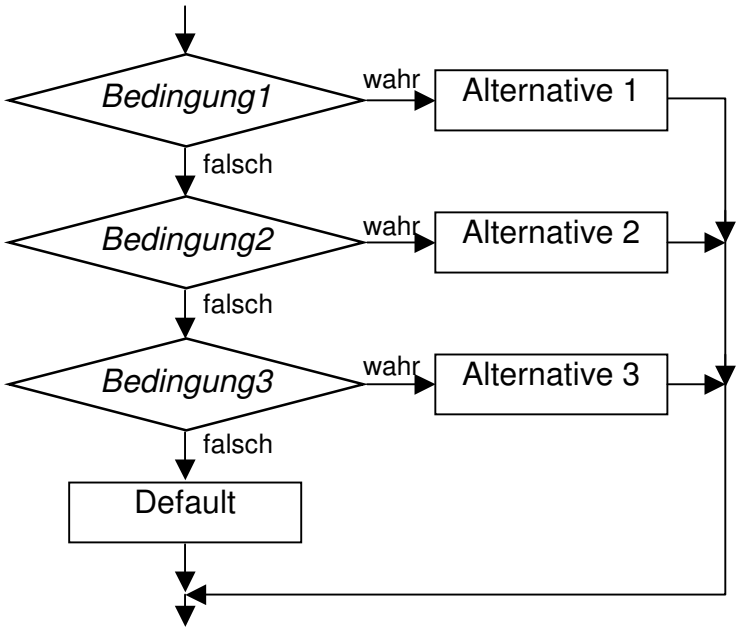
Friedrich-Ebert-Schule Esslingen	µC-Steuerungen mit Visual Basic	Name:
2.2.1	Kontrollstrukturen	Datum:

Obwohl mit Visual Basic objektorientiert programmiert wird, sind alle Elemente einer prozeduralen Programmiersprache enthalten. Dazu zählen insbesondere Abfragen (Verzweigungen), Schleifen und Prozeduren:

Verzweigungen

- **If-Then-Else**

Mit einer If-Then-Else Abfrage kann das Programm abhängig von einer Bedingung (z.B. a >= 5) alternativen Programmcode ausführen.



Syntax:

If *Bedingung1* **Then**
  Alternative1
ElseIf *Bedingung2* **Then**
  Alternative2
ElseIf *Bedingung3* **Then**
  Alternative3
Else
  Default
End If

Bsp.: Einzeilige Version von If-Then

```
If ia < 0.1 Then MsgBox("a ist kleiner 0.1")
```

- **Select-Case**

Select-Case ist eine Mehrfachverzweigung. Es ist möglich auf einfache Weise mehrere Alternativen abzuprüfen. Die Vergleichsbedingung wird dabei unterteilt. Hinter **Select Case** steht die Testvariable. Hinter jedem **Case** steht ein eigner, oder mehrere Vergleichswerte. Select-Case ist nicht so flexibel wie If-Then-ElseIf, oft aber übersichtlicher.

Syntax:

Select Case *n*
Case *1*
  Alternative 1 (wenn n = 1)
Case *2,3,4,5*
  Alternative 2 (wenn n = 2,3,4 oder 5)
Case *6 To 50*
  Alternative 3 (wenn n = 6...50)
Case *Is < 70*
  Alternative 4 (wenn n < 70)
Case Else
  Default
End Select

Schleifen

- **While-Schleifen**

Die While-Schleife wird wiederholt, solange die Bedingung wahr ist !

