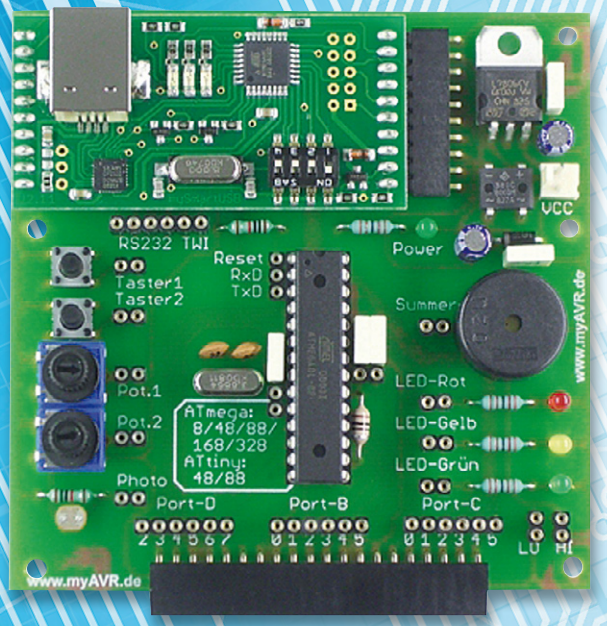
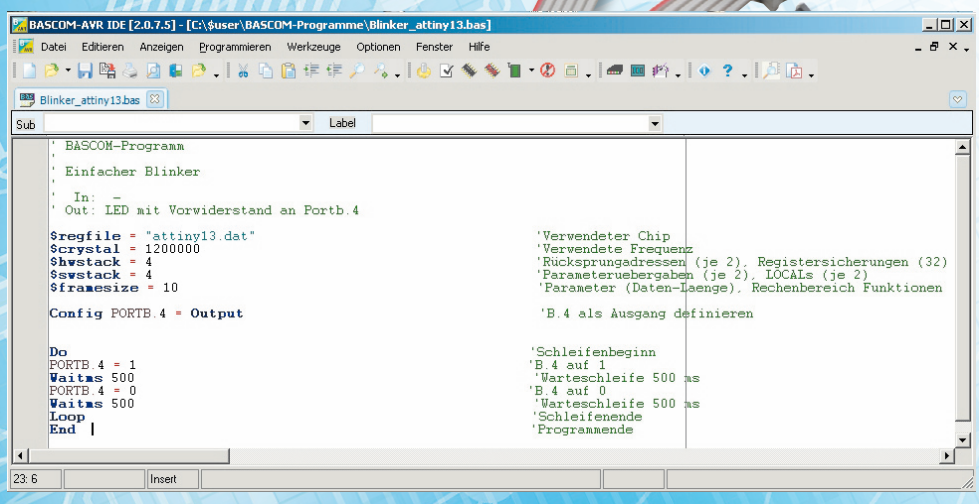
Mikrocontroller-Einstieg

Teil 3: Daten- und Programmstrukturen





mit BASCOM-AVR

Mit einem BASCOM-Programm wird beschrieben, welche Anweisungen vom Mikrocontroller während der Laufzeit ausgeführt werden sollen. Dafür gibt es eine Auswahl verschiedener Programmkonstrukte wie zum Beispiel Verzweigungen und Schleifen, welche durch BASCOM-Schlüsselwörter dargestellt werden. Außerdem benötigt man in einem BASCOM-Programm Möglichkeiten, Eingabewerte, Ausgabewerte oder Zwischenwerte speichern zu können. In diesem Teil unserer Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR" wird beschrieben, wie ein BASCOM-Programm grundsätzlich strukturiert sein sollte, wie Daten in Variablen gespeichert werden und welche grundlegenden Programmstrukturen es gibt.

Struktur eines BASCOM-Programms

Grundsätzlich sollte ein Programm aus 6 Blöcken bestehen:

1. Beschreibung des Programms
   (Zweck, Autor, Datum, Version, Copyright etc.)
2. Compiler-Anweisungen und Deklarationen
   ($REGFILE, $CRYSTAL, CONFIGs, DIMs ...)
3. Initialisierungen (Startwerte ...)
4. Endlosschleife mit dem Haupt-Verarbeitungsteil
5. Unterprogramme (ISRs, Subs, Functions)
6. Tabellen (Data für Read, Lookup)

Dabei sollten viele aussagefähige Kommentartexte sowie Einrückungen von Programmstrukturen verwendet werden. Beides ist zwar für den Compiler irrelevant, aber es erleichtert dem menschlichen Leser des Programms (auch dem Programmierer selbst) das Lesen und Verstehen des Programms.

Variablen

Variablen werden in der Programmierung benutzt, um Zahlen (oder Texte) im Speicher des Mikrocontrollers zu speichern. Im Verlauf des Programms wird mit den Variablen gearbeitet. Eine Variable hat immer einen eindeutigen Namen (bis zu 32 Zeichen lang) und nimmt Daten eines bestimmten Datentyps auf. Der Compiler (oder im Ausnahmefall der Programmierer) legt fest, an welcher Stelle im Speicher die Variable beginnt. Durch den Datentyp wird festgelegt, was für Werte die Variable aufnehmen kann und wie viel Speicherplatz für die Variable reserviert werden muss.

Wird in BASCOM zum Beispiel eine **Byte-Variable** deklariert (DIMensioniert) mit

```
DIM Meine_Variable1 AS Byte
```

dann reserviert der BASCOM-Compiler im Speicher des Mikrocontrollers Platz für eine Byte-Variable mit dem Namen Meine_Variable1. Ein Byte besteht aus 8 Bit. Ein Bit kann entweder 0 oder 1 sein. In den 8 Speicherzellen der Variablen Meine_Variable1 kann also eine Folge von 8 Nullen und Einsen gespeichert werden, wie in Bild 1 zu sehen ist. Die Bits einer Variablen werden von rechts nach links mit 0 bis 7 gezählt. Man kann durch Anhängen eines Punktes und der Bitnummer einzelne Bits einer Variablen ansprechen. Zum Beispiel hat das Bit Meine_Variable1.2 hier den Wert 1.

