Memory Map mit Wandervorschlägen

Diesmal kommen die Adressen 144 bis 152 an die Reihe, die beim C 64 und VC 20 für Kassetten- und Ein-/Ausgabe-Operation verantwortlich sind.

Zu Beginn möchte ich heute einen Nachtrag zu den in Ausgabe 4/85 behandelten Speicherzellen 65/66 bringen. Das kleine Demonstrationsprogramm zur Änderung der Reihenfolge von DATA-Befehlen ist zwar richtig, kann aber unter ganz bestimmten Umständen zu Fehlern führen. Darauf hat mich ein aufmerksamer Leser hingewiesen — ich gebe daher seinen Leserbrief im folgenden direkt wieder. Die Adressen gelten nur für den C 64. Beim VC 20 ist der Sachverhalt aber derselbe.

Korrektur zu Adressen 65 bis 66

Leserbrief von Martin Kröger aus Kiel:

Bei der Behandlung der Adresse 65/66 hat Dr. H. Hauck auf einen Trick hingewiesen, mit dem der Befehl »RESTORE Zeilennummer« simuliert werden kann. Die Idee ist gut — doch die Ausführung hat einen kleinen, aber entscheidenden Fehler: Die Adresse 61/62 darf nicht mit zwei Befehlen, sondern muß mit einem Befehl ausgelesen werden, da bei einem möglichen Page-Wechsel zwischen den zwei Befehlen der Zeiger nicht verbogen, sondern abgeknickt wird.

Was passiert in der ersten Zeile des Demo-Programms?

10 A1=PEEK(61):B1=PEEK(62)

Mit »A1=PEEK(61)« wird der Variablen A1 der Wert des LOW-Bytes des Zeigers 61/62 zugewiesen. Dieser zeigt am Anfang einer Zeile auf das 0-Byte vor der Linkadresse (hier 2048), so daß A1 den Wert (2048 AND 255)=0 erhält. Mit »B1=PEEK (62)« wird der Variablen B1 der Wert des High-Bytes des Zeigers 61/62 zugewiesen. Dieser zeigt aber inzwischen auf das Trennzeichen (»:«) zwischen den beiden Befehlen (hier 2061), so daß B1 den Wert (INT(2061/256))=8 erhält. Als Zeiger auf das aktuelle DATA-Element erhalten wir die erwartete Adresse (A1+B1*256)=2048.

Was aber, wenn Zeilenanfang und Trennzeichen nicht in derselben Page liegen? Dazu setzen Sie bitte den Basic-Anfang um eine Stelle zurück:

POKE 43,0:POKE 2047,0:NEW

Die Zeiger auf den Zeilenanfang und das Trennzeichen werden dadurch ja ebenfalls verändert, so daß A1 jetzt den Wert (2047 AND 255)=255 und B1 den Wert (INT(2060/256))=8 erhält. Als Zeiger auf das aktuelle DATA-Element erhalten wir nun die völlig unbrauchbare Adresse (A1+B1*256)=2303.

Das korrigierte Demo-Programm könnte wie folgt aussehen:

```
10  A1=PEEK(61)+PEEK(62)*256
20  DATA DAS IST DIE 1. ZEILE
30  A2=(61)+PEEK(62)*256
40  DATA DAS IST DIE 2. ZEILE
50  A3=PEEK(61)+PEEK(62)*256
60  DATA DAS IST DIE 3. ZEILE
70  POKE 65,A3 AND 255:
    POKE 66,A3/256:
    READ A$:PRINT A$
80  POKE 65,A1 AND 255:
    POKE 66,A1/256:
    READ A$:PRINT A$
90  POKE 65,A2 AND 255:
    POKE 66,A2/256:
    READ A$:PRINT A$
```

Ich freue mich über Leserbriefe, wie diesen von Herrn Kröger, zeigen sie mir doch, daß meine Erklärungen gelesen und verdaut werden. Und was kann mir besseres passieren, als auf meine eigenen Schlampereien hingewiesen zu werden! Ich habe nämlich den Trick der DATA-Zeilen nicht selbst erfunden, sondern von Sheldon Leemon (eine meiner üblichen Quellen) kritiklos übernommen — und das war falsch.

