Tips & Tricks zum C 128

Tips & Tricks sind das Salz in der Suppe. Hier wieder ein paar Kniffe, um Ihrem C 128 beizukommen.

Diesmal finden Sie hier einen kleinen Kurs, der Sie mit Hilfe einer Befehlserweiterung Einblick in den Video-Chip des C 128 gewähren läßt. Ebenfalls zu finden: Eine etwas andere Anwendung des WINDOW-Befehls.

WINDOW-Befehl einmal anders

Das Programm »WDGRAF V1.01« (Listing 1) demonstriert eine ungewöhnliche Anwendung des WINDOW-Befehls im Revers-Modus des 80-Zeichen-Modus auf dem C 128. Auf den 40-Zeichen-Bildschirm hat das Programm keine Wirkung.

Es stellt Werte zwischen 1 und 24 als Balkendiagramm dar. Dies geschieht durch den WINDOW-Befehl, der den Balken bildet und dann schließlich mit einer Farbe auffüllt.

Die einzelnen Größen der Balken sind in Zeile 410 in Form von DATAs abgelegt und sollten bei Veränderungen im Bereich von 1 bis 24 liegen.

Die Breite der Balken ist in der Variablen GR (Zeile 150) abgelegt und kann beliebig verändert werden. Man sollte aber die Anzahl der einzulesenden DATAs (Variable AZ, Zeile 160) in Zeile 410 mit berücksichtigen und sie entsprechend ändern.

Versuchen Sie doch einmal für die Variable GR in Zeile 150 Werte zwischen 1 und 5. Wie man aus dem Programm ersieht, kann der WINDOW-Befehl des Basic 7.0 durchaus sinnvoll in Programmen verwendet werden, obwohl die Wahl der Parameter doch sehr beschränkt ist und die Windows auch nicht zwischengespeichert werden. (Udo Miller/dm)

```
140 FAST : REM                              FAST-M
    ODUS EINSCHALTEN
150 GR=4: REM                              BREITE D
    ER BALKEN FESTLEGEN
160 AZ=10: DIM A(AZ): REM                   AZ BA
    LKEN AUS DATAS(ZEILE 270) LESEN
170 COLOR 5,RCLR(6): REM                    ZEICHE
    NFARBE AUF RAHMENFARBE
180 PRINT CHR$(27)+"R": REM                 REVERS
    MODUS EINSCHALTEN
190 PRINT CHR$(27)+"M": REM                 'NO-SC
    ROLL MODUS' EINSCHALTEN
200 F=1: SCNCLR       : REM                 FARBE
    AUF WEISS UND SCREEN LOESCHEN
210 :
220 FOR I=1 TO AZ: READ A(I): NEXT : REM
    EINLESEN DER BALKENWERTE(GROESSE)
230 :
240 X=1: Y=23: REM                          FESTLE
    GUNG DER X UND Y KOORDINATE
250 :
260 B=0: DO UNTIL B=AZ: REM                 AUSWER
    TUNG
270 B=B+1
280 WINDOW X,Y-(A(B)-1),X+GR,Y: REM         BALKEN
     ERZEUGEN
290 IF F>15 THEN F=1
300 F=F+1: COLOR 5,F: SCNCLR : REM             WIN
    DOW MIT ZEICHENFARBE AUFFUELLEN
310 CHAR 1,0,RWINDOW(0),"": REM             CURSOR
     POSITIONIEREN
320 PRINT USING "##";A(B): REM              GROES
    SE DES BALKEN AUSGEBEN
330 X=X+GR*2: REM                          VERSCHI
    EBUNG X-ACHSE BERECHNEN
340 LOOP
350 GET KEY A$: REM                         AUF T
    ASTE WARTEN
360 PRINT CHR$(27)+"L": REM                 'NO-SC
    ROLL MODUS' AUFHEBEN
370 PRINT CHR$(19);CHR$(19);: REM           WINDOW
     VERLASSEN
380 END
390 :
400 REM                                    DATEN BA
    LKENGROESSE
410 DATA 8,14,7,11,21,6,13,22,12,18
```

Listing 1. »WDGRAF V1.01«

Ausgabe eines Textes bei einem Reset

Reset-Text (Listing 2) erlaubt die Ausgabe eines beliebigen Testes bei einem Reset, insofern eine Diskette mit diesem Programm in der Floppystation liegt. Das Programm ist lauffähig im C 128-Modus mit einer Floppy 1541, 1570 oder 1571. Der Start des Programms erfolgt durch RUN.

