EPROM-Trans — Die Super-Erweiterung

Der ROM-Speicher des C 64 ließ sich bislang nur mit Steckplatinen im Expansion-Port vergrößern. Mit einem raffinierten Trick geht es auch intern.

Bild 6. So wird EPROM-Trans in den C 64 eingebaut

Wen hat es nicht schon geärgert, daß die Grundversion des C 64 kein Monitorprogramm aufweist, das nach dem Einschalten des Computers sofort zur Verfügung steht? Deshalb wurde eine Möglichkeit gesucht (und gefunden), mit der man Maschinenprogramme von einem festinstalliertem EPROM in den Bereich $C000 bis $CFFF transformieren kann, ohne für das EPROM vorhandenen Speicherplatz in Anspruch zu nehmen.

Um diese Schaltung zu realisieren, wurde bei den C 64-Entwicklern einiges abgeguckt. Es soll hier nicht die Funktionsweise des C 64 im Detail erläutert werden (das würde zu weit führen), doch läßt es sich für das Verständnis dieser Schaltung nicht vermeiden, auf den Speicherbereich $A000 bis $BFFF (hier liegt das Basic-ROM) näher einzugehen.

Wird dieser Bereich angesprochen, so wird das Basic-ROM über das sogenannte PLA-Chip (Programable-Logic-Array), auch Adreßraummanager genannt, unter bestimmten Voraussetzungen (zum Beispiel Lesen) aktiviert, das heißt der $\overline{CE}$ (Chip-Enable)-Anschluß auf Low-Potential gelegt. Drückt man zum Beispiel »RETURN«, so wird zur Adresse $A560 gesprungen. Diese liegt im Speicherbereich, in dem das Basic-ROM liegt. Dieses ROM ist folglich aktiviert, und das dort stehende Maschinenprogramm (Eingabe einer Zeile) wird abgearbeitet.

Wie aus Bild 1 ersichtlich, wurde dieser Bereich ($A000 bis $BFFF) benutzt. Über diesem ROM-Bereich liegt zum einen das »EPROM-Trans« ($A000 bis $AFFF) und zum anderen die EPROMs 1 bis 3 ($B000 bis $BFFF). In letzteren steht jeweils ein beliebiges Programm (bis 4 KByte), welches in den Bereich $C000 bis $CFFF transformiert werden soll. Ich habe zum Beispiel in EPROM 1 den »SMON« und in EPROM 2 »Fast Tape« abgelegt, EPROM 3 ist noch unbenutzt. Das Maschinenprogramm in EPROM-Trans (siehe Listing) soll den Inhalt des jeweils selektierten EPROMs 1 bis 3 von dem Bereich $B000 bis $BFFF in den Bereich $C000 bis $CFFF transformieren.

Wie kann dies jedoch funktionieren, wenn in diesem Bereich wie oben erwähnt, beim Lesen immer das Basic-ROM aktiviert ist? Für die Speichertransformation wird der Inhalt des Basic-ROMs nicht benötigt. Drückt man nun einen der drei Taster (Bild 2), wird die $\overline{CE}$-Leitung des Basic-ROMs auf die Zusatzschaltung umgeleitet. Ist folglich ein Taster gedrückt, dann ist das Basic-ROM nicht mehr ansprechbar; kein Taster gedrückt entspricht dem Normalzustand des Computers.

Wie aus dem Listing ersichtlich, liegt bei der Adresse $0560 (≙ $A560) des Maschinenprogramms beim EPROM-Trans ein Sprungbefehl zur Adresse $A000, wo dann die Speichertransformation startet und mit einem Sprungbefehl nach $C000 endet. Will man diese Speichertransformation starten, muß der Computer zur Adresse $A560 oder $A000 springen. Wie oben schon erwähnt, wird die Adresse $A560 automatisch beim Drücken der »RETURN«-Taste angesprungen. Wird zum Beispiel EPROM 1 durch Betätigen von Taster 1 selektiert und dann »RETURN« gedrückt, startet das Maschinenprogramm in EPROM-Trans. Nun wird mit Adreßbit A12 zwischen den Bereichen $Axxx und $Bxxx hin- und hergeschaltet und so der komplette Inhalt von EPROM 1 nach $C000 bis $CFFF kopiert. Am Ende der Transformation wird das Programm durch den JMP-Befehl gestartet, so daß sich zum Beispiel der SMON gleich mit seiner Registeranzeige meldet. Nach Loslassen des Tasters steht auch das Basic-ROM wieder zur Verfügung.

Ein eventuell vorhandenes Basic-Programm wird während der Speichertransformation selbstverständlich nicht beschädigt. Änderungen sind nur im Bereich $C000 bis $CFFF zu registrieren. Die Transformation weist keine Zeitprobleme auf; wenn man »RETURN« betätigt (≙ Start der Transformation), meldet sich auch gleich der SMON (um beim Beispiel zu bleiben).

In EPROM 1 bis 3 wird das Maschinenprogramm so abgelegt, wie es auch in $C000 bis $CFFF stehen würde, da ja $B000 nach $C000, $B001 nach $C001 und $BFFF nach $CFFF transformiert wird.

In Bild 2 ist die Steuerung der $\overline{CE}$-Eingänge der einzelnen EPROMs und des Basic-ROMs dargestellt.

