TEIL 2 SYNTHETISCHE

Benutzen Sie die synthetischen Steuerzeichen, um Ihre Druckerlistings übersichtlicher

Nachdem ich in der Ausgabe 5/84 des 64'er-Magazins die Erzeugung synthetischer Steuerzeichen und ihre Wirkung bei Bildschirmbetrieb dargestellt habe, soll jetzt von neuen Möglichkeiten für Druckeranwender die Rede sein. Dabei beziehe ich mich auf den Drucker VC 1515, mit dem die Steuerzeichen ausgetestet wurden. Es ist nicht auszuschließen, daß der eine oder andere Druckertyp unterschiedlich reagieren wird. Die wichtigsten Steuerfunktionen dürften jedoch auf allen Geräten die gleichen Reaktionen hervorrufen.

Bevor es nun zur Sache geht, sei mir — vor allem für neu hinzugekommene Leser — ein kurzer Abriß des vorangegangenen Artikels erlaubt:

Zur Cursor-Steuerung, für RVS ON/OFF oder zum Beispiel für Farbumschaltungen stehen dem VC 20- beziehungsweise C 64-Anwender zwei Möglichkeiten zur Verfügung. Erstens kann die CHR$-Funktion genutzt werden (Beispiel ?CHR$(19) bewirkt HOME) und zweitens erlauben reverse Steuerzeichen eine direkte und kurze Eingabe der Steuerbefehle (Beispiel: das reverse S veranlaßt ebenfalls HOME). Nun existieren jedoch einige CHR$-Codes, für die äquivalente Steuerzeichen nicht über zugehörige Tasten abgerufen werden können. So schaltet CHR$(14) beispielsweise auf Kleinbuchstaben um. Mit Hilfe der ASC-Funktion zeigt sich, daß das reverse N den CHR$-Code 14 trägt. Damit kann dieses Zeichen ebenfalls zur Steuerung herangezogen werden. Da außer den konventionellen Steuerzeichen keinerlei reverse Symbole in Strings vorkommen können, muß das reverse N quasi künstlich erzeugt werden (daher der Name »Synthetische Steuerzeichen«). Aber das ist kein Problem! Wir haben bereits ein Eingabeverfahren kennengelernt, das ich aber jetzt nicht wiederholen möchte. Statt dessen möchte ich Ihnen ein anderes Verfahren zeigen, das vielleicht ein wenig übersichtlicher ist, als das im vorigen Heft beschriebene.

Geben Sie die Programmzeile, die ein im String stehendes Steuerzeichen erhalten soll, wie gewohnt ein. Reservieren Sie dabei jedoch mittels Space die Stelle im Textstring, wo später das synthetische Steuerzeichen stehen wird. Schließen Sie die Eingabe der Zeile mit Betätigung der Return-Taste ab. Nun können sie ohne Schwierigkeiten den Cursor auf die bewußte Stelle bewegen, durch gleichzeitiges Drücken von CTRL und RVS ON in den Revers-Mode schalten (es erscheint kein reverses R!) und schließlich den freigehaltenen Platz mit dem entsprechenden Reversezeichen belegen (in unserem Beispiel also mit dem reversen N). Die so ergänzte Programmzeile wird jetzt durch erneutes Drücken der Return-Taste verlassen. Fertig. Auf diese Weise lassen sich sowohl die altbekannten als auch die neuen Steuerzeichen leicht erzeugen und entsprechend ins Programm einfügen. In den weiteren Ausführungen werde ich die synthetischen Steuerzeichen vereinfachend in geschweiften Klammern darstellen (zum Beispiel reverses N = {N}).

