ZX-81-Peripherie:

Transparenter 16-KByte-Speicher

Die Schaltung des Memopak-RAMs

Lange Zeit gaben die Black-Box-Speicher für den ZX 81 Rätsel auf, da Memotech keinerlei Schaltunterlagen freigab. In mühevoller Kleinarbeit gelang es einem ZX-81-Fan jedoch, das Schaltbild des 16-KByte-RAMs zu rekonstruieren.

Seit es ihn gibt, ist der ZX 81 heftig umstritten: Die einen verdammen den Computer als „Spielzeug" in Grund und Boden, wogegen andere den Zwerg mit Vorliebe als Experimental-Computer nutzen. den „anderen" hat die FUNKSCHAU lange zur Seite gestanden, um den ZX 81 besser kennenzulernen. Jetzt wollen wir eines der letzten großen Rätsel lösen, das Hobby-Elektronikern immer wieder Kopfzerbrechen gemacht hat, und der weit verbreiteten 16-KByte-RAM-Erweiterung von Memotech unter die Haube schauen.

Der Adreßbus wird halbiert

Das RAM-Modul enthält drei Baugruppen: Den Speicher, eine Adreßdecodierung und die Spannungsversorgung (*Bild*). Acht dynamische RAM-ICs vom Typ 4116 (oder Vergleichstyp) bilden den Speicher. Diese ICs speichern die Informationen in winzigen Kondensatoren auf dem Chip. Wegen unvermeidbarer Leckströme, die die Kondensatoren entladen, müssen die Informationen jedoch spätestens alle 2 ms aufgefrischt werden. Dies übernimmt automatisch die Z-80-CPU im ZX 81: Sie spendet regelmäßig einen „Refresh"-Impuls an die Speicher-ICs. Nur ein $\overline{WAIT}$-Signal kann die CPU daran hindern.

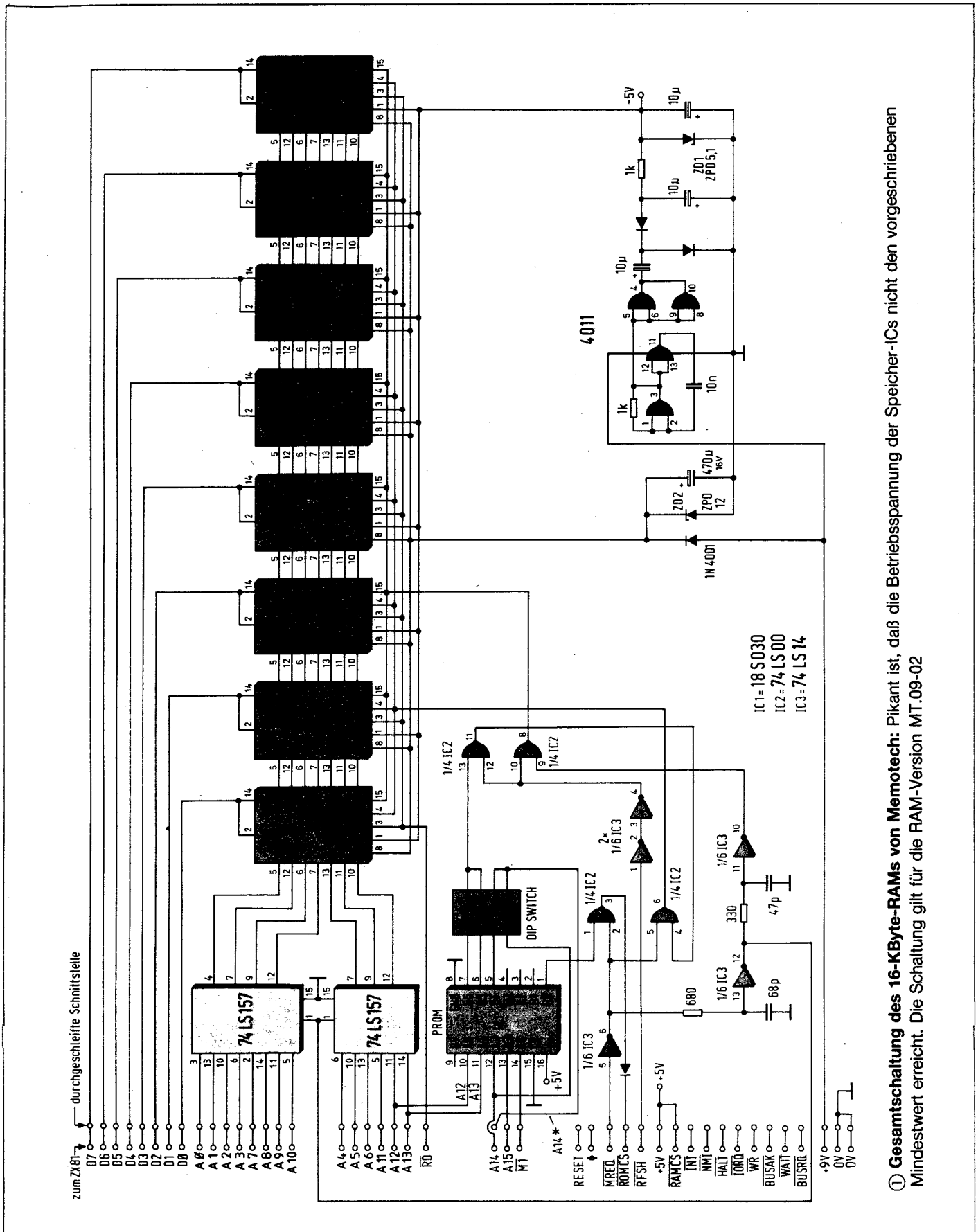
Jedes Speicher-IC hat eine Kapazität von 16 384 Bit (16 K × 1 Bit). Um ein einzelnes Bit adressieren zu können, müßte also jedes IC allein 14 Adreßleitungen haben. Dazu wäre ein großes IC-Gehäuse erforderlich. Um Platz zu sparen, werden die Adreßeingänge der ICs im Multiplex-Verfahren betrieben. Dann reichen sieben Adreßleitungen aus, wobei die in zwei Teile zerlegte Adresse nacheinander eingelesen wird (siehe auch FUNKSCHAU 12/1984, Seite 59). Zwei ICs vom Typ 74LS157 übernehmen im RAM-Modul die Aufspaltung der vom ZX 81 gesendeten 14 Adreßbit in zwei zeitlich versetzte 7-Bit-Blöcke.

Mit dem Anschluß des 16-KByte-RAMs wird das Signal RAM$\overline{CS}$ (ZX-81-Schnittstelle) zwangsweise auf H-Pegel gebracht. Damit ist das computerinterne RAM gesperrt. Dies ist nötig, weil es sonst zu einer Doppel-Adressierung des internen und externen RAMs kommen kann.

Aber auch das ROM des Computers birgt die Gefahr der Doppel-Adressierung. Der Grund dafür steckt in der unvollständigen Adreßdecodierung durch den ZX 81. Sie läßt es zu, daß ein und dieselbe ROM-Speicherzelle unter vier verschiedenen Adressen erreichbar ist. Nur in der Grundversion des ZX 81 stört

Tabelle: Ein PROM im RAM-Modul wirkt als Adreßdecoder zur Freigabe des ROMs oder RAMs in 4-KByte-Blöcken

	PROM-Adreß-eingänge					Datenausgänge								
						ROM $\overline{CS}$	A 14*	RAM $\overline{CS}$	RAM $\overline{CS}$	nicht verwendet				zugehörige Adressen
	$\overline{M1}$	A15	A14	A13	A12	B0	B4	B5	B6	B1	B2	B3	B7	
0	0	0	0	0	0	1	0	1	1	0	1	1	1	0– 4095
1	0	0	0	0	1	1	0	1	1	0	1	1	1	4096– 8191
2	0	0	0	1	0	0	0	1	1	0	1	1	1	8192–12287
3	0	0	0	1	1	0	0	1	1	0	1	1	1	12288–16383
4	0	0	1	0	0	0	0	0	0	1	0	1	1	16384–20479
5	0	0	1	0	1	0	0	0	0	1	0	1	1	20480–24575
6	0	0	1	1	0	0	0	0	0	1	0	1	1	24576–28671
7	0	0	1	1	1	0	0	0	0	1	0	1	1	28672–32767
8	0	1	0	0	0	0	1	1	1	1	1	0	1	32768–36863
9	0	1	0	0	1	0	1	1	1	1	1	0	1	36864–40959
10	0	1	0	1	0	0	1	1	1	1	1	0	1	40960–45055
11	0	1	0	1	1	0	1	1	1	1	1	0	1	45056–49151
12	0	1	1	0	0	0	0	0	0	1	1	1	0	49152–53247
13	0	1	1	0	1	0	0	0	0	1	1	1	0	53248–57343
14	0	1	1	1	0	0	0	0	0	1	1	1	0	57344–61439
15	0	1	1	1	1	0	0	0	0	1	1	1	0	61440–65535
16	1	0	0	0	0	1	0	1	1	1	1	1	1	0– 4095
17	1	0	0	0	1	1	0	1	1	1	1	1	1	4096– 8191
18	1	0	0	1	0	0	0	1	1	1	1	1	1	8192–12287
19	1	0	0	1	1	0	0	1	1	1	1	1	1	12288–16383
20	1	0	1	0	0	0	0	0	0	1	1	1	1	16384–20479
21	1	0	1	0	1	0	0	0	0	1	1	1	1	20480–24575
22	1	0	1	1	0	0	0	0	0	1	1	1	1	24576–28671
23	1	0	1	1	1	0	0	0	0	1	1	1	1	28672–32767
24	1	1	0	0	0	0	1	1	1	1	1	1	1	32768–36863
25	1	1	0	0	1	0	1	1	1	1	1	1	1	36864–40959
26	1	1	0	1	0	0	1	1	1	1	1	1	1	40960–45055
27	1	1	0	1	1	0	1	1	1	1	1	1	1	45056–49151
28	1	1	1	0	0	0	0	1	0	1	1	1	1	49152–53247
29	1	1	1	0	1	0	0	1	0	1	1	1	1	53248–57343
30	1	1	1	1	0	0	0	1	0	1	1	1	1	57344–61139
31	1	1	1	1	1	0	0	1	0	1	1	1	1	61140–65535



① **Gesamtschaltung des 16-KByte-RAMs von Memotech:** Pikant ist, daß die Betriebsspannung der Speicher-ICs nicht den vorgeschriebenen Mindestwert erreicht. Die Schaltung gilt für die RAM-Version MT.09-02

das nicht die Adressierung des RAMs (siehe auch Heft 10/1984, Seite 68). Im RAM-Modul aber muß ein weiterer Adreßdecoder verhindern, daß beim Adressieren des RAMs das ROM dazwischenfunkt. Dieser Decoder sorgt dafür, daß im Adreßbereich 16 KByte bis 32 KByte allein das RAM adressiert wird und das ROM nur noch den Adreßbereich 0 bis 8 KByte einnimmt.

Hinter dem Adreßdecoder steckt ein PROM

Der Adreßdecoder im RAM-Modul ist ein 32-Byte-PROM, also ein festprogrammierter Baustein mit sehr kurzer Zugriffszeit. Die fünf Adreßeingänge des ICs werden vom ZX 81 mit den Signalen A12 bis A15 und mit dem $\overline{M1}$-Signal gespeist. Abhängig vom Pegel auf diesen Leitungen, nehmen die acht Ausgänge des PROMs die vorprogrammierten Zustände ein (*Tabelle*). In Blöcken von 4 KByte läßt sich mit diesen Ausgangssignalen das ROM bzw. das RAM ordnungsgemäß freigeben. Dafür genügen schon vier Ausgangssignale.

Der Weg der PROM-Ausgangssignale wird von der Stellung der vier DIP-Schalter an der Modul-Rückseite bestimmt. Damit ist es möglich, mehrere 16-KByte-RAMs „aufzustocken", wobei ein nahtloses Aneinanderfügen der Adreßbereiche sichergestellt ist (siehe auch Heft 14/1983, Seite 63). Will man z. B. drei Module betreiben, dann müssen beim ersten und zweiten die Schalter 1 und 3 auf ON stehen, bei der letzten dagegen die Schalter 2 und 3.

Ist Schalter 1 geschlossen, dann ist die Adressierung von $\overline{M1}$ unabhängig. Und sobald Schalter 2 geschlossen ist (z. B. bei einem aufgestockten Modul) sind Maschinenprogramme oberhalb von 48 KByte nicht lauffähig (siehe auch Heft 3/1985, Seite 60).

Memotech's Sündenfall

Für die Speicher-ICs ist eine stabilisierte Mindest-Betriebsspannung von 10,5 V vorgeschrieben. So meldet es das Datenblatt. Um so erstaunlicher ist der Mut der Memotech-Entwickler, diesen Mindestwert zu mißachten, und die ICs mit der unstabilisierten ZX-81-Versorgungsspannung von etwa 9 V zu betreiben. Je nach Belastung des Netzteils durch Peripheriegeräte kann es daher zu ernsten Störungen kommen.

Sicherheitshalber sollte man daher in den Computer eine stabilisierte Spannung von 11 V einspeisen. Dann macht sich die Z-Diode ZD2 im RAM-Modul noch nicht bemerkbar und die Speicher-ICs erhalten die vorgeschriebene Betriebsspannung.

Am 5-V-Regler im ZX 81 wird dann allerdings eine erhebliche Verlustleistung frei, so daß der ohnehin knapp bemessene Kühlkörper sehr heiß wird. Ein deutlich größeres gut belüftetes Kühlblech ist daher Pflicht.

Mit einem geänderten PROM ließe sich das RAM-Modul auch als Speicher für andere Z-80-Computer verwenden. Dazu ist freilich ein Programmiergerät nötig. Den $\overline{M1}$-Eingang kann man dann z. B. mit A 11 verbinden, wenn die ROM-/RAM-Freigabe in Blöcken von 2 KByte erfolgen soll.

Relativ einfach ist der Umbau, wenn der Computer bereits die Adreßdecodierung für ein 16-KByte-RAM vornimmt und ein Chip-Select-Signal liefert. Dann darf man das PROM entfernen, und der $\overline{M1}$-Anschluß wird zum $\overline{CS}$-Eingang, wenn die jetzt freien Anschlüsse 7 und 14 verbunden werden.

Peter Hartmann/-ll

ZX-81-Softwaretip:

Bilderbuch

Mit einem kurzen Hilfsprogramm kann auch der ZX 81 den Bildspeicherinhalt auf Band speichern, indem er ihn zuvor einer Basic-Variablen zuweist. Nach dem Wiedereinlesen genügt dann ein PRINT-Befehl, um den ursprünglichen Bildinhalt schnell auf den Bildschirm zu bekommen.

Mit Hilfe des Hex-Laders (*Bild*) wird der Maschinencode des Hilfsprogramms nach GOTO 9996 in der REM-Zeile 1 untergebracht. Ab Zeile 2 wird dann der Platz für die abzuspeichernden Bilder im Variablenbereich reserviert. Dies geschieht mit DIM-Anweisungen – hier z. B. für 3 abzuspeichernde Bilder. Jedes Bild belegt 22 × 32 = 704 Byte plus einen Variablenkopf von 6 Byte.