Mit den 8 Bit eines Bytes lassen sich die Dezimalzahlen 0 bis 255 darstellen. Jedes Bit stellt dabei die Wertigkeit 2^n dar, wobei n = 0 bis 7 sein kann. In Bild 1 also $2^0 + 2^1 + 2^2 = 1 + 2 + 4 = 7$.

„Befüllt" wird eine Variable durch Zuweisung einer Zahl wie z. B. Meine_Variable1 = 7 oder Meine_Variable1 = &b00000111 (Binärdarstellung) oder Meine_Variable1 = &h07 (Hexadezimaldarstellung).

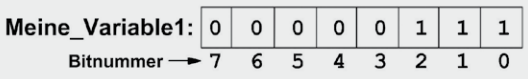


Bild 1: Byte-Variable

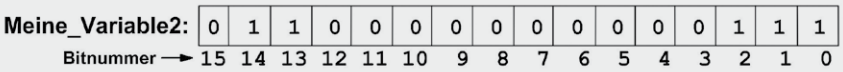


Bild 2: Word-Variable

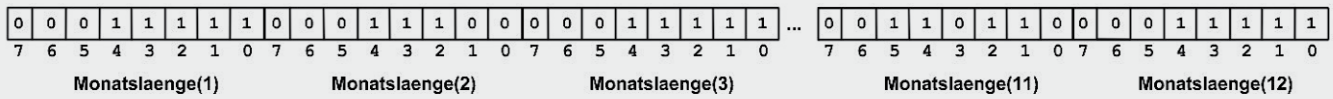


Bild 3: Array

Wenn größere Zahlen als 255 dargestellt werden sollen, dann kann man in BASCOM zum Beispiel eine **Word-Variable** deklarieren mit:

DIM Meine_Variable2 AS Word

Im Speicher werden dann 2 Byte (16 Bit) reserviert und die Speicheradresse bekommt den Namen Meine_Variable2. Die 16 Bit haben die Nummern 0 bis 15 (Bild 2). Eine Word-Variable kann Zahlen von 0 bis 65.535 darstellen.

Zeichenketten benötigen im Speicher 1 Byte Platz je Zeichen plus 1 Byte für das Ende-Kennzeichen. Beispiel einer String-Deklaration:

DIM Vorname AS STRING

Zuweisung zum Beispiel:

Vorname = "Stefan"

Ein **Array** ist eine Aneinanderreihung gleichartiger Variablen, die jeweils mit einem Index angesprochen werden können (Bild 3). Standardmäßig ist der niedrigste Index in BASCOM 1.

Beispiel-Deklaration:

DIM Monatslaenge(12) AS BYTE

Beispiel für eine Zuweisung:

Monatslaenge(1) = 31

Eine Übersicht aller BASCOM-Datentypen zeigt Tabelle 1.

Die linke Spalte (Typ) ist der Variablentyp, der im BASCOM-DIM-Befehl verwendet wird. Der Wertebereich gibt an, welche Werte eine Variable des jeweiligen Typs aufnehmen kann. In der rechten Spalte sieht man, wie viele Bytes im SRAM durch die Variable belegt werden.

Ein Testprogramm zum Thema Variablen könnte so aussehen:

```
' BASCOM-Programm
'
' Variablen in BASCOM
'
$sim                                'Fuer Simulation. Sonst auskommentieren.
$regfile  = "M88def.dat"            'Verwendeter Chip
$crystal = 1000000                  'Verwendete Frequenz
$hwstack = 40                       'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                       'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                     'Parameter (Daten-Laenge), Rechenbereich Funktionen

Dim Meine_variable1 As Byte         'Deklariert eine Byte-Variable
Dim Meine_variable2 As Word         'Deklariert eine Word-Variable
Dim Vorname As String * 20          'Deklariert eine Zeichenkettenvariable für 20 Zeichen
Dim Monatslaenge(12) As Byte        'Deklariert ein Array mit 12 Byte-Variablen

Meine_variable1 = 7                 'Zuweisung einer 7 zur Variablen Meine_variable1

Meine_variable2 = &B0110000000000111    'Hexadezimal: &H6007  Dezimal: 24583

Vorname = "Stefan"                  'Zuweisung der Zeichenkette "Stefan" zur Variablen Vorname

Monatslaenge(1) = 31                'Zuweisung der Zahl 31 zur Array-Variablen Monatslaenge(1)
Monatslaenge(11) = 30               'Zuweisung der Zahl 30 zur Array-Variablen Monatslaenge(11)
Monatslaenge(12) = 31               'Zuweisung der Zahl 31 zur Array-Variablen Monatslaenge(12)

Do                                  'Schleifenbeginn
Loop                                'Schleifenende
End                                 'Programmende
```

Dieses BASCOM-Programm ist vollständig und kompilierbar – auch wenn hier nicht viel an Verarbeitungslogik programmiert ist.

Erläuterungen zum Programm:

Nach den Kommentarzeilen mit dem Programmtitel und den Compiler-Direktiven zu Prozessortyp, Taktgeschwindigkeit und Speicheraufteilung werden einige Variablen mit DIM-Befehlen deklariert und im Anschluss durch Zuweisungen mit Werten gefüllt. Die mikrocontrollertypische Endlosschleife mit DO – LOOP ist leer, da hier zunächst der Schwerpunkt die Variablen-Dimensionierung ist. Die Compiler-Direktive $SIM wird angegeben, wenn ein BASCOM-Programm für den Simulator kompiliert wird. Wenn für das Brennen auf den Mikrocontroller kompiliert wird, ist $SIM zu entfernen oder auszukommentieren.

