Bit-Paar	Farbregister	Speicherplatz
00	Hintergrundfarbe #0 (Bildschirmfarbe)	53281 ($D021)
01	Hintergrundfarbe #1	53282 ($D022)
10	Hintergrundfarbe #2	53283 ($D023)
11	Durch die unteren 3 Bits im Farbspeicher bestimmte Farbe	Farbspeicher

Tabelle 4. Mögliche Farbkombinationen für Multicolor-Zeichen

bens A und das entsprechende Bit-Muster (Bild 2).

Im normalen oder HiRes-Modus wird die Bildschirmfarbe bei jedem 0-Bit und die Zeichenfarbe stets da angezeigt, wo das Bit 1 ist. Beim Mehrfarbenmodus werden die Bits paarweise benutzt (Bild 3).

Im Bildbereich werden die durch AA gekennzeichneten Stellen in der Hintergrundfarbe #1, die durch BB gekennzeichneten Stellen in der Hintergrundfarbe #2 und die durch CC gekennzeichneten Stellen in der Zeichenfarbe dargestellt. Dies wird entsprechend nachstehender Tabelle 4 durch die Bit-Paare bestimmt.

Zeichencode			Hintergrundfarbregister	
Bereich	BIT 7	BIT 6	Nummer	Adresse
0 - 63	0	0	0	53281 ($D021)
64 - 127	0	1	1	53282 ($D022)
128 - 191	1	0	2	53283 ($D023)
192 - 255	1	1	3	53284 ($D024)

Tabelle 5. Im Hintergrundfarbmodus können die ersten 64 Zeichen mit unterschiedlichen Hintergrundfarben dargestellt werden

Hintergrundfarbmodus

In diesem Modus können Sie für jedes einzelne Zeichen sowohl die Hintergrund- als auch die Vordergrundfarbe steuern. So ist es zum Beispiel möglich, auf einem weißen Bildschirm ein blaues Zeichen mit gelbem Hintergrund anzuzeigen.

Für den erweiterten Hintergrundfarbmodus stehen vier Register zur Verfügung. Für jedes Register kann eine der 16 Farben gewählt werden.

In diesem Modus wird über den Farbspeicher die Vordergrundfarbe festgelegt. Die Anwendung ist die gleiche wie beim Standard-Zeichenmodus.

Bei erweitertem Modus ist die Anzahl der verschiedenen anzeigbaren Zeichen jedoch eingeschränkt. Ist der erweiterte Farbmodus eingeschaltet, können nur die ersten 64 Zeichen des Zeichen-ROM (oder die ersten 64 in Ihrem programmierbaren Zeichensatz) benutzt werden. Zwei Bit des Zeichen-Codes werden nämlich für die Wahl der Hintergrundfarbe benutzt.

Der Zeichen-Code (die auf dem Bildschirm gePOKEte Zahl) vom Buchstaben A ist eine 1. Im erweiterten Farbmodus erscheint nach dem POKEn einer 1 ein A. Normalerweise muß nach dem POKEn von 65 das Zeichen mit dem Zeichen-Code (CHR$) 129, also ein invertiertes A, erscheinen. Dies passiert nicht im erweiterten Farbmodus. Es erscheint genau das gleiche A wie vorher, jedoch eine andere Hintergrundfarbe. Entnehmen Sie die Codes der Tabelle 5.

Zum Einschalten des erweiterten Farbmodus wird Bit 6 des VIC-II-Registers mit der Adresse 53265 ($D011) auf 1 gesetzt. Dies geschieht durch folgende POKE-Anweisung:

```
POKE 53265,PEEK(53265)OR 64
```

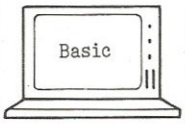
Zum Ausschalten des erweiterten Farbmodus wird Bit 6 des VIC-II-Registers mit der Adresse 53265 ($D011) auf 0 gesetzt. Hierzu dient folgende Anweisung:

```
POKE 53265, PEEK(53265)AND
191
```

Damit sind wir am Schluß des ersten Teils angelangt. Beim nächsten Mal werden wir uns mit dem Bit-Map-Modus beschäftigen. Sie werden dann lernen, wie man HiRes- und Multicolor-Bilder auf den Bildschirm zaubert. (ah)