Das IC 74LS139 besteht aus zwei 1 auf 4 Demultiplexer DM1 und DM2.

Je nach Codierung der Eingangsadresse (A0 und A1) wird der Eingang E auf den entsprechenden Ausgang D0 bis D3 gelegt (Tabelle 1), ansonsten haben die Ausgänge High-Potential.

Realisierung der Hardware

Mit DM1 erfolgt die eigentliche Selektierung der EPROMs. Es bestehen vier Möglichkeiten: Basic-ROM und EPROM 1 bis 3. Der Adreßcode ist abhängig von den Schaltzuständen der Taster T1 bis T3. Aktiviert werden alle EPROMs mit »Active Low« (Low-Potential). $\overline{\text{BASIC}}$ ist die Leitung, die normalerweise das Basic-ROM aktiviert, also dann Low-Pegel hat, wenn im Bereich $A000 bis $BFFF gelesen wird. Dieses Signal ist an die Eingänge E der Demultiplexer DM1 und DM2 geführt. Liegt $\overline{\text{BASIC}}$ auf, kann kein Baustein aktiviert werden, denn die Ausgänge D0 bis D3 von DM1 und DM2 sind high oder entsprechend dem Pegel an E (hier dann auch high). Am $\overline{\text{CE}}$-Eingang vom Basic-ROM steht folglich ein High-Pegel, entsprechendes gilt für EPROM-Trans. Die Ausgänge D1 bis D3 von DM1 werden invertiert, so daß an den betreffenden Eingängen der NAND-Gatter Low-Potential liegt. Bekanntlich ist aber der Ausgang eines NAND-Gatters nur dann low, wenn beide Eingänge high sind. Folglich werden EPROM 1 bis 3 nicht aktiviert. Kommen wir jetzt zum interessanteren Teil: $\overline{\text{Basic}}$ für LOW. Als erstes wollen wir prüfen, ob der Computer »normal« arbeitet, wenn keine Taste gedrückt ist:

a) keine Taste gedrückt:

Wenn $\overline{\text{BASIC}}$ low ist, muß in diesem Fall das Basic-ROM aktiviert werden. Durch die Tasterstellungen liegt an den Eingängen A0, A1 von DM1 Low-Potential. Dies bedeutet, die Information an E ($\overline{\text{BASIC}}$ = low) wird zum Ausgang D0 (vergleiche Tabelle 1) übertragen. Das Basic-ROM wird also aktiviert (unabhängig von A12). EPROM 1 bis 3 können nicht aktiviert werden, da die Ausgänge D1 bis D3 von DM1 High-Potential führen (siehe oben). Aber was ist mit EPROM-Trans? Hier (DM2) wird das Eingangssignal E auf den Ausgang D2 übertragen, wenn A0 low und A1 high ist. A1 ist aber abhängig vom $\overline{\text{CE}}$-Signal des Basic-ROMs. Ist dieses aktiviert (low), kann EPROM-Trans nicht aktiviert werden. Der Computer arbeitet in diesem Fall also völlig normal.

b) Taster T1 wird gedrückt

Wird Taster T1 gedrückt, liegt an A0 von DM1 über den Pull-up-Widerstand (1,5 kOhm) high, A1 bleibt über T2 und T3 auf low. Diese Adreßcodierung bedeutet, daß die Information an E auf den Ausgang D1 übertragen wird. Durch den Inverter liegt also am NAND-Gatter (Pin 1) high.

Als Adresse an DM2 liegt an A1 jetzt high, da das Basic-ROM nicht aktiviert wird. A0 ist abhängig vom Adreßbit A12. Wie erwähnt ist A12 für den Bereich $Axxx low. Die Eingangsinformation (low) wird auf D2 gelegt und EPROM-Trans wird aktiviert. An Pin 2 vom NAND-Gatter liegt Low-Potential (high von D3 invertiert), weshalb EPROM 1 nicht aktiviert wird. Wechselt Adreßbit A12 seinen Zustand (von $Axxx → $Bxxx) wird Eingang E von DM2 mit Ausgang D3 zusammengeschaltet, denn A0=0 und A1=1. EPROM-Trans wird nicht mehr aktiviert (D2 → High). An D3 liegt nun aber Low-Potential welches über einen Inverter auf das NAND-Gatter (Pin 2) geführt wird. An Pin 1 und 2 dieses Gatters liegt folglich high und am Ausgang liegt Low-Pegel. EPROM 1 wird aktiviert.

c) Taster T2/T3 wird gedrückt

Das Prinzip ist das gleiche wie eben beschrieben. Es ändert sich nur die Adreßcodierung von DM1.