So, nach diesem Rückblick geht es weiter mit unserer Wanderung. Heute habe ich wieder ein paar Besonderheiten zu bieten.

Adresse 144 ($90)

Statusvariable ST

Diese Adresse enthält ein Byte, welches mit der Statusvariablen ST von Basic identisch ist. Diese reservierte Variable ist im nebenstehenden Texteinschub 1 »ST-atus« näher beschrieben.

Alle Routinen des Betriebssystems, die mit Ein- und Ausgabe zu tun haben, benutzen diese Speicherzelle zum Abspeichern und Abfragen des Status der Ein-/Ausgabeoperationen.

Genauer gesagt, alle Ein-/Ausgabeoperationen, die mit der Datasette und mit dem Floppy-Gerät beziehungsweise dem Drucker zu tun haben, benutzen die Adresse 144. Im Fachjargon sprechen wir vom Kassettenport und vom seriellen Port.

Der dritte Anschluß des Computers nämlich der RS232 oder User-Port benutzt für den Status die Speicherzelle 663. Jedes Bit der Zelle 144 hat eine eigene Bedeutung.

Kassette:		
Bit 2	(Wert 4)	Kurzer Block
Bit 3	(Wert 8)	Langer Block
Bit 4	(Wert 16)	Lesefehler (nicht korrigierbar)
Bit 5	(Wert 32)	Prüfsummenfehler
Bit 6	(Wert 64)	File-Ende
Bit 7	(Wert 128)	Band-Ende
Floppy/Drucker:		
Bit 0	(Wert 1)	Fehler beim Schreiben
Bit 1	(Wert 2)	Fehler beim Lesen
Bit 6	(Wert 64)	Daten-Ende
Bit 7	(Wert 128)	»Device Not Present«-Fehler

Alle nicht aufgeführten Bits sind nicht benutzt.

Diese Speicherzelle beziehungsweise die Statusvariable ST kann recht nützlich sein. Einige Kochrezepte dafür werden im Texteinschub 1 behandelt.

Adresse 145 ($91)

Zwischenspeicher für Abfrage der STOP-Taste

In den Bildern 1 und 2 ist dargestellt, wie die Tasten des VC 20 und des C 64 miteinander über eine Matrix verbunden sind.

Sechzig mal in der Sekunde unterbricht der Computer seine Arbeit, merkt sich, wo er gerade ist und fragt dann, unter anderem, ob die STOP-Taste gedrückt worden ist. Dadurch wird erreicht, daß die STOP-Taste jederzeit Priorität hat.

Die Abfrage geht so vonstatten, daß das Betriebssystem über das im Bild 1 und 2 gezeigte Spaltenregister 56320 (beim VC 20: 37152) diejenige Tastenspalte anwählt, in welcher sich die STOP-Taste befindet. Aus Bild 1 und 2 sehen wir, daß dies die Spalte mit der Codenummer 127 beziehungsweise 247 ist. Ist in dieser Spalte eine Taste gedrückt, wird an ihrer Stelle eine Null in das Auslese-Register 56321 (VC 20: 37153) geschrieben. Die dadurch entstandene Dualzahl wird in die Speicherzelle 145 gebracht.

Es ist sicher verständlich, daß auf diese Weise nicht nur die STOP-Taste, sondern alle Tasten der Spalte 127 (247) abgefragt werden können. Ein kleines Demonstrationsprogramm kann das beweisen:

```
10 PRINT PEEK (656321); PEEK
(145)
20 GOTO 10
```

Beim VC 20 ist statt 56321 natürlich 37153 einzusetzen.

Das Zahlenband kann durch die Tasten der genannten Spalte — und nur durch diese — beeinflußt werden.

Adresse 146 ($92)

Zeitkonstante beim Lesen vom Band

Die Speicherzelle enthält eine vom Betriebssystem einstellbare Zahl, welche die kleinen Unterschiede in der Aufnahmegeschwindigkeit ausgleicht, die bei verschiedenen Datasetten vorkommen können.

Diese Zeitkonstante steht im Zusamenhang mit der Zahl, die in den Speicherzellen 176/177 steht.

Eine Veränderung der Konstante in Basic ist nicht möglich.