Dieses kleine Programm lässt sich gut im BASCOM-Simulator nachvollziehen, der mit F2 (oder Programmieren – Simulieren) aufgerufen wird. Im Simulator kann man im oberen Teil die Variableninhalte anzeigen lassen, mit F8 (Step into) schrittweise durch das Programm gehen und dabei rechts ein Abbild des SRAM-Speichers sehen (Bild 4). Unterhalb des Programmtextes sieht man im Simulator in der Statuszeile die Anzahl der Programmzyklen und kann diese nach Rechtsklick auf die Statuszeile auch auf null setzen.

Tabelle 1

Datentypen in BASCOM und Deklarations-Syntax

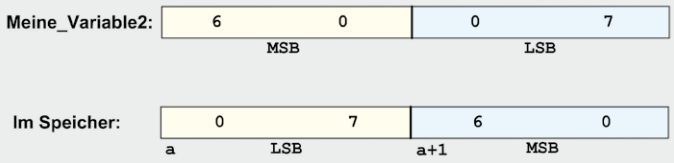
Typ	Wertebereich	Speicherbedarf
Bit	0...1	1 Bit
Byte	0...255	1 Byte (8 Bit)
Word	0...65535	2 Byte
Integer	-32768...+32767	2 Byte
Dword	0...4294967295	4 Byte
Long	-2147483648...2147483647	4 Byte
Single	$1{,}5*10^{-45}$...$3{,}4*10^{38}$	4 Byte
Double	$5*10^{-324}$...$1{,}7*10^{308}$	8 Byte
String*n	1...254 Zeichen	(n+1) Byte; mindestens 2 Byte, max. 254 Byte

DIM variable AS typ
*DIM variable AS STRING*n n=1...254*
DIM variable(n) AS typ n=1...65535

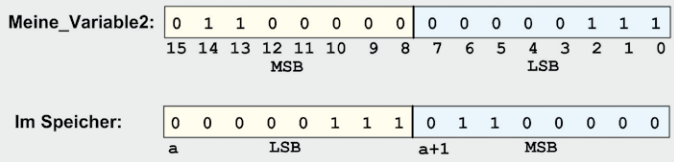
Elektronikwissen

Endianess (Byte-Reihenfolge)

AVR-Mikrocontroller verwenden die Little-Endian-Byte-Reihenfolge beim Abspeichern von Variablen im Speicher. Das bedeutet, dass das niedrigstwertige Byte (LSB = Least Significant Byte) einer mehrere Bytes langen Zahl (Word, Dword ...) an der kleineren Speicheradresse gespeichert wird. Bei einer Word-Zahl &h6007 (hexadezimal 6007) wird also zuerst die 07 im Speicher an einer Adresse a abgelegt und eine Adresse weiter (a+1) die 60. Vergleichbar ist das mit der deutschen Datumsschreibweise, bei der erst der Tag und dann der Monat genannt wird.



In Binärdarstellung sieht das so aus:



Im Gegensatz dazu gibt es die Big-Endian-Byte-Reihenfolge, bei der das höchstwertige Byte (MSB = Most Significant Byte) an der kleineren Speicheradresse gespeichert wird – vergleichbar mit der englischen Datumsschreibweise, bei der erst der Monat und dann der Tag genannt wird (April 1). Im Normalfall muss man sich bei Verwendung von BASCOM nicht um die Byte-Reihenfolge kümmern, aber wenn zum Beispiel mit Overlay-Stukturen, EEPROM-Zugriffen oder serieller Übertragung gearbeitet wird, dann sollte man dieses Thema kennen.