Taster	DM1 A0	DM1 A1	Basic	A12	Basic-ROM	EPROM-Trans	EPROM 1	EPROM 2	EPROM 3
kein Taster gedrückt	0	0	0	0	0	1	1	1	1
	0	0	0	1	0	1	1	1	1
Taster T1 gedrückt	1	0	0	0	1	0	1	1	1
	1	0	0	1	1	1	0	1	1
Taster T2 gedrückt	0	1	0	0	1	0	1	1	1
	0	1	0	1	1	1	1	0	1
Taster T3 gedrückt	1	1	0	0	1	0	1	1	1
	1	1	0	1	1	1	1	1	0

0 ≙ LOW-Potential / 1 ≙ HIGH-Potential

Tabelle 1. Codierung des Demultiplexers (H=High)

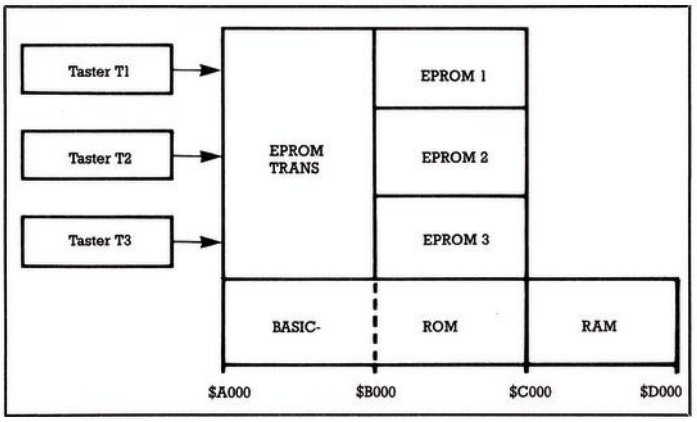


Bild 1. Speicherorganisation mit Zusatzschaltung

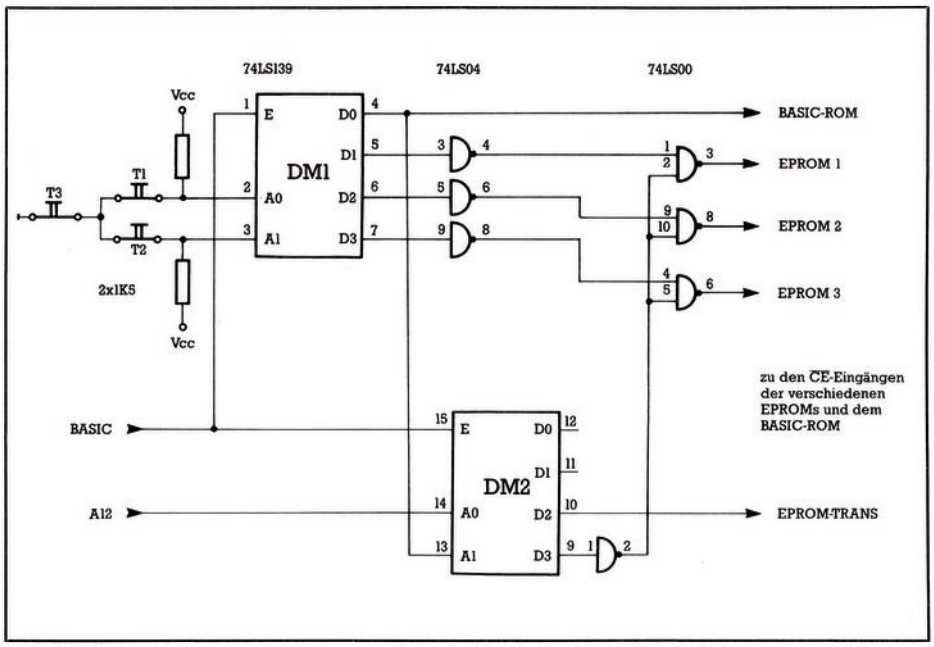


Bild 2. Zusatzschaltung zur Steuerung der verschiedenen EPROMs und dem Basic-ROM

Um ein Kabelgewirr zu vermeiden »zapft« man am besten ein ROM im Computer an, das heißt man erstellt eine Adapterplatine, auf der sämtliche zusätzliche EPROMs sowie das entwendete ROM Platz finden und Adreß- und Datenbus parallel verbunden werden. Diese Platine wird dann in den Steckplatz des herausgenommenen ROMs wie ein einzelnes IC gesteckt, wobei die Anschlüsse des ROMs direkt verbun-

Kernal oder Basic-ROM?

den sind, während die zusätzlichen Anschlüsse der EPROMs (±, $\overline{CE}$, $\overline{OE}$, ...) extern herausgeführt werden. Welches ROM man verwendet, hängt von den ganz individuellen Wünschen ab. Ich habe mich für das Kernal entschieden, da ich so für ein geändertes Betriebssystem gleich einen EPROM-Steckplatz mit vorgesehen habe, so daß zum Beispiel diejenigen, die eine Betriebssystem-Erweiterung benutzen, ohne Änderungen die Adapterplatine nach dem Layout Bild 3 oben verwenden können. Dieser zusätzliche Steckplatz »Kernal neu« (EPROM vom Typ 2764) ist 100%ig zum Kernal verdrahtet, das heißt man kann nur einen der beiden Kernalplätze besetzen. Die folgende Beschreibung bezieht sich auf das Entfernen des Kernal-ROMs.