Eine Programmbeschreibung entnehmen Sie bitte den REM-Zeilen im Programm (Listing 2), die nicht mit eingegeben werden müssen, da sie nicht angesprungen werden.

Sinnvolle Texte zum Ausgeben wären beispielsweise RUN":*" oder GO64. Sie brauchen nach einem Reset nur noch mit <RETURN> quittiert zu werden.

(Thorsten Wewers/dm)

Video-Experimente

Neben den 128 KByte RAM und der schnellen Laufwerk-Schnittstelle gehört die 80-Zeichen-Darstellung sicherlich zu den erfreulichsten Aspekten des C 128.

Um die 80 Zeichen auf dem Bildschirm darzustellen, besitzt der C 128 einen speziellen Video-Controller. Dieser verfügt über einen Speicher von 16 KByte, so daß für die Bildschirmdarstellung kein Basic-Speicherraum verlorengeht. Dieser positive Aspekt wird jedoch getrübt, da die Adressierung dieser 16 KByte über ein Nadelöhr von nur 2 Byte erfolgt.

Der Video-Chip (VIC) besitzt ein Auswahlregister, mit dem — durch POKEs — eines von insgesamt 32 internen Registern angesprochen werden kann. Eine Liste dieser Register und deren Funktionen finden Sie im Anhang E des C 128-Handbuchs.

Nach der Adressierung über die Speicherstelle 54784 ($D600) kann man in das Register neue Werte hineinschreiben (POKE) oder den aktuellen Wert über PEEK abfragen. Diese Ein-/Ausgabe-Operation erfolgt über das zweite Nadelöhr, der Adresse 54785 ($D601).

Nach diesen Erläuterungen nun ein Beispiel, wie Sie die 16 KByte des Video-Speichers ansprechen können:

```
10 AR=54784                 REM Auswahl-Register
20 EA=54785                 REM Ein-/Ausgabe-Reg.
30 AD=80                    REM Adresse (2. Zeile)
40 H=INT (AD/256)           REM High berechnen
50 L=AD-H*256               REM Low berechnen
60 POKE AR,18:POKE EA,H     REM High-Adr.
70 POKE AR,19:POKE EA.L     REM Low-Adr.
80 GETKEY A$:?CHR$(147)     REM Zeichen-Eingabe
90 D=ASC(A$) AND 63         REM in Bildschirm-Code
100 POKE AR,31:POKE EA,D    REM ins DATA-Register
110 POKE AR,30:POKE EA,1    REM Wort-Zaehler = 1
120 GOTO 60                 REM erneut
```

In den ersten beiden Zeilen werden die Adressen des Register- und Ein-/Ausgabe-Bytes definiert. Mit der Variablen AD legt das Programm die Zieladresse fest. Wie eingangs erwähnt, umfaßt das RAM des VIC 16 KByte. Da sich mit einem Byte jedoch nur 256 Speicherzellen adressieren lassen, sind — um den vollständigen Speicherbereich anzusprechen — zwei Bytes nötig. Mit dem ersten Wert erfolgt im Grunde eine grobe Annäherung in Sprüngen zu je 256 Bytes (High-Byte). Den verbleibenden Rest bezeichnet man als Low-Byte. In den Zeilen 60 und 70 werden diese Werte in die Adreßregister gePOKEt. Der erste POKE wählt das Register an, der zweite beschreibt anschließend dieses Register.

