

Tips & Tricks für Profis

Bisher erschien es nicht möglich, auf dem C 64 Daten mit 4800 bit/s zu übertragen. Ein kleines Programm schafft Abhilfe. Außerdem läßt sich der eigentlich nicht nutzbare Bildschirmrahmen nun doch sinnvoll beschreiben.

Im Handbuch des C 64 steht überhaupt nichts, im Programmierhandbuch nichts Genaues — was ist mit den freidefinierbaren Übertragungsraten los?

Beim C 64 handelt es sich bei der RS232-Schnittstelle um eine bitserielle, asynchrone Übertragung. Das heißt ein Byte wird in Bits zerteilt und diese werden dann über eine Leitung in Form von Spannungswechseln gesendet. Unter asynchroner Übermittlung versteht man die Verwendung von Kennzeichen für Byteanfang und Byteende. Bei manchen Computern können die Daten wunschweise auch synchron übermittelt werden. Der Computer sendet dann kein Start- oder Stoppbit mehr, es werden nur noch die Daten übertragen. Vorteil: Die Übertragungszeit verkürzt sich, da die Steuerbits entfallen.

RS232 mit 4800 bit/s

Beim C 64 ist die Übertragung rein softwaremäßig geregelt. Er setzt dazu den CIA #2 ein. Dieser Baustein besitzt zwei 16-Bit-Intervall-Timer, die von einem bestimmten Wert auf Null zählen und dann einen NMI auslösen können. Dies nutzt die RS232-Schnittstelle aus. Die Software holt den Wert für den Timer B, der als Interruptgenerator dient, aus einer im ROM verankerten Tabelle. Es existieren hierzu zwei verschiedene Tabellen, aber dazu später mehr. Diese Werte werden in den Timer und in zwei Speicherstellen gebracht. Danach startet der Timer. Da beim C 64 das Senden und Empfangen über RS232 interruptgesteuert geschieht, werden zwei Puffer (Zwischenspeicher) benötigt. Einer der beiden Puffer steht für die zu sendenden Daten bereit. Die Daten werden nur in den Puffer geschrieben. Das Senden geschieht mit Hilfe des NMI-Interrupts, weil man ihn (im Gegensatz zum IRQ) nicht sperren kann. So werden die Daten in jedem Fall gesendet, es sei denn, man stoppt den Timer oder verbiegt den NMI-Vektor. Da die Daten in den Puffer geschrieben werden und das Beschreiben des Puffers normalerweise schneller geht als das Senden der Zeichen, sollte man, nachdem ein Zeichen gesendet wurde, das nächste Zeichen vorbereiten und warten, bis der Puffer frei ist. Nun kann das Zeichen problemlos in den Puffer geschrieben werden. Beachtet man dies nicht, könnte versehentlich ein noch nicht gesendetes Zeichen überschrieben werden. Die Folge wäre eine fehlerhafte Übertragung. Das Empfangen geschieht auf ähnliche Weise, denn die RDX-Leitung (hier gelangen die Daten vom anderen Computer zum C 64) ist nicht nur mit dem Port B des CIA #2 verbunden, sondern auch mit dem Eingang FLAG der CIA, der einen NMI auslösen kann. Das Empfangen geschieht also unabhängig vom Programm, daher werden die Daten in den Empfangspuffer geschrieben, der genauso groß ist wie der Sendepuffer: 256 Byte. Man sollte sich also darum kümmern, daß die Zeichen rechtzeitig aus dem Puffer gelesen werden. Durch die Kernelroutine READST, oder beim C 64 auch durch Auslesen der Speicherstelle \$90, läßt sich der Status abfragen. Bei gesetztem Bit 2 lief der Empfangspuffer über. Um nun andere Übertragungsraten zu benutzen, könnten natürlich die Werte in den Timer und die entsprechenden Werte in die beiden Speicherstel-

len geschrieben werden. Dies hätte nur den großen Nachteil, daß die so erstellten Programme schwer an andere (Commodore-)Computer anzupassen wären, da diese nicht die gleichen Speicherstellen benutzen. Aber dies ist gar nicht nötig, denn es geht auch sehr viel eleganter!