Man erstellt sich zunächst die beiden Platinen nach den Layouts in Bild 3. Nachdem die angedeuteten Löcher für Bauteile und Drahtbrücken gebohrt (1 mm) sind, beginnt das Bestücken, wobei es ratsam ist, mit den von den Abmessungen niedrigsten Bauteilen zu beginnen. So ergibt sich folgende Reihenfolge: Drahtbrücken, Widerstände, Fassungen. Der Adaptersockel wird unter die IC-Fassung des auf dieser Platine vorgesehenen Platzes für das Kernal festgelötet, da er ja quasi nur eine Verlängerung der Kernal-Pins ist. Dies muß so durchgeführt werden, daß der Adaptersockel möglichst rechtwinklig zur Platine steht und in eine 24polige IC-Fassung paßt. Am einfachsten kann dieses realisiert werden, wenn man den Adaptersockel in eine 24polige Fassung steckt, und dann anlötet und ausrichtet.

Zwischen Zusatzplatine und Adapterplatine werden sieben Verbindungskabel gezogen, wobei die zugehörigen Anschlußpunkte den Bestückungsplänen Bild 4 zu entnehmen sind. Auf der Adapterplatine kann man hierbei eine Steckverbindung (10polig) vorsehen (zum Beispiel: Typ MS-25-10; Völkner-Elektronik, Postfach 5330, 33 Braunschweig; Preis 3,25 Mark), damit die beiden Platinen nicht starr miteinander verbunden sind.

Die Taster T1 bis T3 werden als letztes — eventuell erst nach dem Einbau — nach Bild 4 unten angeschlossen. Die Qualität der Taster ist unbedeutend, da der Faktor des Prellens keine Rolle spielt.

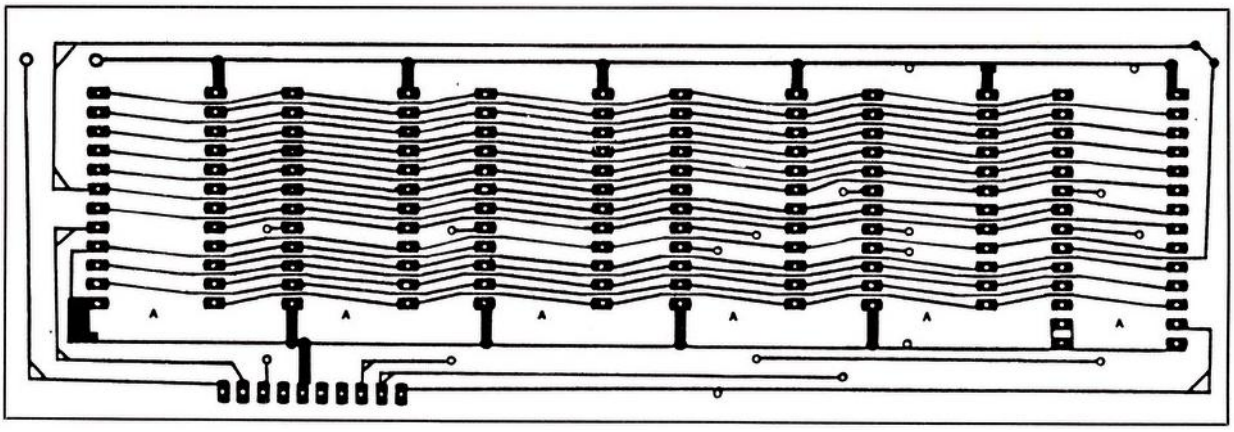
Verwendet man das hier abgebildete Platinenlayout, so müssen EPROM 1 bis 3 vom Typ 2532 sein, EPROM-Trans vom Typ 2716. Hier sollten wir noch einmal einen Blick auf Bild 5 werfen: Bis auf die $\overline{CE}$-Anschlüsse, die zur Zusatzplatine geführt werden, sind die Leitungen fest mit den des Kernal-ROMs verdrahtet (also Adreß- und Datenbus). Die restlichen epromspezifischen Anschlüsse ($\overline{OE}$, V_{pp}, $\overline{PRG}$) werden je nach Typ für EPROM-Lesen fest auf low oder high gelegt. Die Stromversorgung liegt beim Kernal an den Pins 12 (Masse) und 24 (V_{cc} = 5V); A12 entspricht Pin 21.

Einbau in den Computer

Zum Öffnen des Computers (vorher sämtliche Peripherie abziehen inklusive Stromversorgung, Achtung Garantieverlust) müssen die drei Schrauben an der vorderen Unterseite gelöst werden. Hinten sind Ober- und Unterteil des Gehäuses nur zusammengesteckt, so daß durch Anheben des Oberteils sich beide Gehäuseteile trennen lassen (Bild 6).

Jetzt müssen die beiden Steckverbindungen für Tastatur und Leuchtdiode gelöst werden, so daß beide Gehäuseteile völlig getrennt sind.

Als nächstes entfernt man das Kernal-ROM (Steckplatz U4) und steckt dieses in den entsprechenden Sockel auf der Adapterplatine gemäß dem Bestückungsplan.



