Funktionen auf Tastendruck

Mit diesem Beitrag zeigen wir Ihnen, wie auf Tastendruck beliebige Funktionen oder Programme auch während eines Programmlaufs gestartet werden können. Egal ob es sich nun um ein Basic- oder Maschinenprogramm handelt.

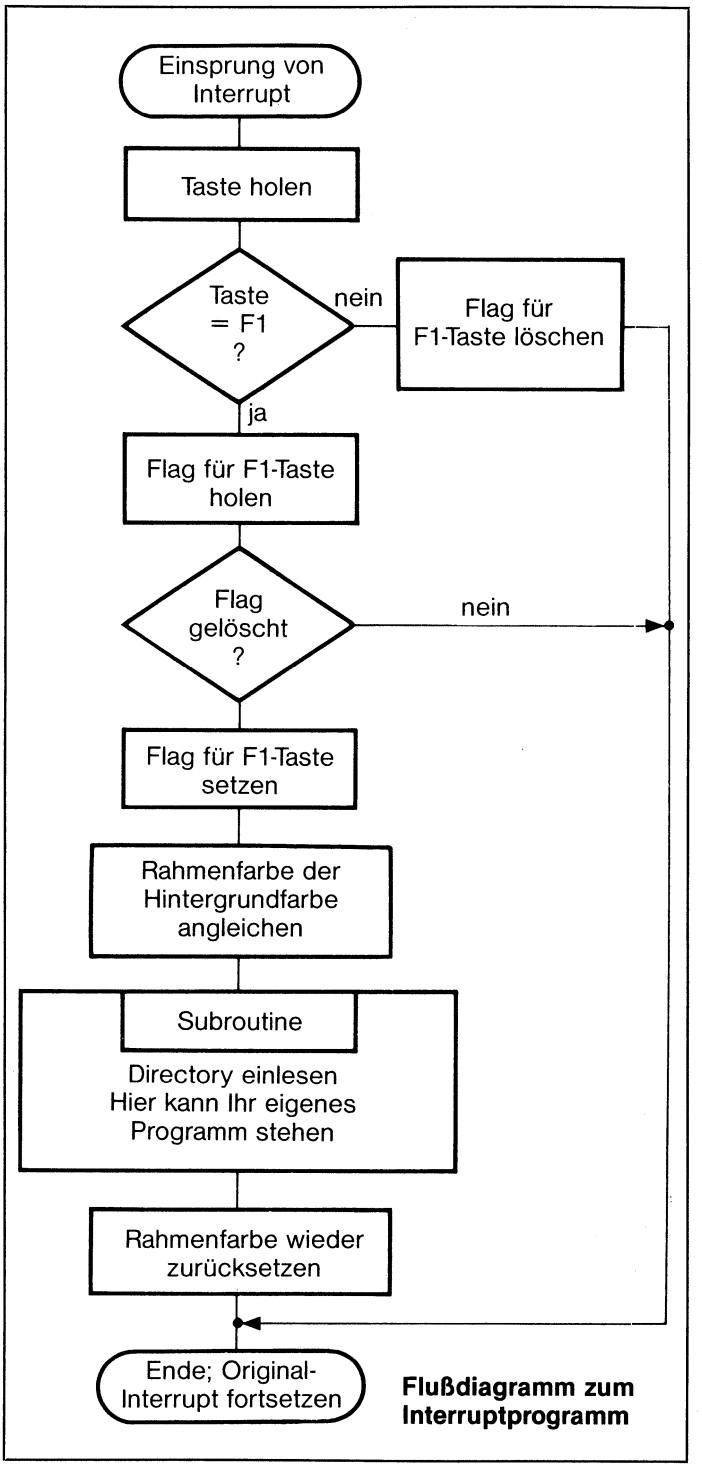
Dieser Artikel soll Ihnen das Thema »Funktionen« anhand eines Beispiels näherbringen. Das Beispielprogramm können Sie mit etwas Assemblerkenntnis auch für Ihre Programme verwenden. Vorschlag: Der F1-Taste wird eine Funktion zugeordnet. Durch Drücken der F1-Taste soll jederzeit das Directory einer Diskette am Bildschirm gezeigt werden, ohne daß ein im Speicher vorhandenes Programm gelöscht wird. Während das Directory geladen wird, soll der Bildschirmrahmen die Farbe des Hintergrundes annehmen.

