

ERLÄUTERUNGEN

\*\*\*\*\*

Bit / Byte . . . . .	A-2
ASCII . . . . .	A-3
STRING / BASIC-SCHLÜSSELWORT / TOKEN . .	A-4
Modulares Programmieren . . . . .	A-6
Maschinen - Loop . . . . .	A-9
Formulare . . . . .	A-10

## Bit / Byte

Bit = Binary Digit (Ziffer im Zweiersystem)

Ein Bit ist die kleinste Einheit der Information, die ein Computer verarbeiten kann.

Diese Informationseinheit wird im allgemeinen mit 1 / 0 dargestellt, was etwa der Aussage JA / NEIN oder EIN / AUS entspricht.

In einem Byte werden 8 Bits zusammengefasst.  
Damit sind 256 Kombinationen von Einsern und Nullen möglich, nämlich von

```
0000'0000
0000'0001
0000'0010
0000'0011
    :
    :
    :
bis 1111'1110
    1111'1111
```

Die Erfahrung zeigt, dass mit diesen 256 Kombinationen alle Zeichen des Alphabets, alle Ziffern sowie Sonder- und Steuerzeichen darstellbar sind.

Mögliche Darstellungsarten:

Binär (Basis 2)	Hexadezimal (Basis 16)	Dezimal (Basis 10)
1011'0110 B	&B6	182

## ASCII

### American Standard Code for Information Interchange

(Amerikanischer vereinheitlichter Schlüssel für Informations-  
Austausch)

Dieser Schlüssel (Code) ordnet jedem Buchstaben (klein und gross), jeder Ziffer, jedem Sonderzeichen und jeder Steuerfunktion eine bestimmte Kombination von 8 Bits = 1 Byte zu (vergleiche Erläuterungen Bit / Byte auf Seite A-2).

Zur Darstellung eines Zeichens genügen allerdings 7 Bits, das 8. Bit hat teilweise Prüffunktion.

Verwendet werden die rechten 7 Bits, d.h. das äusserst linke Bit (höchstwertiges Bit) ist unbedeutend für das darzustellende Zeichen.

Mit den 7 Bits lassen sich 128 Kombinationen erstellen. Im Handbuch zum PC-1500 ist auf den Seiten 144 und 145 ein Auszug aus der ASCII-Tabelle abgedruckt.

Anmerkung: Die Sonderzeichen  $\pi$ ,  $\yen$  und  $\sqrt$  entsprechen nicht den ASCII-Normen, ebenso  $\square$ .

SHARP verwendet diesen Code - wie die meisten Computerhersteller - zur Darstellung des BASIC-Programmes und der Daten im Speicher (Beachten Sie bitte die Erläuterungen zu TOKEN, Seite A-4).

## STRING / BASIC-SCHLÜSSELWORT / TOKEN

Für die richtige Anwendung der mit TOOL1 neu angebotenen BASIC-Befehle FIND und CHANGE ist es wichtig, dass der Benutzer den Unterschied zwischen einem STRING und einem BASIC-Schlüsselwort versteht.

STRING: Unter einem STRING versteht man eine beliebige Reihenfolge von Buchstaben, Ziffern und Sonderzeichen.

Ein STRING kann also ein einzelnes Zeichen, ein Wort oder ein ganzer Text sein.

Im Speziellen spricht man auch von einem NULLSTRING oder LEERSTRING, wenn der STRING kein Zeichen beinhaltet. Der NULLSTRING wird wie folgt dargestellt: "".

BASIC-SCHLÜSSELWORT: Ein BASIC-Schlüsselwort nennt man ein Funktions-Name oder Befehl, also eine Reihenfolge von mindestens zwei Buchstaben. Auf den Seiten 153 bis 159 im Handbuch zum PC-1500 sind alle BASIC-Schlüsselwörter aufgeführt.

TOKEN: Im Gegensatz zu einem Grosscomputer steht bei einem Rechner wie dem PC-1500 nur eine begrenzte Menge Speicherplatz zur Verfügung. Dies vorallem aus Platz- und Leistungsverbrauch-Gründen.

SHARP verwendet deshalb eine Methode, mit der BASIC-Programme weniger Speicherplatz beanspruchen.

Ausdrücke, die ein BASIC-Schlüsselwort darstellen, werden nicht im ASCII-Code (vergleiche Erläuterungen A-3) gespeichert, sondern es wird ein besonderer Code anstelle des BASIC-Schlüsselwortes abgespeichert.

Beispiel: RETURN

Würde RETURN im ASCII-Code gespeichert, würden dazu 8 Bytes gebraucht (inklusive Leerzeichen).

Der Speicherausschnitt würde dann wie folgt aussehen:

ASCII	Hex	Dezimal
R	&52	82
E	&45	69
T	&54	84
U	&55	85
R	&52	82
N	&4E	78
space	&20	32

Gespeichert wird aber das TOKEN &F1'99, also werden nur 2 Bytes benötigt:

ASCII	Hex	Dezimal
	&F1	241
	&99	153

## STRING / BASIC-SCHLÜSSELWORT / TOKEN (Fortsetzung)

---

Auf Grund einer Tabelle werden die TOKEN jeweils für die Anzeige oder den Plotter übersetzt.