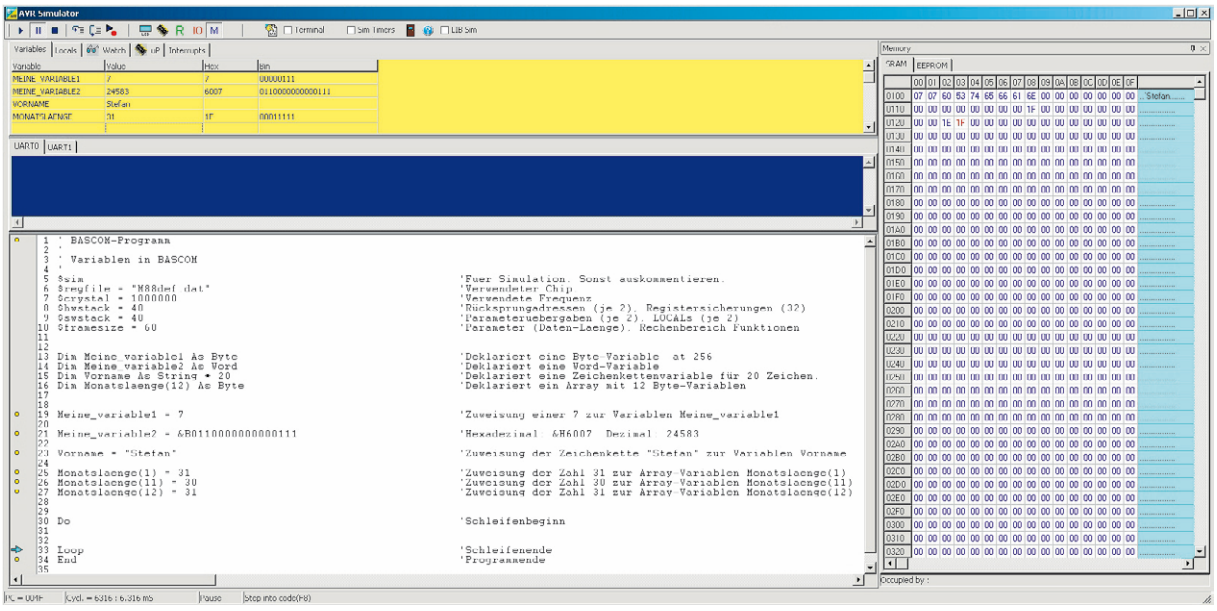


Bild 4: BASCOM-Simulator mit Variablen und Speicherabbild

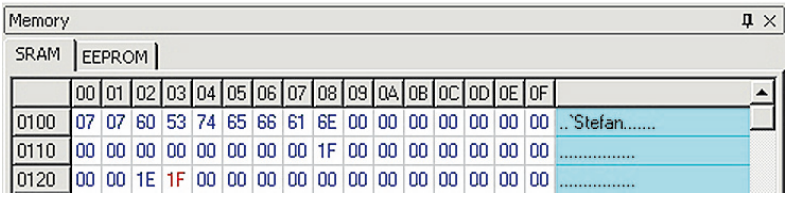


Bild 5: Speicherabbild im Simulator

In Bild 5 wird der SRAM-Speicher in der Simulatordarstellung näher betrachtet.

Man sieht an Adresse &h100 (dezimal 256) die Variable Meine_Variable1 mit dem Inhalt 7. Danach folgt die Word-Variable (2 Byte lang) Meine_Variable2 mit dem Inhalt hexadezimal 6007 in Little-Endian-Speicher-Reihenfolge (siehe „Elektronikwissen").

Ab Speicheradresse &h103 ist die Stringvariable Vorname – hier mit dem Inhalt "Stefan" – Zeichen für Zeichen jeweils im ASCII-Code und mit einem Abschlussbyte gespeichert. Für die String-Variable wurden 20 Byte plus ein Byte Platz reserviert. Deshalb ist die folgende Variable Monatslaenge(1) mit dem Inhalt 31 (hexadezimal 1F) an Speicheradresse &h118 gespeichert. Die Inhalte von Monatslaenge(11) und Monatslaenge(12) sieht man nach einer Reihe unbeschriebener Speicherzellen als hexadezimal 1E und 1F, was dezimal 30 bzw. 31 entspricht. Durch Einsetzen anderer Werte bzw. Dimensionierung anderer Variablen kann man nach Kompilierung im Simulator sehr schön ein Gefühl für die Abspeicherung von Variablenwerten bekommen.

Programmstrukturen

Durch BASCOM-Kontrollstrukturen wird die Ablauflogik eines BASCOM-Programms beschrieben. Für den Programmentwurf sind Struktogramme (siehe „Elektronikwissen") eine wertvolle Hilfe. Sie sollen im Folgenden (in nicht ganz strenger Form) verwendet werden, da sie zu einer strukturierten Denkweise zwingen.

DO – LOOP: Endlosschleife = Wiederholung ohne Bedingungsprüfung

In Bild 6 ist ein einfaches Blinkprogramm als Struktogramm dargestellt. Ein Rechteck symbolisiert eine einzelne Anweisung bzw. einen Anweisungsblock. Eine sequenzielle Anweisungsfolge wird durch mehrere Rechtecke hintereinander dargestellt.

Die farblich unterlegte Klammer stellt eine Schleife dar, die bedingungslos immer wieder durchlaufen wird. Die zwei Anweisungen TOGGLE (Umschalten des Port-Ausgangs) und WAIT (1 Sekunde Warten) werden immer wieder abgearbeitet.

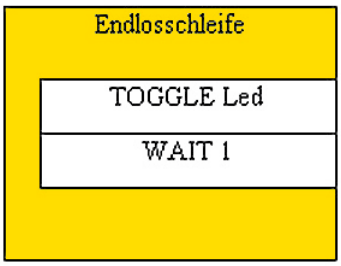


Bild 6: DO – LOOP (Endlosschleife)

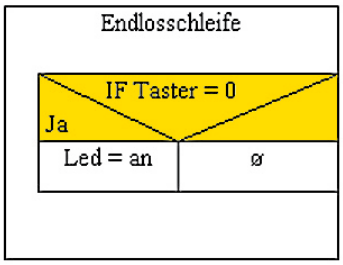


Bild 7: IF – THEN (bedingte Verzweigung)

Elektronikwissen

Struktogramme

Ein wichtiger Teil eines Mikrocontroller-Projektes ist die Beschreibung der Schritte, die der Mikrocontroller abarbeiten soll. Im Rahmen einer strukturierten Programmierung eignen sich für die grafische Darstellung von Programmstrukturen sehr gut sogenannte Struktogramme, welche von Isaac Nassi und Ben Shneiderman entwickelt wurden und daher auch als Nassi-Shneiderman-Diagramme bezeichnet werden. Die Symbole sind in der DIN 66261 beschrieben und eignen sich sehr gut für die anschließende Umsetzung in ein BASCOM-Programm.

Wichtige Konstrukte in Struktogrammen sind

- Anweisung
- Bedingungslose Schleife (Endlosschleife)
- Bedingte Verzweigung
- Zählergesteuerte Schleife
- Schleife mit Bedingungsprüfung am Ende
- Schleife mit Bedingungsprüfung am Anfang
- Fallunterscheidung

Elektronikwissen

Komponenten von BASCOM

Die BASCOM-Entwicklungsumgebung bietet alle Komponenten, die für die Entwicklung und das Testen eines BASCOM-Programms benötigt werden:

1.) **Editor:** Zur Eingabe des BASCOM-Programmtextes (Quellcode) mit ähnlichen Funktionen wie ein Textprogramm. Mit automatischer Farbunterlegung von BASCOM-Befehlswörten, kontextsensitiver Ergänzung und Hilfe-Aufruf (mit F1), je nachdem wo die Schreibmarke steht.