Syntax:
While Bedingung
  Anweisungen
End While

Bsp.: Wartet bis Umschalttaste losgelassen wird.

```
While My.Computer.Keyboard.ShiftKeyDown
    'Hier stehen die Aktionen die ausgeführt werden,
    'wenn die Umschalttaste gedrückt ist!
End While
```

- **Do-Schleifen**

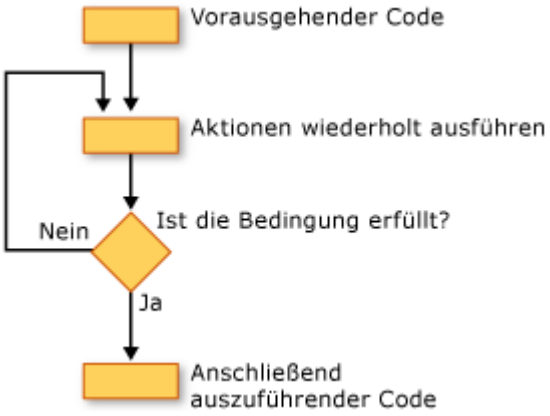
Bei den Do-Schleifen kann die Abfrage der Bedingung sehr flexibel gehandhabt werden.

Syntax:
1) Abweisende Schleife (= Überprüfung der Bedingung vor dem Schleifendurchlauf)

Do { **While** | **Until** } *Bedingung*
Anweisungen
Loop

2) Annehmende Schleife (= Überprüfung der Bedingung nach dem Schleifendurchlauf)

Do
Anweisungen
Loop { **While** | **Until** } Bedingung



- **For-Schleife**

In einer For-Schleife wird der Schleifenrumpf (Anweisungen) so oft ausgeführt, bis die Zählvariable *Counter* vom Start- zum Endwert gezählt wurde.

Syntax:
For *counter* = Startwert **To** Endwert [**Step** schrittweite]
  Anweisungen
Next *counter*

Bsp.: Zeigt nacheinander 4 Messagboxen an!

```
Dim textfeld() As String = {"Hallo", "Welt", "wie", "gehts?"}
For a As Integer = 0 To textfeld.Length - 1
    MsgBox(textfeld(a))
Next a
```

- **For-Each-Schleife**

Bei der For..Each-Schleife durchläuft die Schleifenvariable alle Elemente eines Feldes oder einer Aufzählung. Daher muss sie immer vom gleichen Datentyp wie die Listenelemente sein.

Syntax:
For Each *element* **In** Liste
  Anweisungen
Next *element*

Bsp.: Ergebnis wie oben!

```
Dim textfeld() As String = {"Hallo", "Welt", "wie", "gehts?"}
For Each i As String In textfeld
    MsgBox(i)
Next i
```

Prozeduren

Prozeduren werden benötigt, wenn gleicher Code an verschiedenen Stellen im Programm ausgeführt werden soll. Man unterscheidet die **Prozedurdefinition** und den **Prozeduraufruf**. In der objektorientierten Sprache sind Prozeduren Methoden.

• Unterprogramme

Ein Unterprogramm ist eine Prozedur, die kein Ergebnis zurückliefert. Beim Aufruf können der Prozedur jedoch **Parameter** übergeben werden.
Man unterscheidet Parameter bei denen nur der Wert einer Variablen (ByVal) oder bei denen eine Referenz auf eine Variable (oder Objekt) übergeben wird.

Syntax:
Sub *name*(Parameterliste)
    Anweisungen
End Sub

Bsp.: Prozedurdefinition!

```
Sub sqr(ByVal x As Double)   'Berechnet die Quadratzahl
    Dim Quadrat As Double
    Quadrat = x * x
    MsgBox("Das Quadratvon " + x.ToString + " ist " + Quadrat.ToString)
End Sub
```

Bsp.: Prozeduraufruf

```
Dim wert As Double

wert = InputBox("Geben Sie eine Zahl ein!")
sqr(wert)    'Prozeduraufruf
```

• Funktionen

Eine Funktion liefert immer einen Wert zurück. Der Typ des Rückgabewertes muss bei der Definition angegeben werden.

Syntax:
Function *name*(Parameterliste) **As** datentyp
    Anweisungen
    name = Ergebnis
End Function

Bsp.: Funktionsdefinition!

```
Function sqr(ByVal x As Double) As Double 'Berechnet die Quadratzahl
    sqr = x * x
End Function
```

Bsp.: Funktionsaufruf

```
Dim wert As Double

wert = InputBox("Geben Sie eine Zahl ein!")
MsgBox("Das Quadrat von " + wert.ToString + " ist " + sqr(wert).ToString)
```