Tips & Tricks zum C 16/C 116

In dieser kleinen Sammlung von Tips und Tricks stellen wir Ihnen unter anderem nützliche PEEKs und POKEs, ein Programm zur Veränderung des Zeichensatzes, eine kurze Hardcopy-Routine zum Ausdruck des Textbildschirmes, vier zusätzliche Basic-Befehle sowie eine Möglichkeit zur Erzeugung fast beliebig großer Grafik-Textfenster vor. Außerdem zeigen wir Ihnen anhand eines Beispieles, wie man Zahlen aus dem Hexadezimalsystem in ihr dezimales Äquivalent umwandelt - und umgekehrt.

Für alle neu hinzugekommenen C 16- und C 116-Besitzer haben wir hier nochmals die seit Ausgabe 4/86 veröffentlichten Tips und Tricks zum C 16/C 116 zusammengestellt. Sofern Sie dieser kleinen Sammlung sowie der entsprechenden Rubrik im 64'er noch etwas hinzuzufügen haben, würden wir uns natürlich über Ihren Beitrag freuen.

Veränderung des Zeichensatzes

Der Zeichensatz besteht beim C 16 aus nur 128 Zeichen. Die anderen 128 Zeichen sind die inverse Darstellung der ersten 128 Zeichen. Ein Zeichensatz kann im Abstand von 1024 Byte (also 128 Zeichen mit je 8 Byte) im Speicher stehen (zum Beispiel ab den Adressen 12288, 13312, 14336, 15360).

Nun die Befehle, um ihn zu verändern:

1. »POKE 65298, PEEK (65298) AND NOT 4«
2. »POKE 65299, Anfang Zeichensatz/256«

Nach einer Fehlermeldung sind allerdings keine Zeichen mehr zu erkennen. Sie erscheinen erst dann wieder, wenn erneut ein Befehl eingegeben wird.

Listing 1 zeigt ein Programm zum Kopieren des Zeichensatzes vom ROM ins RAM. Startadresse des Zeichensatzes ist 15360.

Farben auf den Farbtasten verändern

Die Farben der Farbtasten (CTRL- oder Commodore-Taste zusammen mit einer Zahlentaste) sind in den Adressen 275 bis 290 gespeichert. Sie können beliebig geändert werden. Die entsprechenden Steuerzeichen bleiben allerdings gleich!

Beispiel: POKE 275,69.
Statt Schwarz liegt auf der ersten Farbtaste jetzt Grün.

Wiederholfunktion der Tasten

POKE 1344,0:	Wiederholfunktion aus, nur »CRSR«, »SPACE«, »INST/DEL« haben Wiederholfunktion
POKE 1344,64	keine Taste hat Wiederholfunktion
POKE 1344,128	alle Tasten haben Wiederholfunktion (Normalzustand)

Tastaturpuffer

Die Adressen 1319 bis 1328 enthalten die Codes der Tasten, die nicht unmittelbar ausgeführt werden können.

239 enthält den Zähler für den Tastaturpuffer, der angibt, wieviel Codes im Tastaturpuffer abgelegt sind.

Hier ein Beispiel, bei dem ein Programm nachgeladen wird (in einem Programm).

```
POKE 1319,13 : POKE 1320,13 : POKE 239,2
PRINT "DLOAD" : CHR$ (34)
PRINT "RUN"
```

Tastaturabfrage

In der Adresse 198 wird der Code der gerade gedrückten Taste abgelegt. Wenn keine Taste gedrückt wird, ist PEEK (198) gleich 64. Mit PEEK (198) können auch die Funktionstasten abgefragt werden.

Abfragen der »SHIFT«-, »Commodore«- und »CTRL«-Tasten

Mit PEEK (1347) können diese Tasten abgefragt werden.