2.) **Compiler:** Prüft die Syntax des Programmtextes, zeigt gegebenenfalls Fehlermeldungen an und erzeugt im fehlerfreien Fall ausführbaren Maschinencode und weitere Output-Dateien.

3.) **Simulator:** Bietet die Möglichkeit, das BASCOM-Programm ohne Mikrocontroller-Hardware am PC Schritt für Schritt zu durchlaufen und dabei Speicherinhalte, Portzustände, LCD-Ausgaben usw. anzeigen zu lassen. Eingaben können ebenso simuliert werden. Auch die Anzahl der Prozessorzyklen für Programmteile kann gemessen werden.

4.) **Terminalprogramm:** Ermöglicht bei serieller Verbindung zum Mikrocontroller die Eingabe und Ausgabe von Zahlen und Texten am PC.

5.) **Programmer:** Mit diesem Teil von BASCOM wird das kompilierte BASCOM-Programm (im Hex-Format) auf den angeschlossenen Mikrocontroller übertragen – „gebrannt".

6.) **Hilfe:** Die BASCOM-Hilfe umfasst gut 1000 Seiten und bietet neben grundlegenden Themen auch eine detaillierte Befehlsreferenz und sehr viele Beispielprogramme.

Beim Entwickeln eines BASCOM-Programms werden die Struktogramme zu BASCOM-Konstrukten umgesetzt und die notwendigen Details ergänzt.

```
' BASCOM-Programm
'
' Endlosschleife mit DO - LOOP
' LED blinkt
'
' In: -
' Out: LED mit Vorwiderstand an C.0
'
$regfile  = "M88def.dat"                'Verwendeter Chip
$crystal = 1000000                      'Verwendete Frequenz
$hwstack = 40                           'Ruecksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                           'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                         'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portc.0 = Output                 'C.0 als Ausgang definieren
Led Alias Portc.0                       'Aliasname Led für PORTC.0

Do                                      'Schleifenbeginn
  Toggle Led                            'Led (Portc.0) von an zu aus bzw. umgekehrt
  Wait 1                                'warten
Loop                                    'Schleifenende
End                                     'Programmende
```

Erläuterungen:
Das Schleifenkonstrukt DO – LOOP umschließt die Anweisungen, die bedingungslos immer wieder ausgeführt werden. Eine derartige „Endlosschleife" ist typisch für Mikrocontrollerprogramme und fast immer vorhanden.

Mit Led ALIAS Portc.0 wird übrigens ein Aliasname für Portc.0 vergeben, weil sich dadurch das Programm besser lesen lässt. Der TOGGLE-Befehl bewirkt, dass aus 1 am Ausgangspin 0 wird und umgekehrt (to toggle = zwischen zwei Zuständen umschalten).

IF – THEN: Bedingte Verzweigung
Das Struktogramm in Bild 7 stellt innerhalb der Endlosschleife ein Konstrukt dar, bei dem eine Bedingung geprüft wird (wenn der Inhalt von Taster gleich 0 ist ...) und in Abhängigkeit von der Antwort eine Anweisung ausgeführt wird (... dann setze den Inhalt von Led auf den Wert der Konstanten an). Ist die Bedingung nicht erfüllt, dann wird keine zum Abfragekonstrukt gehörige Anweisung ausgeführt.

Die Umsetzung vom Struktogramm zum BASCOM-Programm geschieht nicht immer Wort für Wort. Das Struktogramm ist eher ein Hilfsmittel beim Programmentwurf. Streng genommen sollte ein Struktogramm unabhängig von der benutzten Programmiersprache sein und es sollten eigentlich keine Schlüsselwörter der Ziel-Programmiersprache verwendet werden. Da es hier um die Vorstellung der BASCOM-Konstrukte geht, sind die Struktogramme sehr eng an BASCOM angelehnt.

```
' BASCOM-Programm
'
' Bedingte Verzweigung mit IF - THEN
' Wenn Taster gedrueckt dann LED einschalten
'
'  In: Taster an D.2
' Out: LED mit Vorwiderstand an C.0
'
$regfile  = "M88def.dat"                'Verwendeter Chip
$crystal = 1000000                      'Verwendete Frequenz
$hwstack = 40                           'Ruecksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                           'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                         'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portd.2 = Input                  'D.2 als Eingang definieren fuer Taster
Portd.2 = 1                             'Internen Pullup-Widerstand intern einschalten
Taster Alias Pind.2                     'Aliasname Taster für PIND.2
```

```
Config Portc.0 = Output                  'C.0 als Ausgang definieren fuer LED
Led Alias Portc.0                        'Aliasname Led für PORTC.0

Const Gedrueckt = 0                      'Konstante Gedrueckt fuer 0
Const An = 1                             'Konstante An fuer 1

Do
  If Taster = Gedrueckt Then Led = An    'Wenn Taster an D.2 auf Gnd schaltet: LED an
Loop
End
```

Erläuterungen zum Programm:
Innerhalb der typischen Endlosschleife (DO – LOOP) wird bei jedem Schleifendurchlauf geprüft, ob der Taster gedrückt ist. Wenn das der Fall ist, dann wird die LED an Portc.0 eingeschaltet (logisch 1 an Portc.0).