Ab Zeile 6 darf man das Programm einfügen, das die Bildmuster erzeugt. Der Startbefehl für das Maschinenprogramm (RAND USR 16514) ist dann immer an der Stelle im Programm zu erteilen, ab der ein Bild fertig gezeichnet und bereit zum Abspeichern ist.

Nach dem Start sucht das Maschinenprogramm im Variablenspeicher nach der Variablen A$; es erkennt die Variablen an deren ersten beiden Bytes. Sobald die Variable gefunden wurde, lädt das Programm den Bildspeicherinhalt unter Weglassung der NEWLINE-Codes an jedem Zeilenende in das von der Variablen reservierte Feld. Danach wird automatisch die Variable B$ zur Aufnahme des zweiten Bildes gewählt und ins Basic-Programm zurückgesprungen. Hier läßt sich nun ein zweites Bildmuster mit RAND USR 16514 der Variablen B$ zuweisen.

Je nach RAM-Kapazität lassen sich so bis zu 26 Bilder speichern, wobei nach jedem Maschinenprogramm-Aufruf die nächste Variable in alphabetischer Reihenfolge zur Aufnahme des nächsten Bildes eingerichtet wird. Will man diese Reihenfolge durchbrechen, ist die Speicherzelle 16574 maßgebend: Sie hat für A$ den Inhalt 198 für B$ den Inhalt 199 bis hin zum Inhalt 223 für Z$. Gemäß dieser Zuordnung kann man den Inhalt dieser Speicherzelle mit einem POKE-Befehl gezielt ändern.

Verboten sind jetzt die Befehle RUN, CLEAR und selbstverständlich NEW. Jeder dieser Befehle würde den Variablenspeicher gnadenlos löschen. Per SAVE-Befehl kann jedoch das Programm samt Variablen auf Band gespeichert werden.

Um die Bilder wieder sichtbar zu machen genügt die Eingabe von PRINT A$, PRINT B$ usw. Kommt es beim Aufruf mehrerer Bilder zu der Meldung „Bildschirm voll" genügt zwischendurch ein CLS-Befehl und das nächste Bild wird in voller Pracht ausgegeben.

Hartmut Heinz/-ll

```
   1 REM HIER MINDESTENS 65 BE-
LIEBIGE ZEICHEN EINTIPPEN
   2 DIM A$(704)
   3 DIM B$(704)
   4 DIM C$(704)
   5 POKE 16574,198
   6 REM AB HIER HAUPTPROGRAMM
BIS BIS ZEILE 9994 EINFUEGEN
9995 REM HEX-LADER
9996 LET A$="2A14400118060A7ED424
44D2A10403A6E40ED6120853EC3BE3A6
E4020F43E00110650019ED2A0C402301 2
000ED60233CFE1620F6218E407E3C77C
9CF1EC6"
9997 FOR I=1 TO LEN A$ STEP 2
9998 POKE I/2+16513,(CODE A$(I)-
28)*16+CODE A$(I+1)-28
9999 NEXT I
```

Hilfsprogramm: Jeder Aufruf des Maschinenprogramms speichert einen Bildspeicherinhalt in einer Variablen, die sich auch auf Band abspeichern läßt

ZX-81-Speichererweiterung:

Huckepack-RAM

2 × 16 KByte = (vielleicht) 32 KByte

Zwei 16-KByte-RAMs vom Typ Memopak ergeben zusammengefügt nur dann 32 KByte, wenn die Codier-Schalter der Speicher richtig eingestellt sind.

Gegenüber der Original-Sinclair-RAM-Erweiterung haben die 16-KByte-RAMs von Memotech den Vorteil, daß die Schnittstelle des ZX 81 durchgeschleift wird, sie also trotz aufgestecktem RAM nicht blockiert ist (*Bild 1*). Damit sollten sich eigentlich zwei 16-KByte-RAMs ohne nennenswerte Probleme zu einem 32-KByte-Huckepack-RAM verbinden lassen. Leider ist dem nicht so, solange die vierfach DIL-Schalter an der Rückseite der RAMs nicht die richtige Stellung haben, und leider wurden RAMs verkauft, bei denen die Einstellanweisung fehlt.

Der K-Cursor ist gesucht

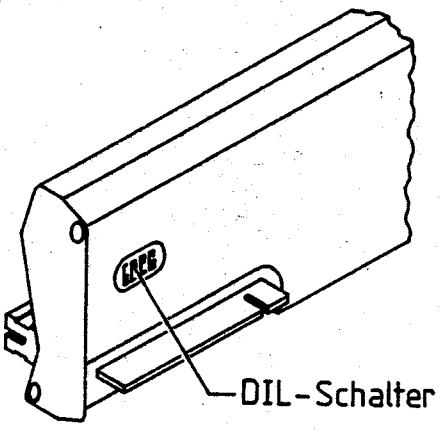
Probeweises Verstellen der Schalter bei eingeschaltetem Gerät führt teils zu einem rhythmisch flimmernden Durcheinander auf dem Bildschirm (*Bild 2 und 3*), teils erscheint der K-Cursor. In *Tabelle 1* ist gegenübergestellt, welche Wirkungen die 16 möglichen Schalterstellungen bei einem einzelnen Memopak-RAM hervorbringen. Hierbei ist anzumerken, daß Bild 2 nur für einige Sekunden stabil bleibt und anschließend einem weißen Bildschirm Platz macht.

Wenn der K-Cursor erscheint, ist der ZX 81 rechenbereit, und man kann in Erfahrung bringen, ob wirklich 16 KByte Speicherplatz verfügbar sind.

Mit dem Kommando

```
PRINT PEEK 16388+256*PEEK 16389
```

wird Speicherzelle 32768 als Obergrenze (RAMTOP) des RAM-Speichers ermittelt. Laut dem ZX-81-Handbuch liegt die Untergrenze des RAMs auf Adresse 16509. RAMTOP gibt nun nicht die oberste nutzbare Adresse an, sondern die erste nicht mehr nutzbare. Daher haben wir 32767 – 16509 = 16258 Byte zur Verfügung – knapp 16 KByte! Hier fallen, ebenso wie in der 1-KByte-Version, die Systemvariablen weg, so daß sich der etwas niedrigere Wert ergibt.

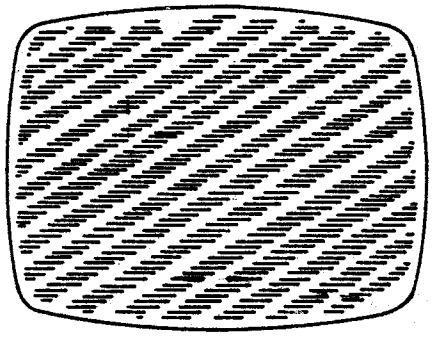


① **16-KByte-RAM von Memotech:** Die Steckerleiste an der Rückseite entspricht der des ZX 81

Zwei Memopaks: Nur 18 Kombinationen führen zum Ziel

Mit zwei 16-KByte-Memopaks gibt es schon 256 Schalterstellungen auszuprobieren. Die Hindernisse sind hier aber noch nicht zu Ende: Denn selbst wenn man Erfolg hat und der K-Cursor erscheint, ist RAMTOP immer 32768.

Wie begegnet man nun diesen Widrigkeiten? Zunächst sollte man wissen, daß 151 von den 256 möglichen Schalterkombinationen die verschiedenartigsten Flimmereien auf dem Bildschirm hervorbringen; der ZX 81 ist dabei nicht rechenbereit. Verbleiben also 105 Stellungen, bei denen der K-Cursor erscheint.



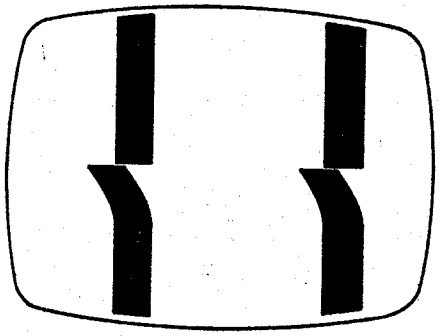
② **Bildsalat:** Zeigt der Bildschirm dieses Muster, ist man auf der falschen Spur

Leider aber heißt das noch lange nicht, daß dann auch 32 KByte Arbeitsspeicher verfügbar sind. Denn bei 23 dieser Stellungen hat man trotz zwei Memopaks nur 16 KByte – und mithin nichts gewonnen. In *Tabelle 2* sind diese Kombinationen durch ein „K" gekennzeichnet. Hierbei gilt, daß unter Memopak 1 die Speichererweiterung zu verstehen ist, die unmittelbar am ZX 81 angeschlossen ist, während Memopak 2 auf Memopak 1 gesteckt wird. Das ist wichtig, weil die Tabelle nicht symmetrisch ist, und damit z. B. die Schalterstellung (6,8) eine andere Wirkung hat als (8,6).

Immerhin gibt es noch andere Varianten: 64 davon bewirken, daß endlich zweimal 16 KByte RAM-Speicher vorhanden sind, jedoch mit einer Lücke von ebenfalls 16 KByte. Die beiden Bereiche liegen zwischen 16 KByte und 32 KByte sowie zwischen 48 KByte und 64 KByte. Zum Rechnen ist freilich wieder nichts, denn das Betriebssystem würde diese Lücke übersehen; dort hineingeschriebene Daten sind selbstverständlich verloren. Tabelle 2 verzeichnet diese Schalterstellungen mit einem „L" (Lücke).

Der nicht gerade große Rest von 18 Schalterkombinationen hat zur Folge, daß tatsächlich 32 KByte Speicher akzeptiert werden – also das gewünschte Ergebnis. In Tabelle 2 ist es mit „S" (Speicher) hervorgehoben.

Der Speicher ist jetzt vorhanden, doch wie nutzt man ihn? RAMTOP nimmt unverdrossen den Wert 32768 an, der aber nur 16 KByte zur Verfü-



③ **ZX-81-Protest:** Diese Darstellung am Bildschirm zeigt's mit Symbolkraft: Der Computer gehorcht nicht mehr

gung stellt. Folglich hat der Rechner von den zweiten 16 KByte „keine Ahnung": und die kann er gar nicht haben!

Beim Einschalten des ZX 81 beginnt nämlich der Z-80-Prozessor bei Adresse 0 den Programmspeicher zu lesen. Hier ist beim ZX 81 das Betriebssystem (ROM) angeordnet. Dieses veranlaßt den Prozessor als erstes RAMTOP zu ermitteln. Dazu wird beginnend bei Adresse 32767 bis hinunter zu Adresse 16384 jede Speicherstelle mit einem Byte geladen. Anschließend werden diese Bytes wieder gelesen und überprüft, ob der zuvor geschriebene Wert vorhanden ist. Die erste Adresse, mit abweichendem Inhalt (255 für nicht vorhandene Speicherstelle), wird als RAMTOP interpretiert.

So findet der ZX 81 zwar die oberste Zelle der ersten 16 KByte, aber nicht die Adresse der zweiten. Man muß also RAMTOP selber hochsetzen; der Wert errechnet sich zu 16 × 1024 × 3 = 49152, denn auch hier werden die ersten 16 KByte vom ZX 81 selbst benutzt, so daß nur 32 KByte übrigbleiben. Folglich muß in Speicherzelle 16388 der Wert „0" stehen (das ist der Fall, wenn man zwei Memopaks angeschlossen hat und den ZX 81 einschaltet). Speicherzelle 16389 verändert man wie folgt:

POKE 16389,192
Zur Kontrolle: 192 × 256 + 0 = 49152.

Aber Vorsicht! Allein das Verändern von RAMTOP hat noch nicht die gewünschte Wirkung. Anschließend muß die Anweisung NEW (CLS bleibt wirkungslos!) gegeben werden.

Sollte man nur ein Memopak besitzen, so empfehlen sich die Schalterstellungen A oder E. Ein zweites Memopak kann mit der gleichen Schalterstellung versehen werden, so daß man auf Anhieb 32 KByte Speicher verfügbar hat. Dann braucht man keine Umstellung beim Neukauf des zweiten Memopaks vornehmen und die beiden Speicher dürfen auch vertauscht werden, ohne daß sich etwas am Ergebnis ändert.

Hans-Peter Gramatke

Stichworte zum Inhalt

16-KByte-RAM, Memopak, Speichererweiterung auf 32 KByte, RAMTOP, ZX-81-Betriebssystem.

Tabelle 1: Die 16 möglichen Schalterstellungen (0 bis F) eines Memopak-RAMs und ihre Wirkung. Hier ist der ZX 81 nur dann funktionsbereit, wenn der K-Cursor erscheint

Schalterstellung	Bez.	Wirkung
▪▪▪▪	0	Bild 2
▪▪▪▪	1	Bild 3
▪▪▪▪	2	Bild 3
▪▪▪▪	3	Bild 3
▪▪▪▪	4	K-Cursor
▪▪▪▪	5	K-Cursor
▪▪▪▪	6	K-Cursor
▪▪▪▪	7	Bild 3
▪▪▪▪	8	K-Cursor
▪▪▪▪	9	K-Cursor
▪▪▪▪	A	K-Cursor
▪▪▪▪	B	Bild 3
▪▪▪▪	C	K-Cursor
▪▪▪▪	D	K-Cursor
▪▪▪▪	E	K-Cursor
▪▪▪▪	F	Bild 3

Tabelle 2: Schalterkombinationen für zwei Memopaks. Nur bei den 18 mit „S" hervorgehobenen Kombinationen stehen lückenlos 32 KByte RAM-Speicher bereit

↓ Schalterstellung von Memopak 1 / Schalterstellung von Memopak 2 →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1					K	K	K		L	L	L		L	L	L	
2																
3																
4				K				K				K				K
5	K	K	K		K	K	K		L	L	L		L	L	L	
6	K	K	K	K	K	K	K	K	S	S	S	K	S	S	S	K
7																
8				L				L				L				L
9	L	L	L		L	L	L		L	L	L		L	L	L	
A	L	L	L	L	L	L	L	L	S	S	S	L	S	S	S	L
B																
C				L				L				L				L
D	L	L	L		L	L	L		L	L	L		L	L	L	
E	L	L	L	L	L	L	L	L	S	S	S	L	S	S	S	L
F																

ZX-81-Hardwarevorstellung:

Kein RAM für alle Fälle

Wer sich für seinen ZX 81 ein 64-KByte-RAM zulegen möchte und glaubt, dann ebensoviel Speicherplatz für Basic-Programme zur Verfügung zu haben – der wird sich wundern.

64-KByte-Speichererweiterungen sind für den ZX 81 in verschiedenen Ausführungen erhältlich. Außergewöhnlich an der hier vorgestellten ist, daß sie, laut Anbieter (Jürgen Schumpich, Ottobrunn), „... volle 64 KByte Speicher für Basic zur Verfügung stellt..." und „... eine 8-Bit-Parallelschnittstelle für Datenverkehr..." hat. Zusätzlich sollen „... zwei Remote-Buchsen zur Motorsteuerung von Kassettenrecordern..." vorhanden sein.