(Ulrich Käfferbitz/tr/bj)

```
10 FORT=0TO56
20 READA
30 POKE15000+T,A
40 NEXT
80 SYS15000
62900 DATA 169,0,141,248,7,133,208,133
62910 DATA 210,169,208,133,209,169,60,133
62920 DATA 211,141,19,255,160,0,177,208
62930 DATA 145,210,200,208,249,230,209,230
62940 DATA 211,165,211,201,64,208,237,169
62950 DATA 192,141,18,255,169,59,133
62960 DATA 52,133,56
62970 DATA 169,246,133,51,133,55,96
```

Listing 1. Verschiebt den Zeichensatz ins RAM

Nützliche Speicherstellen

- WAIT 1,192 wartet darauf, daß eine Taste am Recorder gedrückt wird.
- Der Funktionstastenspeicher steht in den Adressen $0567 bis $05E6
- Die Einsprungadresse für die USR-Funktion steht in den Adressen 1281/1282 ($0501/$0502)

- Basic-Start	43/44 (Low-/Highbyte)
- Anfang der Variablen	45/46 (Low-/Highbyte)
- Anfang der Felder	47/48 (Low-/Highbyte)
- Ende der Felder	49/50 (Low-/Highbyte)
- Stringspeicher Ende	51/52 (Low-/Highbyte)
- Anfang der Strings	53/54 (Low-/Highbyte)
- Basic-Speicher Ende	55/56 (Low-/Highbyte)

- Der Ton-Chip des C 16 läßt sich in Assembler wie folgt programmieren: Soundregister ist Adresse 65297. Jedem Bit kommt eine bestimmte Steuerfunktion zu:

Bit 0 bis 2	= #0 bis #7	Lautstärke = VOL 0 bis VOL 6
Bit 3	= #8	volle Lautstärke (wenn Bit 3 gesetzt ist, sind Bit 0-2 ohne Bedeutung)
Bit 4	= # 16	Stimme 1 an (=1) / aus (=0)
Bit 5	= # 32	Stimme 2 an (=1) / aus (=0)
Bit 6	= # 64	Stimme 2 Rauschen (=1)
Bit 7	= # 128	Bit gesetzt = kein Ton

Der Ton ist in 10 Bit codiert: mögliche Werte: #0 bis #1023

Frequenzen		Low-Byte	High-Byte (0,1)
	Stimme 1	65294	65298
	Stimme 2	65295	65296

- Register 65286: Durch Löschen der Bits 0/1 läßt sich der Bildschirm um bis zu drei Rasterzeilen nach oben verschieben.

Bit 2 gesetzt	Bildschirm vier Rasterzeilen nach unten
Bit 3 gelöscht	Bildschirm vier Rasterzeilen nach unten
Bit 4 gelöscht	Bildschirm ausgeblendet
Bit 5 gesetzt	Char-Speicher verschoben
Bit 6 gesetzt	kein Cursor sichtbar

- Register 65287	Bit 0/1 setzen	Verschieben des Bildschirms um bis zu drei Rasterzeilen nach rechts
	Bit 4 gesetzt	Mehrfarbmodus ein
	Bit 5 gesetzt	Videochip stoppt (keine Bilderzeugung mehr)
	Bit 6 gesetzt	Programmierbare Bildstörung
	Bit 7 gesetzt	38 Zeichen/Zeile
- POKE 65290, 162		sperrt Tastatur
- POKE 65290, 163		entriegelt Tastatur
- Register 65301		Hintergrundfarbe Bit 0-3 (Low-Nibble) Farbe
- Register 65305		Rahmenfarbe Bit 4-6 (High-Nibble) Intensität

(Frank Plachetta/tr/bj)

Windows im Programm

Im Sonderheft zum C 16 wird auf die Möglichkeit hingewiesen, Bildschirmfenster innerhalb eines laufenden Programms zu erstellen. Die beschriebenen Methoden sind jedoch weiter verbesserungsfähig.

Unter Umgehung des Escape-Modus: Die Koordinaten zur Festlegung eines Windows können direkt in die Systemadressen für die aktuelle Bildschirmgröße »gePOKEt« werden.

```
Adresse
2021 ($07E5): unterer Rand  (Zeilenkoordinate/»esc-B«)
2022 ($07E6): oberer Rand   (Zeilenkoordinate/»esc-T«)
2023 ($07E7): linker Rand   (Spaltenkoordinate/»esc-T«)
2024 ($07E8): rechter Rand  (Spaltenkoordinate/»esc-B«)
```

Ein Beispiel: Mit `POKE 2022,2` wird die erste Bildschirmzeile »eingefroren« (zum Beispiel als Titelzeile), indem der obere Rand des aktuellen Bildschirms herabgesetzt wird.