In den Zeilen 80 und 90 folgt nun eine Eingabe, die (mit AND 63) in den Bildschirmcode umgewandelt wird. Das Zeichen landet anschließend im Daten-Register (30). Über diese Adresse erfolgen alle Ein- und Ausgaben des Video-Speichers. Mit der letzten Zeile wird dann das Wortzähler-Register angesprochen. Dies legt einerseits fest, wie oft das Zeichen im Daten-Register ausgeben werden soll. Außerdem startet die Adressierung automatisch die Befehlsausführung (in diesem Beispiel die Ausgabe eines Zeichen in der zweiten Bildschirmzeile).

```
10 REM PROGRAMM ZUM AUSGEBEN EINES
20 REM BELIEBIGEN TEXTES BEI RESET
30 :
80 REM PROGRAMM BITTE IM 40-ZEICHEN-
90 REM ASCII-MODUS EINGEBEN!
110 :
120 REM FARBEN AENDERN/INITIALISIERUNG
130 :
140 COLOR 4,12: COLOR 0,13: PRINT CHR$(11) C
    HR$(142): OPEN 1,0: OPEN 15,8,15
150 :
160 REM KONTROLLE, OB 1/0 BEREITS BELEGT
170 :
180 PRINT#15,"B-A 0 1 0": IF DS<20 THEN PRIN
    T#15,"B-F 0 1 0": GOTO 270
190 GOSUB 680: PRINT "{2DOWN} TRACK 1, SEKTO
    R 0 IST BEREITS BELEGT!"
200 PRINT "{2DOWN} TROTZDEM WEITERMACHEN (J/
    N)?"
210 GET KEY JN$: IF JN$="N" THEN 640: ELSE I
    F JN$<>"J" THEN 210
220 :
230 REM EINGABE DES TEXTES UND
240 REM GGF. KORREKTUR
250 :
260 PRINT#15,"B-F 0 1 0"
270 GOSUB 680: PRINT "{2DOWN}TEXT (ENDE: ←):
    ": INPUT#1,T$: IF T$="←" THEN 790
280 PRINT : PRINT "{3DOWN}KORREKTUR (J/N)?"
290 GET KEY JN$: IF JN$="J" THEN 270: ELSE I
    F JN$<>"N" THEN 290
300 :
310 REM DATEI OEFFNEN, 1 / 0 LESEN
320 :
330 OPEN 2,8,2,"#": PRINT#15,"U1 2 0 1 0"
340 :
350 REM CBM-KENNUNG SCHREIBEN
360 :
370 FAST : PRINT#2,"CBM": FOR I=0 TO 4: PRIN
    T#2,CHR$(0);: NEXT
380 :
390 REM MASCHINENPROGRAMM SCHREIBEN +
400 REM TEXT AUS BILDSCHIRMSPEICHER
410 REM LESEN (IN A$)
420 :
430 I=1224: DO UNTIL PEEK(I)=31 OR I=1351: A
    $=A$+CHR$(PEEK(I)): I=I+1: LOOP
440 FOR I=0 TO 13: READ A: IF A=-1 THEN A=LE
    N(A$)-1
450 PRINT#2,CHR$(A);: NEXT
460 :
470 REM TEXT AUF DISKETTE SCHREIBEN
480 :
490 PRINT#2,A$
500 :
510 REM BEARBEITETEN BLOCK AUF DISKETTE
520 REM SCHREIBEN UND DATEIEN SCHLIESSEN
530 :
540 PRINT#15,"U2 2 0 1 0": PRINT#15,"B-A 0 1
     0": DCLOSE
550 :
560 REM KOMMENTAR
570 :
580 GOSUB 680: SLOW : PRINT "{2DOWN} TEXT ST
    EHT AUF DIESER DISKETTE BEREIT!"
590 PRINT "{DOWN} TRACK 1, SEKTOR 0 IST ALS
    BELEGT GE-"
600 PRINT " KENNZEICHNET! NACH EINEM VALIDAT
    E IST{3SPACE}FOLGENDES ERFORDERLICH:"
610 PRINT "{DOWN} OPEN15,8,15," CHR$(34)"B-A
     0 1 0" CHR$(34)"{3SPACE}{RETURN}"
620 PRINT "{2DOWN} BETAETIGEN SIE DOCH EINMA
    L DIE RESET-{3SPACE}TASTE!"
630 GET KEY G$
640 GOSUB 680: END
650 :
660 REM BILDSCHIRMMASKE ERSTELLEN
670 :
680 PRINT "{CLR,GREY1,RVSON,5SPACE}AUSGABE E
    INES TEXTES BEI RESET{5SPACE}";
690 PRINT "{GREY3,6SPACE}(C) 1986 BY THORSTE
    N WEWERS{7SPACE,BLACK}": RETURN
700 :
710 REM DATAS FUER MASCHINENROUTINE
720 :
730 DATA 120,160,-1,185,23,11,153,104,5,136,
    16,247,88,96
740 :
750 REM NACH ABFRAGE LEEREN BLOCK AUF
760 REM DISKETTE SCHREIBEN / SCHLIESSEN
770 REM ALLER KANAELE / END
780 :
790 GOSUB 680: PRINT "{2DOWN} SOLL DER BLOCK
     GELOESCHT WERDEN (J/N)?"
800 GET KEY JN$: IF JN$="N" THEN 640: ELSE I
    F JN$<>"J" THEN 800
810 FAST : OPEN 2,8,2,"#": PRINT#15,"U1 2 0
    1 0"
820 FOR I=0 TO 255: PRINT#2,CHR$(0);: NEXT
830 PRINT#15,"U2 2 0 1 0": DCLOSE : SLOW : G
    OTO 640
```