Für Basic stehen nur 32 KByte bereit

Geliefert wird dieses RAM-Modul (*Bild*) mit einer Bedienungsanleitung, die nicht einmal eine DIN-A4-Seite ausfüllt, und die, zumindest für Einsteiger, an verschiedenen Stellen ergänzungsbedürftig ist.

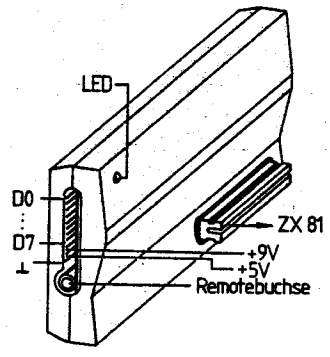
Die Speichererweiterung verfügt über einen eigenen Spannungsstabilisator, so daß der des ZX 81 nicht stärker belastet wird. Zur Inbetriebnahme wird sie an den ausgeschalteten ZX 81 angeschlossen. Nach dem Einschalten stehen dann für Basic 16 KByte zur Verfügung. Um 32 KByte nutzen zu können, ist es nötig, die Variable RAMTOP auf einen 48 KByte entsprechenden Wert heraufzusetzen (16 KByte gehen durch das ROM und das ROM-Doppel verloren):

POKE 16389,192

Wenn nun noch NEW eingegeben wird, stehen 32 KByte Speicher für Basicprogramme bereit. Der Versuch, höhere Werte einzugeben, scheitert am Betriebssystem, das den Wert von RAMTOP nach Eingabe des Befehls NEW wieder auf den 48 KByte entsprechenden Wert heruntersetzt.

Der verbleibende Bereich von 48 KByte bis 64 KByte ist nur durch PEEK und POKE ansprechbar. Dieser 16 KByte umfassende Adreßbereich läßt sich softwaremäßig umgeschalten, so daß zwei voneinander unabhängige Speicherbereiche mit je 16 KByte bereitstehen. Als Umschalter dient die Speicherzelle 16417, deren Datenbit 0 den einen oder den anderen Speicherbereich aktiviert. An der Rückseite der Speichererweiterung ist jedoch neben der durchgeschleiften ZX-81-Steckerleiste noch ein Stecker vorhanden: Wird er gezogen, so sind die oberen 4 KByte des Speicherbereichs „1" nicht mehr ansprechbar. Dieser Adreßbereich ist für weitere Zusatzgeräte vorgesehen.

Welcher Speicherbereich oberhalb von 48 KByte gerade in Betrieb ist, wird durch eine Leuchtdiode an der Vorderseite des Moduls angezeigt. Die Zuordnung zwischen Speicherbereich und Einschaltzustand geht freilich aus der Anleitung nicht hervor; beim Mustergerät leuchtete die LED immer dann, wenn Speicherbereich „1" anzusprechen war. 32 KByte dieser Speichererweiterung sind somit für Basic, die restlichen 32 KByte nur für Maschinenprogramme nutzbar.



64-KByte-RAM: Dieses Modell bietet 32 KByte für Basicprogramme, 32 KByte (2 x 16 KByte) für Maschinenprogramme, eine Schnittstelle zur Ausgabe von 8 Bit und eine Remotebuchse, mit der der Motor eines Kassettenrecorders per Software ein- und auszuschalten ist

Das Umschalten mittels der Speicherzelle 16417 bedeutet Einschränkungen, da diese Speicherzelle auch als 8-Bit-Schnittstelle verwendet wird (Bild). Will man volle 8 Bit zur Ausgabe bringen, so muß man nicht nur auf die Speicherbereich-Umschaltung verzichten, sondern auch auf die Remotebuchse (eine, nicht zwei!), da diese durch Bit 1 derselben Speicherzelle geschaltet wird.

Die Schnittstelle ist pro Leitung mit einer TTL-Last belastbar, wobei es jedoch vorkommen kann, daß bei zu vielen unter Belastung gleichzeitig geschalteten Leitungen der Rechner einfach nicht mehr weiterarbeitet. Hier hilft nur noch Ausschalten – die Daten sind dann natürlich verloren. Daher ist es ratsam nur jeweils eine LS-TTL-Last zu treiben. Am besten verwendet man ein Treiber-IC, z.B. SN74LS244, und man ist dieser Sorge enthoben. Zur Eingabe von Daten ist die Schnittstelle ungeeignet; das Wort „Datenverkehr" trifft also nicht zu.

Die Remotebuchse ist für 3,5-mm-Klinkenstecker ausgelegt. Die Anleitung schweigt sich zwar über die Verkabelung zwischen Kassettenrecorder und RAM-Erweiterung aus, aber wenn man zwei Leitungen zum entsprechenden Eingang des Recorders führt, so funktioniert die Sache praktisch auf Anhieb. Wenn sich der Motor des Recorders nicht schalten läßt, schafft das Vertauschen der Anschlüsse Abhilfe.

Der Motor des Recorders läuft an, sobald Bit 1 der Speicherzelle 16417 den Wert „1" enthält. Das Anhalten freilich ist problematisch; da muß man manchmal von Hand nachhelfen (Stop-Taste). Jedoch liegt das dann nicht am RAM, sondern am Recorder – im Zweifelsfalle hilft da wieder der vorhin erwähnte Treiber mit dem man ein Reed-Relais schaltet, was seinerseits den Recorder in Betrieb setzt.

Fazit: Eine Speichererweiterung, mit der man trotz kleiner technischer Hindernisse gut arbeiten und viel experimentieren kann; zum Preis von 298 DM wird mehr geboten als nur Speicher.

Hans-Peter Gramatke

ZX-81-Hardwaretip:

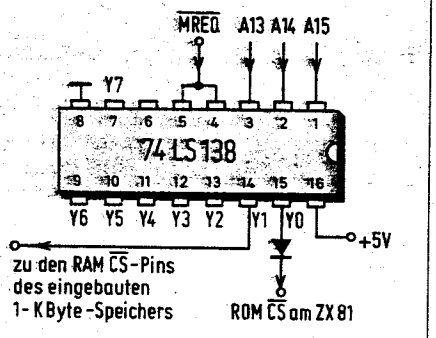
Wiederbelebung

Vollständige Adreßdecodierung des ROMs

Und noch ein Knüller für ZX-81-Fans: Durch „Wiederbelebung" des internen 1-KByte-RAMs wird ein Speicher gewonnen, der nach einem „Absturz" des Computers nicht zwangsweise gelöscht wird. Eine zweite Lösung schafft sogar bis zu 8 KByte sicheren Speicherplatz.

Darüber haben sich insbesondere Freunde der Maschinensprache beim ZX 81 schon oft geärgert: Steigt der Computer aus irgendeinem Grund mitten im Programm aus, dann bleibt einem keine andere Wahl, als resignierend den Netzstecker zu ziehen und mit der Programmiererei wieder von vorne zu beginnen.

Hoffnung auf einen gnädiger gestimmten ZX 81 keimt, wenn man entdeckt, daß die Z-80-CPU einen Reset-Eingang hat (Pin 26). Legt eine Taste diesen Eingang kurz auf L-Pegel, sollte der Computer eigentlich wieder Tritt fassen und ein Wiedersehen mit dem Programm möglich sein. Leider trügt die Hoffnung, denn nach einem L-Impuls an Pin 26 der CPU feiert das Betriebssystem des Computers erst einmal fröhliche Urstände – und führt den mit Adresse 0 beginnenden Programmteil „Urstart" aus.



① **Hürdensprung nach oben:** Diese Schaltung nutzt den stillgelegten 1-KByte-Grundspeicher. Steigt der ZX 81 aus, bleibt der Speicherinhalt oberhalb 32 KByte erhalten

Dabei wird zuerst mit einem out-Befehl der NMI-Generator ausgeschaltet (bewirkt u. a. Einschalten des FAST-Modus) und dann werden, ausgehend von Adresse 32 768 bis herunter zur Adresse 16 383, alle dazwischenliegenden Speicherzellen mit dem Wert 02 geladen. Anschließend prüft die CPU in umgekehrter Richtung, ob die Speicherzellen – nach zweimaligem Decrementieren des Inhalts (je 1 abziehen) – den Wert 0 haben. Die erste Speicherzelle die dann einen von 0 abweichenden Wert hat wird als Ende des RAM-Speichers erkannt und die zugehörige Adresse in der Systemvariablen RAMTOP abgelegt.

Das gleiche macht der ZX 81 jedesmal nach dem Einschalten, er weiß danach stets, wieviel Bytes RAM-Speicher ihm zur Verfügung stehen. Nur bringt uns das keinen Schritt weiter, weil durch den „Speichertest" ein im RAM stehendes Programm zwangsläufig überschrieben wird und dabei verlorengeht.

Das ROM ist der Übeltäter

Das ROM des ZX 81 umfaßt 8 KByte und liegt im Adreßbereich zwischen 0 und 8191. Um jede einzelne Speicherzelle des ROMs ansprechen zu können, sind insgesamt 13 Adreßleitungen nötig ($2^{13} = 8192$).

Woher aber „weiß" das ROM, daß es im Adreßbereich zwischen 0 und 8191 liegt? Das ROM selbst kann das nicht erkennen, jedoch der Sinclair-Logik-Chip (SLC). Er enthält u. a. einen „Adreßdecoder" für das ROM. Der wiederum bewirkt, daß bei Adressen zwischen 0 und 8191 am Ausgang ROM $\overline{CS}$ des SLC L-Pegel herrscht. Und genau mit diesem Pegel wird auch das ROM aktiviert (Pin 20, ROM $\overline{CS}$-Eingang), d. h. es legt die zur jeweiligen Adresse gehörenden Daten auf den Datenbus. H-Pegel an Pin 20 sperrt das ROM. Für den ZX 81 ist es dann scheinbar nicht mehr vorhanden. Das ist z. B. der Fall, wenn ein RAM-Baustein adressiert wird.

Der ROM $\overline{CS}$-Ausgang am SLC und der entsprechende Eingang am ROM sind nicht direkt, sondern über einen Widerstand miteinander verbunden. Deshalb können wir – wie später gezeigt wird – das ROM mit externen Signalen an Pin 20 (bzw. an der Schnittstelle) sperren oder aktivieren, ohne daß der SLC Schaden nimmt. Zunächst ist es jedoch wichtiger zu wissen, daß der Adreßdecoder im SLC nur die für das ROM „interessanten" 13 Adreßleitungen A0 bis A12 abfragt.

Die CPU soll aber nicht nur das ROM, sondern noch jede Menge RAM adressieren können. Sie hat deshalb 16 Adreßleitungen, ausreichend für einen Adreßbereich von 65 536 Byte ($2^{16} = 65\,536$). Solange die CPU jetzt nur auf den 13 Adreßleitungen A0 bis A12 Adressen ausgibt ist alles klar: Dann wird das ROM angesprochen.

Was aber geschieht, wenn die CPU eine höhere Adresse auf den Adreßbus legt z. B. 8200? Unter dieser Adresse müßte sich schon eine RAM-Speicherzelle ansprechen lassen. Nein, denn jetzt treibt das ROM sein Unwesen.

Da der Adreßdecoder im SLC nur die Adreßleitungen A0 bis A12 decodiert, erkennt er nicht, daß bei Adresse 8200 auch auf der Leitung A13 ein Signal liegt (H-Pegel). Allein dieses Signal ermöglicht aber erst eine Unterscheidung zwischen den Adressen 0 bis 8191 und 8192 bis 16 383. Ein Beispiel: Bei Adresse 0 führen die Adreßleitungen A0 bis A13 L-Pegel. Bei Adresse 8192 haben die Leitungen A0 bis A12 ebenfalls wieder L-Pegel, aber A13 führt H-Pegel. Unter der Adresse 8200 wird also ungewollt zusätzlich die Adresse 8 im ROM angesprochen (8200 – 8192 = 8)! Schuld daran ist die „unvollständige Adreßdecodierung" durch den SLC.

Gäbe es unter der Adresse 8200 noch eine RAM-Speicherzelle käme es zu einer verbotenen Doppeladressierung. Insgesamt taucht das störende ROM-Doppel sogar dreimal im Adreßbereich auf: ab 8192, ab 32 768 und ab 40 960. Damit

haben auch die beiden Grenzwerte beim Speichertest ihren Sinn. Sie stecken einen Adreßbereich ab, in dem das ROM-Doppel nicht stören kann.

Jetzt können wir den ZX 81 mit seinen eigenen Waffen schlagen. Der Speichertest zerstört nämlich nur Programme im Adreßbereich zwischen 16 KByte und 32 KByte. RAM-Speicher, der darüber oder darunterliegt, bleibt unbehelligt. Es muß nur gelingen, aus diesen angrenzenden Speicherbereichen das ROM-Doppel zu verbannen. Das ROM bietet uns dazu den Eingang ROM $\overline{CS}$ an. Wir wissen: H-Pegel an diesem Eingang verhindert, daß sich das ROM bemerkbar machen kann.

Wiederbelebung des 1-KByte-RAMs

Wird der ZX 81 mit einer Speichererweiterung betrieben, dann ist das in ihm eingebaute 1-KByte-RAM (zwei ICs 2114 oder ein 4118) automatisch abgeschaltet. Bei einer 16-KByte-Speichererweiterung können wir dieses RAM gemäß *Bild 1* in einem verschobenen Adreßbereich jedoch wieder in Betrieb nehmen, wobei die Speicher-ICs sogar ihren Platz behalten dürfen!

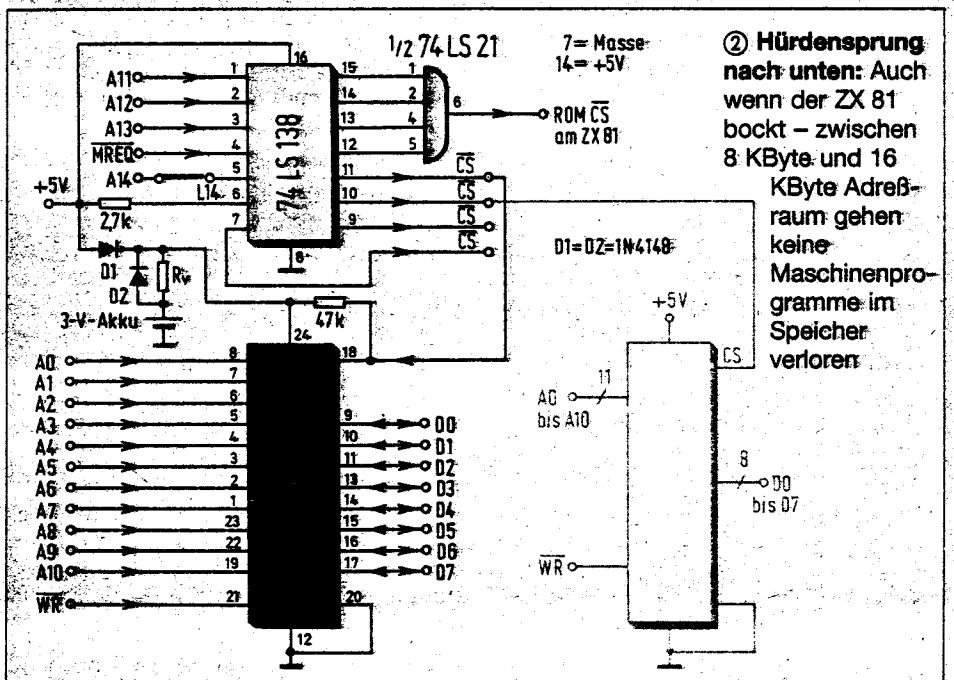
Mit dem Adreßdecoder 74LS138 wird das ROM nachträglich vollständig decodiert bzw. das 1-KByte-RAM freigegeben (Schaltung und Wahrheitstabelle des 74LS138 siehe Heft 12/1983, Seite 73, oder FUNKSCHAU-Sonderheft Nr. 14, Seite 49).