Der Normalzustand (Window = normale Bildschirmgröße) wird wie üblich durch zweimaliges Drücken der Home-Taste beziehungsweise durch `PRINT"Home Home"` wiederhergestellt.

(Gerd Watza/tr/bj)

Hardcopy-Routine

Wenn Sie einen Drucker an Ihren Computer angeschlossen haben, können Sie mit Hilfe dieser kleinen Unterroutine (Listing 2) den Bildschirminhalt zu Papier bringen.

Aufgerufen wird das Programm durch »GOSUB 1000«. Um den Bildschirminhalt herum wird ein Rand gedruckt.

(Jürgen Hagen/tr/bj)

```
1000    REM HARDCOPY-ROUTINE
1020    OPEN 4,4,0
1030    FOR X=1 TO 42:X$=X$+"-":NEXT X
1040    PRINT#4,X$
1050    FOR X=0 TO 24:Y$="!"
1060    FOR Y=1 TO 40:Y%=PEEK(3071+X*40+Y):IF Y%<32 THEN
        Y%=Y%+64
1070    Y$=Y$+CHR$(Y%):NEXT Y
1080    Y$=Y$+"!":PRINT#4,Y$:NEXT X
1090    PRINT#4,X$:PRINT#4:PRINT#4:PRINT#4:PRINT#
        4:PRINT#4
1100    PRINT#4:PRINT#4:PRINT#4:PRINT#4:CLOSE4:X$=
        "":Y$="":RETURN
```

Listing 2. »Hardcopy-Routine« für C 16/C 116

```
        Programm-Beschreibung
Zeile:
1020    Eröffnen eines Drucker-Kanals
1030    Erweitern des Feldes X$ auf insgesamt 42 Bindestriche
1040    Drucken des Feldes X$ als obere Umrandung
1050    Schleife zur Erzeugung von 25 Zeilen mit der Initialisierung der linken Umrandung
1060    Schleife zum PEEKen von 40 Zeichen pro Bildschirmzeile
1070    Ende der Zeichenschleife
1080    Ende der Zeilenschleife mit der rechten Umrandung
1090    Drucken des Feldes X$ als untere Umrandung sowie einige Leerzeilen
1100    Drucken weiterer Leerzeilen, Schließen des Druckers, Löschen der Felder X$ und Y$ sowie ein »RETURN« für den Fall, daß in diese Routine durch ein »GOSUB« verzweigt wurde.
```

Das seltsame Listing

Wer glücklicher Besitzer des C 16-Sonderheftes (3/86) ist, wird sich beim Abtippen des Farbdemos auf Seite 22 etwas gewundert haben: Das Listing befindet sich nämlich in einem absolut »unabtippbaren« Zustand. Unser Umsetzprogramm für die Steuerzeichen (CLR, Farben etc.) hat die Verarbeitung dieses Listings verweigert. Nachfolgend finden Sie nun die fehlerfreie Version (Listing 3). (tr/bj)

```
10 REM ********************************
20 REM *                              *
30 REM *   (C 16)  FARBDEMO    (116)  *
40 REM *           --------           *
50 REM *   CHRISTIAN QUIRIN SPITZNER   *
60 REM *   GRUBERSTR. 53, 8011 POING   *
70 REM *   TELEFON:      08121/81100   *
80 REM *                              *
90 REM ********************************
100 COLOR0,4
110 COLOR4,4
120 PRINT"{CLR,DOWN,BLACK,10SPACE}F A R B D E M O"
130 PRINT"{10SPACE}================{DOWN}"
140 PRINT" H{2SPACE}S W R Z P G B G O B G R B H D H"
150 PRINT" E{2SPACE}C E O Y U R L E R R E O L E U E"
160 PRINT" L{2SPACE}H I T A R U A L A A L S A L N L"
170 PRINT" L{2SPACE}W E . N P E U B N U B A U L K L"
180 PRINT" I{2SPACE}A S . . U N . . G . G . G B E G"
190 PRINT" G{2SPACE}R S . . R . . . E . R . R L L R"
200 PRINT" K{2SPACE}Z . . . . . U . U A B U"
210 PRINT" E{2SPACE}. . . . . E . E U L E"
220 PRINT" I{2SPACE}. . . . . N . N . A N"
230 PRINT" T{2SPACE}. . . . . U ."
240 PRINT
250 FOR J=7 TO 0 STEP -1
260 :  PRINT"{BLACK}"J;
270 :  FOR I=1 TO 16
280 :  COLOR 1,I,J
290 :  PRINT"{SPACE,RVSON,SPACE,RVOFF}";
300 :  NEXT I
310 :  PRINT
320 NEXTJ
330 GETKEYA$
```

64'er

Listing 3. »Farbdemo«. Zeigt alle 121 Farben des C16.