Es wurden schon viele solcher Utilities veröffentlicht. Diese Programme werden aber im Regelfall durch Drücken der STOP/RESTORE-Tasten wieder abgeschaltet. Sollen diese Routinen dagegen gewappnet sein, so versagen sie spätestens nach einem Reset. Im Gegensatz zu diesen »Billigmethoden« soll aber unsere Directory-Funktion der F1-Taste auch nach einem Reset erhalten bleiben.

Um derartige Anforderungen zu realisieren, müssen einige Betriebssystemroutinen zumindest angeschnitten werden. Die Wichtigste ist die Interruptroutine des C 64. Die Interruptroutine ist ein computerinternes Programm, das sehr oft angesprochen wird. Befindet sich der Computer im »Ready«-Modus, wird die Interruptroutine beispielsweise alle 1/60 Sekunde durchlaufen. Wird der C 64 in Anspruch genommen, etwa durch Diskettenoperationen, wird dieses Programm jedoch seltener bearbeitet. Der Interrupt erfüllt viele Aufgaben. Er stellt beispielsweise die Uhrzeit (TI,TI$) nach, läßt den Cursor blinken und übernimmt die Tastaturabfrage. Das Interruptprogramm befindet sich ab Adresse $EA31 im Betriebssystem. Der Computer sucht sich diese Adresse über einen Wegweiser. Dieser Wegweiser steht bei Adresse $0314 und $0315 in der Zero-Page. Ein solcher Wegweiser wird im Fachjargon als Vektor bezeichnet. Da sich der Vektor im RAM befindet, kann dieser verändert werden. Das heißt, daß der Interrupt »umgeleitet« werden kann. Doch davon später.

Wie funktioniert eine Modulerkennung?

Eine zweite gewichtige Betriebsroutine ist die Modulerkennung. Diese benötigt man, damit die geplante Funktion der F1-Taste »resetfest« wird und zugleich immun gegen die Tastenkombination STOP/RESTORE. Die Kennungs-Routine überprüft, ob sich ein Steckmodul im Expansion-Port befindet. Und zwar an der Codefolge »cbm80« ab Adresse $8004. Diese ASCII-Codefolge entspricht den Hex-Zahlen $C3,$C2,$CD,$38 und $30. Findet der Computer diese fünf Zahlen ab Adresse $8004, werden sowohl bei einem Reset als auch beim Drücken der Restore-Taste nicht mehr die Standardroutinen angesprochen. Der C 64 springt die Adressen an, deren Vektoren in $8000/8001 und $8002/8003 stehen. Der erste Vektor ist der Resetvektor. Der Resetvektor zeigt auf die Adresse, die nach einem Reset angesprungen wird. Nach STOP/RESTORE holt sich der C 64 die Sprungadresse des NMI-Vektors aus den Speicherzellen $8002 und 8003. Ist kein Modul eingesteckt, ist der Adreßbereich ab $8000 als RAM freigegeben. Wenn man nun ab Adresse $8004 die oben genannte Codefolge ablegt, nimmt der C 64 an, daß sich ein Modul im Expansion-Port befindet. Durch diesen Trick kann der Reset- und NMI-(RESTORE-) Vektor verändert werden.



Flußdiagramm zum Interruptprogramm

Da unser Programm wegen dieses Tricks mitten im Basic-Speicher steht, muß es vor dem Überschreiben durch lange Basic-Programme oder Stringvariablen gesichert werden. Dies erreicht man, wenn man den Zeiger für den Stringspeicher $33-$34 und den Zeiger für die Speichergrenze auf $8000 setzt.

Doch nun zum Programm selbst. Nach dem Eingeben des Programmes mit Hilfe des MSE kann es mit SYS 32819 oder mit einem Reset (SYS 64738) initialisiert werden. Dabei wird der Interrupt auf die Adresse $8046 umgeleitet und das Programm vor dem Überschreiben geschützt. Solange der Interruptvektor verändert wird, muß der Interrupt gesperrt werden. Das heißt, daß während dieser Zeit die Interruptroutine nicht angesprungen werden darf. Ein Interrupt muß auch verhindert werden, wenn gerade ein Interruptprogramm stattfindet. Die Katze soll sich schließlich nicht in den eigenen Schwanz beißen. Um einen Interrupt zu sperren, gibt es einen speziellen Assemblerbefehl, den SEI (Set Interrupt Flag). Ist das IRQ-Flag (IRQ, Interrupt Request) gesetzt, läßt sich der 6502-Prozessor nicht mehr durch einen Interrupt unterbrechen, um ein anderes Programm auszuführen. Um den Interruptvektor auf unser Beispielprogramm zu richten, muß in Speicherzelle $0314 das Lowbyte ($46) und in $0315 das Highbyte ($80) unserer Programmadresse ($8046) geschrieben werden. Danach kann das Interruptflag wieder gelöscht werden. Der Assemblerbefehl dazu lautet CLI (Clear Interrupt Flag). Wird nun ein Interrupt ausgelöst (von den CIAs), springt der Prozessor zur Adresse $8046. In unserem Beispiel setzt dort die Abfrage der F1-Taste ein. Ist sie gedrückt oder nicht? Eine Antwort darauf liefert die Adresse $CB in der Zero-Page. Denn der Inhalt von $CB gibt Auskunft darüber, welche Taste zuletzt gedrückt wurde. War es die F1-Taste, steht dort eine »4«.