Mit dem Signal $\overline{MREQ}$ wird der Decoder selbst freigegeben, d. h., solange die CPU keinen Speicher anspricht ($\overline{MREQ}$ = H) bleiben alle Ausgänge (Y0 bis Y7) auf H-Pegel. Da sowohl ROM als auch RAM mit L-Pegel an den $\overline{CS}$-Eingängen aktiviert werden, sind somit beide gesperrt. Bei einem Speicherzugriff ($\overline{MREQ}$ = L) mit einer Adresse unter 8192 (A13 bis A15 auf L) führt allein Ausgang Y0 des Adreßdecoders L-Pegel, der, an Pin 20 des ROMs (ROM $\overline{CS}$) gelegt, dieses aktiviert. Bei Adressen über 8191 führt Y0 immer H-Pegel und sperrt damit das ROM. So liegt z. B. im Adreßbereich zwischen 8192 und 16 383 (A13 = H, A14 und A15 = L) allein Y4 auf L-Pegel.

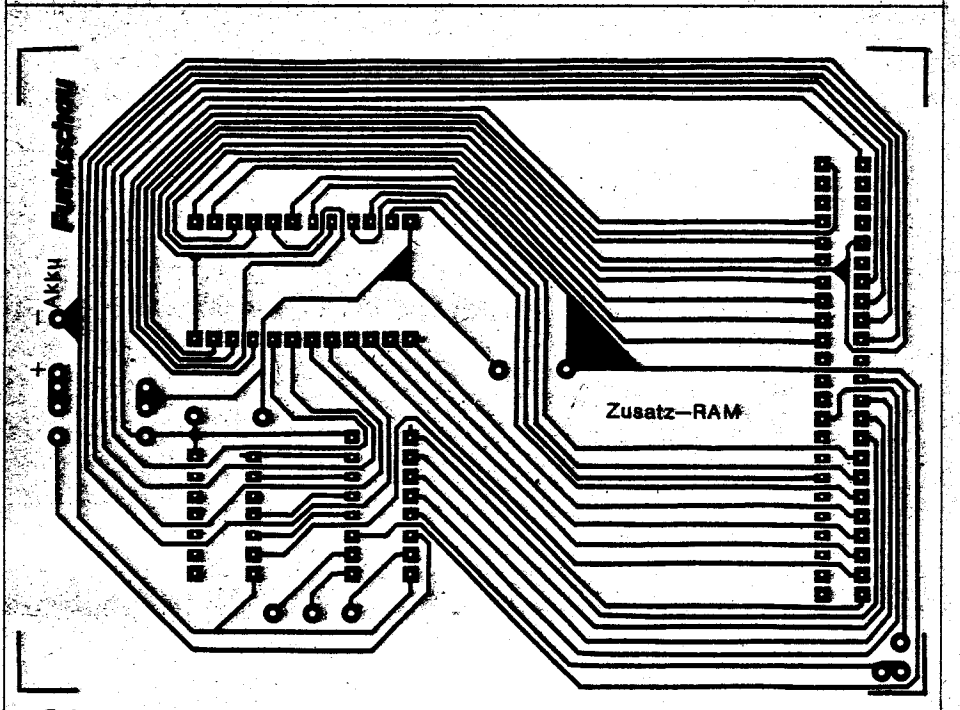
Der uns interessierende Adreßbereich ab 32 768 (bis 40 959) wird angesprochen, wenn A13 und A14 auf L-Pegel liegen und A15 H-Pegel führt. Dann nimmt Ausgang Y1 des Adreßdecoders L-Pegel an.

Soll nun der 1-KByte-Grundspeicher in diesem Adreßbereich wieder in Betrieb genommen werden, dann genügt es, Pin 8 der beiden 2114-ICs hochzubiegen (bzw. Pin 18 und 20 des 4118), und mit Pin 14 (Ausgang Y1) des Adreßdecoders zu verbinden. Das Signal $\overline{MREQ}$ kann man an Pin 19 der CPU abgreifen, die Adreßsignale A13 bis A15 an den Katoden der Tastatur-Dioden D5, D7 und D8. Der Reset-Taster sollte Pin 26 der CPU über einen 100-Ω-Widerstand auf Masse legen.

In den 1-KByte-Zusatzspeicher lassen sich jetzt mit POKE (Adresse zwischen 32 768 und 33 792) Maschinenprogramme unterbringen. Sollte der ZX 81 dann einmal aussteigen, genügt ein Druck auf die Reset-Taste um das gespeicherte Pro-



② **Hürdensprung nach unten:** Auch wenn der ZX 81 bockt – zwischen 8 KByte und 16 KByte Adreßraum gehen keine Maschinenprogramme im Speicher verloren



③ **Layout:** Die Platine ragt hoch genug über ein RAM-Modul hinaus, um bis zu vier 6116-ICs stapeln zu können

gramm wieder zu erreichen. Durch Hochsetzen von RAMTOP (POKE 16389, 132 und anschließend NEW) wird der Speicher auch für das Betriebssystem des ZX 81 erreichbar.

Gepuffert in den Keller gehen

Die Schaltung nach *Bild 2* decodiert den Adreßbereich zwischen 0 und 16 383. Sie kann also auch mit 32-KByte- und (modellabhängig) 64-KByte-Speichererweiterungen genutzt werden. Die geänderte Ansteuerung des Adreßdecoders bewirkt, daß seine Ausgänge jetzt in 2-KByte-Schritten L-Pegel annehmen (bei der anderen Lösung sind es 8-KByte-Schritte). Das erfordert zwar ein Zusammenfassen von vier Ausgängen mit einem UND-Gatter, um ROM $\overline{CS}$ zu gewinnen (4 × 2 KByte = 8 KByte), ermöglicht aber im Adreßbereich von 8192 bis 16 383 das Freigeben von maximal vier RAM-ICs mit je 2 KByte Speicherkapazität (Typ: 6116).

Werden diese ICs batteriegepuffert, hat man sogar einen nichtflüchtigen Speicher. Der ist freilich vom Betriebssystem des ZX 81 und damit für Basic-Programme nicht erreichbar; er kann nur Maschinenprogramme speichern. Anstelle der RAMs lassen sich auch EPROMs oder sogar ein PIO-Baustein adressieren – auch gemischt geht's. Die Batteriepufferung ist selbstverständlich nur für RAMs notwendig.

Bei beiden Schaltungen kann es zu Störungen kommen, wenn die Busse der Z-80-CPU von weiteren Zusatzschaltungen belastet werden. Dann helfen Treiberbausteine weiter: ein 74LS245 für den Datenbus, zwei 74LS241 für den Adreßbus und drei UND-Gatter eines 74LS08 für die Signale $\overline{RD}$, $\overline{WR}$ sowie $\overline{MREQ}$.

Wird die Schaltung nach Bild 2 hinter einer Speichererweiterung aufgesteckt, arbeitet sie nur dann einwandfrei, wenn bei der RAM-Erweiterung alle benutzten Leitungen durchgeschleift sind. Auf keinen Fall darf die Schaltung gemeinsam mit Peripheriegeräten betrieben werden, die selbst den Adreßbereich zwischen 8 KByte und 16 KByte vollständig nutzen (Doppeladressierung).

Das zeigt die Praxis

Die Schaltung nach Bild 1 eignet sich gut zum Einbau in den ZX 81. Sie erzwingt freilich immer einen Betrieb gemeinsam mit dem 16-KByte-RAM-Modul, da der ZX 81 ohne dieses Modul keinen Speicher mehr zum Ablegen der Systemvariablen vorfindet. Das Ergebnis wäre ein Balkengewirr am Bildschirm. Für einen Test kann man z. B. POKE 33000, 99 eingeben. PRINT PEEK 33000 muß dann den Wert 99 auf den Bildschirm bringen – auch nach NEW oder einem Drücken der Reset-Taste!

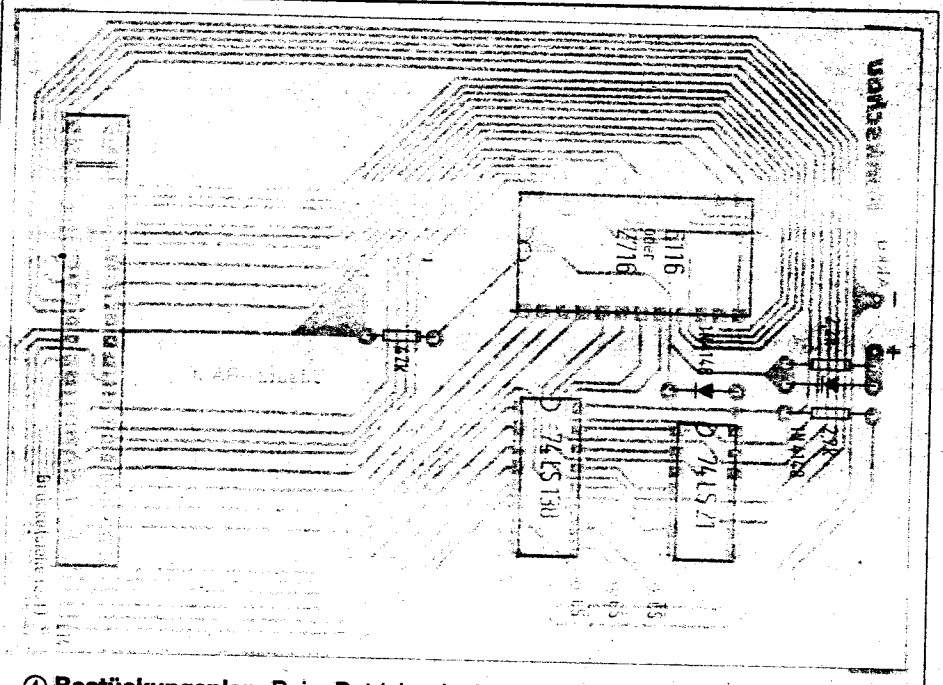
Leider sind Maschinenprogramme oberhalb der Adresse 32 767 nicht lauffähig. Wurde ein Programm dorthin gerettet, dann muß man es nach einem Systemabsturz in einen Speicherbereich unterhalb dieser Adresse kopieren. Erst dann läßt sich das Programm erneut starten.

Beide Probleme kennt die Schaltung nach Bild 2 nicht! Sie hat dafür ein anderes. Da die Schaltung nicht zum Einbau vorgesehen ist, wird man sie zwangsläufig an der Schnittstelle des ZX 81 anschließen. Geschieht das unmittelbar, ist alles in Ordnung. Steckt jedoch ein RAM-Modul (Memotech) dazwischen, fehlt plötzlich das Signal A14. Dieses Signal muß man sich dann mit einer eigenen Leitung aus dem Computer holen. Die Brücke auf der Platine darf in diesem Fall nicht eingelötet werden, denn das Signal A14 darf nicht auf die Steckerseite des RAM-Moduls gelangen (Systemabsturz), sondern allein über die ZX-81-Schnittstelle auf die Buchsenseite des Moduls.

Während die Schaltung nach Bild 1 Zusatzspeicher im Adreßbereich 32 768 bis 33 792 schafft, liegt das RAM der Schaltung nach Bild 2 im Adreßbereich 8192 bis 10 240. Ein Test läßt sich mit einer dieser Adressen wie zuvor beschrieben durchführen.

Wer mit 2 KByte RAM nicht auskommt, kann bis zu vier 6116 ICs parallel schalten, d. h., Pin für Pin übereinander löten (stapeln). Lediglich die $\overline{CS}$-Anschlüsse sind dann getrennt mit den Ausgängen des Adreßdecoders auf der Platine zu verbinden. Werden für die RAM-ICs CMOS-Ausführungen gewählt, ist sogar eine Batterie- bzw. Akku-Pufferung möglich. Die Platine ist dafür vorbereitet, Anschlußhinweise geben der Schalt- und Bestückungsplan. R_v ist so zu bemessen, daß der Ladestrom etwa 1/50 der Akkukapazität beträgt (Ladungserhaltung). Im Akkubetrieb legt der 47-kΩ-Widerstand $\overline{CS}$ des 6116 auf H-Pegel um das RAM abzuschalten (Standby). Anderenfalls wäre der Akku bald leer.

M. Longobardi, H.-G. Schmitz, S. Schall



④ **Bestückungsplan:** Beim Betrieb mit einem RAM-Modul muß das Signal A14 über eine Extra-Leitung aus dem Computer auf das Lötauge L14 gegeben werden; die Drahtbrücke muß dann fehlen (siehe Text)

ZX 81 à la carte (1):

Messer und Gabel

Teil 1: 6-KByte-RAM-Erweiterung

Zu Beginn einer Bauanleitungsserie für den ZX 81 stellen wir das Grundbesteck vor: RAM-Erweiterung und I/O-Port. Beide sind gemeinsam auf einer Basisplatine untergebracht.

Mehr über die Bauanleitungsserie „ZX 81 à la carte" verrät der Kasten auf Seite 73. Hier im ersten Teil geht es um die Beschreibung der ZX-81-Schnittstelle und um die RAM-Erweiterung. Im Teil 2 folgt das Ein-/Ausgabe-Interface und die Platine für beide Baugruppen. Mit dieser Basisplatine läßt sich bereits eine Menge anfangen.

Das Tor nach draußen: die ZX-81-Schnittstelle

Der ZX 81 hat an seiner Rückseite eine Kontaktleiste, die normalerweise für eine Speichererweiterung und den Drucker vorgesehen ist. Wie ein Blick auf die Anschlußbelegung der Kontaktleiste zeigt (*Bild 1*), sind hier tatsächlich alle für Hardware-Erweiterungen wichtigen Signale herausgeführt.

○ Betriebsspannung 9V/5V/0V (Masse): Als Ausgang verwendet liefert der 9-V-Anschluß die unstabilisierte Gleichspannung des Rechnernetzteils. Falls die Belastung nicht zu groß wird, läßt sich diese Spannung auch für externe Hardware verwenden. Andererseits kann über diese Leitung auch eine Stromversorgung des Rechners erfolgen, falls für umfangreiche Erweiterungen ohnehin ein anderes Netzteil erforderlich wird.

Die stabilisierte 5-V-Versorgung ist für externe Schaltungen nur bedingt geeignet, da bei zu hoher Belastung der 5-V-Regler im Rechner sehr heiß wird, und seine Temperaturschutz-Schaltung ansprechen kann.