Die Drucker-»Synthies«

Nun zum eigentlichen Thema: Ich könnte fast wetten, daß einige Druckerbesitzer unter unseren Lesern inzwischen beim Experimentieren mit synthetischen Steuerzeichen nicht schlecht gestaunt haben. Denn auch hier eröffnen sich neue Möglichkeiten, an die bislang nicht zu denken war. So sind zwar keine spektakulären Effekte in laufenden Programmen zu erzielen, dafür jedoch hat man erstmals die Möglichkeit, Druckerlistings zu manipulieren. Zuvor möchte ich Ihnen jedoch eine Liste der Steuerzeichen geben. Es ist zu beachten, daß die mit *) gekennzeichneten Steuerzeichen nicht zu den synthetischen zählen, da sich diese direkt über Tasten eingeben lassen. Sie wurden nur der Vollständigkeit halber mit in die Tabelle aufgenommen.

```
TEST⇦LAUF


10 REM GRAPHIK-MODUS
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST⇦LAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 1

Steuerzeichen	CHR$-Code	Wirkung
{H}	8	Umschaltung auf Grafik-Modus
{J}	10	Zeilenvorschub
{M}	13	RETURN. Nachfolgende Zeichen werden nicht mehr gedruckt.
{N}	14	Umschaltung auf Breitschrift
{O}	15	Umschaltung auf normale Schriftbreite
{P}	16	Festlegung der Druckstartposition
*){Q}	17	Umschaltung auf Kleinbuchstaben
*){R}	18	RVS ON
{Z}	26	Zeichenwiederholung (Grafik)
{[}	27	Punktadresse für Druckstart
{SHIFT M}	141	SHIFT RETURN. Nachfolgende Zeichen werden in einer neuen Zeile gedruckt.
*){SHIFT Q}	145	Umschaltung auf Großbuchstaben
*){SHIFT R}	146	RVS OFF

Tabelle. Liste der Steuerzeichen

Nun zu den Erläuterungen derjenigen Zeichen, die sinnvolle Anwendungen erlauben.

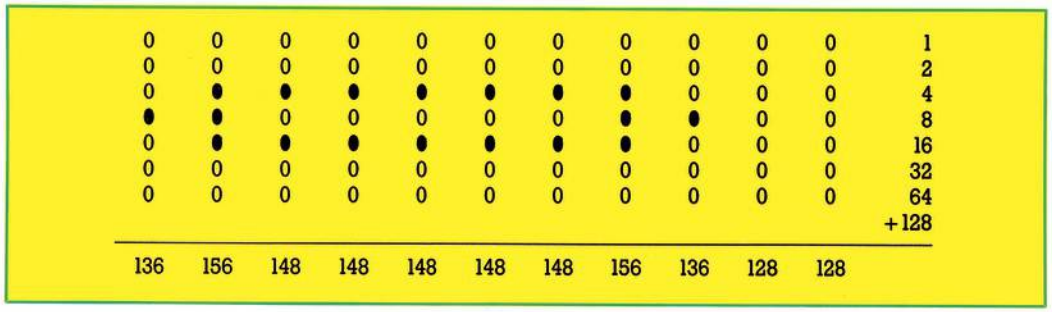
Grafik im Listing

Das reverse H gestattet die Umschaltung des Druckers in den Grafik-Modus. Alle nachfolgenden Zeichen im String werden — sofern ihr jeweiliger CHR$-Code größer als 127 ist — als Grafikinformationen interpretiert. In diesem Modus bleibt der Drucker so lange, bis er mit {N} oder {O} auf Schriftbetrieb zurückgeschaltet wird. Betrachten Sie bitte das Beispiel 1.

Zwischen »TEST« und »LAUF« befindet sich ein selbstdefiniertes Grafikzeichen, das sowohl im Programmlauf als auch im Listing ausgegeben wird. Auch wenn Sie es zunächst nicht glauben — Sie sehen ein Original-Listing und nicht etwa gePRINTete Textzeilen. Ich will Ihnen den Trick ver-

STEUERZEICHEN

zu machen. Es gibt interessante Möglichkeiten.



0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	2
0	●	●	●	●	●	●	●	0	0	0	4
●	●	0	0	0	0	0	●	●	0	0	8
0	●	●	●	●	●	●	●	0	0	0	16
0	0	0	0	0	0	0	0	0	0	0	32
0	0	0	0	0	0	0	0	0	0	0	64
											+128
136	156	148	148	148	148	148	156	136	128	128	

Bild 1. Vergrößertes Grafiksymbol aus Beispiel 1

raten. In einer Vergrößerung sieht das Grafiksymbol aus wie im Bild 1 gezeigt.

Die Umrechnung des dualen Bitmusters in Dezimalzahlen liefert diejenigen CHR$-Codes, durch die das Zeichen spaltenweise beim Drucken entsteht. Man muß also nur wiederum synthetische Zeichen finden, die die gesuchten CHR$-Codes tragen. Bei diesem Beispiel ergibt sich:

"TEST {H}	{SHIFT H}	{SHIFT MINUS}	{SHIFT T}	{SHIFT T}	{SHIFT T}	
8	136	156	148	148	148	
{SHIFT T}	{SHFIT T}	{SHIFT MINUS}	{SHIFT H}	{SHIFT*}	{SHIFT* }	{O}LAUF "
148	148	156	136	128	128	15

Mit dieser Folge von reversen Zeichen erzielen Sie also den Ausdruck, der im Beispiel 1 abgebildet ist.

Folgt dem {H} keine Grafikinformation, das heißt ist der CHR$-Code kleiner als 128, so kommt es zum Abbruch des Listings. Diese Tatsache kann man sich bewußt zunutze machen, wenn man ohne größeren Programmieraufwand einen Drucker-Listschutz in sein Programm einbauen will. Der Bildschirmbetrieb leidet darunter nicht, da {H} dort nur eine Verriegelung der Umschaltung zwischen Groß- und Kleinschrift bewirkt.

Zeilenvorschub

Die Beispiele 2.1 und 2.2 zeigen daß {J} einen Zeilenvorschub mit Rückwagenlauf verursacht. In 2.1 befindet sich das reverse J zwischen »TEST« und »LAUF«, bei 2.2 am Schluß des Strings. Da man das zweite Anführungszeichen eines Strings weglassen kann, sofern keine weiteren Basic-Befehle mehr in dieser Zeile folgen sollen, ergibt sich so die Möglichkeit, echte Leerzeilen mit {J} im Listing zu erzeugen

