Basic-Befehle im Griff

Gerade für den Anfänger ist es nicht immer leicht, sich alle Basic-Befehle zu merken. Man denke nur an ausgefallene Befehle wie CMD oder POS(X).

Exbasic Level II kennt als Abhilfe den Befehl HELP, mit dem eine Liste sämtlicher Exbasicbefehle auf den Bildschirm gebracht werden kann. Die folgenden kurzen Programme (drei Lösungsmöglichkeiten für dasselbe Problem), die sich als Hilfsprogramme (Utility) zum Einbau in ein zu entwickelndes längeres Hauptprogramm verstehen, simulieren diesen Befehl HELP für das reine Basic, das heißt, sie geben eine Liste aller Befehle auf dem Bildschirm aus.

Wie geschieht das nun im einzelnen?

Eine unmittelbare Ausgabe aller Basic-Befehle auf dem Bildschirm per PRINT-Anweisungen hätte zwar gegenüber den neuen zu besprechenden drei Methoden den Vorteil, daß man diese Liste alphabetisch ordnen könnte. Sie wäre aber viel zu langsam (in der Größenordnung 5 Sekunden) und das generierte Basic-Programm wäre viel zu lang: Eine Liste aller aneinandergereihten Basic-Befehle, mit Trennzeichen zwischen Befehl und Befehl, umfaßt 331 Bytes. Einschließlich PRINT-Befehle, Hochkommata und Zeilennumerierungen würde ein solches Programm also mindestens 375 Bytes benötigen.

Nun enthält aber das Betriebssystem bereits eine Tabelle aller Basic-Befehle: 49310-49565 ($C09E-$C19D). Diese hat lediglich den Nachteil, daß sie keine Trennzeichen verwendet, sondern das Ende eines Basic-Befehlswortes dadurch kennzeichnet, daß der ASCII-Code des betreffenden Zeichens um den Wert 128 (Bit 7 gesetzt) erhöht wird. Das Vorhandensein einer solchen Tabelle wollen wir in den folgenden drei Vorschlägen zur Simulation von HELP ausnützen.

Methode 1: »HELP« in Basic

Das Basic-Programm nach Listing 1 ist der kürzeste unserer drei Vorschläge (nur 80 Bytes lang), es hat aber den Nachteil, daß es zum Aufbau der Basic-Befehlsliste mehr also vier Sekunden benötigt. Es wird (im Direktmodus oder vom Hauptprogramm aus) per GOSUB500 aufgerufen und PRINTet Zeichen für Zeichen der Tabelle 49310 — 49565, wobei es darauf achtet, daß immer wenn ein Befehlswort zu Ende ist, das betreffende Zeichen vor dem Ausdrucken (auf dem Bildschirm) zuerst normalisiert wird und daß dann zusätzlich noch ein Befehlsworttrennzeichen eingefügt wird (in Listing 1 an der Stelle X$=" . ", welches Symbol vom Leser beliebig abgeändert und individuellem Geschmack angepaßt werden kann). Nachteilig ist bei allen drei zu besprechenden Vorschlägen, daß man sich sinnvollerweise mit der durch die CBM-Tabelle vorgegebenen Unordnung zufriedengeben muß.

```
500 REM:HELP
510 FORI=0TO254:X=PEEK(49310+I):IFX>99TH
ENX=X-128:X$=".":GOTO530
520 X$=""
530 PRINTCHR$(X)+X$;:NEXT:RETURN
READY.
```

Listing 1. Simulation von HELP als reines Basicprogramm, Länge 80 Bytes, Ausführungszeit 4 s