Adresse 147 ($93)

Flagge für LOAD oder VERIFY

Diese Flagge dient dem Betriebssystem, um zu unterscheiden, ob eine LOAD-Operation nur LOADen oder aber VERIFYen soll.

Sie ist identisch mit der Flagge des Basic-Übersetzers in Speicherzelle 10. Genauere Hinweise bitte ich der Beschreibung von Zelle 10 in Ausgabe 12/84, Seite 132 zu entnehmen.

Adresse 148 ($94)

Flagge für Floppy/Drucker Ausgang

Das Betriebssystem benutzt diese Speicherzelle, um anzuzeigen, daß ein Zeichen im Ausgabepuffer steht, welches zum Floppy-Gerät oder zum Drucker geleitet werden soll. Diese Flagge setzt alle am seriellen Port angeschlossenen Geräte in den Zustand »Listen«, das heißt bereit zu sein, Daten aufzunehmen.

Adresse 149 ($95)

Zeichen im Ausgabepuffer

In dieser Speicherzelle wird das Zeichen abgelegt, welches als nächstes über den Serial Port zum Floppy-Gerät oder zum Drucker transportiert wird, sobald die Flagge in 148 die Bereitschaft anzeigt.

Adresse 150 ($96)

Arbeitsspeicher für die Band-Leseroutine

Diese Speicherzelle wird zur Zwischenspeicherung von Da-

ten beim Lesen einer Kassette benutzt.

Adresse 151 ($97)

Zwischenspeicher des X-Registers

Maschinen-Programmierer kennen das X-Register des Mikroprozessors. Beim Lesen eines Zeichens von der Datasette wird der Inhalt des X-Registers in dieser Adresse zwischengespeichert.

Adresse 152 ($98)

Anzahl der offenen Files

Ein File, oder auf Deutsch gesagt, eine Datei, wird mit dem Befehl OPEN eröffnet. Nach OPEN folgt die Nummer der Datei; sie ist beliebig wählbar bis maximal 255. Als zweites folgt die Nummer des Gerätes, mit dem die Verbindung hergestellt werden soll.

Es ist erlaubt, mehrere Dateien gleichzeitig geöffnet zu halten, vorausgesetzt die Nummern der Dateien sind verschieden.

In Speicherzelle 152 wird festgehalten, wieviel Dateien gleichzeitig geöffnet sind. Dieses kleine Programm zeigt es uns deutlich:

```
10  FOR K=10 TO 22
20  PRINT PEEK (152),K
30  OPEN K,0
40  NEXT K
```

Mit der FOR...NEXT-Schleife der Zeilen 10 und 40 eröffnen wir 13 Dateien hintereinander, und zwar — wie Zeile 30 uns deutlich macht — mit der Tastatur. Die Tastatur hat die Nummer 0, der Drucker die Nummer 4, das Floppy-Gerät die Nummer 8 und die Datasette die Nummer 1. Ich habe die Tastatur gewählt, obwohl es keinen Sinn ergibt, weil sie die vielen Eröffnungen ohne zu unterbrechen akzeptiert.

Nach RUN sehen wir links untereinander den Inhalt von 152, also die Anzahl der eröffneten Dateien. Rechts steht jeweils die Nummer der eröffneten Datei.

Nach der 10. Datei bricht das Programm ab und druckt uns die Fehlermeldung TOO MANY FILES aus.

Das heißt es sind gleichzeitig nur 10 Dateien betreibbar. Wenn wir oben in Zeile 10 die Zahl 22 durch 19 ersetzen, läuft das Programm fehlerfrei.

Eine Datei, die unter einer bestimmten Nummer eröffnet ist, kann nicht noch einmal eröffnet werden. Fügen Sie bitte dem Programm noch die folgende Rücksprungzeile hinzu:

```
50  GOTO 10
```

In der 10. Zeile sehen wir jetzt die 10 als Inhalt von 152 und als

Texteinschub 1

ST-atus

Neben den Befehlen (wie PRINT) und den Funktionen (wie COS) hat Basic auch noch drei fest definierte Variable, nämlich TI, TI$ und ST.

Von den dreien ist ST wohl am seltensten anzutreffen, Grund genug, hier ein wenig darüber zu berichten.