Den Code einer jeden Taste können Sie mit der folgenden Basic-Anweisung leicht ermitteln:

```
FOR I=1 TO 10000 : PRINT PEEK(203) : NEXT I
```

Drücken Sie danach die gewünschte Taste, erscheint die zugehörige Zahl am Bildschirm.

Das Beispielprogramm reagiert nun folgendermaßen: Ist die F1-Taste nicht gedrückt, also der Inhalt der Speicherzelle ungleich 4, wird in Speicherzelle $8009 der Wert $FF geschrieben und die Adresse $EA31 angesprungen. Wie schon erwähnt, steht ab $EA31 die normale IRQ-Routine des C 64. Warum in Speicherzelle $FF geschrieben wird, soll später erklärt werden.

Wurde die F1-Taste gedrückt, wird das Beispielprogramm bearbeitet. Dabei verändert der Bildschirm seine Rahmenfarbe und das Unterprogramm »Directory« ab Adresse $8074 wird abgearbeitet. Ab Adresse $8074 kann jedes beliebige Unterprogramm stehen. Voraussetzung ist nur, daß dieses mit RTS abgeschlossen wird. In diesem Beispiel ist es eben das Zeigen eines Diskettendirectories.

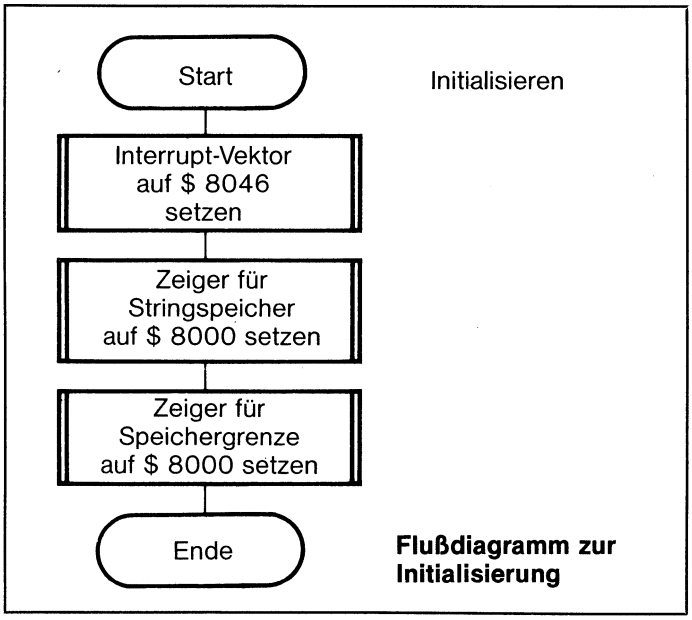
Stellen Sie sich jetzt vor, es würde kein Directory gelesen, sondern irgendeine Funktion aufgerufen, die einen sehr kleinen Zeitbedarf hat. Eventuell eine Einstellung von Bildschirmfarben, wo verfügbare Farben durch Drücken der F1-Taste gezeigt werden. Alle 1/60 Sekunde würde dann die Farbe wechseln. Unmöglich! Deshalb ist in die Tastenabfrage noch ein Trick eingebaut, der das verhindert. Dazu wird nach dem Drücken von F1 ein Flag gesetzt (Speicherzelle $8009 auf $00) und erst dann die Funktion aufgerufen. Ist beim nächsten Interrupt $8009 immer noch »0«, so heißt das, daß die F1-Taste in der Zwischenzeit noch nicht losgelassen wurde. Die Funktion soll dann nicht nochmal ausgeführt, sondern mit der originalen IRQ-Routine fortgefahren werden.