Gekürzter Auszug aus dem Buch »Alles über den C 64« aus der Commodore Sachbuchreihe, Markt & Technik Verlag, Hans-Pinsel-Str. 2, 8013 Haar, Bestnr.: ISBN 3-89090-379-7, Preis 59 Mark

Variablen — Schwerarbeiter in der Datenverarbeitung



Sind Sie Computerneuling? Dann sind Sie in dieser Rubrik genau richtig. An dieser Stelle beginnt ein Kurs, der die Funktionen und Eigenschaften des Commodore-Basic Version 2.0, das im C 64 eingebaut ist, erklärt. Im ersten Teil erfahren Sie alles Wissenswerte zu den bisher vielleicht noch unbekannten Variablen.

Legen Sie zunächst das Handbuch des C 64 neben sich, denn hier soll das dort Geschriebene weiter vertieft und ausführlich erklärt werden. Lesen Sie zunächst die Kapitel über die Variablen durch (S. 35,98 bis 100). Wir werden im Laufe des Kurses immer wieder auf das Handbuch verweisen und darauf aufbauen.

Vielleicht haben Sie sich schon des öfteren gefragt, wie es möglich ist, daß der Computer Unmengen von Daten in relativ kurzer Zeit verarbeiten kann. Des Rätsels Lösung liegt in den sogenannten Variablen verborgen, die im folgenden ihren vielleicht noch etwas rätselhaften Charakter verlieren werden.

Vielleicht erinnert sich der eine oder andere noch entfernt an den Mathematikunterricht in der Schule und möchte jetzt beim Gedanken an diese ungeliebten Wesen aus der Algebra am liebsten nicht mehr weiter lesen. Doch keine Angst, schließlich soll hier nicht höhere Mathematik betrieben werden, vielmehr erfolgt eine leicht verständliche Erklärung der für die Programmierung so wichtigen Variablen.

Sicher haben Sie schon des öfteren mit den im Handbuch des C 64 erklärten Befehlen gearbeitet. Als Beispiel soll der PRINT-Befehl herangezogen werden. Geben Sie doch einmal die Anweisung

```
10 PRINT "HALLO"
```

ein. Wenn dieses kleine Programm mit RUN gestartet wird, erhalten Sie auf dem Bildschirm das Wort »HALLO« angezeigt. Leider kann diese Ausgabe jetzt nicht mehr verändert werden, ohne das Programm zu ändern. Man spricht deshalb von einer Textkonstanten (ständig in derselben Weise vorhanden). Soll nun beispielsweise eine ganz einfache Berechnung angestellt werden, bei der ständig neue Werte erforderlich sind, reicht diese Möglichkeit der Datenverarbeitung nicht mehr aus.

Erste Gehversuche

Für diese komplexeren Aufgaben werden Variablen benötigt. Diese bezeichnen ein Datenfeld (Speicherbereich für bestimmte Daten), dem Werte oder Texte zugewiesen werden können, die sich ständig durch Ihre Eingabe oder den Programmablauf ändern. Um dieses Phänomen näher zu betrachten, sollte die Funktion der beiden Basic-Befehle INPUT und LET aus dem Handbuch entnommen werden. Man kann also einer Variablen einen beliebigen Wert zuweisen und mit dieser Variablen

rechnen, Bildschirmausgaben gestalten oder die Variable, was immer als erstes erfolgen sollte, mit einem Wert belegen. Wie das nun genau funktioniert? Anhand eines kleinen Beispielprogramms können Sie das selbst erforschen:

```
10 PRINT "ERSTE ZAHL: ";
20 INPUT EZ%
30 PRINT "ZWEITE ZAHL: ";
40 INPUT ZZ
50 ERG = EZ% + ZZ
60 TEXT$ = "ERGEBNIS"
70 PRINT:PRINT
80 PRINT TEXT$,ERG
```

Nachdem Sie das Programm eingetippt haben, starten Sie es mit dem Befehl RUN und geben dann, wie gewünscht, die erste Zahl ein. Schließen Sie Ihre Eingabe mit <RETURN> ab. Der eingegebene Wert ist jetzt automatisch der Variablen EZ% zugewiesen. Doch warum steht hinter der Variablen ein Prozentzeichen? Der Computer arbeitet jetzt nicht etwa mit Prozenten, sondern faßt die Variable EZ durch den Zusatz »%« als Integer-Variable auf.