Vier nützliche Basic-Befehle

Das Programm »BASICTOOL« (Listing 4) ist ein sogenannter »DATA-Lader«, der vier kurze Maschinenprogramme erzeugt. Hierdurch werden einige zusätzliche Funktionen auf dem C 16 implementiert:

OLD

Mit diesem Befehl kann man Basic-Programme, die unbeabsichtigt durch »NEW« oder einen Reset gelöscht wurden,

wieder lauffähig machen. Er hilft bisweilen auch, wenn Programme fehlerhaft von Kassette geladen wurden.

Aufgerufen wird er einfach mit: SYS 1618

SWAP

Dieser Befehl wurde für den C 64 bereits in der 64'er, Ausgabe 12/85, vorgestellt. Ich habe ihn für den C 16 umgeschrieben.

Er dient dazu, zwei Strings miteinander zu vertauschen. Dies geschieht durch Vertauschen der Stringdeskriptoren, es entsteht also kein »Stringmüll«. Die gefürchtete Garbage Collection wird verhindert.

Es gilt folgender Syntax für den Aufruf: SYS 1569 (A$,B$) A$ und B$ können hierbei zwei beliebige Stringvariable sein.

BLOAD und BSAVE

Diese Befehle simulieren die entsprechenden Befehle des C 128. Sie dienen dazu, von einem Basic-Programm aus einen ganzen Speicherblock zu laden oder zu speichern.

Man kann so zum Beispiel eine Grafik speichern oder ein Maschinenprogramm nachladen. Auch Programme, die mit dem S-Befehl des eingebauten Monitors gespeichert wurden, können geladen werden.

Es gilt folgende Syntax:

BLOAD: SYS 1536 "name", g, 1

BSAVE: SYS 1548 "name", g, 1, aa, ea+1

Hierbei ist g die Gerätenummer (1 oder 8 für Kassette oder Disk), aa die Anfangs- und ea die Endadresse des Speicherbereichs.

Beispiel: aa = 8192, ea = 16384 zum Speichern der HiRes-Grafik.

(Michael Schmand/tr/bj)

```
120 DATA 20,6B,A8,A5,0A,A6,2B,A4
130 DATA 2C,4C,D5,FF,20,6B,A8,20
140 DATA DE,9D,84,D8,85,D9,20,DE
150 DATA 9D,A6,14,A8,A9,D8,4C,D8
160 DATA FF,20,8E,94,20,2C,93,20
170 DATA 1A,93,A5,64,85,D0,A5,65
180 DATA 85,D1,20,91,94,20,2C,93
190 DATA 20,1A,93,A0,00,B1,D0,85
200 DATA D2,B1,64,91,D0,A5,D2,91
210 DATA 64,C8,C0,03,D0,EF,20,8B
220 DATA 94,60,A9,01,A8,91,2B,20
230 DATA 18,88,20,4B,88,68,68,4C
240 DATA 9A,8A,00,00,00,00,00,00
260 FOR I=1536 TO 1639
270 READ D$:POKE I,DEC(D$):C=C+DEC(D$)
280 NEXT:IF C<>12205 THEN PRINT" DATA FEHL
    ER!":END
460 PRINT" OBJEKTCODE SPEICHERN (J/N)?"
470 GET KEY C$:IF C$<>"J"THEN END
480 PRINT" CASETTE ODER DISK{4SPACE}(C/D)?
    ":GET KEY C$
490 IF C$="D"THEN U=8:ELSE U=1
500 SYS 1548"MINITOOL.OBJ",U,1,1536,1640
```

Listing 4. Vier nützliche Befehle für den C 16/C 116

Beliebige große Grafikfenster

Dieses Maschinenprogramm steuert einen Rasterzeilen-Interrupt, durch den man im »GRAPHIC 2«- oder »GRAPHIC 4«-Modus ein fast beliebig großes Grafikfenster erzeugen kann. Die hierzu notwendige Routine ist bereits im ROM des C 16 vorhanden, denn auch im normalen Split-Screen-Modus muß ja ein Raster-Interrupt stattfinden. Man kann also diese Routine einfach kopieren und ein wenig ändern. Wie dies am einfachsten geschieht, wird im folgenden beschrieben:

1. Geben Sie zunächst die folgende Basic-Zeile ein:

10 SYS 4293

Dies wird der spätere Einsprung in die Initialisierungs-Routine. Achten Sie darauf, daß zwischen »SYS« und der Adresse höchstens ein Space steht, sonst kommen Sie mit dem Speicherplatz nicht hin.

2. Rufen Sie jetzt den Monitor auf und geben Sie ein:

T CE0E CEC4 100E

Damit wird die Interrupt-Routine ins RAM kopiert. Hierbei bleibt das Low-Byte der Interrupt-Startadresse gleich ($OE), so daß später nur die High-Bytes der Interrupt-Vektoren geändert werden müssen, um den Interrupt auf unsere eigene Routine zu lenken.

3. Machen Sie nun noch vom Monitor aus folgende Änderungen: A 1015 JSR $1060.

Damit wird der Raster-Interrupt auf die eigene Routine umgelenkt.

4. Zur Initialisierung müssen Sie jetzt noch folgendes kurzes Maschinenprogramm eingeben:

```
A 10C5  SEI
A 10C6  LDA#$10   (Interrupt auf eigene Routine lenken)
A 10C8  STA$0313
A 10CB  STA$0315
A 10CE  CLI
A 10CF  LDA#$11   (Basic-Start auf $1100 heraufsetzen)
A 10D1  STA $2C
A 10D3  LDA#$00
A 10D5  STA $1100
A 10D8  PLA
A 10D9  PLA
A 10DA  JMP $8A7B  (NEW/CLR ausführen)
```

5. Verlassen Sie jetzt den Monitor mit »X« und machen Sie im Direktmodus folgende Eingaben:

POKE 4185,129:POKE 4203,131:POKE4238,131

Damit ist das kleinstmögliche Grafikfenster eingestellt.

6. Rufen Sie wieder den Monitor auf. Speichern Sie das fertige Programm mit:

S "GRAFIKFENSTER",8,1001,10DD

Dadurch wird jetzt die vorher eingegebene Basic-Zeile zusammen mit der kopierten Maschinenroutine als zusammenhängendes Programm gespeichert und später als solches auch wieder geladen.

Kassettenbenutzer müssen statt der »8« natürlich eine »1« einsetzen.

Das so gespeicherte Programm können Sie nun einfach mit »LOAD "GRAFIKFENSTER",8« (beziehungsweise mit »LOAD "GRAFIKFENSTER",1 für Datasette) laden und mit »RUN« starten. Dabei wird automatisch der Basic-Start um 256 Byte nach oben gesetzt. Danach können Sie Basic-Programme eingeben oder laden, als ob nichts geschehen wäre. Sobald Sie aber »GRAPHIC 2« oder »GRAPHIC 4« eingeben, sehen Sie, daß jetzt nicht mehr fünf, sondern neun Textzeilen sichtbar werden. Dies wurde durch die POKE-Befehle erreicht, die einfach dafür sorgen, daß bei einer früheren Rasterzeile als sonst in den Textmodus zurückgeschaltet wird. Weitere POKE-Möglichkeiten entnehmen Sie bitte der Tabelle. Sie können damit jederzeit von Basic aus die Größe des Grafikfensters ändern.

Wichtiger Hinweis:

Sowohl bei der Eingabe als auch beim Laden des Programms muß der Basic-Start unbedingt auf seinem Normalwert sein! Ansonsten liegt die am Anfang eingegebene

Basic-Zeile nicht an der richtigen Stelle im Speicher. Drücken Sie also am besten einmal auf den Reset-Knopf. Dies ist besonders wichtig für Besitzer eines Plus/4 oder eines C 16 mit RAM-Erweiterung, da hier beim erstmaligen Einschalten der Grafik der Basic-Start automatisch um 12 KByte heraufgesetzt wird!