Mit CONST kann man feste Werte vorgeben. Der Compiler setzt an jeder Stelle, an der der Konstantenname vorkommt, den Konstantenwert ein. Konstanten verbrauchen keinen zusätzlichen Speicherplatz im Programmspeicher oder im Variablenspeicher.

Da hier bei Erfüllung der Bedingung nur EINE Anweisung ausgeführt werden soll, wurde hier das EINZEILIGE IF-Konstrukt verwendet. Wenn mehrere Anweisungen ausgeführt werden sollen, muss hinter THEN eine neue Zeile begonnen und die Anweisungsfolge mit END IF abgeschlossen werden.

IF – THEN – ELSE: bedingte Verzweigung mit Alternative
Da das Programm mit der einfachen Bedingungsabfrage die Eigenart hat, dass die LED zwar beim Drücken des Tasters eingeschaltet, aber nie wieder ausgeschaltet wird, sieht man in Bild 8 die Verzweigung mit Alternative. Die LED soll nur genau so lange an sein, wie der Taster auch tatsächlich gedrückt wird. In der BASCOM-Umsetzung sieht man die Bedingungsabfrage (IF) und die beiden Möglichkeiten THEN und ELSE.

```
' BASCOM-Programm
'
' Bedingte Verzweigung mit IF - THEN - ELSE
' Wenn Taster gedrueckt dann LED einschalten sonst ausschalten.
'
'  In: Taster an D.2
' Out: LED mit Vorwiderstand an C.0
'
$regfile  = "M88def.dat"                 'Verwendeter Chip
$crystal = 1000000                       'Verwendete Frequenz
$hwstack = 40                            'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                            'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                          'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portd.2 = Input                   'D.2 als Eingang definieren
Portd.2 = 1                              'Internen Pullup-Widerstand einschalten
Taster Alias Pind.2                      'Aliasname Taster für PIND.2

Config Portc.0 = Output                  'C.0 als Ausgang definieren
Led Alias Portc.0                        'Aliasname Led für PORTC.0

Const Gedrueckt = 0                      'Konstante Gedrueckt fuer 0
Const An = 1                             'Konstante An fuer 1
Const  Aus = 0                           'Konstante Aus fuer 0

Do
  If Taster = Gedrueckt Then             'Wenn Taster an D.2 auf Gnd schaltet..
    Led = An                             'LED anschalten
  Else                                   'sonst ..
    Led = Aus                            'LED ausschalten
  End If                                 'Ende des IF-Konstruktes
Loop
End
```

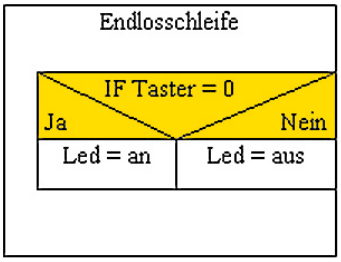


Bild 8: IF – THEN – ELSE (bedingte Verzweigung mit Alternative)

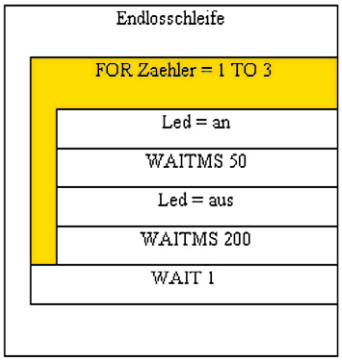


Bild 9: FOR – NEXT (zählergesteuerte Schleife)

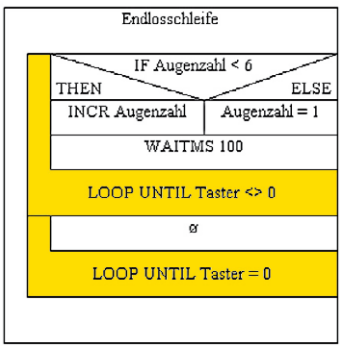


Bild 10: DO – LOOP UNTIL (fußgesteuerte Schleife)

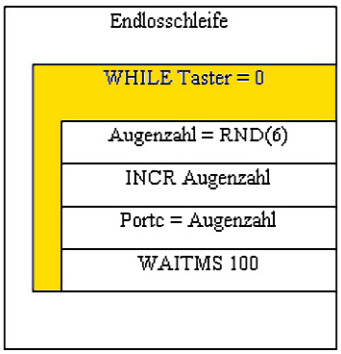


Bild 11: WHILE – WEND (kopfgesteuerte Schleife)

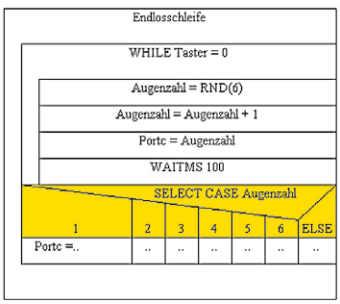


Bild 12: SELECT – CASE (Fallunterscheidung)

Erläuterungen:
Wenn die Bedingung der IF-Abfrage (Taster = gedrueckt) erfüllt ist, dann wird die Anweisung hinter THEN ausgeführt (Led = An), sonst die Anweisung hinter ELSE (Led = Aus). Abgeschlossen wird das IF-Konstrukt mit END IF.

FOR – NEXT: zählergesteuerte Schleife
Mit dem Struktogramm in Bild 9 soll ein Programm entworfen werden, bei dem eine LED drei Mal blitzt – es zeigt eine Schleife, die drei Mal durchlaufen wird.

In BASCOM wird ein FOR–NEXT-Konstrukt für eine Schleife benutzt, die eine bestimmte Anzahl von Durchläufen haben soll.