Rechnen kann er auch ...

Das klingt jetzt vielleicht schon wieder unheimlich kompliziert, ist aber nur halb so schlimm. Natürlich sollen Sie mit den Integer-Variablen nicht im Halbdunkel des Variablendschungels herumtappen, sondern diese auch wirklich verstehen. Dazu bauen Sie einen weiteren Befehl in Ihr kleines Programm ein: Das Programm muß durch gleichzeitiges Drücken der Tasten <RUN/STOP + RESTORE> abgebrochen werden. Mit LIST erreichen Sie eine Ausgabe aller Zeilen am Bildschirm. Tippen Sie danach folgende Zeile ein: »25 STOP«. Nun starten Sie das Programm erneut mit RUN. Nach dem Eingeben der ersten Zahl, Sie sollten eine Zahl mit mehreren Stellen hinter dem Komma verwenden (beispielsweise 10.487), meldet sich der Computer mit <BREAK IN 25>. Dort steht schließlich ein ziemlich hartnäckiger STOP-Befehl. Mit dem PRINT-Befehl kann die Integer-Variable EZ% am Bildschirm betrachtet werden. Dazu geben Sie einfach »PRINT EZ%« ein. Seltsamerweise fehlen aber alle Stellen hinter dem Komma (Ausgabe: 10). Hat der vielgepriesene Computer etwa eine Fehlschaltung? Natürlich nicht, Sie können beruhigt davon ausgehen, daß Ihr C 64 richtig arbeitet. Nun haben Sie bereits einen Einblick in die Funktion und Handhabung der Integervariablen. Es handelt sich um eine Zahl, bei der, ohne Rundung, die Stellen nach dem Komma einfach abgeschnitten werden. Eine Integervariable enthält also immer nur ganzzahlige Werte, unabhängig von der Eingabe.

Als nächstes entfernen Sie den STOP-Befehl mit der Eingabe der Zeilennummer 25 und drücken die <RETURN>-Taste. Dann starten Sie das Programm erneut. Daß es völlig sinnlos ist, bei der ersten Zahl Kommastellen einzugeben, wissen Sie inzwischen. Doch wie sieht das nun bei der zweiten aus? Diese Variable verträgt beides, sowohl Integer- als auch Fließkomma-Werte. Eine Variable (BS,A,ZZ) ohne nähere Deklaration (% und $ sind solche Deklarationen) faßt der Computer als Fließkommavariable auf. Mit dem Namen ZZ ist eine derartige Fließkommavariable bereits in Zeile 40 des obigen Programms verwendet worden. Eines allerdings vertragen weder Integer noch Fließkomma-Variablen: die Eingabe von Buchstaben, sogenannte alphanumerische Zeichen (zu den alphanumerischen Zeichen zählen alle per Tastendruck darstellbaren Symbole).

Dafür stellt die Programmiersprache Basic eine weitere Variablenart zur Verfügung, die Sie in unserem kleinen Beispielprogramm bereits verwenden. Es handelt sich um die Zeichenketten-Variablen, die im fachchinesisch auch als Strings bezeichnet werden. Ein String kann alle darstellbaren Zeichen enthalten. Also auch Zahlen aller Art.
Zum Beispiel:
A$=" 64'ER MAGAZIN"

Der Computer erkennt einen String an einem Dollarzeichen, das am Ende des Variablennamens steht. Doch Vorsicht, verfallen Sie nicht der Versuchung jetzt einfach an die numerische Variable in Zeile 40 ein Dollar-Zeichen anzuhängen, an dem der Computer erkennt, daß es sich um eine Stringvariable handelt. Bei der Eingabe ist noch kein Unterschied zu erkennen. Doch bei der Berechnung des Ergebnisses in Zeile 50 bekommen Sie dann ernsthafte Schwierigkeiten. Der Computer steigt mit der Meldung »TYPE MISMARCH ERROR IN 50« aus dem Programm aus, auch wenn Sie nur Zahlen eingegeben haben. Woher soll der Computer auch wissen, daß Sie in eine Stringvariable nur Zahlen eingegeben haben? Der C 64 kann also einen String nie zu Berechnungen heranziehen, da dort seiner Meinung nach alphanumerische Zeichen stehen müßten, und mit denen kann und will er bei Rechenoperationen, stur wie ein Computer nun mal ist, nichts zu tun haben.