○ Adreßleitungen A0...A15: Es steht der komplette Adreßbus der Z-80-CPU (CPU: Central Processing Unit – Zentraleinheit) zur Verfügung. Beim Verwenden von Adressen für externe Schaltungen muß geprüft werden, ob nicht gleichzeitig auch die internen Speicher angesprochen werden (wegen unvollständiger Adreßdecodierung). Die Leitungen sind H-aktiv (H: high), das bedeutet, daß eine logische 1 durch einen hohen Spannungspegel realisiert wird.

○ Datenleitungen D0...D7: Hier kann auf den 8-Bit-Datenbus zugegriffen werden. Die Leitungen sind ebenfalls H-aktiv.

○ ROM $\overline{CS}$/RAM $\overline{CS}$ (chip-select – Chip-Anwahl). Während eines Zugriffs der Zentraleinheit auf die internen Speicher gehen die Signale ROM $\overline{CS}$ bzw. RAM $\overline{CS}$ auf logisch 0 (low: L-aktiv). Da diese Leitungen intern durch Widerstände von den Ausgängen des Sinclair-Logik-Chips entkoppelt sind, kann durch ein von außen zugeführtes H-Signal der jeweilige Speicher abgeschaltet werden, ohne daß dieses IC Schaden nimmt.

○ Systemtaktfrequenz Φ: Die Taktfrequenz des Z 80 A beträgt 3,23 MHz. Dieses durch ein Keramikfilter stabilisierte Signal kann für Zähler- und Zeitmeßanwendungen extern verwendet werden.

○ $\overline{INT}$ (interrupt request – Unterbrechungsaufforderung): Wenn $\overline{INT}$ auf maskierbaren, d. h. durch einen Software-Befehl unterdrückbaren Interrupt aus. Da dieses Signal vom Betriebssystem verwendet wird, kann es von Erweiterungsschaltungen nicht sinnvoll benutzt werden.

○ $\overline{NMI}$ (nonmaskable interrupt): Nichtmaskierbarer Interrupt (L-aktiv), ist ebenso wie $\overline{INT}$ nicht verwendbar.

○ HALT: Dieser L-aktive Ausgang zeigt an, daß die CPU einen Software-HALT-Befehl ausführt. Eine externe Verwendung ist beim ZX 81 ebenfalls nicht unproblematisch.

○ $\overline{MREQ}$ (memory request – Speicheraufforderung): Während eines Speicherzugriffs durch die CPU geht dieser Ausgang auf logisch 0.

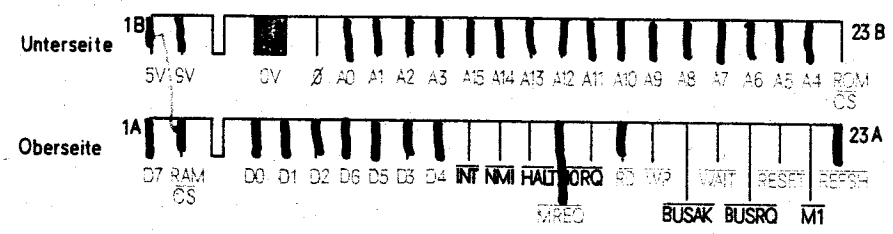
○ $\overline{IORQ}$ (input/output-request – Ein-/Ausgabe-Aufforderung): Die Z-80-CPU hat spezielle Ein-/Ausgabe-Befehle, mit denen 256 verschiedene Ein-/Ausgabe-Adressen angesprochen werden. Dabei geht $\overline{IORQ}$ auf logisch 0. Da diese Befehle auch für die Tastaturabfrage und Bildschirmausgabe des ZX 81 benutzt werden, ist eine externe Verwendung von $\overline{IORQ}$ nicht zweckmäßig.

○ $\overline{RD}$/$\overline{WR}$ (read/write – lesen/schreiben): Diese Ausgänge (L-aktiv) zeigen, ob die CPU einen Lese- bzw. Schreibzugriff auf einen Speicher (bzw. I/O-Baustein) ausführt. Daher ist gleichzeitig auch $\overline{MREQ}$ (bzw. $\overline{IORQ}$) aktiv.

○ BUSRQ/$\overline{BUSAK}$ (bus request/bus acknowledgement – Busanforderung/Quittung): Durch ein L-Signal auf dem BUSRQ-Eingang werden Adreß-, Daten- und Steuerbus von der CPU freigegeben (Bestätigung durch L-Signal auf dem $\overline{BUSAK}$-Ausgang) und können extern verwendet werden. Diese Leitungen sind beim ZX 81 nicht sinnvoll nutzbar.

○ $\overline{WAIT}$ (warten): Diese Eingangsleitung (L-aktiv) kann benutzt werden, um langsame Speicher- oder I/O-Bausteine einsetzen zu können. Solange $\overline{WAIT}$ aktiv ist, wartet die CPU.

○ $\overline{RFSH}$ (refresh – auffrischen): Dieses Signal (L-aktiv) ist für den Auffrisch-Zyklus bei dynamischen Speichern wichtig.



① **ZX-81-Schnittstelle:** Nur die farblich hervorgehobenen Signale lassen sich für Hardwareerweiterungen sinnvoll nutzen

○ $\overline{M1}$: Während eines Befehl-Lesezyklusses wird dieses Signal logisch 0. Für Erweiterungen ist es nicht sinnvoll anwendbar.
○ $\overline{RESET}$ (rücksetzen): Durch ein L-Signal wird die CPU auf Adresse 0 rückgesetzt. Dadurch wird leider auch ein im ZX 81 vorhandenes Programm gelöscht. Aus- und erneutes Einschalten der Spannungsversorgung hat dieselbe Wirkung.

Werden die Ausgangssignale des ZX 81 verwendet, so muß beachtet werden, daß die Belastung nicht zu groß wird. Aus diesem Grund ist das Zwischenschalten von Verstärkern (Pufferung) sinnvoll.

Ein Adreßdecoder plaziert die Speicherblöcke

In der Grundversion hat der ZX 81 nur eine geringe Speicherkapazität von 1 KByte RAM. Mit zusätzlichen 6 KByte RAM ist man für viele Anwendungsfälle gerüstet. Durch Verwenden von Speicher-ICs des Typs HM 6116 (2K × 8 Bit) kann der Speicher auch in Einzelschritten von jeweils 2 KByte ausgebaut werden. Damit der Computer die Speichererweiterung auch nutzen kann, muß sich der externe RAM-Bereich nahtlos an den internen Bereich anschließen. Daher ist es nötig, sich die Speicherorganisation des ZX 81 für eine 6-KByte-RAM-Erweiterung klar zu machen (*Bild 2*).

Eine Eigentümlichkeit dieses Computers ist die dezimale Angabe von Adressen und Speicherinhalten am Bildschirm. Im Bild sind die Adressen sowohl dezimal als auch hexadezimal aufgeführt. Die vierte Spalte zeigt die dazugehörigen logischen Pegel auf den Adreßleitungen.

HEX	DEZ	Y	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
5FFF	24575	Y7	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
5C00	23552		0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
5BFF	23551	Y6	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1
5800	22528		0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
57FF	22527	Y5	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1
5400	21504		0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
53FF	21503	Y4	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1
5000	20480		0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4FFF	20479	Y3	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
4C00	19456		0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
4BFF	19455	Y2	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1
4800	18432		0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
47FF	18431	Y1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
4400	17408		0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
43FF	17407	Y0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
4000	16384		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3FFF	16383		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2000	8192		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1FFF	8191		0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0000	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

② **Speicherorganisation:** Die 6-KByte-RAM-Erweiterung nimmt den Adreßbereich 17408 bis 23551 in Anspruch. Y0 bis Y6 sind die Signale eines Adreßdecoders zum Auswählen von Speicherblöcken (siehe Text). Y7 gibt den I/O-Port (Teil 2) frei

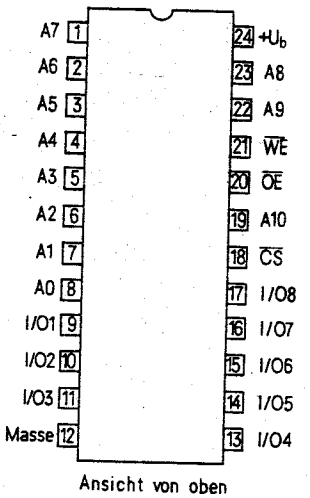
Da die Z-80-CPU bei RESET den ersten (Maschinen-)Programmbefehl auf Adresse 0 erwartet, liegt das ROM für Betriebssystem und Basic-Interpreter (8 KByte) zwischen 0 und 8191. Überraschenderweise erscheint es nochmals zwischen 8192 und 16383. Der Grund dafür ist eine unvollständige Adreßdecodierung, d. h. für die Zuordnung der Speicheradressen im ROM zu den von der CPU ausgegebenen Adreßsignalen werden nur die Leitungen A0...A12 verwendet. Dadurch „sieht" das ROM z. B. 0 und 8192 als dieselbe Adresse. Man kann sich dies an den dezimalen Zahlen 1293 und 7293 klar machen: Betrachtet man nur die letzten drei Ziffern, so erhält man beide Male dieselbe Zahl, nämlich 293.

Von 16384 bis 17407 schließt sich das interne RAM an. Die RAM-Obergrenze wird beim Einschalten vom Computer mit einem Testprogramm ermittelt und als Systemvariable RAMTOP gespeichert (deshalb muß auch eine Speichererweiterung nahtlos ans interne RAM anschließen). Durch die Befehlseingabe

PRINT PEEK 16388 + 256*PEEK 16389

erhält man ihren Zahlenwert in dezimaler Schreibweise (siehe auch FS 11/83 Seite 78). Er stellt die Adresse des ersten nicht existierenden Bytes dar; ohne Speichererweiterung ergibt sich also die Zahl 17408.

In *Bild 3* sind das Anschlußschema und die Funktionstabelle eines Speicherbausteins HM 6116 gezeigt. Dieser Speicher erfordert elf Adreßleitungen (A0...A10). Wenn man ihn direkt mit den entsprechenden Leitungen des Adreßbusses verbindet, so würde er gemeinsam mit dem internen ROM zwischen Adresse 0 und 2047 (2 KByte)



Ansicht von oben

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Betriebsart
1	X	X	nicht angesprochen
0	0	1	Lesen
0	1	0	Schreiben
0	0	0	Schreiben

③ **Speicher-IC:** Der Speicherbaustein HM 6116 ist ein statisches 2 KByte RAM (2 K × 8). X bedeutet: Signalpegel gleichgültig

angesprochen. Erwünscht ist er allerdings zwischen den Adressen 17408 und 19455.

Erreichen läßt sich das, indem man die Adreßleitungen A11...A14 zur Unterscheidung heranzieht. Im Bereich von 16384 bis 24575 ist stets A14 = 1 und A13 = 0. Betrachtet man weiterhin A12, A11 und A10 als dreistellige Binärzahl, so entspricht jeder solchen Zahl ein 1 KByte großer RAM-Bereich. Ein 3-zu-8-Decoder 74LS138 kann in Abhängigkeit von dieser Zahl (C B A in *Bild* 4) jeweils einen seiner Ausgänge Y0...Y7 auf logisch 0 schalten (Spalte 3 in Bild 2).

Führt man seinem Freigabe-Eingang G1 (H-aktiv) das Signal A14, den Eingängen G2A und G2B (L-aktiv) die Signale MREQ und A13 zu, so ist gewährleistet, daß der Decoder nur während eines Speicherzugriffs auf die Adressen 16384 bis 24575 angesprochen wird.

A15 wird nicht benutzt. Das hat zur Folge, daß sich die gesamte Speicheraufteilung zwischen den Adressen 32768 und 65535 wiederholt (unvollständige Adreßdecodierung).

Da der 3-zu-8-Decoder jeweils 1-KByte-Blöcke auswählt, die Speicher-ICs aber zwei KByte umfassen, müssen jeweils zwei Ausgangsleitungen über eine ODER-Verknüpfung zusammengefaßt werden, um den jeweiligen Speicher anzusprechen. Das bedeutet, Y1 oder Y2 aktivieren den CS-Eingang (siehe Bild 3) des ersten Speicher-ICs (logisch 0 an CS aktiviert das IC); Y3 oder Y4 bzw. Y5 oder Y6 bewirken dasselbe für die anderen Speicher.

Wenn Y0 = 0 ist, ist das interne RAM an der Reihe. Das bedeutet umgekehrt, daß während Y0 = 1 externe Bausteine angesprochen werden. Daher ist dieses Signal zum Abschalten des internen RAMs geeignet. Der durch Y7 = 0 erfaßte Bereich wird für die in Teil 2 beschriebene Schaltung benutzt.

Die Schaltung der Speichererweiterung

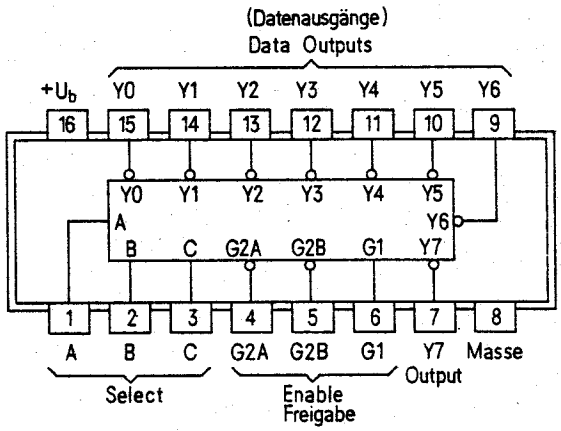
Bild 5 zeigt das Schaltbild der 6-KByte-Speichererweiterung. Die Stromversorgung erfolgt über den 9-V-Ausgang des ZX 81 und einen Spannungsregler 7805. Zum Abblocken von Impulsstörungen auf den Versorgungsleitungen dienen die Kondensatoren C1...C7.

ZX 81 à la carte

Wie kein anderer Heimcomputer reizt der ZX 81 Anwender dazu, mit selbstgebauter Hardware die Fähigkeiten des Maschinchens an der Schnittstelle Computer/Umfeld auszuprobieren. Gewiß – auch Programmieren allein macht Spaß, aber mit Spielen oder nützlichen Programmen ist das Leistungsvermögen eines Computers längst nicht ausgeschöpft: Er kann zum Tongenerator, Morse-Geber, RTTY-Sender/Empfänger, Lauflicht oder zur Dunkelkammeruhr werden, wenn – ja wenn dafür die passende Hardware neben der Software bereitsteht.

Mit diesem Heft startet die FUNKSCHAU eine lose Bauanleitungsserie (samt Anwendersoftware), die den ZX 81 zu munteren Aktivitäten in seinem Umfeld befähigt. Wer mitmacht, ist von den oft stundenlangen stummen Programmierdialogen via Tastatur und Bildschirm erlöst und deshalb wird auch „Freak's Frau" mehr Verständnis für das Hobby zeigen.

Die einzelnen Bauvorschläge sind zum Teil aufeinander angewiesen; so ist z. B. die jetzt und im nächsten Heft beschriebene Basisplatine Voraussetzung zum Anschluß aller folgenden Hardwarezusätze. Damit Sie dann leichter erkennen, daß Sie wieder einen Teil der Serie vor sich haben, lautet die Kopfzeile über der jeweiligen Überschrift immer „ZX 81 à la carte". Die nachgestellte Ziffer gibt an, um welchen Teil es sich handelt.