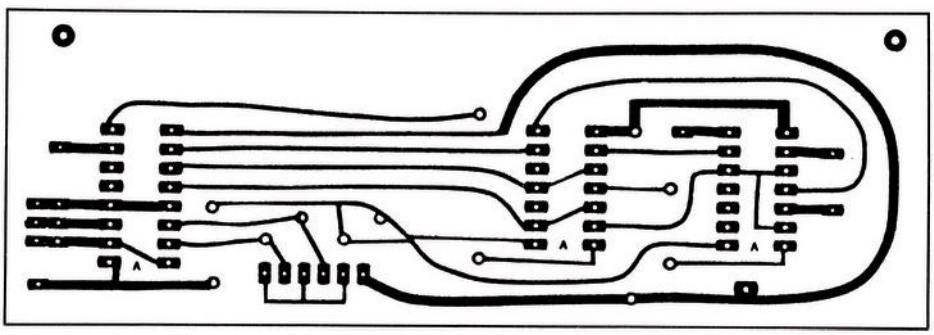


Bild 3. Layout der Adapterplatine (oben); Layout der Zusatzplatine (unten)

Sollte das ROM fest eingelötet sein, so ist dies unter Verwendung von Entlötlitze und einer guten Lötstation (nicht mit irgendeinem Lötkolben) herauszulöten. An diesem freigewordenen Platz wird dann ein 24poliger IC-Sockel eingesetzt.

Um die Leitungen $\overline{BASIC}$ und Basic-ROM zu erhalten, ist im Computer die Leiterbahn nach Bild 7 aufzutrennen. Hierzu nimmt man einen

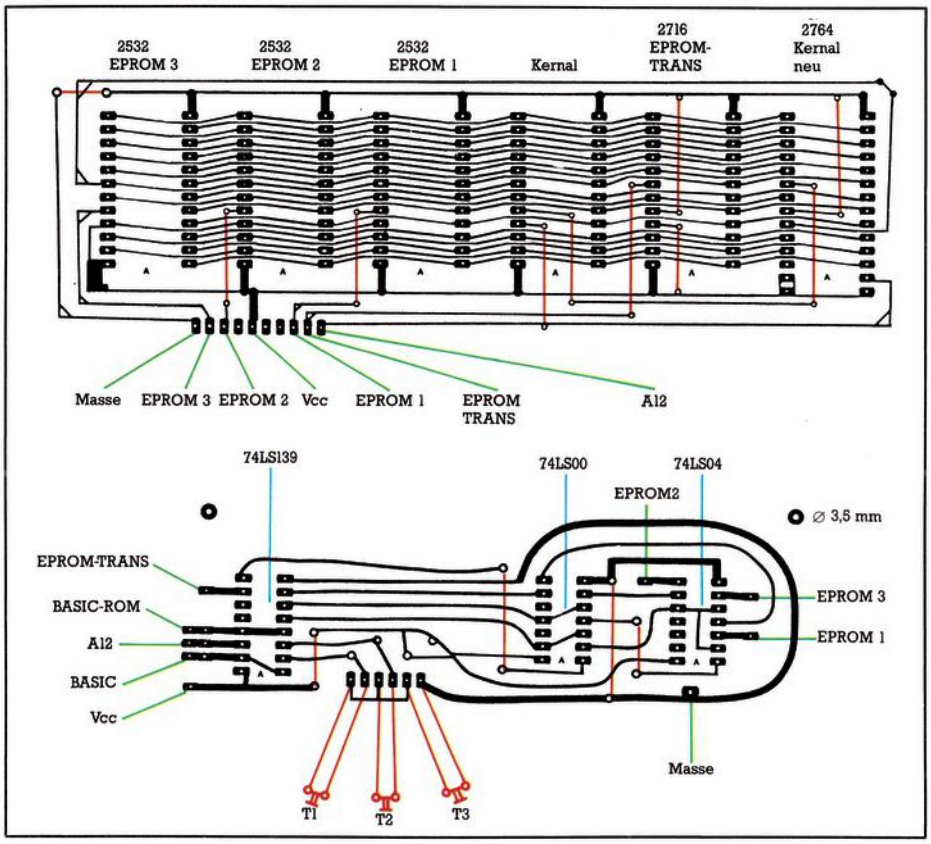


Bild 4. Bestückungspläne: Adapterplatine (oben); Zusatzplatine (unten). Die Kerben auf den ICs zeigen immer nach unten. Abgebildet ist die jeweilige Lötseite der Platinen.