Tabelle:

In nachfolgender Tabelle ist aufgeführt, welchen Effekt man erzielt, wenn man durch POKE-Befehle die Größe des Grafikfensters ändern will. Dazu müssen Sie jeweils in die Speicherstellen 4203 und 42038 den Wert von X, und in Speicherstelle 4185 den Wert von X minus 2 POKEn.

X	Anzahl Textzeilen	Wirkung
131	9	kleinstmögliches Grafikfenster
139	8	
147	7	
155	6	
163	5	normal
171	4	
179	3	
187	2	

Ein besonderer Effekt läßt sich erzielen, wenn man den Wert in 4238 auf X plus 1 setzt. Hier wartet nämlich das Interrupt-Programm, bis der Rasterstrahl den eigentlichen Umschaltpunkt erreicht hat. Dadurch wird ein flimmerfreies Umschalten möglich. Erhöht man nun diesen Wert, so wartet der Interrupt mit dem Einschalten des Textbildschirms noch eine Rasterzeile »lang«, obwohl der Grafikbildschirm bereits ausgeschaltet wurde. Resultat ist eine schwarze Trennlinie zwischen Grafik und Text, die entsteht, während sich der Video-Chip im »Niemandsland« zwischen Text und Grafik befindet. (Michael Schmand/tr/bj)

Warum Hexadezimal?

Hier wollen wir einen leicht verständlichen Lösungsweg zur Umrechnung der beiden Zahlensysteme Dezimal und Hexadezimal erarbeiten.

Wir sind von Kind auf an das dezimale Zahlensystem gewöhnt. In der Computertechnik wird jedoch auch häufig das hexadezimale Zahlensystem verwendet. Warum? Um diese Frage zu beantworten, müssen wir etwas ausholen.
Wie Sie vielleicht wissen, kennt der Computer im Prinzip nur das binäre Zahlensystem, also die Kombination von 0 und 1. Aus diesen beiden Ziffern wird jede Zahl gebildet. Beispiel:

binär 00001101 = dezimal 13 = hexadezimal D
binär 1100000000000000 = dezimal 49152 = hexadezimal C000

In Fachzeitschriften finden Sie oft Abkürzungen oder andere Schreibweisen, um klarzustellen, welches Zahlensystem gemeint ist. Beispiel:

binär = bin = %
hexadezimal = hex = $
dezimal = dez = #

Die Zeichen %, $ und # werden vor allem von Programmen benutzt, die irgendetwas mit Assembler (Maschinensprache) oder mit dem Betriebssystem des Computers zu tun haben, zum Beispiel Maschinensprache-Monitore. Wenn Sie mehr darüber wissen wollen, empfehle ich Ihnen das 64'er-Sonderheft 8/85 (Assembler für Anfänger und Fortgeschrittene). Doch zurück zu den Zahlensystemen:

Die ersten Computer konnten nur mit binären Werten »gefüttert« werden, eine sehr anstrengende Sache für die Programmierer. Eine wesentliche Vereinfachung ergab das hexadezimale System. Jeweils 4 Binär-Ziffern konnten ersetzt werden durch eine einzige Hex-Ziffer. Beispiel:

bin 1101 = hex D = dez 13
bin 1000 = hex 8 = dez 8

Das liegt daran, daß das binäre Zahlensystem die Basis 2 besitzt, sich jede Zahl also in 2er-Potenzen darstellen läßt, während das hexadezimale System die Basis 16 hat. Da 2 hoch 4 den Wert 16 ergibt, können 4 Binär-Ziffern durch eine Hex-Ziffer ersetzt werden.

Unser C 16 und auch fast alle anderen Heimcomputer sind 8-Bit-Computer. Der C 16 kann 64 KByte Speicherplatz adressieren, das sind 2 hoch 16 = dezimal 65536 = hex FFFF. Sie sehen, überall taucht die 2 auf, beziehungsweise Potenzen von 2. Und das macht die Verwendung von Hex-Zahlen so einfach, weil man sehr oft mit runden Hex-Zahlen zu tun hat. Das hexadezimale Zahlensystem bietet jedoch nicht nur von seiten der Eingabe Vorteile, sondern erlaubt es auch, sich häufig verwendete Kombinationen zu merken (zum Beispiel $A9 = 169 für den Assembler-Befehl »LDA #«, also lade Akkumulator mit dem Wert der im Programmablauf folgenden Speicherzelle. Genaueres finden Sie, wie schon erwähnt, im Assembler-Sonderheft 8/85).