```
TEST
LAUF


10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST
LAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 2.1

```
TESTLAUF

10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF

50 PRINT#4:CLOSE4

READY.
```

Beispiel 2.2

Breitschrift — Schmalschrift

Während {N} beim Bildschirmbetrieb auf Kleinbuchstaben umschaltet, zeigt der Drucker eine andere Wirkung. Er wird veranlaßt, alle nachfolgenden Zeichen in doppelt breiter Schrift auszugeben. Es ist also bei der Anwendung von {N} Vorsicht geboten, da ein für den Bildschirm konzipiertes Programm beim Drukkerlisting unerwünschte Steuerungen zur Folge haben kann. In diesen Fällen sollte man {N} wieder durch den konventionellen Befehl CHR$ ersetzen.

Wollen Sie jedoch die Breitschrift zur übersichtlichen Gestaltung eines Listings nutzen, so können Sie ruhig das {N} hier und da im Programm verstecken. Es bleibt demjenigen, der das Programm lesen soll, sowieso verborgen. Wenn nur eine Hervorhebung einzelner Worte, Befehle oder Zeilen gewünscht wird, muß die Breitschriftphase entsprechend mit {O} beendet werden. Im Beispiel 3 sehen Sie dieses Vorhaben realisiert. Hier bewirkt das {N} vor dem Wort »Test« die Umschaltung auf Breitschrift, während ein {O} vor »Lauf« diesen Modus beendet.

```
TESTLAUF


10 REM BREITSCHRIFT
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 3

Druckstartposition

Die Druckstartpositionierung — vergleichbar mit TAB — wird normalerweise mit CHR$(16) und nachgestellter Startadresse vorgenommen. Also etwa:

PRINT#4,CHR$(16)"10TEST"

Diese Zeile druckt beginnend in Spalte 10 nur das Wort »TEST« aus. Die im String vorangestellte Zahl dient dabei als Startadresse. Das kürzere Äquivalent sieht nun wie folgt aus:

PRINT #4,"{P}10 TEST"

Doch Vorsicht! Obwohl diese Anweisung während

des Programmlaufs korrekt ausgeführt wird, verschluckt sie der Drucker im Listing vollständig. Das 4. Beispiel zeigt, daß von {P}10 zwischen »TEST« und »LAUF« nicht mehr zu bemerken ist.