Auswirkungen für den Anwender bei FIND oder CHANGE:  
\*\*\*\*\*

Die BASIC-Zeile

```
PRINT "SIN = ";SIN X
```

sieht im Speicher wie folgt aus:

ASCII	Hex	Dezimal	
	&F0	240	} TOKEN für PRINT
	&97	151	
"	&22	34	} STRING SIN
S	&53	83	
I	&49	73	
N	&4E	78	
space	&20	32	
=	&3D	61	} TOKEN für SIN
space	&20	32	
"	&22	34	
;	&3B	59	
	&F1	241	
	&7D	125	
X	&58	88	

- Mit FIND "SIN" wird der STRING SIN zwischen den Anführungszeichen gefunden, nicht aber die Funktion SIN.
- Mit FIND !SIN! wird das TOKEN für SIN, also die Funktion SIN gefunden, nicht aber der STRING SIN zwischen den Anführungszeichen.
- CHANGE verhält sich genau gleich.

## Technik der modularen Programmierung

Modulare Programmierung nennt man die Technik, bei der man Programme in einzelne Blöcke (Module) unterteilt. Die einzelnen Module übernehmen dann üblicherweise eine Funktion oder eine Gruppe von Funktionen des gesamten Programms.

- Voraussetzungen:
- Die einzelnen Funktionen des Programms müssen klar umschrieben und getrennt werden.
  - Die Schnittstellen müssen exakt definiert sein, d.h. es muss genau festgelegt werden, welche Daten in welcher Form von einem Modul ins andere übergeben werden.

Man unterscheidet in der modularen Programmierung zwei grundsätzlich verschiedene Methoden:

1. Ein Hauptprogramm (Main-Modul), mehrere Unterprogramme (Subroutinen):  
In diesem Fall hat das Hauptprogramm meist nur Steuerfunktion. Eingabe, Verarbeitung der Daten und Ausgabe werden in den Unterprogrammen erledigt.

Vorteil: Das Programm kann laufend um neue Funktionen ergänzt werden.

2. Verschiedene gleichwertige Module:  
In diesem Fall ist das Programm einfach in mehrere Teile gegliedert.

Vorteil: Längere Programme werden übersichtlicher.

### Vorteile der modularen Programmierung:

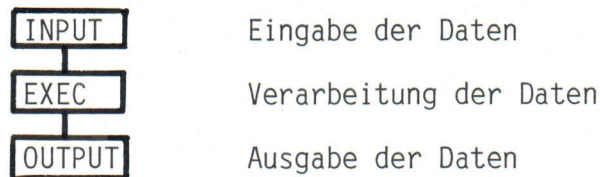
- Arbeitsteilung:
- Die einzelnen Module können separat erstellt und getestet werden.
  - Es können mehrere Personen gleichzeitig am selben Programm arbeiten.
- Flexibilität:
- Anpassungen, zum Beispiel an neue Peripheriegeräte, können bei geeignet gewählter Aufgabenteilung durch Aenderung eines einzelnen Moduls realisiert werden.
  - Denkbar ist auch die Ergänzung des Programms um ein weiteres Modul.
- Übersichtlichkeit:
- Kleinere Programm-Module sind übersichtlicher bei der Erstellung, der Fehlersuche und bei Anpassungen.
- Wartung:
- Auswirkungen von Aenderungen oder Massnahmen zu Anpassungen sind einfacher abzuschätzen.
- Einlesen / Aufzeichnen:
- Es können einzelne Module in den Speicher eingelesen, bzw. auf Band aufgezeichnet werden, was Zeitersparnisse bedeutet (siehe FLOAD P, Seite 2-9 und FSAVE P, Seite 2-11, nur mit TOOL2).

## Modulare Programmierung mit dem PC-1500

Der Aufbau des PC-1500 erlaubt es, verschiedene BASIC-Programme gleichzeitig im Speicher zu haben, man spricht dann von Programm-Moduln.

Diese Programm-Module können zusammen ein ablauffähiges Programm bilden oder jedes für sich als unabhängiges Programm funktionieren.

Der Aufbau eines Programms aus mehreren Moduln könnte zum Beispiel wie folgt aussehen:



Ein Programm zur Berechnung der reellen Lösungen einer quadratischen Gleichung könnte dann etwa so aussehen:

```
10:"A"REM INPUT
20:INPUT "A = ";A
30:INPUT "B = ";B
40:INPUT "C = ";C
50:GOTO "B"
10:"B"REM EXEC
20:D=B*B-4*A*C
30:IF D<0THEN 70
40:D=√D
50:X=(-B+D)/(2*A)
60:Y=(-B-D)/(2*A)
70:GOTO "C"
10:"C"REM OUTPUT
20:IF D<0THEN 60
30:PRINT "X1 =";X
40:PRINT "X2 =";Y
50:GOTO 70
60:PRINT "DIS.<0"
70:END
```

Eingabe der Daten

Verarbeitung der Daten

Ausgabe der Daten

Man beachte, dass jedes Modul nur einen Einstieg (Entrypoint) und einen Ausstieg (Backpoint) hat.

Zwar lässt es der Aufbau des Rechners ohne weiters zu, dass jede Zeile jedes Moduls angesprungen wird, doch widerspricht dies den Regeln der modularen Programmierung.