Der Anlaß ist natürlich, daß der Wert von ST immer in der Speicherzelle 144 steht, die ja heute in der Memory Map vorkommt.

Bei der Beschreibung wurde schon erwähnt, daß ST den Status nach der letzten Ein- beziehungsweise Ausgabeopration angibt, beschränkt allerdings nur auf Operationen mit der Datasette und der an einem gemeinsamen Ausgang angeschlossenen Floppy und Drucker.

Dementsprechend zeigt die Tabelle bei der Speicherzelle 144 diese beiden Fälle.

Wichtig ist noch zu erwähnen, daß nicht nur die in der Tabelle gezeigten Zahlen für ST auftreten, sondern auch Kombinationen davon. So ergibt zum Beispiel ein zu kurzer Block (4) und ein gleichzeitig aufgetretener Prüfsummenfehler (32) ein Wert von 36.

Kassettenoperationen

Zuerst testen wir mit einem Datei-Programm auf »File-Ende«. Geben Sie bitte folgendes Programm ein:

```
10  OPEN 1,1,1,"DATEI"
20  PRINT#1, "QWERTY"
30  CLOSE 1
40  END
```

Zur Erinnerung: nach dem OPEN-Befehl folgt zuerst die Nummer der Datei (ich nehme hier 1), dann die Gerätenummer (1= Datasette) und schließlich die Sekundäradresse (1= Schreiben).

Jetzt kommt der Lesevorgang:

```
50  OPEN 2,1,0"DATEI"
60  FOR K=1 TO 10
70  GET#2,A$
80  PRINT A$;ST
90  NEXT K
100 CLOSE 2
```

In Zeile 50 eröffnen wir wieder eine Datei (diesmal Nummer 2) für die Datasette, jetzt aber zum Lesen (Sekundäradresse = 0). Die Schleife der Zeilen 60 und 90 schreiben uns 10 mal ein Zeichen (A$) und den Wert von ST auf den Bildschirm.

Jetzt geht es los. Mit RUN starten wir den ersten Teil des Programms. Nach dem Schreibvorgang und der READY-Meldung (nach Zeile 40) müssen Sie das Band zurückspulen und mit GOTO 50 ab Zeile 50 weiterfahren. Jetzt wird die Datei gelesen.

Wir erhalten untereinander die sechs Buchstaben von Zeile 20, daneben für ST lauter Nullen. Am Ende allerdings erscheint eine 64. Das ist der in der Tabelle angegebene Wert von ST für »File-Ende«.

Da die FOR-NEXT-Schleife zu lang ist, schießen wir mit dem Lesen über das File-Ende hinaus. Normalerweise kennen wir natürlich die Länge einer Datei nicht. Deshalb ist es besser mit GOTO zurückzuspringen und das File-Ende abzufragen.

Löschen Sie bitte Zeile 60 und 90 und fügen Sie als Rücksprung und Prüfung ein:

```
85  IF ST=64 THEN 100
90  GOTO 70
```

Statt nach ST können wir natürlich genau so gut nach PEEK(144) fragen.

Ein erneutes GOTO 50 bringt das erwünschte Resultat.

Um den vorhin schon erwähnten »kurzen Block« zu sehen, müssen wir einen entsprechenden Fehler künstlich erzeugen.

Löschen Sie bitte den ersten Teil des Programms bis einschließlich Zeile 40. Wir behalten also nur den Leseteil ab Zeile 50. Dann laden wir dieses Programm (Band vorher am besten wieder zurückspulen) mit SAVE "DATEI" nicht als Datei, sondern als ganz gewöhnliches Programm. Wenn es geladen ist, bitte das Band wieder zurückspulen.

Mit RUN starten wir jetzt das Lese-Programm, welches eine Datei sucht, aber nur ein Programm findet, allerdings mit dem richtigen Namen. Natürlich findet es einen Fehler und wir erhalten als Ausdruck:

```
36    oder manchmal auch 4
64                       64
```

Die Zahl 36 entsteht aus 32+4, das bedeutet Prüfsummenfehler + Block zu kurz. Danach folgt wie vorher das File-Ende.