Listing 2. »Reset-Text«

Starten Sie das Programm. Sobald Sie eine Taste drücken, wird dieses Zeichen in den Bildschirmspeicher gePOKEt. Sie werden jedoch schnell feststellen, daß die Zeichen nicht immer an der gleichen Bildschirmposition ausgegeben werden, sondern oft um eine Stelle nach rechts verschoben sind. Vermutlich ist es unmöglich, eine entsprechende Basic-Routine zu schreiben.

Hier hilft das nächste Programm (Listing 3), das Sie nun bitte eingeben. Da jede DATA-Zeile mit einer Prüfsumme endet, meldet das Programm sofort Eingabefehler mit der dazugehörigen Zeilennummer. Nachdem das Ladeprogramm (bitte zuerst speichern) korrekt abgelaufen ist, können Sie durch SYS 4864 vier neue Basic-Befehle aufrufen.

Einfacher mit PEEK% und POKE%

Falls Sie nun eines der 32 VIC-Register adressieren möchten, dann müssen Sie nach jedem PEEK- oder POKE-Befehl ein »%« hinzufügen. Indem Sie die folgenden Zeilen ändern, können Sie das kleine Test-Programm etwas vereinfachen:

```
 60    POKE% 18,H
 70    POKE% 19,L
100    POKE% 31.D
110    POKE% 30,1
```

Nach dem Programmstart werden Sie zwar feststellen, daß das Zeichen nun nicht mehr hin- und herspringt. Dafür erscheinen aber meist zwei Zeichen auf dem Bildschirm. Dies ist offenkundig ein Hardware-Fehler des Chips, dem man aber mit einem Trick begegnen kann. Falls Sie — an Stelle des POKE-Befehls in den Wort-Zähler — eines der Adreßregister (18,19) ansprechen, dann gibt der VIC ebenfalls das Zeichen aus, und diesmal nur einmal. Übrigens wird dieser Trick auch im Betriebssystem des C 128 angewendet.

Ändern Sie hierzu die Zeile 110 in:

```
110 POKE 18,H      REM Adresse erneut
```

Anschließend erhalten Sie eine Routine, mit der Sie die 16 KByte des Videospeichers gezielt verändern können. Betrachten Sie dies jedoch nur als Beispiel, da Sie das kleine Basic-Grab durch zwei weitere spezielle PEEK-/POKE-Anweisungen ersetzen können.

Folgt auf einen PEEK- oder POKE-Befehl ein »#«-Zeichen, dann führt die Anweisung direkt in das Video-RAM des C 128.