Die RS232-Schnittstelle öffneten wir bisher folgendermaßen:

```
lda #2 ;Logische Filenummer = 2
ldx #2 ;Gerätenummer der RS232
ldy #3 ;Sekundäradresse (ohne Funktion)
jsr setpar ;Werte setzen
lda #2 ;Länge des Filenamens
ldx # < fadr ;Adresse des Filenamens (low)
ldy # > fadr ;(high)
jsr setnam ;Name für OPEN setzen
jsr open ;Datei öffnen
ldx # < ebuf ;Zeiger für Empfangspuffer
ldy # > ebuf ;und
stx $f7
sty $f8
ldx # < abuf ;Ausgabepuffer setzen
ldy # > abuf
stx $f9
sty $fa
fadr .byte %00000110 ;Wert für Steuerregister
      .byte %00000000 ;Wert für Befehlsregister
```

Übrigens werden die Zeiger auf die Puffer auch vom Betriebssystem gesetzt, das die Puffer an das obere (Basic-)RAM-Ende legt. Dies ist auch der Grund, weswegen beim Öffnen der Schnittstelle ein CLR ausgeführt wird, denn im gleichen Speicherbereich liegen ja auch die Stringvariablen. Wenn wir jetzt zum Beispiel sequentielle Dateien senden oder empfangen wollten, könnte es bei langen Dateien passieren, daß die Puffer und die Datei sich überlappen, so daß Datenverluste entstehen. Darum legen wir die Puffer in uns angenehme Speicherbereiche, was man entweder vor oder nach dem OPEN-Befehl machen kann. Vorher deswegen, da das Betriebssystem die Pufferadressen erst festlegt, nachdem es in den Speicherstellen nachgesehen hat, ob das vom Programm schon gemacht wurde. Ist das High-Byte der Pufferzeiger ungleich Null, wird davon ausgegangen, daß die Zeiger bereits auf die Puffer deuten. Deshalb wird nichts mehr verändert.

Übersicht der Werte für:

Steuerregister	Befehlsregister
Bit 7:	Bit 7,6,5:
0 1 Stoppbit	000 keine Parität
1 2 Stoppbits	001 gerade Parität
	011 ungerade Parität
	101 8. Datenbit = 1
	111 8. Datenbit = 0
Bit 5,6:	Bit 4:
00 8 Datenbits	0 Vollduplex
01 7 Datenbits	1 Halbduplex
10 6 Datenbits	
11 5 Datenbits	
Bit 0,1,2,3:	Bit 0 (Handshake):
0000 selbstdef. Übertr.	0 3-Line (Software-)
0001 50 bit/s	1 X-Line (Hardware-)
0010 75 bit/s	
0011 110 bit/s	
0100 134,5 bit/s	
0101 150 bit/s	
0110 300 bit/s	
0111 600 bit/s	
1000 1200 bit/s	
1001 1800 bit/s	
1010 2400 bit/s	

Und nun geht's los!

Der Filename kann auch vier Zeichen lang sein. Die ersten beiden Zeichen behalten ihre Bedeutung bei, aber die nächsten beiden beinhalten den Wert für die Übertragungsrate.

Der Filename sieht also wie folgt aus:

»Steuerregister« »Befehlsregister« »Übertragungsrate-low«
»Übertragungsrate-high«

Wie gesagt, das untere Nibble vom »Steuerregister« muß %0000 sein! Die Werte für »Übertragungsrate-low« und »Übertragungsrate-high« berechnet man wie folgt:

$\text{Übertragungsrate-high} = \text{INT}(\text{Prozessortakt} / \text{Übertragungsrate} / 2 - 100) / 256$

$\text{Übertragungsrate-low} = \text{INT}(\text{Prozessortakt} / \text{Übertragungsrate} / 2 - 100 - \text{Übertragungsrate-high} \times 256)$

Der Prozessortakt beträgt bei der deutschen PAL-Version 985259 Hertz, bei der amerikanischen NTSC-Version 1022730 Hertz. Das heißt nichts anderes, als daß die Programme auf amerikanischen Geräten schneller laufen als auf deutschen. Da dies natürlich auch Auswirkungen auf die Übertragungskapazität hat, sind im Betriebssystem zwei Tabellen für den Timer B vorhanden. Wenn Sie nun Programme mit selbstdefinierten Übertragungsraten schreiben, sollten Sie immer daran denken, daß diese Programme entweder nur auf dem deutschen oder dem amerikanischen C 64 richtig laufen, wenn Sie nur eine Tabelle verwenden. Durch Auslesen der Speicherstelle \$02A6 können Sie ganz einfach feststellen, ob Sie eine PAL- oder eine NTSC-Version vor sich haben. Bei einer PAL-Version steht in besagter Speicherstelle der Wert 1, bei NTSC eine Null.