④ **Adreßdecoder:** Das IC 74LS138 ist ein 3-zu-8-Decoder. Abhängig von den Eingangssignalen C, B, A, nimmt einer der Y-Ausgänge 0-Pegel an. G1 und G2* sind Freigabe-Eingänge, wobei G2* die UND-Verknüpfung der Signale G2A und G2B ist

Eingänge / Inputs					Ausgänge							
Freigabe		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

Alle vom Computer kommenden Signale werden mit Bustreibern 74LS245 (IC1 bis IC3) gepuffert (Ausnahme A11 und A12, die beide nur eine LS-TTL-Last zu treiben haben).

A0...A10 gehen nach der Pufferung direkt an die Anschlüsse der Speicher-ICs (IC5 bis IC7). Die Leitungen A14', A13' und MREQ' aktivieren den Adreßdecoder 74LS138 (IC4) wie zuvor erläutert. In Abhängigkeit von dem aus A10', A11 und A12 gebildeten Eingangswort wird eine seiner acht Ausgangsleitungen auf logisch 0 geschaltet (siehe Bild 4).

Die ODER-Verknüpfung erfolgt für jeweils zwei dieser Ausgangsleitungen über die Germanium-Dioden D2...D7. Silizium-Dioden sind wegen ihrer höheren Flußspannung von 0,7 V ungeeignet, da dann eine logische 0 Spannungswerte bis zu 1,1 V erreichen kann (maximale Ausgangsspannung für logisch 0 bei TTL-Gattern 0,4 V zuzüglich0,7 V), ein Speicher-IC am CS-Eingang aber nur maximal 0,8 V als logisch 0 akzeptiert.

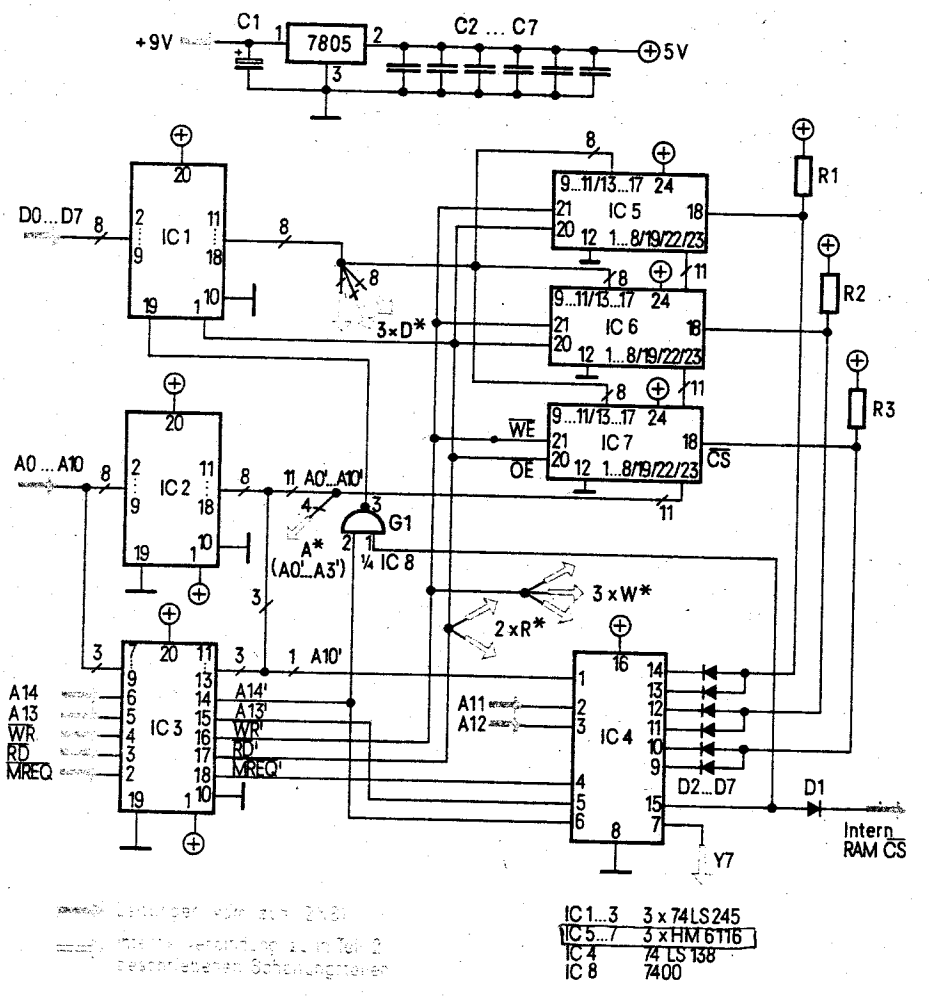
Das $\overline{RD}$'-Signal wird dem Anschluß $\overline{OE}$ (output enable – Freigabe der Ausgänge zum Lesen), das $\overline{WR}$'-Signal dem Anschluß $\overline{WE}$ (write enable – Freigabe zum Einschreiben von Daten) von IC5 bis IC7 zugeführt (siehe Bild 3).

Um zu verhindern, daß das interne RAM gleichzeitig mit dem externen Speicher auf den Datenbus zugreift, wird es während Y0 = 1 über die Diode D1 abgeschaltet. Solange Y0 = 0 oder A14 = 0 ist (siehe Bild 2) wird der Datenbus vom internen RAM bzw. ROM belegt. Dann wird der Datenbustreiber IC1 abgeschaltet (der Tri-State-Ausgang wird hochohmig), um einen Doppelzugriff zu vermeiden. Die hierfür nötige ODER-Verknüpfung erfolgt über das NAND-Gatter G1, das in negativer Logik (Y0 und A14 sind hier L-aktiv) eine ODER-Funktion realisiert.

Da der Datenbuspuffer IC1 in Abhängigkeit von einem Schreib- bzw. Lesezugriff die Daten aus dem Computer heraus oder in den Computer hinein leiten soll, muß seine Duchlaßrichtung über die $\overline{RD}$'-Leitung umgeschaltet werden (Pin 1 von IC 1): $\overline{RD}$' = 0 bedeutet Lese-Operation, $\overline{RD}$' = 1 bedeutet Nichtlese-Operation. Die zu den Punkten D*, A*, W* und R* weisenden Pfeile beziehen sich auf den Schaltungsteil, der in Teil 2 besprochen wird.

Oskar Merker

(Wird fortgesetzt)



⑤ **6-KByte-RAM-Erweiterung:** Die Platine zu dieser Schaltung folgt in Teil 2

ZX-81-Softwaretip:

Abfragen des Speichers

Mit folgender Eingabe läßt sich herausfinden, wieviel Platz ein Programm im Speicher einnimmt:

```
PRINT PEEK 16396+
256* PEEK 16397–16509
```

Erläuterung: Ein Programm beginnt immer ab Adresse 16509. Die Systemvariable D-FILE unter den Adressen 16396 und 16397 gibt dagegen die Adresse an, mit der der Bildschirmbereich anfängt, der sich direkt an den Programmbereich anschließt (siehe auch Handbuch, Kapitel 27). Daher ergibt die Differenz zwischen D-FILE und 16509 den Programmumfang.

Auf die gleiche Weise erhält man den Speicherumfang des Variablenbereichs als Differenz von E-LINE und VARS:

```
PRINT PEEK 16404–PEEK
16400+256* (PEEK 16405–PEEK
16401)–1
```

Diesmal müssen die Inhalte von zwei Systemvariablen berücksichtigt werden, weil sowohl Anfang als auch Ende dieses Speicherbereichs nicht festliegen. Die Eins muß abgezogen werden, weil grundsätzlich ein Byte mit Inhalt 128 im Variablenbereich vorhanden ist.

Den Umfang des noch freien Speicherbereichs, vielleicht die interessanteste Information beim Programmieren, erfährt man näherungsweise nach folgender Eingabe:

```
PRINT PEEK 16386–PEEK
16404+256*(PEEK 16387–PEEK
16405)
```

Hierbei wird die Differenz zwischen ERR-SP und E-LINE ermittelt. E-LINE sagt aus, wo der noch ungenutzte Speicherbereich beginnt. ERR-SP gibt an, wo der GOSUB-Stapel anfängt, wodurch das Ende des freien Speicherbereichs in etwa gegeben ist.

Michael Schramm

ZX 81 à la carte (2)

Messer und Gabel

Teil 2: I/O-Port und Softwareschalter

Die im vorangegangenen Heft begonnene Beschreibung der Basisplatine wird jetzt abgeschlossen. Damit stehen die RAM-Erweiterung und das I/O-Interface zur Anwendung bereit.

Der Nachteil der ZX-81-Schnittstellen für die Tastatur, den Drucker, Bildschirm und den Kassettenrecorder ist, daß kein direkter Kontakt zum Umfeld besteht; stets muß der Mensch als Vermittler zusätzlich in Aktion treten. Wünschenswert wäre aber eine direkte Verbindung, um z. B. ein Gerät unmittelbar durch den Computer ein- oder ausschalten zu können. Umgekehrt kann auf diesem Weg eine unmittelbare Datenerfassung, z. B. der Zimmertemperatur, erfolgen.

Für diesen Zweck stehen eine Reihe spezieller ICs zur Verfügung, wie der Z-80-PIO (Parallel In/Out). Bei diesem IC ist jedoch ein Programmieren der verschiedenen Betriebsarten erforderlich. Dem Anfänger erschwert das die Bedienung des I/O-Ports. Daher wurde im folgenden ein anderes Konzept verwendet.

Mit PEEK und POKE werden Daten ein- und ausgegeben

Speicherstellen können in Basic mit dem Befehl PRINT PEEK (Adresse) gelesen, mit POKE (Adresse, Zahl) geladen werden. Ersetzt man das Speicher-IC durch ein Port-IC, kann auf die gleiche Weise eine Ein- oder Ausgabe realisiert werden. Daher schließt sich der I/O-Bereich an den Adreßbereich der Speichererweiterung an.

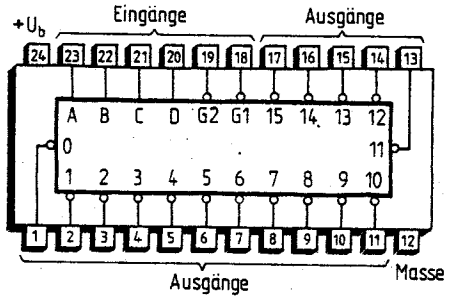
Zum Adressieren des Ports steht vom Adreßdecoder 74LS138 (IC 4 der Speichererweiterung) das Signal Y7 zur Verfügung, das im Adreßbereich 23552...24575 logisch 0 ist (siehe Teil 1). Mit diesem Signal läßt sich ein 4-zu-16-Decoder 74LS154 über die Eingänge G1/G2 aktivieren (*Bild 1*). Werden an seine Eingänge A, B, C und D die Adreßleitungen A0', A1', A2' und A3' angelegt, so wählt er in Abhängigkeit von der Adreßkombination eine seiner 16 Ausgangsleitungen an (L-aktiv, logisch 0). Auf diese Weise können die Adressen 23552....23567 einzeln angesprochen werden (die Basisplatine nutzt vorerst nur die drei Adressen 23556 bis 23558). Wegen unvollständiger Adreßdecodierung wiederholt sich das Ansprechen dieser Adressen 64mal bis zur Adresse 24575, was aber nicht weiter stört.

Der Datenbus: Pipeline zum I/O-Port

Beim Ansprechen der Port-Adresse (23558) müssen zur Dateneingabe die von außen kommenden Daten auf den Datenbus gelegt werden. Hierfür läßt sich sehr gut der Bus-Treiber 74 LS 245 zweckentfremden (*Bild 2*).

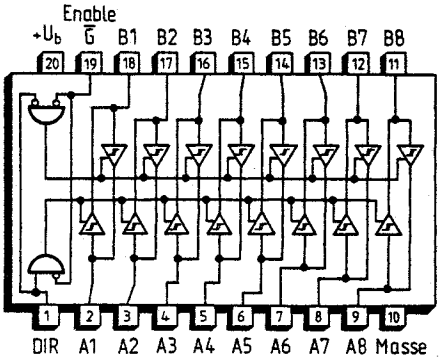
Legt man seinen Anschluß 1 an Masse (logisch 0), so ist eine Datenübertragung von außen (B) zum Datenbus (A)

① 4-zu-16-Decoder 74LS154: Die vier Eingangssignale A bis D bestimmen, welcher der 16 Ausgänge logisch 0 annimmt



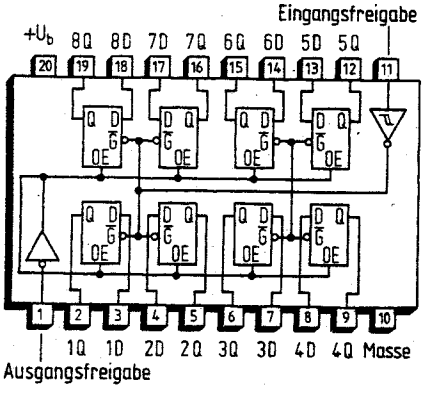
Eingänge					Ausgänge											
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

möglich. Dies geschieht allerdings nur, wenn an Pin 19 (Enable-Freigabe) ein L-Signal (logisch 0) anliegt. Ist das Signal an Pin 19 logisch 1, so sind die IC-Ausgänge hochohmig (Tristate) und belasten den Datenbus nicht.



Pin 19	Pin 1	
Freigabe Enable $\overline{G}$	Richtungs-steuerung DIR	Datenfluß
0	0	von B nach A
0	1	von A nach B
1	X	hochohmig

② **Bustreiber 74LS245:** Er läßt sich gut zur Dateneingabe von der Außenwelt (B) zum Datenbus (A) zweckentfremden, wenn Pin 1 mit Masse verbunden wird



Pin 1	Pin 11		
Ausgangs-freigabe	Eingangs-freigabe $\overline{G}$	D	Ausgang
0	1	1	1
0	1	0	0
0	0	X	Q_0
1	X	X	Z

③ **8-Bit-Zwischenspeicher:** Das IC 74LS373 speichert die nur kurzfristig auf dem Datenbus liegende Information Q_0: alte Daten. Z: Ausgänge hochohmig

Die Eingabe-Daten müssen vorhanden sein, wenn die Port-Adresse angesprochen wird, denn ein Zwischenspeichern erfolgt nicht. Dies ist in den meisten Anwendungsfällen allerdings kein Nachteil.

Da andererseits bei der Datenausgabe die Daten von der CPU nur sehr kurze Zeit auf den Datenbus gebracht werden (einige Mikrosekunden lang), ist in diesem Fall ein Zwischenspeichern erforderlich.