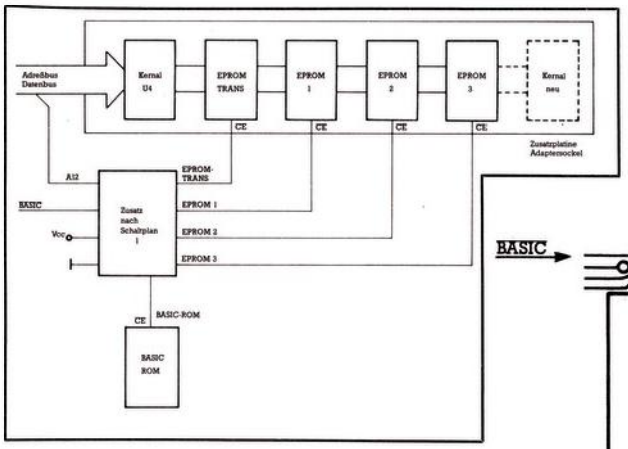


Bild 5. Komplettes Blockschaltbild zum Anschluß der Zusatzplatine

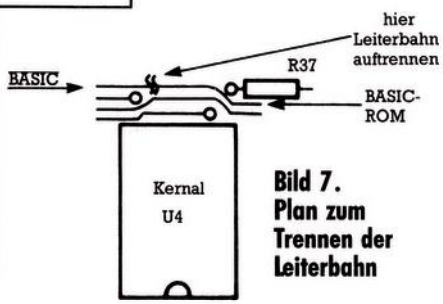


Bild 7. Plan zum Trennen der Leiterbahn

kleinen Schraubenzieher und unterbricht die Leiterbahn durch vorsichtiges Kratzen. Die grüne Schutzschicht auf der Leiterbahn wird an beiden Seiten (rechts und links von der Unterbrechung) entfernt, so daß das Kupfer sichtbar wird. Nun wird an diesen Stellen jeweils ein flexibles Kabel mit etwas Fingerspitzengefühl angelötet. Die eine Leitung entspricht dann $\overline{\text{BASIC}}$ (vom PLA), die andere Basic-ROM (zu $\overline{\text{CE}}$ dieses ROMs) und sind als solche bezeichnet auf der Zusatzplatine zu finden und entsprechend zu verdrahten (eventuell mit Steckverbindung auf der Zusatzplatine).
Diejenigen, die anstatt des Kernals das Basic-ROM anzapfen

Fortsetzung auf Seite 140

```
10 FOR I=32768 TO 32947                                    <213>
20 READ X:POKE I,X: S=S+X: NEXT                            <222>
30 DATA 162,  0,189,  0,176,157,  0,192,23
   2,208,247,162,  0,189,  0,177,157                       <180>
40 DATA   0,193,232,208,247,162,  0,189,
   0,178,157,  0,194,232,208,247,162                       <099>
50 DATA   0,189,  0,179,157,  0,195,232,20
   8,247,162,  0,189,  0,180,157,  0                       <154>
60 DATA 196,232,208,247,162,  0,189,  0,18
   1,157,  0,197,232,208,247,162,  0                       <102>
70 DATA 189,  0,182,157,  0,198,232,208,24
   7,162,  0,189,  0,183,157,  0,199                       <185>
80 DATA 232,208,247,162,  0,189,  0,184,15
   7,  0,200,232,208,247,162,  0,189                       <058>
90 DATA   0,185,157,  0,201,232,208,247,16
   2,  0,189,  0,186,157,  0,202,232                       <153>
100 DATA 208,247,162,  0,189,  0,187,157,
     0,203,232,208,247,162,  0,189,  0                     <137>
110 DATA 188,157,  0,204,232,208,247,162,
     0,189,  0,189,157,  0,205,232,208                     <013>
120 DATA 247,162,  0,189,  0,190,157,  0,2
   06,232,208,247,162,  0,189,  0,191                      <149>
130 DATA 157,  0,207,232,208,247, 76,  0,1
   92,247                                                  <213>
140 IF S<>    25763 THEN PRINT "DATA-FEHLE
   R !": END                                               <232>
150 POKE 34144,76:POKE 34145,0:POKE 34146,
   160                                                     <229>
160 PRINT "O.K."                                           <219>
```

Listing 2. Basic-Lader des Programms

```
,A000  A2 00     LDX #00      Block 1
,A002  BD 00 B0  LDA B000,X
,A005  9D 00 C0  STA C000,X
,A008  E8        INX
,A009  D0 F7     BNE A002
,A00B  A2 00     LDX #00      Block 2
,A00D  BD 00 B1  LDA B100,X
,A010  9D 00 C1  STA C100,X
,A013  E8        INX
,A014  D0 F7     BNE A00D
,A016  A2 00     LDX #00      Block 3
,A018  BD 00 B2  LDA B200,X
,A01B  9D 00 C2  STA C200,X
,A01E  E8        INX
,A01F  D0 F7     BNE A018
,A021  A2 00     LDX #00      Block 4
,A023  BD 00 B3  LDA B300,X
,A026  9D 00 C3  STA C300,X
,A029  E8        INX
,A02A  D0 F7     BNE A023
,A02C  A2 00     LDX #00      Block 5
,A02E  BD 00 B4  LDA B400,X
,A031  9D 00 C4  STA C400,X
,A034  E8        INX
,A035  D0 F7     BNE A02E
,A037  A2 00     LDX #00      Block 6
,A039  BD 00 B5  LDA B500,X
,A03C  9D 00 C5  STA C500,X
,A03F  E8        INX
,A040  D0 F7     BNE A039
,A042  A2 00     LDX #00      Block 7
,A044  BD 00 B6  LDA B600,X
,A047  9D 00 C6  STA C600,X
,A04A  E8        INX
,A04B  D0 F7     BNE A044
,A04D  A2 00     LDX #00      Block 8
,A04F  BD 00 B7  LDA B700,X
,A052  9D 00 C7  STA C700,X
,A055  E8        INX
,A056  D0 F7     BNE A04F
,A058  A2 00     LDX #00      Block 9
,A05A  BD 00 B8  LDA B800,X
,A05D  9D 00 C8  STA C800,X
,A060  E8        INX
,A061  D0 F7     BNE A05A
,A063  A2 00     LDX #00      Block 10
,A065  BD 00 B9  LDA B900,X
,A068  9D 00 C9  STA C900,X
,A06B  E8        INX
,A06C  D0 F7     BNE A065
,A06E  A2 00     LDX #00      Block 11
,A070  BD 00 BA  LDA BA00,X
,A073  9D 00 CA  STA CA00,X
,A076  E8        INX
,A077  D0 F7     BNE A070
,A079  A2 00     LDX #00      Block 12
,A07B  BD 00 BB  LDA BB00,X
,A07E  9D 00 CB  STA CB00,X
,A081  E8        INX
,A082  D0 F7     BNE A07B
,A084  A2 00     LDX #00      Block 13
,A086  BD 00 BC  LDA BC00,X
,A089  9D 00 CC  STA CC00,X
,A08C  E8        INX
,A08D  D0 F7     BNE A086
,A08F  A2 00     LDX #00      Block 14
,A091  BD 00 BD  LDA BD00,X
,A094  9D 00 CD  STA CD00,X
,A097  E8        INX
,A098  D0 F7     BNE A091
,A09A  A2 00     LDX #00      Block 15
,A09C  BD 00 BE  LDA BE00,X
,A09F  9D 00 CE  STA CE00,X
,A0A2  E8        INX
,A0A3  D0 F7     BNE A09C
,A0A5  A2 00     LDX #00      Block 16
,A0A7  BD 00 BF  LDA BF00,X
,A0AA  9D 00 CF  STA CF00,X
,A0AD  E8        INX
,A0AE  D0 F7     BNE A0A7
,A0B0  4C 00 C0  JMP C000
-----------------------------------
   :                    Sprung: Start
   :
,A560  4C 00 A0  JMP A000
-----------------------------------
```