Um es nochmals zu verdeutlichen: PEEK% und POKE% verändern lediglich die 32 Register des VIC. Mit den zwei Anweisungen PEEK#(..) und POKE#(..) hingegen gelangen Sie direkt in das Video-RAM. Sollten Sie aber hinter dem VIC mehr als 16 KByte RAM vermuten (zum Beispiel POKE 20453,12), dann meldet sich der Computer mit einem »ILLEGAL QUANTITY ERROR«.

Ausgehend von diesen vier Anweisungen (normale PEEKs und POKEs bleiben Ihnen natürlich erhalten) können Sie nun sehr einfach den Video-Controller des C 128 erforschen.

Die VIC-Speicheraufteilung

Grundsätzlich gliedert sich der VIC-Speicher in vier Abschnitte von zwei oder vier KByte Länge. Die Adreßpositionen dieser Teile können über entsprechende Register des Controllers verschoben werden. Nach dem Einschalten legt der Computer folgende Einteilung fest:

```
    0 -  2048 :   Bildschirmspeicher
 2049 -  4095 :   Farbspeicher
 4096 -  8191 :   Frei
 8192 - 12287 :   Zeichensatz A
12288 - 16383 :   Zeichensatz B
```

Der Befehl POKE# 0,1 schreibt beispielsweise ein A in die linke obere Bildschirmecke.

Wesentlich interessanter ist jedoch der Attributspeicher. Wie Sie der nachfolgenden Tabelle entnehmen können, hat dabei jedes Bit eine spezielle Bedeutung.

Bit :	Wert :	Funktion	Bit :	Wert :	Funktion
0	1	Intensität	4	16	Blinken
1	2	Blau	5	32	Unterstreichen
2	4	Grün	6	64	Invers
3	8	Rot	7	128	Zeichensatz A/B

Hierzu ein Beispiel: Schreiben Sie ein Zeichen in die linke obere Bildschirmecke. Wenn Sie nun POKE# 2048,16+32 eingeben, blinkt das Zeichen, wird jedoch zusätzlich unterstrichen.

Interessant ist auch das siebte Bit. Vielleicht haben Sie sich schon einmal gewundert, daß Grafikzeichen — nach dem Umschalten auf Groß/Kleinschreibung — weiterhin auf dem Bildschirm sichtbar bleiben. Die Auswahl zwischen den beiden Zeichensätzen erfolgt über das siebte Bit des Attributspeichers, so daß der C 128 gleichzeitig 512 unterschiedliche Zeichen darstellen kann.

Der Umgang mit den Zeichensätzen ist jedoch etwas umständlich. Jedem Zeichen sind acht Bytes im Video-Speicher zugeordnet. Die Startadresse eines Zeichens ermitteln Sie einfach, indem Sie den Bildschirmcode mit 16 multiplizieren. Zu diesem Wert müssen Sie anschließend noch die Startadresse des Zeichensatzes addieren. Ausgehend von dieser Adresse legen die folgenden Bytes das Aussehen dieser acht »Mikro-Zeilen« eines Zeichens fest.

Mit POKE 8192+2*16+7,255 wird beispielsweise die unterste »Zeile« des B (1. Zeichensatz) mit einer Linie versehen. Das Zeichen erscheint anschließend unterstrichen auf dem Bildschirm.

Vielleicht schreibt einer der Leser ein Programm, mit dem

man neue Zeichensätze entwerfen kann. Allerdings sollte das Programm einen Zusatz enthalten, damit man diese Daten leicht mit anderen Programmen verbinden kann.

Zuletzt noch zwei Anregungen. Das Register 7 des VIC bestimmt die Vertikal-Synchron-Position des Controllers. Den Originalwert können Sie mit PRINT PEEK%(7) abfragen (Wert = 32). Falls Ihr Bildschirm etwas nach unten verschoben ist, können Sie dies einfach mit POKE% 7,33 korrigieren. Bei kleineren Werten rutscht der Bildschirm entsprechend nach unten.