Diese Aufgabe kann ein 8fach-Latch 74LS373 (Latch – Zwischenspeicher) übernehmen (*Bild 3*). Seine Ausgänge werden durch ein L-Signal an Pin 1 aktiviert, ein H-Signal (logisch 1) schaltet die Ausgänge in den hochohmigen Zustand. H-Pegel an Pin 11 (Enable) veranlaßt das IC, die an den Eingängen anliegenden Daten zu übernehmen und zu speichern. Dieses Signal muß also vom Computer geliefert werden, sobald die Ausgabe-Daten auf dem Datenbus vorhanden sind.

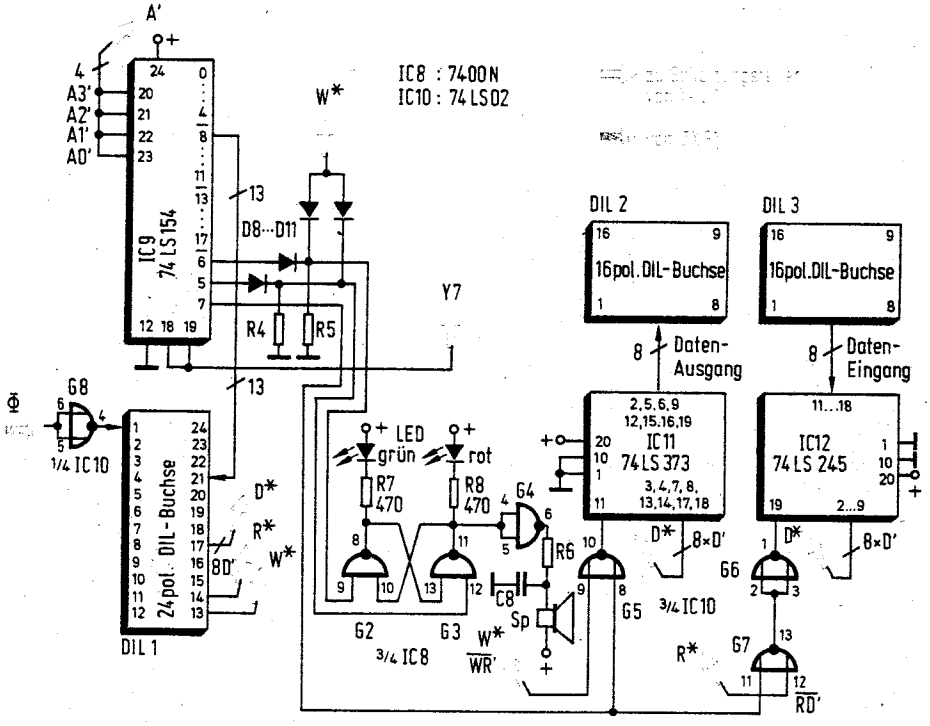
Schaltung des I/O-Ports

In *Bild 4* ist die Port-Schaltung gezeigt. Sie ergänzt die Schaltung der Speichererweiterung aus dem vorange-

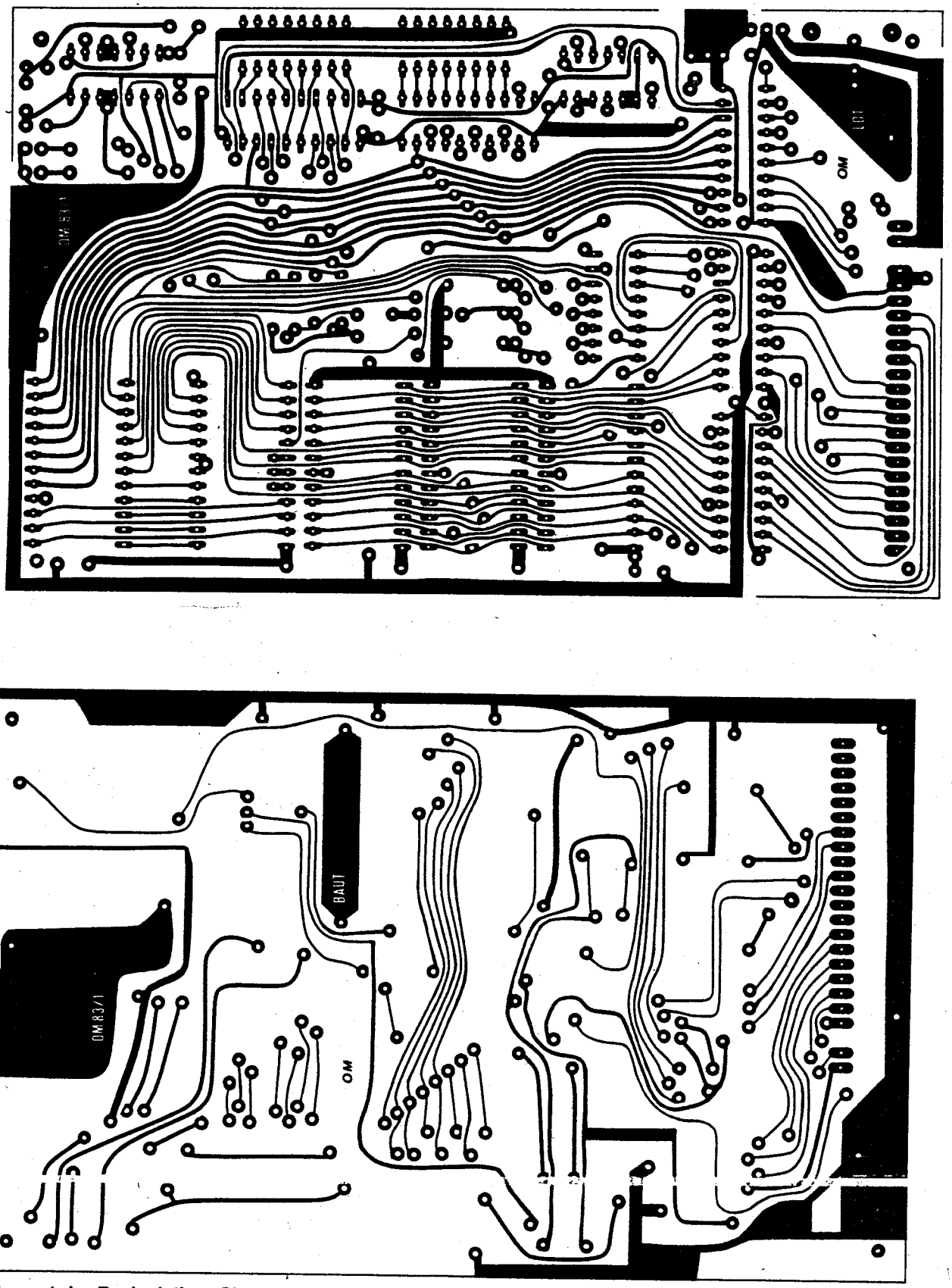
23556 Einschalten LED grün
23557 Einschalten LED rot
23558 I/O-Port (jeweils unidirektional)

DIL 1		DIL 2	(Ausgabe)
1	Takt (Ø)	1	D 0
2	$\overline{RD}$'	2	D 1
3	$\overline{WR}$'	3	D 2
4	23552	4	D 3
5	23553	5	D 4
6	23554	6	D 5
7	23555	7	D 6
8	23567	8	D 7
9	23566		
10	23565	9...16 Masse	
11	23564		
12	23563		
13	23562		
14	23561	DIL 3	(Eingabe)
15	23560	1	D 0
16	23559	2	D 1
17	D 0'	3	D 2
18	D 1'	4	D 3
19	D 2'	5	D 4
20	D 3'	6	D 5
21	D 4'	7	D 6
22	D 5'	8	D 7
23	D 6'		
24	D 7'	9...16 Masse	

⑤ **Portadressen und Stiftbelegung der DIL-Buchsen:** DIL 1 dient dem Anschluß weiterer Hardware, DIL 2 der Dateneingabe und DIL 3 der Datenausgabe



④ **I/O-Port und Softwareschalter:** Mit der 6-KByte-RAM-Erweiterung aus Teil 1 (die Pfeile führen zu dort angegebenen Schaltungspunkten) ist das Schaltbild der Basisplatine jetzt komplett



⑥ **Layout der Basisplatine:** Oben die Lötseite und unten die Bestückungsseite der doppelseitig kaschierten Leiterplatte im Europakarten-Format

gangenen Heft, daher sind die Bauteile fortlaufend durchnumeriert.

Im Adreßbereich 23552...24575 wird IC 9 (4-zu-16-Decoder) aktiviert, die Adressen A0', A3' wählen dann wie erläutert eine der 16 Ausgangsleitungen an. Wenn dabei die Port-Adresse 23558 angesprochen wird, führt Pin 7 L-Signal. Falls gleichzeitig $\overline{WR}$' logisch 0 ist (wenn z. B. ein POKE-Befehl gegeben wurde), wird über das NOR-Gatter G 5 (da negative Logik: UND-Verknüpfung) das 8fach-Latch IC 11 angesprochen (H-Pegel an Pin 11) und es speichert die auf dem Datenbus liegenden Informationen. Sie stehen dann an der Buchse DIL 2 extern zur Verfügung.

Falls $\overline{RD}$' und Pin 7 von IC 9 ein L-Signal führen (z. B. bei einem PEEK-Aufruf) wird auf gleiche Weise über G 7 das IC 12 adressiert. G 6 dient als Inverter, da IC 12 an Pin 19 in diesem Fall L-Pegel benötigt. Dadurch werden die an DIL 3 von außen kommenden Daten in den Computer übernommen.

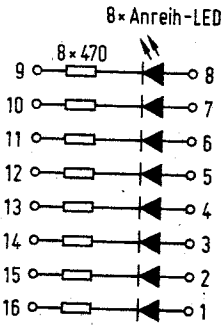
Eine kleine Zugabe: Software-Schalter

Mit den beiden NAND-Gattern G 2 und G 3 ist ein Set-Reset-Flipflop aufgebaut. Der Schaltzustand des Flipflops ist durch zwei Leuchtdioden an den Ausgängen erkennbar.

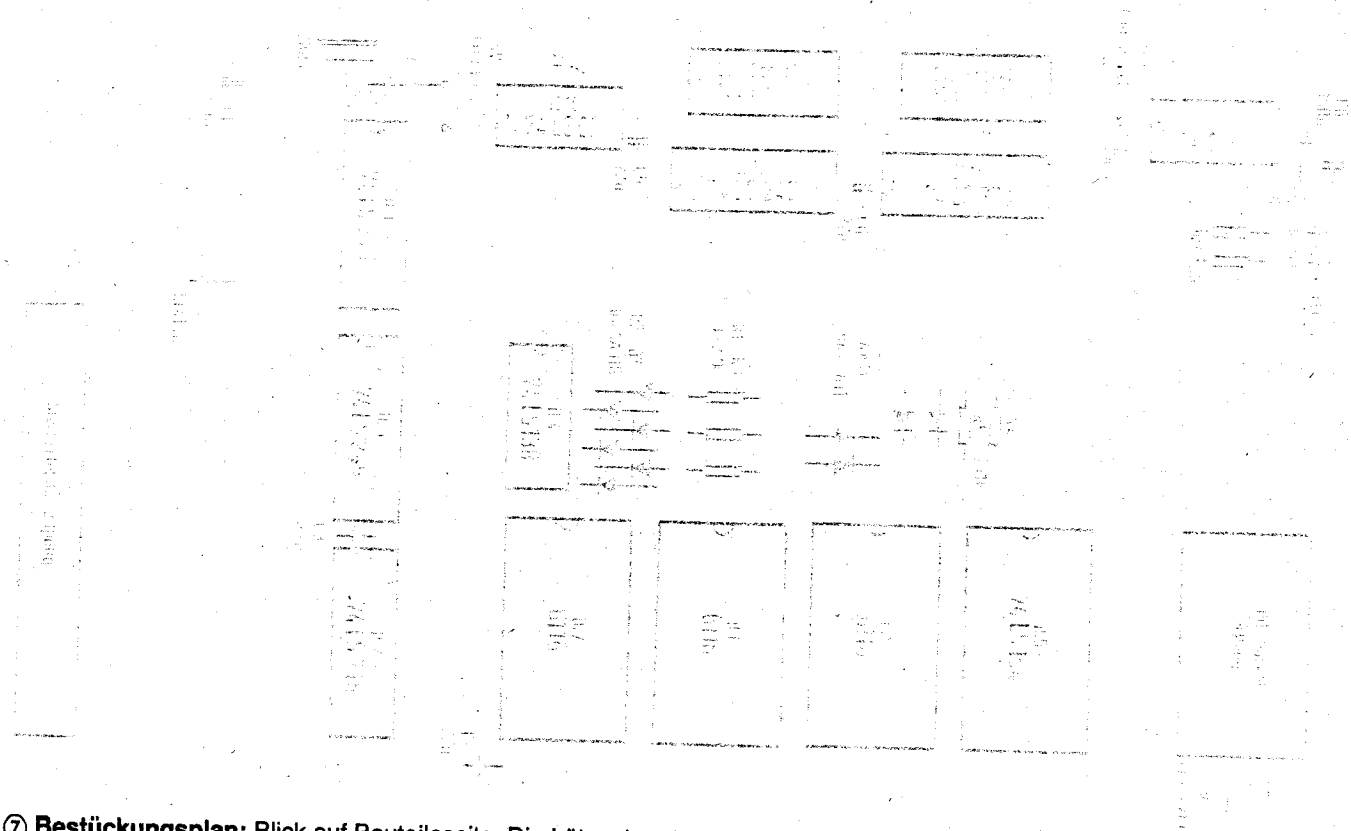
Durch einen POKE-Befehl auf Adresse 23556 wird das Flipflop gesetzt, durch einen gleichartigen Befehl auf Adresse 23557 rückgesetzt. Realisiert wird dies, indem die entsprechenden Ausgangsleitungen von IC 9 über die Dioden D 8...D 11 mit dem $\overline{WR}$'-Signal UND verknüpft werden (negative Logik, da L-aktiv). Dieser Schalter kann also, wie auch der Name sagt, von einem Programm aus bedient werden.

Zusätzlich treibt der eine der beiden Schalterausgänge über G 4 einen Lautsprecher. Jedesmal, wenn umgeschaltet wird, ist im Lautsprecher ein Klick-Geräusch zu hören. Erfolgt die Umschaltung schnell genug, entsteht ein Ton. Seine Frequenz ist gleich der Umschaltfrequenz; auf diese Weise ist also eine programmgesteuerte Tonerzeugung möglich. Der Wert von R 6 bestimmt die Lautstärke.

Die verbleibenden 13 decodierten Adreßsignale sind zusammen mit dem Datenbus und dem $\overline{RD}$'- sowie $\overline{WR}$'-



⑧ **DIL-Stecker mit LEDs:** In die Buchse DIL 2 gesteckt, zeigen die LEDs den Binärcode einer mit POKE eingegebenen Zahl



⑦ **Bestückungsplan:** Blick auf Bauteileseite. Die Lötpunkte der hier scheinbar „in der Luft" hängenden Bauelemente befinden sich auf der Lötseite

Signal an die Buchse DIL 1 geführt. Auch der CPU-Takt steht an dieser Buchse zur Verfügung. Hier lassen sich weitere Zusatzschaltungen anschließen, die in späteren Heften der FUNKSCHAU veröffentlicht werden. Portadressen sowie die Stiftbelegung der DIL-Buchsen sind in *Bild 5* zusammengefaßt.