Nun ein Beispiel: Die Übertragungsrate soll 1800 Bit/s betragen. Also erhalten wir für »Übertragungsrate-high«: $\text{INT}((985250/4800/2-100)/256) = 0$ und für »Übertragungsrate-low«: $\text{INT}(985250/4800/2-100-0 \times 256) = 2$. Der Filename lautet also: %XXXX0000, %XXXXXXXX, %00000010, %00000000. Achtung: Für die X-Bits setzen Sie die entsprechenden Werte aus der Tabelle ein.

Übungen:

Wir möchten ein sequentielles File mit 4800 bit/s, 7 Datenbits, 1 Stoppbit, 3-Line-Handshake und keiner Parität senden. Also: Zuerst das sequentielle File in den Speicher laden. Jetzt müssen wir noch die RS232-Schnittstelle öffnen, wobei wir auf die Abfrage auf PAL- oder NTSC-Version verzichten. Das Programm zum Öffnen ist bis auf den Filenamen und dessen Länge von dem altbekannten nicht zu unterscheiden.

```
lda #2      ;logische Filenummer = 2
ldx #2      ;Geraetenummer der RS232
ldy #3      ;Sekundaeradresse
jsr setlfs  ;Werte setzen
lda #4      ;der Filename ist 2 Zeichen lang
ldx #fadr   ;Adresse des Filenamens (low)
ldy #fadr   ;(high)
jsr setnam  ;Name für OPEN setzen
jsr open    ;Datei oeffnen
ldx #2      ;Datei mit der logischen Filenummer
jsr chkout  ;#2 als Standardausgabegerät
```

Nun müssen Sie einen Zeiger auf den Anfang des in den Speicher geladenen sequentiellen Files setzen und die Pufferzeiger entsprechend ändern.

```
(Zeiger »pt« auf Dateianfang setzen)
sende ldy #0 ;aktuelles Zeichen aus dem
lda (pt),y ;File in den Akku holen
(Zeichen nun bearbeiten, zum Beispiel in ASCII-Code
wandeln)
tax          ;Zeichen merken
wloop lda $02a1 ;ist das vorherige Zeichen
and #1      ;bereits gesendet worden?
bne wloop   ;nein: =>warten
txa         ;gemerktes Zeichen zurueckholen
jsr chrout  ;auf RS232 ausgeben
```

Der Zeiger »PT« muß nun erhöht und auf Fileende getestet werden. Ist die Datei noch nicht zu Ende, muß wieder zum Label SENDE gesprungen werden.

```
jsr clrchn ;Standardwerte Ein-/Ausgabe
lda #2     ;setzen und Datei schließen
```

```
jsr close
rts          ;das war's!
fadr .byte %00100000
.byte %00000000
.byte %00000010
.byte %00000000
```

Das folgende Programm (Listing 1, bitte mit dem MSE eingeben) sendet sequentielle Files mit 4800 Bit/s, 8 Datenbits, 1 Stoppbit, keine Parität, Vollduplex und 3-Line-Handshake.

Die Syntax lautet: SYS 52480,konv-flag," Name"

Wenn konv-flag = 0, werden die Zeichen erst in ASCII-Code gewandelt und dann gesendet. Bei konv-flag = 128 erfolgt die Übertragung unverändert.

Mit dem Namen ist der Titel der sequentiellen Datei gemeint. »S,R« wird vom Programm nicht automatisch ergänzt. Sie müssen es also noch an den Namen anhängen. Beispiel: SYS 52480,128," TEST,S,R"

Die Datei TEST wird unkonvertiert gesendet.

(Martin Müller/aw)

```
Name : screencopy $cf11 cf11 cfff
cf11 : 78 a2 23 a0 cf 8e 14 03 7f
cf19 : 8c 15 03 58 a9 41 8d f2 bc
cf21 : cf 60 ad 8d 02 49 07 29 16
cf29 : 07 f0 03 4c 31 ea ad 0e 30
cf31 : dc 29 fe 8d 0e dc a9 01 84
cf39 : a2 08 a0 01 20 ba ff a9 53
cf41 : 06 a2 ed a0 cf 20 bd ff 1d
cf49 : 20 c0 ff a2 01 20 c9 ff 56
cf51 : a9 40 20 d2 ff a9 00 20 0a
cf59 : d2 ff ad 88 02 85 f8 a9 2b
cf61 : 00 85 f7 a9 19 8d ec cf a8
cf69 : a9 28 8d eb cf a0 00 20 49
cf71 : f3 cf b1 f7 49 80 91 f7 86
cf79 : 20 f9 cf c9 20 d0 04 a9 af
cf81 : df d0 21 c9 a0 f0 1d 49 e2
cf89 : 80 29 7f c9 20 b0 05 09 65
cf91 : 40 4c a5 cf c9 40 90 0c 54
cf99 : c9 60 b0 05 09 80 4c a5 70
cfa1 : cf 18 69 40 20 d2 ff a0 b9
cfa9 : 00 20 f3 cf b1 f7 49 80 b1
cfb1 : 91 f7 20 f9 cf e6 f7 d0 3b
cfb9 : 02 e6 f8 ce eb cf d0 22
cfc1 : a9 28 8d eb cf a9 20 20 6a
cfc9 : d2 ff ce eb cf d0 f6 ce c9
cfd1 : ec cf d0 94 20 cc ff a9 27
cfd9 : 01 20 c3 ff ee f2 cf ad fc
cfe1 : 0e dc 09 01 8d 0e dc 4c 15
cfe9 : 31 ea 00 00 42 49 4c 44 b8
cff1 : 20 41 78 a2 34 86 01 60 60
cff9 : a2 37 86 01 58 60 00 00 81
```

