»PRINT AT« und »RESTORE N«

Die hier vorgestellte kleine Basic-Erweiterung bringt zwei häufig benötigte Funktionen, nämlich die PRINT-Ausgabe an einer wählbaren Bildschirmposition und das Setzen des DATA-Zeigers auf einen bestimmten Datensatz. Zusätzlich ist noch die Möglichkeit gegeben, beliebige Bildschirmzeilen zu löschen.

In vielen Leserbriefen wurde nach einem geeigneten »PRINT AT« gefragt. Da bislang keine optimale Lösung angeboten wurde, habe ich nun einen »PRINT AT-Ersatz« geschrieben. Diese Ersatzroutine arbeitet sehr zuverlässig und ist denkbar einfach anzuwenden. Das 146 Bytes lange Maschinenprogramm enthält außerdem eine Routine, mit der man beliebig viele Bildschirmzeilen löschen kann und eine Erweiterung des Restorebefehls, nämlich »RESTORE N«. Es ist nun möglich, die Data-Zeile, aus der das nächste Statement gelesen werden soll, selbst zu bestimmen, ohne die nichtbenötigten Daten zu überlesen. Der Datazeiger kann zum Beispiel mit »RESTORE 1100« direkt auf Zeile 1100 gestellt werden und beim ersten »READ« wird das erste Statement aus Zeile 1100 gelesen. Beim Umnumerieren eines Programmes mit diesem Befehl ist die Zeilenangabe von Hand zu ändern, da Renumberroutinen diesen Befehl nicht berücksichtigen. Der Restore-Befehl kann nach wie vor auch ohne Angabe einer Zeilennummer verwendet werden.

Für die Bildschirmzeilen-Löschroutine steht ein Kurzbefehl zur Verfügung. Die Syntax ist »£A,B«. In A steht die Nummer der obersten, in B die Nummer der untersten zu löschenden Zeile. Beispiele:

£0,22 löscht den ganzen Bildschirm
£0,0 löscht die oberste Zeile
£4,9 löscht BS-Zeilen 4 bis 9 (5. bis 10. von oben)

Das besondere an dieser Einrichtung ist, daß die Cursorposition dabei nicht verändert wird. Folgt der »£«-Befehl auf eine IF..THEN-Abfrage, so ist vor dem »£«-Befehl ein »:« zu setzen. Wird er vergessen, meldet der Interpreter einen »Syntax Error«. Die Werte von A und B müssen zwischen 0 und 22 liegen.

Auch die PRINT AT-Routine wird mit einem Kurzbefehl aufgerufen. Das Kurzzeichen ist der Klammeraffe (»@«) oder besser das »AT«-Zeichen. Die Syntax lautet: »@, X, Y; Ausdruck«. X steht für waagerecht, Y für senkrecht.

X darf Werte zwischen 0 und 21, Y Werte zwischen 0 und 22 erreichen. Andernfalls wird ein »Illegal Quantity Error« ausgegeben. Für X und Y sind möglich:

1. Direkte Zahlenangabe
2. Wertangabe in Form einer Variablen
3. Eine Formel als Koordinate.

Der Ausdruck kann Text (in Hochkommata eingeschlossen, Strichpunkt kann entfallen) oder eine Stringvariable (Strichpunkt unbedingt erforderlich) sein. Der Nullpunkt der Koordinaten ist die obere, linke Ecke. Einige Beispiele:

@0,0"V C - 2 0"	Text beginnt oben links
@X,Y"V C - 2 0"	Text erscheint an gegebener Position
@A+B,3+A;A$	String A$ wird gedruckt
@0,10;	Cursor wird für »Input« positioniert

Auch hier ist bei einer IF...THEN Abfrage zwischen »THEN« und dem »@«-Befehl ein »:« zu setzen.

Das Programm läuft sowohl auf der VC 20-Grundversion als auch mit Erweiterung, nicht jedoch mit der 40-Zeichen-Karte.

Der Basiclader (Listing) lädt das Maschinenprogramm ans Speicherende. Hierzu wird der für Basic zur Verfügung stehende Speicherplatz um 256 Bytes verkürzt. Dies übernimmt Zeile 10. In Zeile 20 wird die Anfangsadresse berechnet und dann das Maschinenprogramm in den Speicher gepoket. Nebenbei wird eine Prüfsumme gebildet. Sollte diese nicht gleich 17442 sein, wird der Programmablauf in Zeile 70 abgebrochen. Ist der Prüfsummencheck OK (dies meldet das Programm), wird das Maschinenprogramm aufgerufen (Zeile 90). Abschließend meldet der VC 20 Interpreter »READ« und die Befehle stehen nun zur Verfügung. Wenn das Maschinenprogramm an einer bestimmten Stelle des Speichers stehen soll, kann dies durch Löschen der Programmzeile 10 und Ändern der Zeile 20 in »20 ES=[ADRESSE]« erreicht werden. Einen geeigneten Platz bietet zum Beispiel die 3KByte-Erweiterung, wenn Block 1 erweitert (8KByte oder 16KByte) ist. Auch der Kassettenpuffer (AB 828) ist verwendbar, jedoch nur bei Diskettenbetrieb empfehlenswert, da das Programm dann bei jeder Kassettenoperation gelöscht wird.

(Jürgen Reinert/ev)

```
1 REM ************************
2 REM * PRINT AT / RESTORE N *
3 REM *       (C) 1983       *
4 REM * BY JUERGEN REINERT    *
5 REM ************************
6 :
7 :
10 POKE56,PEEK(56)-1:CLR
20 ES=PEEK(55)+256*PEEK(56)+2
30 HB=INT((ES+11)/256)
40 LB=ES+11-HB*256
50 FORI=0TO145:READA:POKEI+ES,A
60 S=S+A:NEXT
70 IFS<>17442THENPRINT"■DATA ERROR■":END
80 PRINT"■OK.■"
90 POKEES+1,LB:POKEES+6,HB:SYSES:END
100 FORI=0TO145:READA:NEXT
110 DATA169,11,141,8,3,169,8
120 DATA141,9,3,96,169,199,72
130 DATA169,173,72,32,115,0,240
140 DATA244,201,140,240,8,201
150 DATA64,240,38,208,76,234
160 DATA234,32,115,0,240,26,201
170 DATA58,240,22,32,138,205
180 DATA32,247,215,32,19,198
190 DATA56,165,95,233,1,164
200 DATA96,176,1,136,76,39,200
210 DATA76,29,200,32,115,0,32
220 DATA158,215,224,22,176,24
230 DATA138,72,32,253,206,32
240 DATA158,215,104,224,23,176
250 DATA11,168,24,32,240,255
260 DATA32,121,0,76,160,202,76
270 DATA72,210,234,234,234,201
280 DATA92,240,4,56,76,237,199
290 DATA32,115,0,32,235,215,224
300 DATA23,176,232,164,20,192
310 DATA23,176,226,32,141,234
320 DATA202,228,20,16,248,76
330 DATA121,0,0,0,0
READY.
```

PRINT AT und RESTORE N für den VC 20