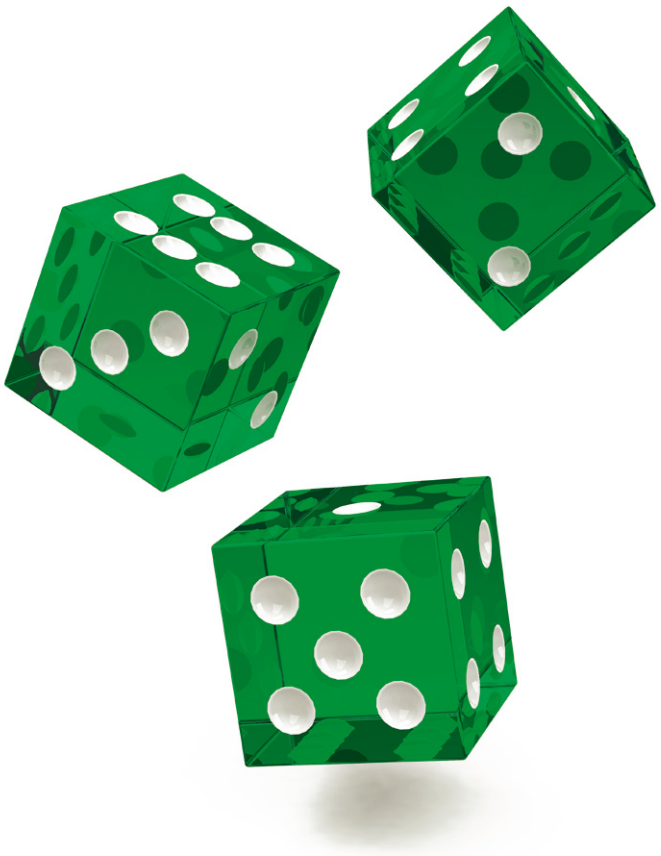
```
' BASCOM-Programm
'
' Zaehlergesteuerte Schleife mit FOR - NEXT
' LED blitzt jeweils 3 mal
'
'  In: -
' Out: LED mit Vorwiderstand an C.0
'
$regfile  = "M88def.dat"              'Verwendeter Chip
$crystal = 1000000                    'Verwendete Frequenz
$hwstack = 40                         'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                         'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                       'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portc.0 = Output               'C.0 als Ausgang definieren fuer LED
Led Alias Portc.0                     'Aliasname Led für PORTC.0

Dim Zaehler As Byte                   'Zaehlvariable Zaehler als Byte-Variable

Const An = 1                          'Konstante An fuer 1
Const Aus = 0                         'Konstante Aus fuer 0

Do
  For Zaehler = 1 To 3                'Zaehlschleife von 1 bis 3
    Led = An                          ' 3 mal blitzen
    Waitms 50
    Led = Aus
    Waitms 200
  Next                                'Ende der Zaehlschleife
  Wait 1
Loop
End
```

Erläuterungen:
Mit FOR – NEXT werden alle Anweisungen umklammert, die mehrfach ausgeführt werden sollen. Hinter FOR steht die Regel für die Anzahl der Durchläufe (Zaehler = 1 TO 3).

DO – LOOP UNTIL: Schleife mit nachfolgender Bedingungsprüfung
Das Struktogramm in Bild 10 zeigt ein Schleifenkonstrukt, bei dem im Anschluss an Anweisungen eine Schleifenbedingung geprüft wird und die Schleife bei Erfüllen der Bedingung beendet wird.

Durch das BASCOM-Programm wird ein elektronischer Würfel realisiert, bei dem „gewürfelt" wird, bis der Taster nicht mehr gedrückt wird (Taster <> 0 wobei <> für ungleich steht). Dann wird so lange eine leere Schleife durchlaufen (="gewartet"), bis die Taste gedrückt wird (Taster = 0). Wie in der gesamten Artikelserie wird hier von einem Taster ausgegangen, der in geschlossenem Zustand eine Verbindung zwischen dem Eingangspin zu 0V (GND) herstellt.

```
' BASCOM-Programm
'
' Fußgesteuerte Schleifen mit DO - LOOP UNTIL
' Wuerfelprogramm: Dualcode-Würfel.
'
'  In: Taster an D.2
' Out: LEDs mit je 1 Vorwiderstand an C.0, C.1, C.2
'
$regfile  = "M88def.dat"              'Verwendeter Chip
$crystal = 1000000                    'Verwendete Frequenz
$hwstack = 40                         'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                         'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                       'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portd.2 = Input                'D.2 als Eingang definieren
Portd.2 = 1                           'Internen Pullup-Widerstand einschalten
Taster Alias Pind.2                   'Aliasname Taster für PIND.2

Config Portc = Output                 'PortC als Ausgang definieren

Const Gedrueckt = 0                   'Konstante Gedrueckt fuer 0

Dim Augenzahl As Byte
Augenzahl = 1

Do
  Do                                  'Wuerfeln..
    If Augenzahl < 6 Then
      Incr Augenzahl                  'Incr = Inkrementieren = Um 1 erhöhen
    Else
      Augenzahl = 1
    End If
    Portc = Augenzahl
    Waitms 100
  Loop Until Taster <> Gedrueckt      ' .. bis Taste losgelassen wird
  Waitms 50

  Do                                  '  "Warten" ..
  Loop Until Taster = Gedrueckt       ' ..bis Taste gedrueckt wird
  Waitms 50
Loop
End
```



Erläuterungen:
Zwischen DO und LOOP stehen jeweils die Anweisungen, die mehrfach durchlaufen werden, bis die Bedingung hinter UNTIL erfüllt ist.