Die normale Blocklänge entspricht der Länge des Bandpuffers, in den die Datasette einspeichert. Er ist 191 Bytes lang. In unserem Fall war offenbar der Block nicht ganz voll.

Der Prüfsummenfehler tritt dann auf, wenn eine der Überprüfungen von Kassettenoperationen einen Fehler gefunden hat. Der Blockfehler, auch der des zu langen Blocks, interessiert wohl weniger. Aber ein durch die Prüfungen gefundener Fehler könnte, frühzeitig noch vor dem Ausstieg des Programms entdeckt, abgefangen und ausgenutzt werden.

Die Formel dafür, ins obige Program eingebaut ist:

```
85 IF ST<64 OR ST>8 THEN..(zum Beispiel LIST)
```

Statt LIST kann man natürlich auch etwas anderes nehmen.

Floppyoperationen

Bei der Floppy bedeutet ST=64, »Daten-Ende«, das ist etwa dasselbe wie bei der Datasette. Um es zu überprüfen nehmen wir dasselbe Programm wie vorher, nur müssen wir die Datei-Zeilen der Floppy anpassen. Das sieht dann so aus:

```
10  OPEN 1,8,1,"DATEI,S,W"
20  PRINT#1,"QWERTY"
30  CLOSE 1
40  END

50  OPEN 2,8,0,"DATEI,S,R"
60  FOR K=1 TO 10
70  GET#2,A$
80  PRINT A$;ST
90  NEXT K
100 CLOSE 2
```

Das Ergebnis sieht hier so aus:

```
64
66
66
```

Die 64 ist natürlich wie erwartet der »Wert« für Daten-Ende. Die 66 ist 64+2, entstanden dadurch, daß wir über das Datenende hinausgelesen haben. Die 2 bedeutet »Fehler beim Lesen« (in den englischen Beschreibungen heißt es »Read Time Out«). Ähnliches gilt für ST=1, das bededutet »Fehler beim Schreiben« (englisch: Write Time Out), nur weiß ich leider nicht, wie es vorzuführen ist. Wie bei der Datasette kann das Überlesen natürlich mit der Abfrage IF ST=64 THEN...und GOTO... gestoppt werden.

Interessant ist noch der Status beim Fehler »DEVICE NOT PRESENT«, den wir dadurch erzeugen, daß wir ein Programm, oder die Directory, aus der Floppy laden wollen, ohne daß dieses Gerät angeschlossen oder eingeschaltet ist. Nach der Fehlermeldung geben wir direkt ein:

```
PRINT ST oder PRINT PEEK(144)
```

und wir erhalten die Zahl 128.

Wie man allerdings in einem Basic-Programm durch Abfrage von ST=128 die Fehlermeldung »Device Not Present« und den dann folgenden Programmabbruch vermeiden kann, weiß ich leider auch nicht. Die einzige Möglichkeit, die ich kenne, verwendet die sogenannte Interrupt-Methode. Diese ist von Christoph Sauer im letzten Teil seines Kurses »Der gläserne VC 20« in Heft 3/85, Seite 155, beschrieben worden.