Wäre die F1-Taste in der Zwischenzeit losgelassen worden, hätte in der Speicherzelle $CB eine Änderung stattgefunden. Unsere IRQ-Routine hätte das erkannt und in $8009 den Hex-Wert $FF geschrieben.

Damit die Funktions-Belegung der F1-Taste nicht durch einen Reset zerstört werden kann, wurde das Prinzip der Modulerkennung genutzt. Der Resetvektor zeigt auf die Adresse $800A. Ab dieser Adresse wurde einfach der Anfang des Originalresets nachgebildet. Nachdem der Arbeitsspeicher neu initialisiert und alle Vektoren (auch Interruptvektor) mit ihren Standardwerten belegt wurden, springt »unser« Reset nach $8033 und initialisiert wieder die F1-Taste. Anschließend wird der Prozessor wieder auf seinen gewohnten Weg geschickt ($FCFB).

Unempfindlich gegen Reset

Der NMI-Vektor wurde auf die Adresse $801F »verbogen«. Ab dieser Adresse ist der Original-NMI nachgebaut. Bis auf den Sprung an die Adresse $FD15, der weggelassen wurde. Dort wird nämlich die Interruptadresse korrigiert und die F1-Taste abgeschaltet, was die ganzen Anstrengungen zunichte machen würde.



Flußdiagramm zur Initialisierung

Das Beispielprogramm kann natürlich beliebig erweitert oder geändert werden. Es wäre zum Beispiel möglich, die Buchstabentasten über die CTRL-Taste mit einer vierten Funktion zu belegen. Ähnlich SHIFT/CBM. Ein anderer Einsatz wäre der Aufruf einer Hardcopy. Versuchen Sie einfach mal, eine Hardcopyroutine über ????? aufzurufen, statt die Directory-Funktion. Sie müssen dazu nur den JSR $8074-Befehl ab Adresse $806A so ändern, daß die von Ihnen gewählte Routine abgearbeitet wird. Der Befehl lautet dann vielleicht JSR $C000. Haben Sie keine Angst vor Maschinensprache! Experimentieren Sie doch einfach mit diesem Programm. Eine einfache Übung: Stellen Sie mit der F1-Taste die Farben ein.

(Christian Quirin Spitzner/hm)

```
programm : directory          8000 80ef
-------------------------------------

8000 : 0a 80 1f 80 c3 c2 cd 38  1c
8008 : 30 ff 8e 16 d0 20 a3 fd  37
8010 : 20 50 fd 20 b7 e4 20 15  29
8018 : fd 20 33 80 4c fb fc 20  db
8020 : bc f6 20 e1 ff d0 09 20  86
8028 : a3 fd 20 18 e5 6c 02 a0  e0
8030 : 4c 72 fe 78 a2 46 a0 80  64
```

```
8038 : 8e 14 03 8c 15 03 58 a9  41
8040 : 80 85 34 85 38 60 a5 cb  f5
8048 : c9 04 f0 08 a9 ff 8d 09  33
8050 : 80 4c 31 ea ad 09 80 c9  59
8058 : ff d0 f6 a9 00 8d 09 80  44
8060 : ad 20 d0 48 ad 21 d0 8d  9d
8068 : 20 d0 20 74 80 68 8d 20  49
8070 : d0 4c 31 ea a9 01 20 c3  bb
8078 : ff a9 24 8d f0 03 a9 30  35
8080 : 8d f1 03 a9 01 a2 08 a0  82
8088 : 00 20 ba ff a9 02 a2 f0  5e
8090 : a0 03 20 bd ff 20 c0 ff  76
8098 : a9 40 20 90 ff a2 01 20  d5
80a0 : c6 ff 20 90 ff 20 cf ff  c0
80a8 : 20 cf ff 20 cf ff 20 cf  d1
80b0 : ff c9 00 f0 31 a9 01 85  21
80b8 : cc a2 01 20 c6 ff 20 cf  a6
80c0 : ff a8 20 cf ff 48 98 aa  0f
80c8 : 68 20 cd bd a9 20 20 d2  2d
80d0 : ff 20 cf ff c9 00 d0 08  c3
80d8 : a9 0d 20 d2 ff b8 50 cb  09
80e0 : 20 d2 ff b8 50 eb a9 01  8d
80e8 : 20 c3 ff 20 cc ff 60 00  3c
```