Listing 1.
RS232 mit
4800 Bit/s.
Bitte mit dem
MSE eingeben.

Änderung der Tastaturbelegung

Es kommt des öfteren vor, daß man anstelle der alten Tastaturbelegung eine andere benutzen möchte, um beispielsweise Programme leichter eingeben zu können. Oder man möchte anstelle der amerikanischen QWERTY- eine QWERTZ-Tastatur haben. Eine Lösungsmöglichkeit wäre, die Zeichen Y und Z im Zeichensatz umzudefinieren. Die gebräuchlichste Lösung dieses Problems ist, das ROM ins RAM zu kopieren und die Zeichen in der Tastaturliste auszutauschen. Will man dies jedoch in Basic ausführen, ist das sehr zeitaufwendig und schwierig.

Das folgende Programm (Listing 2) macht es Ihnen sehr leicht. Bis auf die Tasten <SHIFT>, <CBM>, <CTRL> und <RESTORE> kann jeder Taste des C 64 ein beliebiger CHR\$-Code zugeordnet werden. Der Tasten-Code stimmt mit dem jeweiligen Bildschirm-Code überein. Die Syntax des neuen Befehls lautet: !A,B

A ist der Bildschirm-Code, dem der CHR\$-Code B zugeordnet wird. Wollen wir beispielsweise das Zeichen A in ein B umwandeln, muß folgender Befehl angewandt werden: !1,66 (und !2,65, falls das B in ein A geändert werden soll).

Die Zahl 1 ist der Bildschirm-Code des Zeichens A, dem der Wert 66 (CHR\$-Code von B) zugeordnet wird. Es ist dabei unwichtig, ob das B vorher geändert war oder nicht. Der CHR\$-Code wird aus dem ROM gelesen.

Um eine QWERTZ-Tastatur zu erhalten, sind folgende Befehle einzugeben:

!25,90 (wandelt Y in Z um)

!26,89 (wandelt Z in Y um)

Soll die Cursor-Steuerung auf die Funktionstasten gelegt werden, so benötigt man folgende Befehle:

!39,145 (<F1> = <CRSR>-hoch)

```
0 DATA 120,162,53,142,8,3,162,192,142,9,3,
169,0,133,251,133,253,169,160 <253>
1 DATA 133,252,169,224,133,254,162,32,160,
0,177,251,145,251,177,253,145 <037>
2 DATA 253,200,208,245,230,254,230,252,202
,208,238,169,53,133,1,88,96,32 <099>
3 DATA 115,0,201,33,240,6,32,121,0,76,231,
167,32,115,0,32,158,183,134,2 <063>
4 DATA 160,0,224,64,144,14,224,128,144,8,2
24,192,144,2,200,200,200,200,200 <149>
5 DATA 200,152,170,189,121,235,133,251,189
,122,235,133,252,165,2,41,63,170 <128>
6 DATA 189,135,192,133,2,32,253,174,32,158
,183,138,164,2,145,251,169,53 <020>
7 DATA 133,1,76,174,167,234,46,10,28,20,18
,14,21,26,29,33,34,37,42,36,39 <248>
8 DATA 38,41,62,17,13,22,30,31,9,23,25,12,
64,48,64,54,57,60,2,7,0,51,63 <193>
9 DATA 1,4,5,6,49,40,47,43,44,55,56,59,8,1
1,16,19,24,27,32,35,45,50,64,53,64,3 <190>
10 FOR I=49152 TO 49350:READ A:POKE I,A:NE
XT <198>
11 REM AKTIVIEREN SYS49152 <215>
12 REM SYNTAX !A,B <027>
```

Listing 2. Leichtes Umdefinieren der Tastaturbelegung

!40,17 (<F3> = <CRSR>-unten)

!41,157 (<F5> = <CRSR>-links)

!63,29 (<F7> = <CRSR>-rechts)

Beim Abtippen von »DATA-Wüsten« lohnt es sich, die Komma-Taste zu verschieben. Der Befehl !31,44 verlegt diese auf die <->-Taste.