Methode 2: »HELP« per Maschinensprogramm

Das in Listing 2 aufgeführte Maschinensprogramm zur Ausgabe der Basic-Befehlsliste auf dem Bildschirm ist zwar ein wenig länger als das Basic-Programm in Listing 1, nämlich 106 Bytes lang. Dafür benötigt es aber zum Einlesen der Data-Zeilen deutlich weniger als 1/3 Sekunde und erzeugt die Basic-Befehlsliste nach dem Aufruf per SYS700 in einem kaum wahrnehmbaren Bruchteil einer Sekunde. Das eigentliche Maschinenprogramm ist nur 21 Bytes lang und kann (voll verschiebbar) überall abgelegt werden, wo es nicht stört, zum Beispiel in einem durch Herabsetzen der RAM-Grenzen geschützten Maschinensprachbereich oder im Kassettenpuffer oder eben auch dort, wo wir es hingelegt haben, nämlich in den für den Benutzer freien Bereich 678-767 (02A6-02FF). Wir haben es als mit SYS700 aufrufbar gestaltet. Bei Ablage an anderer Stelle aaaa müßte POKE700+I,X in Zeile 610 von Listing 2 durch POKEaaaa+I,X ersetzt und das Programm per SYSaaaa aufgerufen werden. Legt man das Programm nicht in den Kassettenpuffer, sondern wie hier vorgeschlagen, nach 700ff., dann kann man das generierende Basic-Programm aus Listing 2 natürlich nach dem Einlesen des Maschinenprogramms wieder löschen (ein Maschinenprogramm für größere Löschvorhaben wurde im 64'er Ausgabe 5/84, Seite 85, veröffentlicht). Listing 3 gibt eine ausführlich kommentierte Darstellung des Maschinensprogramms in Assemblernotierung.

Methode 3: Maschinenprogrammerzeuger ohne Datenzeilen

Will man das Maschinenprogramm in den Kassettenpuffer legen, so muß man es gegebenenfalls (bei zwischenzeitlicher Ausführung von LOAD, SAVE oder VERIFY) oder, wenn sich

```
600 REM:HELP
610 FORI=0TO20:READX:POKE700+I,X:NEXT
620 DATA 160,255,200,185,158,192,16,7,41
,127,32,71,203,169,46,32,71,203,208,238,
96
READY.
```

Listing 2. Simulation von HELP per Maschinenprogrammlader. Länge 106 Bytes. Einlesen 0,3 Sekunden. Ausführung »augenblicklich«, Ansprung SYS700.

mehrere Hilfsprogramme den Kassettenpuffer teilen) vor jedem Aufruf neu generieren. Die Schwierigkeiten, die dadurch entstehen, daß der Basic-Datenzeiger per RESTORE nur immer an den Datenanfang gesetzt werden kann, lassen sich beispielsweise durch ein in Computer persönlich, Ausgabe 10/84, Seite 52, beschriebenes Hilfsprogramm beseitigen, mit welchem der Datenzeiger an beliebige Stellen gesetzt werden kann. Das nun noch zu beschreibende Programm nach Listing 4 vermeidet die Schwierigkeiten durch ausschließliche Verwendung von POKE-Anweisungen anstelle von DATA-Zeilen. Um den Aufwand so gering wie möglich zu halten, wurde angestrebt, mit möglichst weniger POKE-Anweisungen auszukommen. Das wurde mit einem Trick erreicht: Das für die Rückübersetzung der Token beim Ausdrucken nach dem LIST-Befehl zuständige Maschinenunterprogramm des Betriebssystems enthält fast alle Befehlssequenzen, die wir für unsere Zwecke benötigen. Das Programm in Listing 4 lädt den Maschinenprogrammabschnitt 50929 bis 51008 (C6F1 — C740), der zugegebenermaßen viel für unsere Zwecke überflüssigen Ballast enthält, mit einer einfachen FOR-NEXT-Schleife in den Kassettenpuffer, (Es wird mit GOSUB700 angesprungen). Jeder zweite und weitere Ansprung kann mit GOSUB730 erfolgen und wird dann in kaum wahrnehmbaren Bruchteilen einer Sekunde ausgeführt.