```
TEST      LAUF


10 REM DRUCKSTARTPOSITION
15 REM IST IM LISTING
20 REM NICHT SICHTBAR
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 4

Damit verbietet sich die Anwendung des synthetischen {P} bei Programmen, die zum Beispiel zur Veröffentlichung vom Drucker dokumentiert werden sollen.

Für die Zeichen {Z}Zeichenwiederholung und }[] (Punktadresse für Druckerstart) haben sich bislang noch keine sinnvollen Einsatzmöglichkeiten ergeben. Sie sollen daher jetzt nicht weiter diskutiert werden. Auch auf die Beschreibung von {Q}, {R}, {SHIFT Q} und {SHIFT R} möchte ich verzichten, da sie keine synthetischen Zeichen sind und sich damit «ganz normal« verhalten.

Ich persönlich benutze die synthetischen Steuerzeichen gern zur Gestaltung von Listings. Wie so etwas aussehen kann, habe ich Ihnen im abschließenden Beispiel 5 zusammengestellt.

Vielleicht versuchen Sie einmal, die versteckten und unsichtbaren Steuerzeichen herauszufinden.

(Jürgen Wagner)

```
100 REM DEMO ZUR OPTISCHEN STRUKTURIERUNG
110 REM      VON DRUCKER-LISTINGS MIT
120 REM"     SYNTHETISCHEN
130 REM"     STEUERZEICHEN


200 REM"*  HAUPTPROGRAMM  *

210 FORI=1TO64
220 PRINT"64'ER-MAGAZIN
230 GOSUB300:NEXT:"
240 END


300 REM"*  UNTERPROGRAMM  *

310 FORJ=1TO50:NEXT
320 RETURN:"


400 REM"ORIGINAL-LISTING!"

READY.
```

Beispiel 5

Autostart in Theorie und Praxis

Welcher Anwender hat sich nicht schon immer gewünscht, seine Programme »so einfach wie möglich« in den Computer zu bringen. Geräte der oberen Preis-/Leistungsklasse sind zu diesem Zweck mit einem Autostart-Mechanismus ausgerüstet. Dabei wird das Programm ohne weiteres Zutun nach dem Anschalten des Gerätes vom Massenspeicher (Floppy) in den Computer geladen. Hier soll jetzt die Realisierung eines solchen Autostarts auf einem C 64, beginnend mit der dazu nötigen Theorie, beschrieben werden.

Es gibt beim C 64 mehrere Möglichkeiten, einen Autostart herbeizuführen. Ich will Ihnen hier eine weniger bekannte Methode vorstellen:

Die Stack-Manipulation.

Der Stack belegt beim C 64 den Bereich $0100 bis $01ff (256 bis 511). Er wird unter anderem zum Ablegen von Rücksprungadressen benutzt. Nach Beendigung einer Routine sucht sich der Prozessor vom Stack zwei Bytes (nämlich die Rücksprungadresse in das Hauptprogramm) und springt die Adresse, die sich aus diesen beiden Bytes ergibt, an. Eigentlich ist daraus schon zu ersehen, was zu tun ist: Man müßte diese Rücksprungadresse so ändern, daß das Programm nicht an die eigentliche Rücksprungstelle springt, sondern auf eine eigene Routine. Jetzt jedoch tauchen schon die ersten Probleme auf: Wie soll der Stack geändert werden, ohne daß man dafür ein Extra-Programm braucht? Wie soll sich das Programm nach dem Ladevorgang automatisch starten?