Einige der Zeichen können jedoch nicht geändert werden. Es sind die Bildschirmcodes von 33 bis 41 sowie der Code 63. Diese Zeichencodes sind anderweitig belegt.

Zeichencode	Neue Belegung
33	<CRSR>-links
34	<CRSR>-hoch
35	<DEL>
36	<HOME>
37	<STOP>
38	<RETURN>
39	<F1>
40	<F3>
41	<F5>
63	<F7>

Im Klartext heißt das, daß Sie, wenn Sie den Bildschirm-Code 33 mit dem Befehl !33,65 ändern wollen, nicht das Ausrufrungszeichen, sondern die <CRSR>-links-Taste in ein A umwandeln.

Die Initialisierung der Routine erfolgt durch SYS 49152. Danach steht Ihnen der Befehl !A,B zur Verfügung. Der Wertebereich von A und B liegt zwischen 0 und 255. Löschen läßt sich die Tastenbelegung, indem man die ursprüngliche Belegung aus dem ROM kopiert. Dies geschieht durch: POKE 1,55:SYS 49152 (Andreas Schindler/dm)

Laufschrift im Bildschirmrahmen

Seit einiger Zeit ist es in Mode gekommen, im Titelbild mancher Programme mindestens eine Laufschrift darzustellen. Nachdem sich die Programmierertechnik hier immer weiter fortentwickelt hat, so daß sich die Laufschriften jetzt schon hochauflösend und flimmerfrei bewegen, bleibt immer noch ein großes Problem: Jede Laufschrift benötigt Platz auf dem Bildschirm. Will man zum Beispiel eine Laufschrift in ein HiRes-Bild einblenden, mußte bisher immer das Bild so weit angepaßt werden, daß eine oder zwei Zeilen für die Laufschrift freibleiben. Hier stellt sich die Frage: Muß das so sein?

```
Name : laufschrift c000 c0b0
c000 : 78 a9 5c a0 c0 8d 14 03 47
c008 : 8c 15 03 a9 81 8d 1a d0 a3
c010 : a9 00 8d 0e dc a9 08 a2 5f
c018 : 0f 9d 00 d0 ca ca 10 f9 47
c020 : a9 18 a2 00 9d 00 d0 18 cb
c028 : 69 30 e8 e8 e0 10 d0 f4 bc
c030 : a9 e0 8d 10 d0 a0 ff a2 06
c038 : 07 98 9d f8 07 ad 86 02 0e
c040 : 9d 27 d0 88 ca 10 f2 a9 02
c048 : ff 8d 17 d0 8d 1d d0 8d 0e
c050 : 15 d0 58 60 01 02 04 08 30
c058 : 10 20 40 80 ad 19 d0 8d 9a
c060 : 19 d0 ad 12 d0 c9 f7 90 eb
c068 : 0d a9 31 8d 12 d0 a9 13 bc
c070 : 8d 11 d0 4c 81 ea a9 f7 49
c078 : 8d 12 d0 a9 1b 8d 11 d0 7c
c080 : a2 0e a0 07 de 00 d0 bd df
c088 : 00 d0 c9 ff f0 08 ca ca 73
c090 : 88 10 f1 4c 31 ea ad 10 67
c098 : d0 59 54 c0 8d 10 d0 39 51
c0a0 : 54 c0 f0 ea a9 7f 9d 00 fb
c0a8 : d0 d0 e3 00 06 52 78 14 d6
```

Listing 3.
Laufschriften im
Bildschirmrahmen.
Bitte mit dem MSE
eingeben.