WHILE – WEND: Schleife mit vorausgehender Bedingungsprüfung
Bei einer „kopfgesteuerten" Schleife wie in Bild 11 wird am Anfang der Schleife die Bedingung geprüft. Eventuell wird eine WHILE-Schleife nicht ein einziges Mal durchlaufen.

```
' BASCOM-Programm
'
' Kopfgesteuerte Schleife mit WHILE - WEND
' Wuerfelprogramm: Dualcode-Würfel.
'
'  In: Taster an D.2
' Out: LEDs mit je 1 Vorwiderstand an C.0, C.1, C.2
'
$regfile  = "M88def.dat"              'Verwendeter Chip
$crystal = 1000000                    'Verwendete Frequenz
$hwstack = 40                         'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                         'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                       'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portd.2 = Input                 'D.2 als Eingang definieren
Portd.2 = 1                           'Internen Pullup-Widerstand einschalten
Taster Alias Pind.2                   'Aliasname Taster für PIND.2

Config Portc = Output                 'PortC als Ausgang definieren

Dim Augenzahl As Byte

Do
  While Taster = 0                    'Solange Taste gedrueckt...
    Augenzahl = Rnd(6)                'Zufallszahl zwischen 0 und 5
    Augenzahl = Augenzahl + 1         'Zahlen 1 bis 6
    Portc = Augenzahl
    Waitms 100
  Wend                                ' .. Wuerfeln
Loop
End
```

Erläuterungen:
Die WHILE-Schleife wird nur ausgeführt, wenn die Bedingung (Taster = 0) erfüllt ist. Vor jedem Schleifendurchlauf wird erneut geprüft, ob es einen weiteren Durchlauf gibt.

SELECT – CASE: Fallunterscheidung
Für den Entwurf eines eleganteren Würfel-Programms, bei dem die Augenzahl nicht binär dargestellt wird, sondern durch entsprechende Leuchtdioden, wird die Augenzahl in Bild 12 durch eine Fallunterscheidung abgefragt. (Es wurden im Struktogramm nicht alle Zweige der Fallunterscheidung ausformuliert, da es um den Entwurf geht.)

```
' BASCOM-Programm
'
' Fallunterscheidung mit SELECT - CASE
' Wuerfelprogramm
'
'  In: Taster an D.2
' Out: LEDs mit je 1 Vorwiderstand an C.0, C.1, C.2, C.3, C.4, C.5
'
$regfile  = "M88def.dat"              'Verwendeter Chip
$crystal = 1000000                    'Verwendete Frequenz
$hwstack = 40                         'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                         'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                       'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Portd.2 = Input                 'D.2 als Eingang definieren
Portd.2 = 1                           'Internen Pullup-Widerstand einschalten
```

```
Taster Alias Pind.2                 'Aliasname Taster für PIND.2

Dim Augenzahl As Byte

Do
  While Taster = 0                  'Solange Taste gedrueckt...
    Augenzahl = Rnd(6)              'Zufallszahl zwischen 0 und 5
    Augenzahl = Augenzahl + 1       'Zahlen 1 bis 6
    Portc = Augenzahl
    Waitms 100
  Wend                              ' .. Wuerfeln

Select Case Augenzahl               'Fallunterscheidung
  Case 1 : Portc = &B0000_0001
  Case 2 : Portc = &B0000_0010
  Case 3 : Portc = &B0000_0100      'Mit Unterstrich übersichtliche Binärdarstellung
  Case 4 : Portc = &B0000_1000      'Werte der Bits nach Belieben anpassen
  Case 5 : Portc = &B0001_0000
  Case 6 : Portc = &B0010_0000
  Case Else : Portc = &B1111_1111   'In jedem anderen Fall
End Select                          'Ende Fallunterscheidung

Loop
End
```

Erläuterungen:
Während des Würfelvorgangs wird die Augenzahl als Binärzahl angezeigt. Wird die Schleife nicht noch einmal ausgeführt, weil Taster nicht mehr 0 ist, dann wird durch das SELECT-Konstrukt die Augenzahl ausgewertet und dem Ausgangsport Portc ein Muster zugewiesen (eine bestimmte LED wie hier oder eine Anzahl LEDs oder die übliche Würfel-Anzeige der Augenzahl). Mit der BASCOM-Funktion RND generiert das Programm Zahlen zwischen 0 und 5 in „zufälliger" Reihenfolge.

Ausblick

In diesem Teil der BASCOM-Artikelserie wurden Programmstrukturen dargestellt, durch die in einem BASCOM-Programm die Anweisungen in einer vorbestimmten Reihenfolge abgearbeitet werden. Da es bei der Mikrocontroller-Programmierung externe und interne Ereignisse gibt, die mit hoher Priorität verarbeitet werden sollen und die daher den Programmablauf kurzzeitig unterbrechen dürfen, wird im nächsten Teil der Artikelserie das wichtige Konzept der Interrupts dargestellt. ELV

Weitere Infos:

- Stefan Hoffmann: Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM. Systematische Einführung und Nachschlagewerk mit vielen Anregungen. ISBN 978-3-8391-8430-1
- www.bascom-buch.de
- www.mcselec.com
- www.atmel.com

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics	–	–
Atmel AVRISP mkII Programmer	JV-10 03 55	€ 39,95
oder myAVR Board MK2	JV-10 90 00	€ 49,–
Netzteil für myAVR Board MK2	JV-10 90 01	€ 6,95
ATtiny13	JV-10 03 39	€ 1,95
ATmega8	JV-05 29 71	€ 3,20
ATmega88	JV-10 07 62	€ 3,95
100-nF-Kondensator	JV-10 03 17	€ 0,08
Batteriehalter für 3x Mignon	JV-08 15 30	€ 0,75
Batterieclip für 9-V-Block-Batterie	JV-08 01 28	€ 0,30
BASCOM-Buch	JV-10 90 02	€ 54,–

Preisstellung Februar 2013 – aktuelle Preise im Web-Shop