(Fred Behringer/ev)

```
      LDY #$FF       ZEICHENZAEHLER
NEXT  INY            NAECHSTES ZEICHEN
      LDA $C09E,Y    IN AKKU
      BPL NORM       WORTENDE ?
      AND #$7F       DANN NORMALISIERT
      JSR $CB47      ZEICHEN AUSGEBEB
      LDA #$2E       WORTTRENNZEICHEN
NORM  JSR $CB47      ZEICHEN AUSGEBEB
      BNE NEXT       ZEICHEN IN AKKU = 0 ?
ENDE  RTS            DANN ZURUECK ZU BASIC
```

Listing 3. Assemblerdarstellung des nach Listing 2 erzeugten Maschinenprogramms

```
700 REM:HELP
710 FORI=0TO80:POKE828+I,PEEK(50929+I):N
EXT
720 POKE836,55:POKE892,169:POKE893,166:P
OKE897,6:POKE903,180:POKE909,96
730 POKE782,255:SYS898:RETURN
READY.
```

Listing 4. Simulation von HELP als Maschinenprogramm für Kassettenpuffer, ohne DATA-Zeilen. Länge 106 Bytes, Einlesen 1,5 s, Ausführung »augenblicklich«.

Genau betrachtet: RS232/V.24-Schnittstelle

Eine kurze und bündige Beschreibung der RS232-Schnittstelle Ihres C 64. Was machen die Signale, wie sind die Pin-Belegungen?

Bei der RS232-Schnittstelle werden die Daten Bit für Bit übertragen, im Gegensatz zur Centronics- oder IEEE-488-Norm, bei der ganze Bytes übergeben werden. Die Bits werden als eine Folge von Spannungsimpulsen mit einer bestimmten Dauer übertragen. In der Praxis werden dabei Pakete von 5 bis 8 Datenbit übertragen, die von einem Startbit und 1 bis 2 Stop-Bit eingerahmt sind (Bild 1). Das Startbit hat grundsätzlich logischen Low- und die Stop-Bits High-Pegel. Vor dem Stop-Bit kann ein sogenanntes Paritäts-Bit vereinbart werden, das die Anzahl der High-Zustände im Datenwort immer gerade oder ungerade macht.

Beispiel: Sind in einer 8-Bit-Übertragung 5 Bit gesetzt, wird das Paritäts-Bit ebenfalls gesetzt, wenn gerade Parität vereinbart wurde.

Päckchenweise Übertragung

Um die Störungs-Anfälligkeit der Übertragung zu mindern, wird logisch »Eins« (gesetztes Bit) nicht durch +5V (TTL-Pegel) realisiert, sondern mit einer Spannung von —3 bis —12V und logisch »Null« mit +3 bis +12V (RS232 nach DIN 66020). Eine andere Norm ist die RS232/TTY, die gegen äußere Störungen recht unempfindlich ist. Bei dieser Norm werden die logischen Zustände durch das Fließen oder Fehlen eines Stromes (20mA) dargestellt. Der C 64 hat zwar die nötige Software für eine RS232-Schnittstelle im Betriebssystem integriert, verfügt aber nicht über die entsprechenden Spannungspegel. Im C 64 gibt es nur zwei Spannungen: +5V (TTL) und 9V Wechselspannung. Es ist also ein Interface zur Spannungskonvertierung nötig. Links in Bild 2 finden Sie den Schaltplan eines solchen Interfaces (Bauanleitung in Ausgabe 3/85). Rechts im Bild die diskrete Lösung, für die Konvertierung von 0/5V auf ±12V (oben) und von ±12V auf 0/5V (unten). Beachten Sie, daß jede Sende- und Empfangsleitung die entsprechende Transistorschaltung braucht.

Mit einer Masse- und einer Datenleitung könnte schon eine Übertragung von Texten an einen Drucker erfolgen. Was ist aber, wenn die Datenübertragung schneller ist, als der

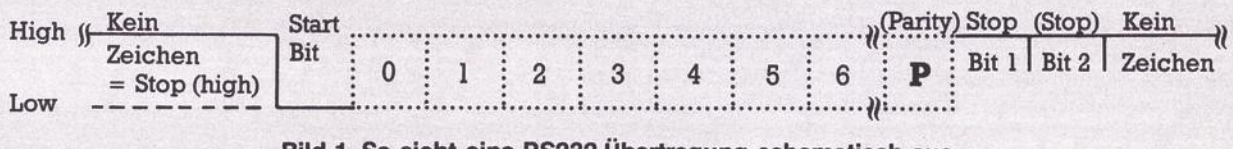

Bild 1. So sieht eine RS232-Übertragung schematisch aus