Was ist das Hexadezimalsystem?

In unserem normalen Zahlensystem repräsentiert jede Stelle einer Zahl eine Zehnerpotenz. Ein Beispiel: Die Zahl 4714 läßt sich auch als Summe von Zehnerpotenzen schreiben

$4714 = 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 4 * 10^0 = 4 * 1000 + 7 * 100 + 1 * 10 + 4 * 1$

Beim Hexadezimalsystem wird nun jede Stelle einer Zahl nicht mehr durch eine Zehner-, sondern durch eine Sechzehnerpotenz repräsentiert. Auch hier wieder ein Beispiel: Die Hexadezimalzahl 0324 bedeutet nichts anderes als $3 * 16^2 + 2 * 16^1 + 4 * 16^0$ (=3*256+2*16+4*1).

Dies hat aber noch weitere Konsequenzen:

Im Dezimalsystem wird eine Stelle immer von 0 bis 9 (insgesamt 10 Ziffern) durchgezählt, bevor die nächste Stelle um eins erhöht wird. Also

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14...

Im Hexadezimalsystem jedoch wird eine Stelle um 16 Werte erhöht, bevor zur nächsten Stelle ein Wert hinzugefügt wird. Da aber unsere Ziffern von 0 bis 9 dazu nicht ausreichen, wurden zusätzlich die Buchstaben A bis F herangezogen. Sie vertreten die Zahlenwerte 10 bis 15 (von 0 bis 15 sind es 16 Werte!). Es bedeuten:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14... 1A 1B 1C...

Die Dezimalzahl 10 ist also gleichwertig mit dem Hexadezimalwert A. Damit wären wir auch schon bei der Umrechnung.

Dezimal - Hexadezimal

Wenn wir eine Dezimalzahl in Hexadezimal umrechnen wollen, so bauen wir den Hex-Wert Stelle für Stelle von links nach rechts auf.

Nehmen wir also an, wir möchten die Dezimalzahl 41717 in Hexadezimal umrechnen. Dazu teilen wir sie erst einmal durch 16^3

$41717 : 16^3 = 10{,}1848145$

Uns interessiert hier nur die Vorkommastelle 10. Sie ist gleichbedeutend mit dem Hex-Wert **A**. Er bildet die letzte Stelle unserer Hexadezimalzahl (die Zählung beginnt rechts und endet links).

Nun müssen wir von unserer Dezimalzahl $10 * 16^3$ abziehen.

Also $41717 - 10 * 16^3 = 757$

Um die nächsten Stellen unserer Hex-Zahl zu erhalten, führen wir diese Prozedur nun noch mit 16^2 und 16^1 durch:

$757 : 16^2 = 2{,}9703125$ (=**2**)
$757 - 2 * 16^2 = 245$
$245 : 16^1 = 15{,}3125$ (=**F**)
$245 - 15 * 16^1 =$ **5**

Als endgültige Umrechnung der Zahl 41717 ins Hexadezimalsystem erhalten wir also **$A2F5**.

Hexadezimal - Dezimal

Diese Umrechnung ist schon wesentlich einfacher. Um die Hex-Zahl $A2F5 wieder zurückzurechnen, geht man wie folgt vor:

$A * 16^3 + 2 * 16^2 + F * 16^1 + 5 * 16^0$

Da man aber mit den Buchstaben A und F nicht rechnen kann, müssen diese als Dezimalzahlen angegeben werden.