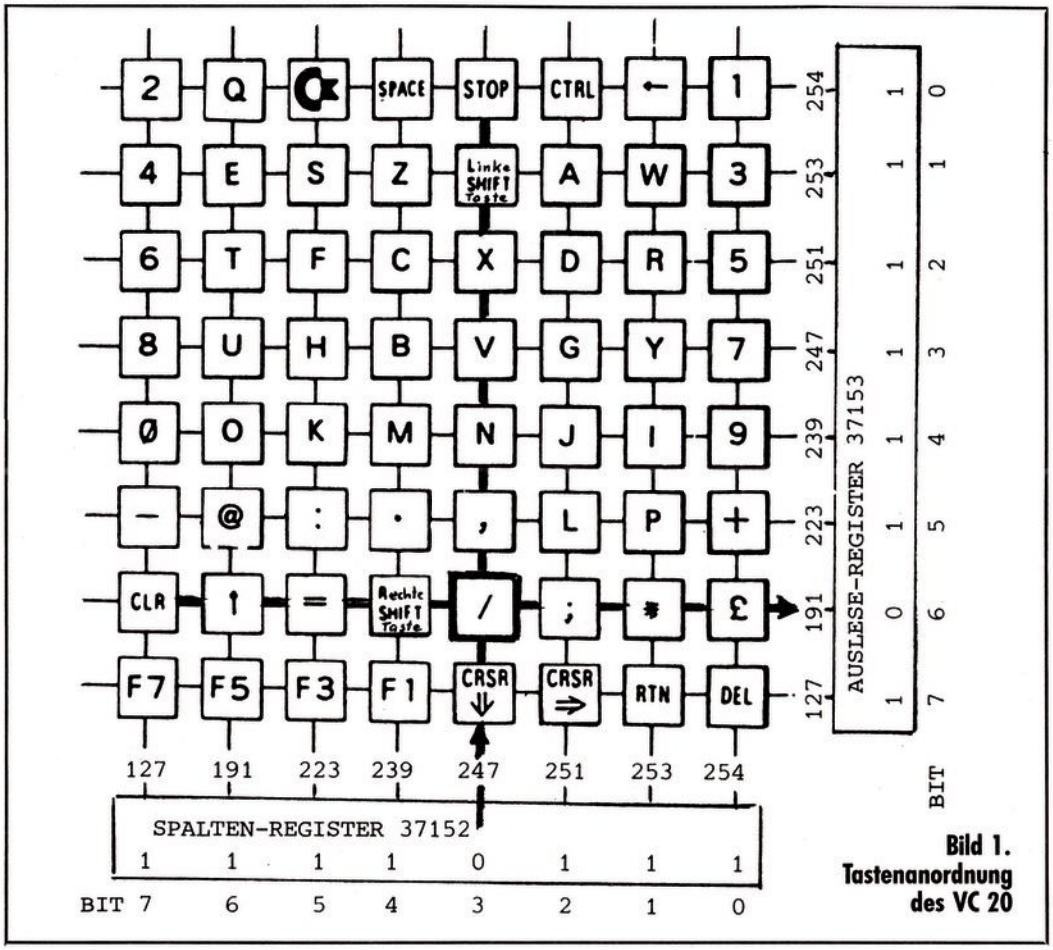

Bild 1. Tastenanordnung des VC 20

Sie werden vielleicht fragen, warum ich das so ausführlich beschreibe. Nun, in einem Programm kann es sicher sehr nützlich sein, die Zelle 152 mit PEEK nach der Datei-Lage abzufragen und entsprechend Maßnahmen zu treffen, ehe die Fehlermeldung das Programm abbricht.

Mit POKE 152,0 aber müssen Sie aufpassen. Es ersetzt nämlich nicht (!!) den CLOSE-Befehl. Probieren Sie es aus: Um das kleine Programm oben per Drucker auszudrucken, brauchen wir:

OPEN 1,4: CMD 1: LIST

Wenn Sie jetzt die Zeile 152 auf 0 POKEn und dann LIST eintippen, wird trotzdem wieder auf dem Drucker gelistet und nicht auf dem Bildschirm. Die vorgeschriebene Schließmethode mit PRINT #1:CLOSE 1
geht jetzt aber auch nicht mehr, denn das Betriebssystem ist ja im Glauben, daß keine Datei eröffnet ist — schöner Schlamassel!

Erst eine Neueröffnung bringt alles wieder in die Reihe. Also Vorsicht mit der Anwendung der Speicherzelle 152. Eine Möglichkeit, alle Dateien auf einen Schlag zu schließen, gibt es aber doch:

SYS 65511 besorgt das, sowohl beim C 64 als auch beim VC 20.

(Dr. Helmut Hauck/ah)

neue Dateinummer (K) wieder die 10. Das Programm bleibt aber stehen und meldet FILE OPEN. Es hat recht, denn die Datei 10 ist bereits als erste eröffnet, aber nicht wieder geschlossen worden.

Das Betriebssystem macht das so, daß jede der Dateinummern in eine Tabelle geschrieben wird, die in den Speicherzellen 601 bis 610 stehen. Bei jedem OPEN-Befehl wird dort nachgeschaut, ob die Filenummer existiert. Wenn ja wird die Fehlermeldung FILE OPEN ERROR ausgegeben. Bei jedem CLOSE-Befehl wird die entsprechende Nummer aus der Tabelle gelöscht.