Listing 4.
Editor-Programm zur
Rahmen-Laufschrift.
Beachten Sie die
Eingabehinweise
auf Seite 99.

```
100 REM ***** <209>
110 REM * * * * * <159>
120 REM * DEMO - LAUFSCHRIFT * <100>
130 REM * * * * * <179>
140 REM * VON N.NEBEL 21.2.1986 * <036>
150 REM * * * * * <199>
160 REM ***** <013>
170 REM <232>
180 POKE 56,62:CLR:ZA=53248 <064>
190 PRINT "{CLR,DOWN}2 TEXTE EINGEBEN, JEWE
ILS HOECHST. 24 Z." <233>
200 INPUT "1. ";A$:INPUT "2. ";B$ <220>
210 PRINT "{CLR}"A$:"PRINT" {HOME,2DOWN}"B$ <181>
220 FOR I=15872 TO 16382:POKE I,0:NEXT <101>
230 POKE 56334,0:POKE 1,51 <073>
240 FOR A=0 TO 7:FOR B=0 TO 2 <243>
250 AD=ZA+B*PEEK(1024+A*3+B) <139>
260 A2=ZA+B*PEEK(1104+A*3+B) <166>
270 FOR I=0 TO 7:POKE 15911+A*64+B*I*3,PEE
K(A2+I) <091>
280 POKE 15872+A*64+B*I*3,PEEK(A2+I):NEXT <143>
290 NEXT:NEXT <039>
300 POKE 1,55:POKE 56334,1 <181>
310 POKE 53280,0:POKE 16383,255:SYS 49152 <038>
```

Nein, denn wozu gibt es den weitgehend ungenutzten Bildschirmrand, der lange Zeit immer nur die Farbe annehmen konnte, die in \$D020 angegeben war. Vor einiger Zeit kam jemand auf die Idee, diesen Platz dem VIC durch geschicktes Umschalten in den 24-Zeilen-Modus (durch Löschen des Bit ROW SELECT im Register \$D011) als nutzbaren Raum zu entlocken. Die Euphorie über diese Entdeckung wurde jedoch rasch wieder gebremst: Außer dem einen Byte am Ende des VIC-Adreßraums, das nur langweilige Streifen erzeugt, kann der VIC in diesem Bereich nur Sprites darstellen. So lag die einzige Nutzung dieses zusätzlichen Raumes darin, ein paar einsame Sprites über den Rand hinauswandern zu lassen. Diese Sprites können aber auch sinnvoller eingesetzt werden!

Bei einigermaßen geschicktem Einsatz der acht zur Verfügung stehenden Sprites läßt sich eine brauchbare Laufschrift auf dem Rahmen erzeugen. Es sind aber durchaus auch zwei Laufschriften möglich. Genau dies tut nun das vorliegende Programm (Listing 3, bitte mit dem MSE eingeben). Es scrollt acht Sprites, deren Inhalt in \$3E00 bis \$3FFF abgelegt ist, von rechts nach links über den Bildschirm. Das dazugehörige kleine Basic-Programm (Listing 4) berechnet aus den eingegebenen Texten die Daten für diese Sprites und schreibt sie gleich in den richtigen Bereich.

Nachdem das Basic-Programm etwa zehn Sekunden lang gerechnet hat, erscheinen die eingegebenen Texte als Laufschriften ober- und unterhalb des Bildschirms. Ein erneutes Einschalten ist nach <RUN/STOP+RESTORE> durch SYS 49152 möglich, da die Daten im Speicher erhalten bleiben. Achtung: Vor dem erneuten Starten des Basic-Teils muß die Interrupt-Routine durch <RUN/STOP+RESTORE> oder

Fortsetzung auf Seite 88

Beispiel: FRQ 1200

2) FILTER 49165 bis 49255,0

Funktion: Einstellen der Resonanz und Einschalten der Stimmen.

Syntax: FILTER re,st1,st2,st3,ex

re= Resonanz (0 bis 15)

st= Stimmen (0 aus/1 ein)

ex= Extern (0 aus/1 ein)

Beispiel: FILTER 15,1,0,1,0

Listing 7. Modulnummer: 47

Anzahl der Befehle: 1

1) FMOD 49152 bis 49213,0

Funktion: Festlegen der Lautstärke und der Filterart.

Syntax: FMOD la,ho,ba,ti

la= Lautstärke (0 bis 15)

ho= Hochpaß (1 ein/0 aus)

ba= Bandpaß (1 ein/0 aus)

ti= Tiefpaß (1 ein/0 aus)

Beispiel: FMOD 15,1,0,1

Listing 8. Modulnummer: 48

Anzahl der Befehle: 1

1) FIND 49152 bis 49419,0

Listet alle Zeilen in denen sich der angegebene String befindet.

Beispiel: FIND PRINT

Listing 9. Modulnummer: 49

Anzahl der Befehle: 1

1) KEY 49152 bis 49402,143

Diese Routine arbeitet mit dem Kassettenpuffer. Dorthin wird der Tastaturabfrage-Vektor umgestellt. Jede Funk-

tionstaste wird vierfach (normal, (SHIFT), (CBM), (CTRL)) mit je maximal 35 Zeichen belegt. Diese werden ab \$CDBE gespeichert.