Listing zum Directory auf Tastendruck. Beachten Sie den MSE

```
8000  0a 80                    ; reset-vektor
8002  37 80                    ; nmi-vektor
8004  c3 c2 cd 38 30           ; cbm80 (bewirkt 'modul-start')
8009  ff
------------------------------ reset
800a  8e 16 d0    stx $d016
800d  20 a3 fd    jsr $fda3
8010  20 50 fd    jsr $fd50    ; arbeitsspeicher initialisieren
8013  20 b7 e4    jsr $e4b7
8016  20 15 fd    jsr $fd15    ; hardware und i/o vektoren setzen/holen
8019  20 33 80    jsr $8033    ; initialisierung der f1-taste
801c  4c fb fc    jmp $fcfb    ; zurueck zum original-reset
------------------------------ nmi
801f  20 bc f6    jsr $f6bc    ; flag fuer stop-taste setzen
8022  20 e1 ff    jsr $ffe1    ; stop-taste abfragen
8025  d0 09       bne $8030    ; nicht gedrueckt, dann $8030
8027  20 a3 fd    jsr $fda3    ; i/o initialisieren
802a  20 18 e5    jsr $e518    ; bildschirm loeschen (i/o initialisieren)
802d  6c 02 a0    jmp ($a002)  ; zum basic-warmstart
8030  4c 72 fe    jmp $fe72    ; weiter im orginal nmi
------------------------------ initialisierung der f1-taste
8033  78          sei          ; setzen des interrupt-dissable-bit
8034  a2 46       ldx #$46     ;
8036  a0 80       ldy #$80     ;
8038  8e 14 03    stx $0314    ; interrupt auf adresse
803b  8c 15 03    sty $0315    ;  $8046 setzen
803e  58          cli          ; loeschen des interrupt-dissable-bit
803f  a9 80       lda #$80     ; programm vor ueberschreiben sichern
8041  85 34       sta $34      ; zeiger fuer stringspeicher auf $8000
8043  85 38       sta $38      ; zeiger speichergrenze auf $8000
8045  60          rts
------------------------------ erweiterung der interrupts
8046  a5 cb       lda $cb      ; aktuelle taste holen
8048  c9 04       cmp #$04     ; = f1-taste ?
804a  f0 08       beq $8054    ; ja, dann $8054
804c  a9 ff       lda #$ff     ; flag fuer
804e  8d 09 80    sta $8009    ; f1-taste loeschen
8051  4c 31 ea    jmp $ea31    ; zurueck zum interrupt
8054  ad 09 80    lda $8009    ; f1-taste schon laenger gedrueckt ?
8057  c9 ff       cmp #$ff
8059  d0 f6       bne $8051    ; ja, dann zurueck zum interrupt
805b  a9 00       lda #$00     ; setzt flag
805d  8d 09 80    sta $8009    ; fuer gedrueckte f1-taste
8060  ad 20 d0    lda $d020    ; rahmenfarbe lesen
8063  48          pha          ; farbe auf stapel legen
8064  ad 21 d0    lda $d021    ; hintergrundfarbe lesen
8067  8d 20 d0    sta $d020    ; rahmen angleichen;
806a  20 74 80    jsr $8074    ; beliebige funktion (z.b. directory)
806d  68          pla          ; rahmenfarbe von stapel holen
806e  8d 20 d0    sta $d020    ; rahmenfarbe in den originalzustand
8071  4c 31 ea    jmp $ea31    ; zurueck zum interrupt
------------------------------directory (hier kann beliebige funktion
                                ausgefuehrt werden.)
8074  a9 01       lda #$01
8076  20 c3 ff    jsr $ffc3    ; close 1
8079  a9 24       lda #$24
807b  8d f0 03    sta $03f0
807e  a9 30       lda #$30
8080  8d f1 03    sta $03f1
8083  a9 01       lda #$01
8085  a2 08       ldx #$08
8087  a0 00       ldy #$00
8089  20 ba ff    jsr $ffba    ; fileparameter (1,8,0) setzen
808c  a9 02       lda #$02
808e  a2 f0       ldx #$f0
8090  a0 03       ldy #$03
8092  20 bd ff    jsr $ffbd    ; filenamenparameter (2,15,3)
8095  20 c0 ff    jsr $ffc0    ; open
8098  a9 40       lda #$40
809a  20 90 ff    jsr $ff90    ; status setzen  st=64
809d  a2 01       ldx #$01
809f  20 c6 ff    jsr $ffc6    ; eingabegeraet = 1 setzen
80a2  20 90 ff    jsr $ff90    ; status setzen
80a5  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80a8  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80ab  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80ae  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80b1  c9 00       cmp #$00     ; zeichen = 0 ?
80b3  f0 31       beq $80e6    ; ja, dann ende
80b5  a9 01       lda #$01
80b7  85 cc       sta $cc      ; cursorblinken ausschalten
80b9  a2 01       ldx #$01
80bb  20 c6 ff    jsr $ffc6    ; chkin  eingabegeraet = 1 setzen
80be  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80c1  a8          tay          ; akku ins y-register
80c2  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80c5  48          pha          ; zahl auf stapel legen
80c6  98          tya          ; y->akku
80c7  aa          tax          ; akku->x
80c8  68          pla          ; zahl vom stapel holen
80c9  20 cd bd    jsr $bdcd    ; dezimalzahl ausgeben
80cc  a9 20       lda #$20
80ce  20 d2 ff    jsr $ffd2    ; leerzeichen ausgeben
80d1  20 cf ff    jsr $ffcf    ; basin (zeichen holen)
80d4  c9 00       cmp #$00     ; zeichen = 0 ?
80d6  d0 08       bne $80e0    ; nein, dann zur zeichenausgabe
80d8  a9 0d       lda #$0d
80da  20 d2 ff    jsr $ffd2    ; zeilenvorschub ausgeben
80dd  b8          clv
80de  50 cb       bvc $80ab
80e0  20 d2 ff    jsr $ffd2    ; zeichenausgabe
80e3  b8          clv
80e4  50 eb       bvc $80d1    ; naechstes zeichen holen ($80d1)
80e6  a9 01       lda #$01
80e8  20 c3 ff    jsr $ffc3    ; close 1
80eb  20 cc ff    jsr $ffcc    ; clrch
80ee  60          rts          ; zurueck zum interrupt
```

