war der Wert zu Anfang 1, ist er nachher 0