Beispiel: KEY3,30,14, "Poke53281,0:Poke53280,0",13

Der erste Parameter gibt die Funktionstastennummer ((CBM):9-12, (SHIFT):13-16) an, der Text in Anführungszeichen wird direkt ausgegeben, die Zahlen sind CHR\$-Codes. Die Funktionstastenabfrage wird eingeschaltet, das heißt beim Drücken der (F3)-Taste erscheint ein schwarzer Bildschirm, grüne Zeichen und der Klein-Großschrift-Modus wird eingeschaltet.

Beispiel: KEY

Die Funktionstastenbelegung wird im Anführungszeichenmodus angezeigt. (RETURN) wird mit dem Pfeil nach links gekennzeichnet. Die Funktionstastenabfrage wird eingeschaltet.

Beispiel: KEY3,0

Die Belegung der (F3)-Taste wird gelöscht.

Beispiel: KEY0

Die Funktionstastenabfrage wird abgeschaltet.

Listing 10. Modulnummer: 40

Anzahl der Befehle: 2

1) PAGE 49152 bis 49211,0

Syntax: PAGEx,y,z

Legt Arbeitsbereich (Page) x von Zeile y bis Zeile z fest

Beispiel: Page 2, 100, 200

2) L 49212 bis 49251,0

Syntax: Lx

Listet Page x

Beispiel: L2

```
Name : 48erw .ass          c000 c10b
c000 : 20 79 00 c9 00 d0 03 4c 41
c008 : 08 af a2 00 c9 2c f0 13 78
c010 : 9d 00 02 20 73 00 c9 00 90
c018 : f0 09 c9 2c f0 05 e8 e0 21
c020 : 0a d0 ed e8 a9 00 9d 00 3c
c028 : 02 20 9b a6 a5 2b 85 7a b5
c030 : a5 2c 85 7b a9 01 85 fb 6d
c038 : 20 79 00 85 f4 4c 43 c0 ac
c040 : 20 73 00 20 73 00 a2 00 e0
c048 : 20 73 00 c9 00 d0 02 c6 77
c050 : fb 85 14 20 73 00 c9 00 75
c058 : d0 04 c5 fb f0 3a 85 15 3c
c060 : 20 73 00 c9 00 f0 d9 dd 1e
c068 : 00 02 d0 27 e8 bd 00 02 03
c070 : d0 ee a9 0d 20 d2 ff a5 a7
c078 : 7a 48 a5 7b 48 20 13 a6 0e
c080 : 20 99 c0 68 85 7b 68 85 8b
c088 : 7a 20 73 00 c9 00 f0 b0 b1
c090 : 4c 89 c0 a2 00 4c 60 c0 8b
c098 : 60 a0 01 84 0f b1 5f f0 f7
c0a0 : 40 20 2c a8 c8 b1 5f aa fd
c0a8 : c8 b1 5f c5 15 d0 04 e4 8b
c0b0 : 14 f0 02 b0 2c 84 49 20 1f
c0b8 : cd bd a9 20 a4 49 29 7f 0a
c0c0 : 20 47 ab c9 22 d0 06 a5 b4
c0c8 : 0f 49 ff 85 0f c8 f0 11 49
c0d0 : b1 5f d0 0e a8 b1 5f aa 12
c0d8 : c8 b1 5f 86 5f 85 60 d0 67
c0e0 : b8 60 10 dc c9 ff f0 d8 7a
c0e8 : 24 0f 30 d4 38 e9 7f aa 61
c0f0 : 84 49 a0 ff ca f0 08 c8 27
c0f8 : b9 9e a0 10 fa 30 f5 c8 c5
c100 : b9 9e a0 30 b7 20 47 ab 27
c108 : d0 f5 60 00 00 00 00 eb
```