Listing 1. Speichertransformation $Bxxx nach $Cxxx.

Texteinschub # 2

Tape Header

Wenn ein Programm oder eine Datei auf Band gespeichert wird, setzt der Computer vor das Programm einen Vorspann, der auf englisch »Tape-Header« genannt wird. Da dieser Name weit verbreitet ist, will ich ihn hier beibehalten. Der Tape-Header ist 192 Byte lang. Er enthält alle wichtigen Angaben über das nachfolgende Programm.

Beim Laden eines Programms wird der Tape Header im Kassettenpuffer gespeichert, für den die Speicherstellen 828 bis 1019 reserviert sind. Von dort kann der Inhalt des Tape-Headers gelesen und analysiert werden.

Bevor wir das versuchen, will ich erst seine Zusammensetzung erklären.

Im **ersten Byte** steht eine Kennzahl für den Typ des Programms. Diese Kennzahl ist abhängig von der Sekundär-Adresse, die beim SAVEn eingegeben worden ist. Die Arten der Sekundär-Adressen und ihre Bedeutung ist im anderen Texteinschub »Files-Geräte-Namen-Nummern« genau beschrieben. Es gibt zwei Kennzahlen: 1 und 3.

In Anlehnung an die Erklärung der Sekundär-Adresse kann man die Kennzahl generell dadurch beschreiben, daß ein Programm mit Kennzahl 1 immer an den Anfang des zur Verfügung stehenden Programm-Speichers geladen wird. Hauptsächlich kommt das für Basic-Programme in Frage.

Eine Kennzahl 3 bewirkt, daß das Programm an diejenige Stelle des Programmspeichers geladen wird, wo es vor dem SAVEn gestanden hat. Das ist hauptsächlich der Fall bei Programmen in Maschinensprache.

In Verbindung mit der Bedeutung der Sekundär-Adresse kann man den Zusammenhang wie Tabelle 1 zeigt darstellen:

In **Byte 2 und 3** steht in Low/High-Darstellung die Adresse, ab der das Programm im Speicher des Computers stand, als es gespeichert wurde.

In **Byte 4 und 5** steht die entsprechende End-Adresse des Programms.

Ab **Byte 6 bis Byte 192** steht der Name des Programms. Er darf also maximal 187 Zeichen lang sein. Bei LOAD werden allerdings nur 16 Zeichen auf dem Bildschirm dargestellt.

Jetzt wollen wir das alles mit einem kleinen Experiment überprüfen.

Schreiben Sie bitte ein kleines Programm, es braucht nicht sehr sinnvoll zu sein, wie zum Beispiel:

```
10 REM TAPE HEADER
20 REM TEST PROGRAMM
```

Nehmen Sie ein leeres Band und laden das Programm mit einem Namen, der länger sein soll als 16 Zeichen, zum Beispiel:

```
SAVE "TEST PROGRAMM FUER INHALT TAPE HEADER"
```

Nach Drücken der RECORD- und PLAY-Tasten der Datasette meldet der Computer:

```
FOUND TEST PROGRAMM FU
```

Es werden also nur 16 Zeichen inklusive Leerzeichen gedruckt. Sobald das Programm geladen ist, schauen wir im Kassettenpuffer nach, was in den ersten fünf Bytes steht, danach lesen wir die restlichen Bytes des Puffers.

Geben Sie direkt, ohne Zeilennummer, ein:

```
FOR I=0 TO 4: PRINT PEEK (828+I);: NEXT,
```

Sie erhalten die Zahlen 1 1 8 41 8 (beim VC 20 mit 3-K-Speichererweiterung 1 1 4 41 4)

Danach geben wir wiederum direkt ein:

```
FOR I=5 TO 192: PRINT CHR$(PEEK(828+I));: NEXT
```

Beim VC 20 geben Sie in der FOR...NEXT-Schleife eine kleine Zahl ein, da der Bildschirmspeicher beim VC 20 kleiner ist als beim C 64.