$10 * 16^3 + 2 * 16^2 + 15 * 16^1 + 5 * 16^0$

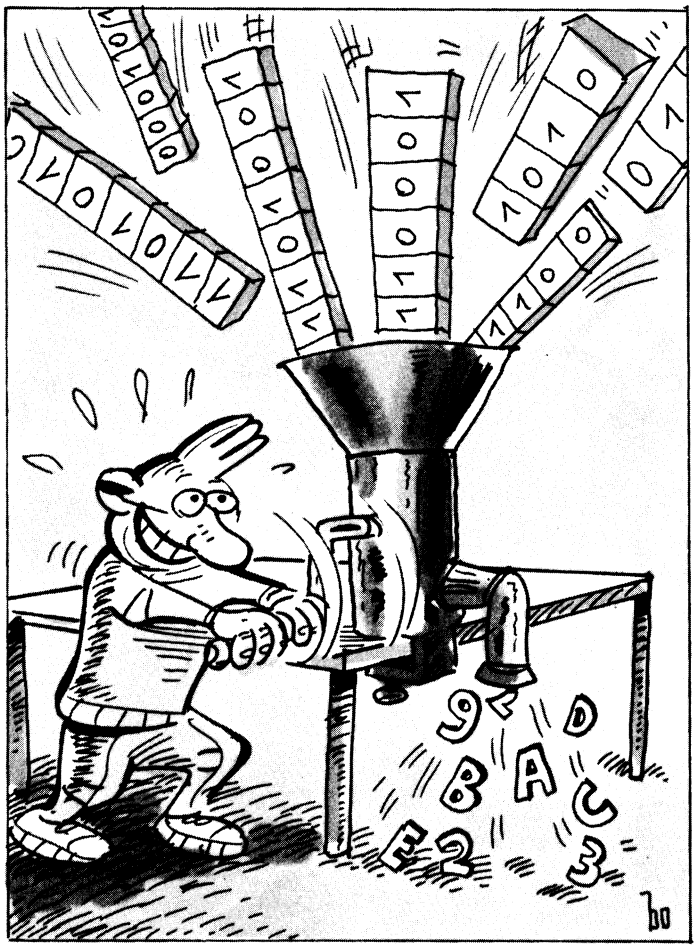
Wenn Sie dies auf Ihrem C 16 einmal ausrechnen, so werden Sie als Ergebnis wieder die Zahl 41717 erhalten!

Als Abschluß unseres kleinen Kurses könnten Sie einmal versuchen, ein Basic-Programm zu schreiben, das diese Berechnungen ausführt.

Kleine Merkhilfe

Gerade wenn man anfängt, sich mit dem hexadezimalen Zahlensystem zu beschäftigen (und dafür gibt es einige gute Gründe), ist man froh über ein paar kleine Eselsbrücken. Am Anfang kommt man immer wieder ins Stolpern, weil man sich nicht so schnell an den Dezimalwert von zum Beispiel Hex C erinnern kann. Dies geht wesentlich leichter, wenn man die Anfangsbuchstaben der beiden Systeme gegenüberstellt.

C = **Z**wölf
D = **D**reizehn
F = **F**ünfzehn (bj)



Ausführliche Informationen zu ausgewählten Themen finden C64-Anwender in zwei weiteren aktuellen 64'er SONDERHEFTEN

SONDERHEFT 07/86: PEEKs UND POKEs

Die wichtigsten Speicherstellen des C64, C16 und C128 werden ausführlich erklärt und die Unterscheidungsmerkmale klar herausgestellt. Man erfährt, was die einzelnen Speicherstellen bedeuten und wie man mit ihnen umgeht. Für Assembler-Programmierer gibt es eine ausführliche Beschreibung, wie man in Maschinensprache Berechnungen mit dem C64 durchführt. Hervorzuheben ist das Xref 7.0 Listing, das eine Crossreferenz-Liste für C128-Programme aufstellt, die in Basic 7.0 geschrieben sind. Top Tool ist eine äußerst leistungsfähige Programmierhilfe für den C64. Über 30 Seiten Tips & Tricks für C64 und C128.



Jetzt für DM 14,— überall im Zeitschriftenhandel!

SONDERHEFT 06/86: GRAFIK

Grafik-Programmierung des C64, C128 und C128 im C64-Modus. Schwerpunktthema: »Giga-CAD« — ohne komplizierte Berechnungen am Bildschirm dreidimensional konstruieren. Mit »Giga-CAD« lassen sich Grafiken mit einer Auflösung von 640 x 400 oder 1000 x 640 Punkten berechnen und konstruieren. Jede Menge Spitzen-Listings zum Abtippen: Sprite-Editor mit Animationseffekt / Erweiterungen zu Hi-Eddi / Die schnellste Grafik-Erweiterung / Routinen zum Ein- und Überblenden von Grafik-Bildern im Hi-Res-Modus / Hardcopies für Koala-Pad- und Blazing-Paddles-Bilder / Eine Basic-Erweiterung für Seikosha-Drucker / Plot- und Sprite-Basic.



Nur noch bis 25.08.86 erhältlich!