Versuchen Sie auch einmal POKE% 9,8 (Originalwert = 7).

(Heino Velder/dm)

```
10 REM ********************************
20 REM *                              *
30 REM * PEEK & POKE ZUM VIDEO-CHIP   *
40 REM *                              *
70 REM ********************************
80 :
85 REM PROGRAMM-AUFRUF: SYS 4864
90 :
100 READ D$  : AD=DEC(D$)     : ZL=1001
110 FOR I=1 TO 10 : READ D$   : D=DEC(D$)
120 SU=SU+D  : POKE AD,D      : AD=AD+1
130 NEXT     : READ P$        : PR=DEC(P$)
140 D=PR-32768 : IF D=>0 THEN PR=D
150 IF PR<>SU THEN PRINT "FEHLER IN",ZL: STOP
160 ZL=ZL+1: SU=0: IF D<0 THEN 110 : ELSE END
170 :
180 :
190 :
1000 DATA  1300
1001 DATA A9,1F,8D,04,03,A9,13,8D,05,03,  02AD
1002 DATA A9,95,8D,08,03,A9,13,8D,09,03,  032B
1003 DATA A9,4D,8D,0A,03,A9,13,8D,0B,03,  02E7
1004 DATA 60,20,0D,43,A2,FF,E8,BD,00,02,  0418
1005 DATA F0,22,C9,20,F0,F6,C9,30,90,04,  056E
1006 DATA C9,3A,90,16,A2,FF,E8,BD,00,02,  04F1
1007 DATA D0,FA,BD,00,02,9D,01,02,CA,10,  0403
1008 DATA F7,A9,3A,8D,00,02,60,20,29,14,  0326
1009 DATA C9,C2,F0,03,4C,DA,78,85,FD,20,  05BE
1010 DATA 0E,14,D0,F6,B0,02,85,FD,C8,20,  0504
1011 DATA 34,14,A5,17,48,A5,16,48,20,D7,  0346
1012 DATA 77,20,15,88,A5,FE,48,A5,FC,48,  0508
1013 DATA A5,FD,85,FE,A9,00,85,FC,20,BE,  062D
1014 DATA 13,A8,68,85,FC,68,85,FE,68,85,  057C
1015 DATA 16,68,85,17,4C,D4,84,68,AA,20,  03F0
1016 DATA 29,14,C9,97,F0,03,4C,A2,4A,85,  044D
1017 DATA FE,85,FC,20,0E,14,D0,F4,B0,02,  0537
1018 DATA 85,FE,8A,48,C8,20,34,14,20,D7,  047C
1019 DATA 77,20,15,88,20,5C,79,20,F4,87,  03C4
1020 DATA A9,00,AC,00,FF,8D,00,FF,24,FE,  0502
1021 DATA 10,0F,A5,67,A6,17,D0,06,A6,16,  037A
1022 DATA E0,20,90,1D,4C,28,7D,A2,12,A5,  03F7
1023 DATA 17,20,CC,CD,E8,A5,16,20,CC,CD,  052C
1024 DATA A2,1F,24,FC,10,0B,A5,67,20,CA,  03F2
1025 DATA CD,A2,12,24,FC,30,07,20,DA,CD,  049F
1026 DATA 8C,00,FF,60,20,CC,CD,8C,00,FF,  052F
1027 DATA A0,FF,C6,3E,20,34,14,4C,93,13,  03FD
1028 DATA C8,20,2B,14,C9,20,F0,F8,C9,25,  04E6
1029 DATA F0,05,C9,23,D0,01,18,60,E6,3D,  044D
1030 DATA D0,02,E6,3E,A0,00,2C,A0,01,8D,  03F0
1031 DATA 01,FF,B1,3D,8D,03,FF,60,18,98,  048D
1032 DATA 65,3D,85,3D,90,02,E6,3E,60,00,  837A
```

Listing 3. Eine Befehlserweiterung, die den Video-Chip unterstützt

Tips & Tricks für Einsteiger

Wie lese ich innerhalb eines Basic-Programms das Diskettenінhaltsverzeichnis ein? Wie berechne ich mit dem C 64 Primzahlen? Wie rechne ich mit hexadezimalen Zahlen? Auf diese und einige weitere Fragen sollen Sie hier Antwort bekommen.