Jetzt erscheint der volle Programmname, gefolgt von nicht sichtbaren Leerstellen. Wenn Sie in der letzten Direkteingabe den CHR$-Teil weglassen, dann druckt die Zeile die ASCII-Codes aus, und Sie sehen dann die Leerstellen als Zahl 32.

Diese Resultate habe ich zur besseren Übersicht in Tabelle 2 dargestellt.

Die Kennzahl in Byte »1« können Sie dadurch verändern, daß Sie dem oben verwendeten SAVE-Direktbefehl nach dem langen Namen ein ,1,1 anhängen. Im Ausdruck steht dann die Kennzahl »3«.

Übrigens, wenn Sie in den Speicherzellen 195/196 nachschauen, finden Sie dort denselben Wert wie in den Zellen 829/830, so wie die Beschreibung es in der Memory Map erklärt.

Vielleicht fragen Sie jetzt nach dem Nutzen dieser ausführlichen Erklärung. Nun, hauptsächlich kann man damit Programme, die eigentlich wegen LOAD ERROR nicht mehr ladbar sind, retten. Oder aber man kann durch Verändern der Zahlen in den Bytes 2 bis 5 nachträglich die Adressen ändern, in die das Programm geladen wird. Die erste Anwendung werde ich erklären, sobald wir zu den Adressen des Kassetten-Puffers selbst kommen.

Das Problem des LOAD oder SAVE mit geänderten Adressen ist aber zu umfangreich für einen Texteinschub innerhalb dieses Kurses. Es wäre eigentlich einen eigenen kleinen Beitrag wert.

Sekundär Adresse	Kennzahl	Bedeutung
0 oder leer	1	Basic-Programm
1	3	Maschinen-Programm
2	1	Basic-Programm mit End-Of-Tape-Marke
3	3	Maschinen-Programm mit End-Of-Tape-Marke

Tabelle 1

Adresse	828	829	830	831	832	833 etc.
Byte Nr.	1	2	3	4	5	6 etc.
Bedeutung	Kennzahl	Low Byte	High Byte	Low Byte	High Byte	Namen in ASCII-Code
Resultat	1	1	8	41	8	T etc.
bei C 64	1	(2089)				(2049)
Resultat	1	1	4	41	4	T etc.
bei VC 20		(1065)				(1025)

Tabelle 2

Fortsetzung von Seite 45

möchten, finden den Anschluß BASIC an Pin 20 vom Sockel des Basic-ROMs beziehungsweise der Adapterplatine, und Basic-ROM an Pin 20 des ROMs selber, wo die Leitungen dann anzuschließen sind. Am einfachsten ist dies wohl durch Herausbiegen des Pins am Basic-ROM (Achtung: nicht abbrechen) zu bewerkstelligen, so daß dieser nicht im Sockel steckt und ein Kabel angelötet werden kann.

Die Zusatzplatine ist so ausgelegt, daß man sie mit an den Schrauben, die auch zur Befestigung der Tastatur dienen, installieren kann. Ob man sie in der linken oder rechten Seite anbringt, ist völlig belanglos, nur sollte man die Taster auf der Gegenseite befestigen, beziehungsweise vorher prüfen, ob die Einbautiefe der Taster gering genug ist, so daß diese sich nicht mit der Zusatzplatine berühren.

Die Taster werden an dem oberen Gehäuseteil über der Tastatur montiert, wobei man darauf achten sollte, daß genügend Platz für die ausgewählten Taster zur Verfügung steht (Durchmesser; am besten vom Inneren des Gehäuses anpassen).

Ist die Adapterplatine so aufgesteckt, daß die Kerben der ICs in Richtung Kassettenport zeigen und mit EPROM-Trans, dem Kernal und mindestens einem EPROM 1 bis 3 bestückt, sind wir fertig und können den Computer wieder zusammenbauen.

Eventuell kann die Abschirmung — dies ist die Pappe mit der Alu-Schicht — so aufgeschnitten werden, daß beim Verlegen der Kabel von Adapter- zur Zusatzplatine keine Probleme auftreten.

Die Steckverbindung Tastatur — Computer kann nur in einer Richtung aufgesteckt werden; bei der Leuchtdiode zeigt das rote Kabel zum Ein-Aus-Schalter.

Gehäuse zusammenschrauben, fertig.

Ich betreibe diese Schaltung seit etwa 6 Monaten und möchte sie nicht mehr missen. Was ich damit aber zum Ausdruck bringen will, ist, daß die Stromversorgung den zusätzlichen Belastungen gewachsen ist; auch sind bis dato noch keine Wärmeprobleme aufgetreten.

Mir ist bekannt, daß sich viele C64-Besitzer schwer dazu durchringen können, ihren Computer äußerlich oder innerlich (Achtung! Garantieverlust) zu verändern. Diesmal ist es aber, glaube ich, eine Überlegung wert. Alle, die nicht die Möglichkeit zum Herstellen der Platinen haben, können sich an den Verlag wenden. Lesen Sie dazu unseren Hardware-Leserservice. (Peter Rausche/aw)