Ein solcher String befindet sich in unserem kleinen Programm in Zeile 60. Dort wird der Stringvariablen TEXT$ die Zeichenkette »Ergebnis« zugewiesen. Wird TEXT$ irgendwo im Programm aufgerufen, erfolgt die Verarbeitung der Variablen mit dem zugewiesenen Inhalt.

Sie haben soeben eine neue Art der Zuweisung von Werten an Variablen kennengelernt. Der Variablen werden direkt im Programm Werte oder Zeichenketten zugeordnet. Natürlich können auch Integer- und Fließkomma-Variablen mit festen Werten vorbelegt werden. Eines muß jedoch besonders beachtet werden: Sobald die vorbelegte Variable, hier TEXT$, im weiteren Programmverlauf mit dem Befehl INPUT verwendet wird, der die Eingabe der verwendeten Variablen zuweist, erhält die Variable TEXT$ natürlich einen neuen Inhalt. Dasselbe gilt für die Integer- und Fließkomma-Variablen. Mit den Variablennamen können Sie Ihrer Fantasie beinahe völlig freien Lauf lassen.

Auf starken Widerstand stoßen Sie bei Ihrem C 64 allerdings, wenn Sie Befehlswörter (das sind alle Basic-Befehle und Anweisungen, zum Beispiel RUN, PRINT, INPUT, etc.) als Variablen verwenden wollen. Diese finden nur bei der Programmsteuerung Verwendung. Um ein Programm übersichtlich zu gestalten, und um auch nach zwei Wochen noch zu wissen, was abläuft, sollten Sie immer Variablennamen verwenden, die deren Inhalt näher bezeichnen. So wäre es doch unsinnig, eine Variable, die eine Summe beinhalten soll, mit NAME zu titulieren. Nennen Sie diese Variable doch SUMME. Der C 64 wäre kein Computer, wenn er nicht noch ein paar Besonderheiten mit seiner Variablenverwaltung zu bieten hätte, mit der er selbst den findigen Programmierer noch des öfteren an der Nase herumführt. Um diese Spitzfindigkeiten des Computers zu erklären, geben Sie bitte das folgende kleine Programm ein:

```
10 SUMME1 = 50 + 25
20 SUMME2 = 60 + 40
30 PRINT SUMME1
40 PRINT SUMME2
```

Mit dem bisher Gelernten können Sie mit Sicherheit behaupten, SUMME1 enthält den Wert 75 und SUMME2 den Wert 100. Weit gefehlt. Nach der Programmausführung steht am Bildschirm statt den Werten 75 und 100 lediglich zweimal der Wert 100. Für dieses Phänomen gibt es ebenfalls eine logische Erklärung. Auch hier ist die Ursache des Fehlers, wie übrigens in den meisten Fällen, beim Programmierer zu suchen, der Computer arbeitet richtig. Er erkennt eine Variable nur an den ersten beiden Zeichen des Namens. Das bedeutet, daß Sie die ersten beiden Buchstaben immer unterschiedlich benennen müssen, damit der Computer sie auch unterscheiden kann. Im obigen Beispiel arbeitet der C 64 intern nur mit der Variablenbezeichnung »SU«.

Um dieses Manko zu beseitigen, könnte man logischerweise die Variablen in 1.SUMME und 2.SUMME umbenennen. Warum auch nicht. Versuchen Sie es doch einmal. Leider, so werden Sie feststellen, führt auch dieser Versuch nur zu einem SYNTAX ERROR IN 101. Ein SYNTAX ERROR in Zeile 101? So lang ist das Programm