Häufig bekommen wir von unseren Lesern Anfragen, wie man von einem Basic-Programm aus das Inhaltsverzeichnis einer Diskette einlesen kann. Die wirklich einfachste Lösung soll hier einmal vorgestellt werden. Sie kann problemlos in eigene Basic-Programme eingebaut und mit »GOSUB 1000« aufgerufen werden:

```
1000 OPEN 1,8,0,"$" : GET#1,A$,A$
1010 GET#1,A$,A$ : IF ST=64 THEN CLOSE 1 : RETURN
1020 GET#1,A$,B$ : PRINT ASC(A$+CHR$(0))+256*ASC(B$+
CHR$(0));
1030 GET#1,A$ : PRINT A$; : IF A$< >" " THEN 1030
1040 PRINT : GOTO 1010
```

Wem es gelingt, eine bessere Version zu entwerfen, soll uns sofort schreiben! (tr)

Ein paar kleine Effekte

1. Diese Routine gibt einen vorher festgelegten Namen in dreieckiger Form auf dem Bildschirm aus.

```
0 A=53270:A$="PETER":L=LEN(A$)
1 FOR R=1 TO L : PRINT TAB(20)MID$(A$,R,L):POKE A,R:NEXT R
```

Der Ausdruck (Bildschirm) sieht folgendermaßen aus:

```
PETER
ETER
TER
ER
R
```

Der Name ist also horizontal und vertikal lesbar.

2. Mit dieser Routine wird folgendes erreicht:

a) Diskname und ID werden extern eingegeben, das heißt beide können fest im Programm gespeichert werden.
b) Diskname und ID erscheinen beim Listen des Directory untereinander!

```
10  INPUT "NAME";N$
20  INPUT "ID"; I$
30  OPEN 1,8,15,"N:□"+N$+","+I$:CLOSE 1
```

Das Zeichen »□« erhält man folgendermaßen:
Geben Sie ein: OPEN 1,8,15,"N:"
Zurückfahren auf das zweite Anführungszeichen (Cursor nach links) und <CTRL+9, SHIFT+M, CTRL+0> drücken.
3. Dieser Trick erlaubt das Anhängen von Buchstaben an Directory-Namen.

```
OPEN 1,8,15,"R:Prgname{shift-space} Buchstaben = Prgname"
```

Prgname = Programmname

Natürlich funktioniert das auch beim Speichern:

```
SAVE "Programmname {shift+space} Buchstaben",8,1
```

Beim Laden braucht nur der Programmname angegeben zu werden.

(Stefan Wichmann/tr)

Input-Zeichen selbst wählen

Wenn man bei INPUT ein anderes Zeichen als das »?« haben möchte, muß man vor dem INPUT-Befehl folgende POKEs eingeben:

```
POKE 631,20:POKE 632,20:POKE 633,ASC("gewünschtes Zei-
chen"): POKE 634,32:POKE 198,4
```

(Axel Stämmler/tr)

Schnelles Primzahlenprogramm in Basic

Das Programm (Listing 1) arbeitet nach dem Prinzip des Eratosthenes. Hierbei werden die geraden Zahlen gar nicht erst gestrichen, sondern durch einen STEP-Befehl übersprungen. Nun werden alle Vielfachen der Zahlen gestrichen. Alle Zahlen, die übrigbleiben, sind Primzahlen.

Die Primzahl 2 wird nur deshalb ohne Rechnung ausgegeben (Zeile 30), da es sich um die einzige gerade Primzahl handelt. Man müßte sonst in der Ausgabeschleife mit einer Verzögerung rechnen.

Übrigens ist in der IF-Abfrage in Zeile 30 der Punkt hinter dem Gleichheitszeichen kein Fehler. Der Computer rechnet hier mit dem Punkt schneller als mit einer Null. Das Pro-