Das Quellisting zum Directory auf Tastendruck

Star SG-10 und Textomat

Haben Sie Probleme, den Star SG-10 mit Secus-Interface an Textomat anzupassen? Verzweifeln Sie nicht, hier kommt die Lösung.

Der erste und wichtigste Schritt ist die Eingabe folgender Basic-Kommandos:

OPEN 4,4,25 <Return>
PRINT # 4

Die dabei verwendete Sekundäradresse »25« teilt dem Drucker mit, daß er den vollen Star-Zeichensatz fest einstellen soll. Die PRINT-Anweisung muß gegeben werden, damit der Drucker den Lock-Zustand ausführt. Um diesen Status wieder aufzuheben, muß der Drucker ausgeschaltet werden.

Als Zweites laden Sie Textomat wie gewohnt und nehmen im Menü-Punkt »Druckeranpassung« die entsprechenden Änderungen vor.

Beim Punkt »Drucker« wählen Sie die »1«. Die ASCII-Werte entnehmen Sie dem Interface-Handbuch auf Seite 14 oder dem Star-Handbuch auf den Seiten 217 bis 223. Für den Bereich »Steuerzeichen« empfehle ich folgende Befehlssequenzen:

Zeichenabstand:	di10	1b4201	(Pica)
	di12	1b4202	(Elite)
	di15	1b4203	(entspricht 17 Zeichen/inch = Schmalschrift)
Zeilenabstand:	ab1	1b4106	(1zeiliger Abstand)
	ab2	1b410c	(2zeiliger Abstand)
	ab3	1b4118	(3zeiliger Abstand)

Die anderen Steuerzeichen können Sie leicht selbst einsetzen, wobei Sie den Seitenwechsel nicht mit dem Wert 0c belegen sollten, da der Drucker dann immer zwei Seiten vorschiebt.

Sie sollten auch noch beachten, daß die DIP-Schalter am Drucker richtig eingestellt sind. Im Interface-Handbuch steht zwar, daß die Schalter 2-2, 2-3 und 2-4 auf OFF-Stellung sein sollen, ich habe aber die Erfahrung gemacht, sie besser in ON-Stellung zu lassen. Wichtigstes Argument hierfür ist das nochmalige Vorschieben um eine Zeile. Des weiteren stelle ich den 1. und 7. Schalter auf OFF.

Ich hoffe, Sie freuen sich jetzt genauso wie ich an dem tollen Schriftbild Ihres SG-10 oder SG-15 in Verbindung mit dem Textverarbeitungsprogramm Textomat.

(Klaus Croll/gk)