Listing 8. »FIND«

```
Name : 49erw .ass          c000 c179
c000 : 20 79 00 d0 0c a9 0d 20 79
c008 : d2 ff 20 96 c0 60 4c 08 05
c010 : af 20 7e c0 20 9e b7 e0 1f
c018 : 00 d0 0b a9 48 8d 8f 02 ac
c020 : a9 eb 8d 90 02 60 e0 11 fd
c028 : b0 e4 ca 8a 0a aa bd a6 88
c030 : 03 85 fb e8 bd a6 03 85 3a
c038 : fc a0 00 20 79 00 f0 39 56
c040 : c9 2c d0 ca 20 73 00 c9 de
c048 : 22 d0 1f 20 73 00 c9 3a 71
c050 : f0 0e c9 00 f0 23 c9 22 4d
c058 : d0 06 20 73 00 4c 3b c0 73
c060 : 91 fb c8 c0 24 d0 e4 4c 2e
c068 : 79 c0 98 48 20 9e b7 68 17
c070 : a8 8a 91 fb c8 c0 24 d0 06
c078 : c2 a9 00 91 fb 60 a2 00 8e
c080 : bd eb c0 9d 3c 03 e8 e0 58
c088 : 8e d0 f5 a9 3c 8d 8f 02 a3
c090 : a9 03 8d 90 02 60 20 7e d1
c098 : c0 a2 00 8a 48 48 e8 a9 b8
c0a0 : 00 20 cd bd a9 2e 20 d2 0e
c0a8 : ff a9 01 85 d4 68 0a aa 7b
c0b0 : bd a6 03 85 fb bd a7 03 84
c0b8 : 85 fc a0 00 a7 36 85 01 48
c0c0 : b1 fb f0 11 c9 0d 00 02 19
c0c8 : a9 5f 20 d2 ff c8 c0 23 13
c0d0 : f0 03 4c c0 c0 a9 0d 20 3b
c0d8 : d2 ff a9 37 85 01 68 aa 52
c0e0 : e8 e0 10 d0 b6 a9 91 20 96
c0e8 : d2 ff 60 a5 9d f0 0e a5 6b
c0f0 : cb a2 00 dd c6 03 f0 08 20
c0f8 : e8 e0 04 d0 f6 4c 48 eb 36
c100 : c5 c5 f0 f9 85 c5 8a ae 31
c108 : 8d 02 d0 04 0a 4c 7c 03 46
```

Listing 9. »KEY«

```
c110 : e0 01 d0 0a 18 69 01 0a cb
c118 : 38 e9 01 4c 7c 03 e0 03 78
c120 : d0 06 18 69 08 4c 7c 03 01
c128 : 18 69 0c 0a aa bd a6 03 72
c130 : 85 fb bd a7 03 85 fc a0 a9
c138 : 00 b1 fb f0 13 c9 0d d0 83
c140 : 07 8d 77 02 a9 01 85 c6 72
c148 : 20 d2 ff c8 c0 25 d0 e9 36
c150 : a5 c5 4c 48 eb be cd e2 a6
c158 : cd 06 ce 2a ce 4e ce 72 a1
c160 : ce 96 ce ba ce de ce 02 a7
c168 : cf 26 cf 4a cf 6e cf 92 5c
c170 : cf b6 cf da cf 04 05 06 27
c178 : 03 00 00 00 00 00 00 00 7c
```

Listing 9. (Schluß)

```
Name : 40erw .ass          c000 c064
c000 : 20 9e b7 e0 0a 90 05 a2 f8
c008 : 0e 20 37 a4 8a 0a 0a 48 3a
c010 : 20 fd ae 20 8a ad 20 f7 65
c018 : b7 68 48 aa a5 14 9d bc 56
c020 : 02 a5 15 9d bd 02 20 fd 56
c028 : ae 20 8a ad 20 f7 b7 68 b0
c030 : aa a5 14 9d be 02 a5 15 22
c038 : 9d bf 02 60 20 4a e5 20 3d
c040 : 9e b7 8a 0a 0a 48 aa bd a7
c048 : bc 02 85 14 bd bd 02 85 c6
c050 : 15 20 13 a6 68 aa bd be 5f
c058 : 02 85 14 bd bf 02 85 15 26
c060 : 20 c9 a6 60 e6 7b a9 fc 05
```

Listing 10. »PAGE«

Fortsetzung von Seite 79

durch Eingabe von »SYS 58648:SYS 64789:SYS 64931« (klappt nicht immer) abgeschaltet werden.

Mögliche Änderungen des Programms: Am einfachsten läßt sich die Farbe der Schrift ändern. Hierzu ruft man die Routine nur mit einer anderen Cursorfarbe auf. Zum Verschieben der Laufschriften in Y-Richtung muß der Inhalt der Speicherstelle 49174 verändert werden (normal = 8). In

Adresse 49206 steht die Blocknummer des letzten Sprites (normal = 255). Wird diese verändert, müssen natürlich auch im Basic-Teil die Zahlen 15911 und 15872 in Zeile 270/280 verändert werden. Auch mit einem geänderten Zeichensatz läßt sich die Laufschrift einsetzen. Dazu muß die Variable ZA in Zeile 180 auf den Beginn des neuen Zeichensatzes zeigen. Setzt man hier 55296 ein, erscheint die Laufschrift in Klein-schrift.

(Niklas Nebel/dm)