Wir können aber auch eine 0 in die Speicherzelle 152 POKEn, wodurch dem Betriebssystem vorgegaukelt wird, daß keine Datei eröffnet ist. Schieben Sie im Programm einfach die Zeile ein:

45 POKE 152,0

und das Programm läuft ewig weiter.

Die Speicherzelle 152 ist also der Wächter über die Anzahl der eröffneten Dateien. Steht sie auf 0, dann wird eine Neueröffnung am Anfang der Tabelle ab 601 eingetragen. Die Tabelle ihrerseits ist der Wächter über Exklusivität der Dateinummern. Ich zeige Ihnen das noch genauer, wenn wir zu 601 kommen.

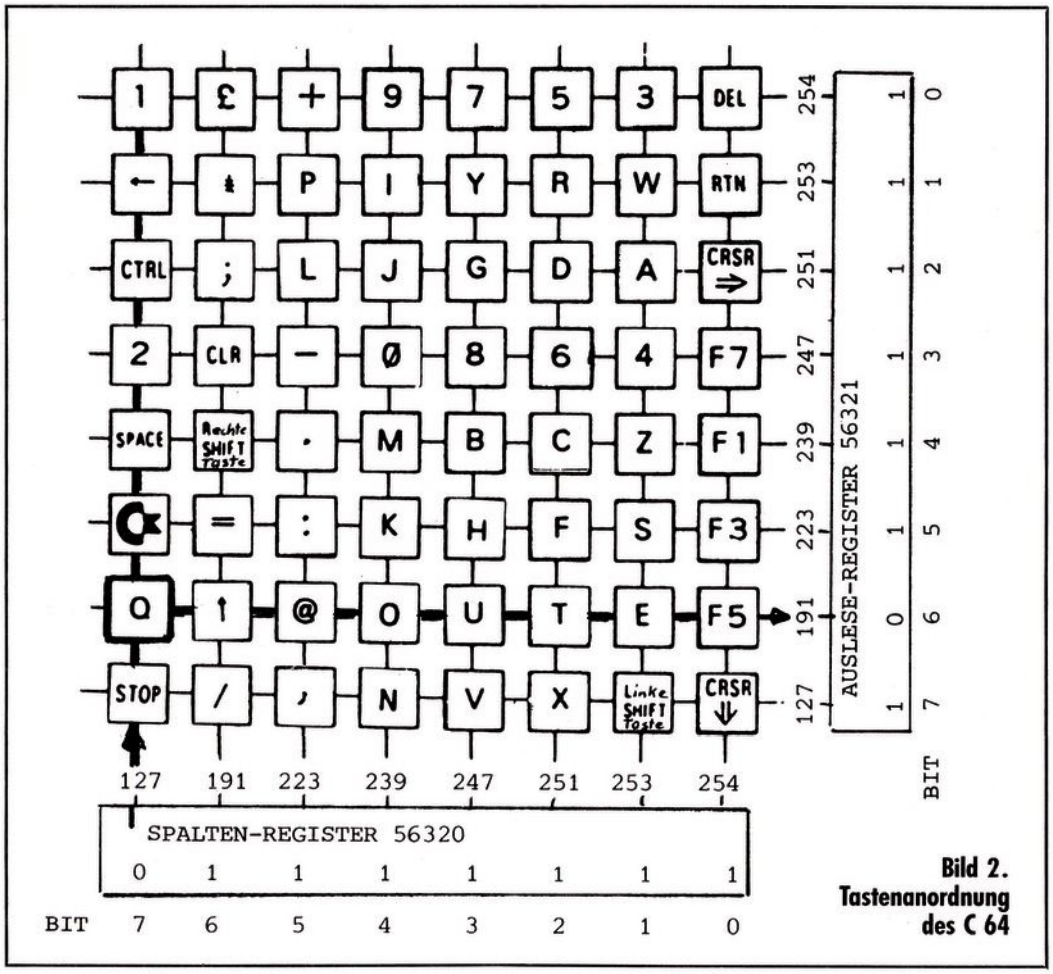

Bild 2. Tastenanordnung des C 64