nun wirklich nicht. Holen Sie sich die einzelnen Zeilen noch einmal mit der Anweisung LIST auf den Bildschirm. Die Zeile 101 existiert tatsächlich und sogar eine Zeile mit der Nummer 202 hat sich eingeschlichen. Wie kann es zu einer derartigen Verunstaltung kommen? Die Antwort liegt beim Interpreter. Der Interpreter gehört zu den Innereien (gemeint sind die verschiedenen Bauteile), die für den Ablauf Ihres Programms zuständig sind. Da der Computer die von Ihnen eingegebenen Befehle in dieser Form nicht versteht, müssen diese in eine »maschinengerechte« Sprache übersetzt (interpretiert) werden. Der Interpreter fungiert als Dolmetscher zwischen Ihnen und dem Computer. Damit der Interpreter das Programm auch in der richtigen Reihenfolge abarbeiten kann, benötigt er Zeilennummern. Vielleicht ist Ihnen schon klar geworden, warum der C 64 bei 1.SUMME und 2.SUMME aussteigt. Die Zahl am Anfang des Variablennamens verbindet der Interpreter mit der Zeilennummer, sie ist für ihn Bestandteil der Nummer.

Mit Recht werden Sie jetzt behaupten, daß diese Zahl zwar das Programm verfälscht, aber doch eigentlich nicht zu dem produzierten Syntax-Error führen kann. Dafür sorgt nämlich der Punkt nach der Zahl (1.SUMME). Der Computer verträgt an vorderster Front (das heißt an erster Stelle des Variablennamens) keinerlei Sonderzeichen, im weiteren Namensaufbau dürfen diese, außer dem Prozent- und Dollarzeichen sowie Komma, Punkt, Nummernzeichen (#), Pfeile (↑, ←), Ausrufungszeichen und Doppelpunkt, ohne weiteres verwendet werden.

C 64 — das Feldergenie

Ihr C 64 ist zum Glück mit Integer-, Fließkomma- und String-Variablen noch nicht ganz ausgelastet. Die neuen Zauberwörter heißen DIM und Array (Bild 1). Mit der Anweisung DIM können Sie Felder definieren. Felder (vom Aufbau her als Tabellen zu verstehen) kennen Sie aus der Mathematik oder ganz einfach aus der Tageszeitung, beispielsweise die Aktienkurse (Bild 2). Diese sind in einem sogenannten zweidimensionalen Feld aufgebaut. Eine Spalte (1. Dimension) beinhaltet den Firmennamen, die zweite und dritte Spalte (2. Dimension) den Tageskurs. Genauso ist auch ein Feld in Basic aufgebaut. Natürlich können Sie in Basic selbst wählen, wieviele Dimensionen (Spalten) Ihr Feld haben soll. Dabei müssen Sie allerdings beachten, daß dabei sehr schnell der Speicher des C 64 voll wird (DIM A$ (300,300,300,300) führt zu einem »OUT OF MEMORY ERROR«).

Ein Feld besteht aus mehreren Zeilen und Spalten. Die genaue Anzahl wird in der DIM-Anweisung (siehe C 64-Handbuch S. 118) festgelegt. Mit der Anweisung DIM BSP$(20) (BSP$ ist der Name der Feldvariablen, 20 gibt die Anzahl der Feld-Elemente an) definieren Sie ein Feld BSP$ mit 21 Elementen (erstes Element: 0). Diese Elemente sind alle vom Typ String und einzeln ansprechbar. Sämtliche Elemente können mit unterschiedlichen Daten belegt werden. Sie können auch ohne weiteres Felder vom Typ Integer oder mit Fließkomma-Variablen erstellen. Wo liegt nun der Vorteil dieser Methode? Anhand einiger Beispiele wird dies veranschaulicht werden. Nehmen Sie noch einmal die Aktienkurse Ihrer Tageszeitung zur Hand. Stellen Sie sich vor, Sie müßten ein Programm erstellen, mit dessen Hilfe alle Kurse erfaßt, danach abgespeichert, mit alten verglichen und ausgewertet werden. Wenn Sie nun für jedes Element dieses umfangreichen Kursfeldes eine eigene Variable definieren müßten, würden Sie recht bald die Übersicht verlieren.

Hier wäre es am besten drei Felder anzulegen. Ein String-Feld für die Firmennamen und je ein Fließkomma-Feld für den Tages- und Vortageskurs. Wenn die 50 interessantesten Aktien verarbeitet werden sollen, könnte eine derartige Information etwa so aussehen:

```
10 DIM FIRMEN$(49)
20 DIM TAGES(49)
30 DIM VRTAG(49)
```

Die einzelnen Elemente der drei Felder können mit einer Integervariablen indiziert (zugeordnet) werden. Durch den Befehl »PRINT FIRMEN$(3)« wird das vierte (das Feld beginnt mit Element Null) Element auf dem Bildschirm ausgedruckt. Wie Sie diese Felder sinnvoll einsetzen können, erfahren Sie in einer späteren Ausgabe, denn hierzu sind ausführliche Programmbeispiele vonnöten, zu denen erst noch die erforderlichen Basic-Kenntnisse im Laufe der Zeit vermittelt werden.

Verlassen wir nun das etwas trocken-theoretische Metier der Felder und wenden wir uns der Computer-internen Abfallbeseitigung zu.

Müll und Computer...

Die gibt's tatsächlich. Allerdings haben die C 64-Müllmänner nicht nur ein paar Stäubchen zu entfernen, sondern viele Kilobyte internen Speicher zu säubern. Wie kann nun eigentlich in einem Computer, der doch von außen ziemlich sauber aussieht, überhaupt Schrott entstehen? Dafür ist wieder einmal der bereits angesprochene Interpreter verantwortlich. Eine Variable, egal welcher Art, gilt dem Computer als bekannt, sobald sie zum ersten Mal im Programm auftaucht. Für die Variable wird dann ein bestimmter Speicherbereich reserviert. Bei der nächsten Zuweisung an dieselbe Variable wird diese jedoch nicht etwa am selben Platz abgelegt (und überschrieben), sondern an einer neuen, bisher noch nicht belegten Stelle abgelegt. Auf diese Weise belegt der Computer ständig neuen Speicherplatz für ein und dieselbe Variable. In einem Programm, in dem eine große Anzahl von Variablen und Feldern verwendet wird, ist der verbleibende freie Speicherplatz irgendwann voller »Variablenmüll«. Dieser muß natürlich weggeräumt werden. Wenn Sie eine solche Prozedur bereits miterlebt haben, wissen, Sie wie lange das dauern kann. Der Computer verabschiedet sich ohne jede Meldung, für den Einsteiger oftmals der erste Schock mit seinem neuen Gerät. Nach einiger Zeit arbeitet der C 64 weiter, so als ob nichts gewesen wäre.

In der Zwischenzeit aber hat die interne Müllabfuhr Akkordarbeit geleistet. Je mehr Variablen und Felder Sie in einem Programm verarbeiten, desto schneller kann es zu dieser im Computer-Chargon als »Garbage Collection« bezeichneten Situation kommen. Doch keine Sorge, bei den ersten Beispielprogrammen, die Sie zu Lernzwecken eintippen, wird dies noch nicht der Fall sein.

Nachdem Sie nun einen Einblick in die Welt der Variablen erhalten haben, lassen Sie Ihr neu erworbenes Wissen noch einmal Revue passieren, vertiefen Sie Ihre Kenntnisse durch selbstgeschriebene kleine Beispielprogramme. Die meisten Fehler lernen Sie zu vermeiden, indem Sie hartnäckig Ihren Computer mit eigenen Programmen füttern. (rf)

Variablentyp	Merkmale (Deklaration)	Inhalte
String	$	alphanumerisch
Fließkomma	keine	numerisch
Integer	%	numerisch
Felder	Anweisung DIM vorangestellt;	alle Typen zugelassen; mit Indexfeld abschließen

Bild 1. Die Variablenarten im Überblick

AKTIEN	17.7.	16.7.
AAch.u.Mü.Bet.	2020,—	2010,—
AAch.u.Mü.Vers.	1560,—	1580,—
Adca-Bank	158,—	156,—
Adlerwerke	290,—	290,—
*Agrob St.	175,—	175,—
do. Vorzüge	130,—	138,—
Aigner, Etienne	200,—	200,—
*Aktbr.Kaufb.	1077,—	1077,—
Allianz Lebensver.	6150,—	6390,—
Audi	621,—	632,—

Bild 2. Kurs-Feld in der Süddeutschen Zeitung