Aufbautips und Inbetriebnahme

Die Schaltung wird auf einer zweiseitig kupferkaschierten Europakarte aufgebaut (*Bild 6*). Selbstverständlich ist dabei ein Teilausbau möglich. Durch Weglassen der jeweils nicht benötigten ICs kann die Platine auf die Speichererweiterung (auch teilweise) oder den I/O-Port bzw. Softwareschalter beschränkt werden. Ein Nachrüsten ist jederzeit möglich.

Um bei der Platinenherstellung eine exakte Passung zwischen Vorder- und Rückseite zu erhalten, empfiehlt sich das folgende Verfahren: Die Europakarte erhält vor der Belichtung vier Bohrungen (1 mm ∅) jeweils 5 mm von den Kanten entfernt. Auf einem Holzbrett werden dann vier Stifte in einem Rechteck (9 cm × 15 cm) angeordnet und darauf die Platine und die Vorlage aufgesteckt. Hierzu hat die Platinenvorlage in den Ecken vier Lötaugen, jeweils 5 mm vom Rand entfernt, die durchstochen werden.

Bohrungen werden mit 0,8 mm Durchmesser ausgeführt, mit Ausnahme der Anschlüsse für Steckleiste, Siebkondensator und Spannungsregler (1 mm). Anschließend werden alle von der Bestückungsseite aus sichtbaren Bohrungen durchkontaktiert (Ausnahme: Bohrungen der Steckerleiste). Danach erfolgt das Bestücken (ICs auf Sockel) gemäß *Bild 7*.

Beim Einlöten der Steckerleiste ist genügend Abstand zum Platinenrand einzuhalten, damit das Anstecken an den ZX 81 ohne Schwierigkeiten möglich ist. Die A-Kontaktreihe wird von der Bestückungsseite her verlötet. Für den Spannungsregler sollte schließlich ein geeignetes Kühlblech vorgesehen werden (Befestigungsbohrungen sind auf der Platine vorhanden).

Es empfiehlt sich, die Platine zunächst ohne eingesteckte ICs in Betrieb zu nehmen. Wenn keine Kurzschlüsse vorhanden sind, muß auf dem Bildschirm der Cursor erscheinen. Dann werden die ICs eingesetzt (Spannungsversorgung zuvor ausschalten!). Es dauert jetzt etwas länger, bis der Cursor wieder erscheint, da infolge der Speichererweiterung nun ein größerer Speicherbereich auf RAMTOP hin geprüft wird.

RAMTOP (PRINT PEEK 16388 + 256 * PEEK 16389) liefert bei richtiger Funktion 23552. Falls gewünscht, lassen sich auch nur ein oder zwei Speicher-ICs einsetzen. Dabei ist auf die richtige Reihenfolge zu achten, da der Computer einen durchgehenden Speicherbereich benötigt. RAMTOP ist in diesem Fall 19456 bzw. 21504.

POKE 23556, Zahl schaltet die grüne LED, POKE 23557, Zahl die rote LED ein (Zahl kann zwischen 0 und 255 liegen). Ein an den Lautsprecherausgang angeschlossener Hörer gibt dabei jedesmal ein Klickgeräusch ab.

Um den IN-Port zu testen, wird ein Achtfach-DIL-Schalter in den Sockel DIL 3 gesteckt. PRINT PEEK 23558 liefert den Dezimalwert der eingestellten Binärkombination. Zur Überprüfung des OUT-Ports (DIL 2) sind acht Anreih-LEDs geeignet, die zusammen mit den Vorwiderständen in einen 16poligen DIL-Stecker gelötet werden (*Bild 8*). Mit POKE 23558, Zahl leuchten die LEDs gemäß dem Binärwert der eingegebenen Zahl.

Damit ist die Platine einsatzbereit. Anwendungsbeispiele und passende Erweiterungsschaltungen folgen in späteren Heften.

Oskar Merker

Stückliste der Gesamtschaltung

IC 1, 2, 3, 12:	4 × 74LS245
IC 5, 6, 7:	3 × HM 6116LP-3
IC 10:	1 × 74LS02
IC 9:	1 × 74LS154
IC 4:	1 × 74LS138
IC 11:	1 × 74LS373
IC 8:	1 × 7400 N
	1 × 7805 (Spannungsregler)
	1 × LED rot 3 mm
	1 × LED grün 3 mm
	8 × Anreih-LED rot (für OUT-Kontrolle)
D1:	1 × 4148 (o. ä. Si-Diode)
D2...D11:	10 × AA 116 (o. ä. Ge-Diode)
R6...9	3 × 470 Ω
	8 × 470 Ω (für OUT-Kontrolle)
R1...3:	3 × 22 kΩ
R4, 5:	2 × 1 kΩ
C1:	1 × 470 µF/16 V
C2...6:	5 × 22 nF
C7, 8:	2 × 150 nF
Sp:	1 × Lautsprecher 50 Ω (geeignet auch Postkapsel 32 Ω)
	1 × Steckerleiste 2 × 23pol., Raster 2,54
	1 × DIL-Schalter 8pol.
	1 × DIL-Stecker 16pol.
	5 × Stecksockel 24pol.
	5 × Stecksockel 20pol.
	3 × Stecksockel 16pol.
	2 × Stecksockel 14 pol.
	6 × Steckstifte

ZX 81 à la carte (3)

Software voller Klang

Wer die Grundplatine zu dieser Serie nachgebaut hat, kann jetzt allein mit Software dem ZX 81 Töne entlocken. Selbst eine Miniorgel läßt sich so verwirklichen.

Die in den beiden FUNKSCHAU-Heften 12 und 13 (1983) vorgestellte Grundplatine zu dieser Serie trägt auch einen softwaregesteuerten Schalter zum Betrieb eines Lautsprechers. Mit Hilfe von Programmen ist der ZX 81 nun in der Lage, Töne zu erzeugen. Dadurch lassen sich z. B. akustische Marken in einen Programmablauf einbauen. Sogar eine elektronische Miniorgel läßt sich „programmieren".

Basic setzt der Tonhöhe Grenzen

Das Prinzip der Tonerzeugung ist einfach: Der Software-Schalter wird durch das Steuerprogramm lediglich in schneller Folge umgeschaltet. Da es sich bei diesem Schalter um ein Set-Reset-(SR-)Flipflop handelt, läßt sich damit das Tastverhältnis beliebig einstellen (*Bild* 1). Das folgende kleine Programm erzeugt bereits einen Ton:

```
10 POKE 23556,0
20 POKE 23557,0
30 GOTO 10
```

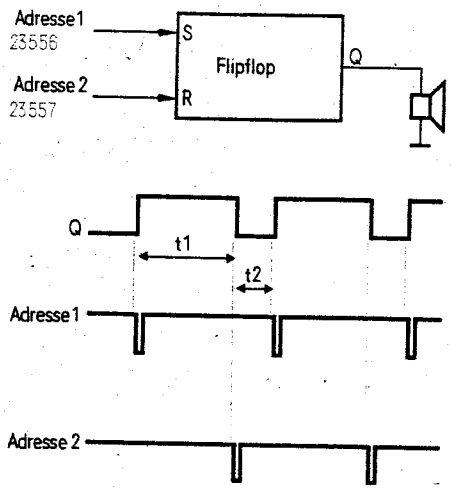
Befindet sich der ZX 81 im SLOW-Modus, so klingt der Ton ein wenig jaulend. Der Grund dafür ist, daß der Computer 50mal pro Sekunde die Programmabarbeitung unterbricht, um ein Fernsehbild darzustellen. Wird auf den FAST-Modus umgeschaltet, fehlt das Jaulen und der Ton klingt höher: Eine definierte Tonerzeugung ist also nur im FAST-Modus möglich.

Es ist unschwer zu erkennen, daß die Tonfrequenz von den Zeiten t1 und t2 (Bild 1) bestimmt wird. Für die Grundfrequenz der stark oberwellenhaltigen Schwingung gilt: $f = 1/(t1 + t2)$. Die Frequenz wird also durch die Zeit bestimmt, die zum Abarbeiten der Programmbefehle benötigt wird. Man kann das leicht testen, indem man zwischen die POKE-Befehle weitere (beliebige) Befehle einschiebt, z. B.:

```
15 LET A = 0
25 LET A = 0
```

Die Frequenz wird dadurch niedriger. Im SLOW-Modus wird zusätzlich Zeit für die Bildausgabe benötigt; deshalb ist die Tonhöhe dann stets geringer als im FAST-Modus.

Will man die Tonfrequenz erhöhen, so erkennt man sehr schnell, daß das in Basic nicht geht: Nach dem Löschen der Zeilen 15 und 25 hat man bereits das schnellstmögliche Basicprogramm zur Tonerzeugung. Ein weiteres Steigern der Tonhöhe ist nur mit einem Maschinenprogramm anstelle des Basic-Programms möglich. Maschinenprogramme werden erheblich schneller abgearbeitet als Basic-Programme.



① **Software-Schalter:** Unter den beiden Adressen erteilte Set-Reset-Impulse bestimmen das Tastverhältnis des Tonsignals am Ausgang Q

```
100 REM 00000000000000000000000
        00000000000000000000000
110 REM S-E: PROGRAMM POKEN
120 PRINT "START : ";
130 INPUT S
140 PRINT S;" ENDE : ";
150 INPUT E
160 PRINT E
170 FOR N=S TO E
180 PRINT N;
190 INPUT K
200 POKE N,K
210 PRINT " - "; PEEK N
220 NEXT N
230 STOP
300 REM S-E: SPEICHERINHALT
310 PRINT "START : ";
320 INPUT S
330 PRINT S; " ENDE : ";
340 INPUT E
350 PRINT E
360 FOR N=S TO E
370 PRINT N;" - ";PEEK N
380 NEXT N
390 STOP
```

② **Hilfsprogamm:** Dieses Programm dient der (dezimalen) Eingabe von Maschinenprogrammen und zu deren Kontrolle

Tonerzeugung per Maschinenprogramm

Mit der USR-Funktion läßt sich beim ZX 81 ein zuvor geschriebenes Maschinenprogramm aufrufen. Zum Einschreiben eines solchen Programms in das RAM gibt es verschiedene Methoden (siehe auch Sinclair-Handbuch Kapitel 26).

Für unser Vorhaben ist die Unterbringung in einem (an erster Stelle im Programm stehenden) REM-Kommentar zweckmäßig. Hierzu werden nach dem REM-Befehl mindestens so viele (beliebige) Zeichen eingetippt, wie das spätere Maschinenprogramm Befehle hat. Anschließend wird an die Stelle der Zeichen das gewünschte Maschinenprogramm mit POKE-Befehlen eingeschrieben.

Da Maschinenprogramme jetzt öfters benötigt werden, ist es sinnvoll, das Hilfsprogramm aus *Bild 2* einzutippen und abzuspeichern. Es enthält neben dem „Platzhalter" für das Maschinenprogramm (Zeile 100) eine Hilfsroutine zum Poken (Zeile 120...230) und zum Kontrollieren (Zeile 300...390; Aufruf: RUN 300) des Maschinenprogramms.

Mit Hilfe des POKE-Programms kann sofort ein kleines Maschinenprogramm zur Tonerzeugung geladen werden (*Bild 3*). Startadresse ist 16514 (Adresse im Speicher, in der der Code für das erste Zeichen im REM-Kommentar abgelegt ist) und die Endadresse ist 16533. Aufgeführt ist die jeweilige Speicheradresse, sowie die Befehlsangabe in Assembler- und Maschinensprache (dezimal).

Ist der letzte Maschinenbefehl eingegeben worden, meldet sich der Rechner mit 9/230 (bei Bildschirmüberlauf weiter mit CONT). Nach Drücken der CONT-Taste läßt sich das Maschinenprogramm überprüfen.

Der Inhalt des REM-Kommentars hat sich jetzt verändert. Die nunmehr dargestellten Symbole entsprechen den Maschinenbefehlen. Nun muß das Programm noch um die Zeile

400 LET A = USR 16514

ergänzt werden (Aufruf des Maschinenprogramms aus dem Basic-Programm).

Bevor das Programm mit RUN 400 (FAST-Modus) gestartet wird, sollte es unbedingt abgespeichert werden. Das erspart viel Tipp-Arbeit, falls das Maschinenprogramm infolge eines Eingabefehlers „abstürzen" sollte. Wenn man das Hilfsprogramm nicht mehr benötigt, darf man es bis auf die Zeile 100, die ja das Maschinenprogramm enthält, aus dem RAM tilgen.

Adresse	Mnemonik	Eingabe (dezimal)
16514	ld a,255	62
16515		255
16516	ld b,255	6
16517		255
16518	djnz,−2	16
16519		254
16520	ld (23556),a	50
16521		4
16522		92
16523	ld b,255	6
16524		255
16525	djnz,−2	16
16526		254
16527	ld (23557),a	50
16528		5
16529		92
16530	dec a	61
16531	jrnz,−17	32
16532		239
16533	ret	201

③ **Tonerzeugung:** Das Maschinenprogramm erzeugt Töne deutlich höherer Frequenz, als es ein Basic-Programm vermag

Ladebefehle ersetzen POKE-Befehle

Wie *Bild 4* zeigt, ist die Tonerzeugung in Maschinensprache ähnlich aufgebaut wie in Basic: Den POKE-Befehlen von dort entsprechen die Ladebefehle unter den Adressen 16520 bzw. 16527 (*ld (NN),a*) zum Umschalten des Flipflops.

Da das Maschinenprogramm sogar zu schnell ist, müssen zwischen die Schaltbefehle noch Programmschleifen zur Verzögerung eingebaut werden. Hierbei wird der Inhalt des b-Registers bis auf 0 dekrementiert (heruntergezählt). Der entsprechende Anfangswert des b-Registers (Speicherstelle 16517 bzw. 16524) bestimmt die hierfür nötige Zeit und damit die Frequenz.

Nach jeder Schwingungsperiode wird der Akku-Inhalt dekrementiert, sein Anfangswert (Speicherstelle 16515) bestimmt also die Tondauer (genauer: die Anzahl der Schwingungszüge, die Tondauer ist also frequenzabhängig).

Ein Berechnen der Frequenz und Tondauer ist im Prinzip möglich, wenn man die Zeiten aufaddiert, die zur Abarbeitung der einzelnen Maschinenbefehle benötigt werden. Da die Z-80A-CPU mit einer Taktfrequenz von 3,23 MHz arbeitet, dauert z. B. das Abarbeiten des dec-Befehls 1,24 µs. Weil aber die bedingten Sprungbefehle (*jrnz, djnz*) unterschiedlich lange dauern, je nachdem ob die Bedingung erfüllt ist oder nicht, ist dieses Verfahren nicht ganz einfach; Experimentieren führt meist schneller zum Ziel.

Basic-Programme rufen das Maschinenprogramm

Akustische Marken im Ablauf eines Programmes haben einen hohen Aufmerksamkeitswert. Dabei ist es besonders günstig, wenn das hierzu benutzte Maschinenprogramm überhaupt nicht in Erscheinung tritt, sondern „versteckt" wird.

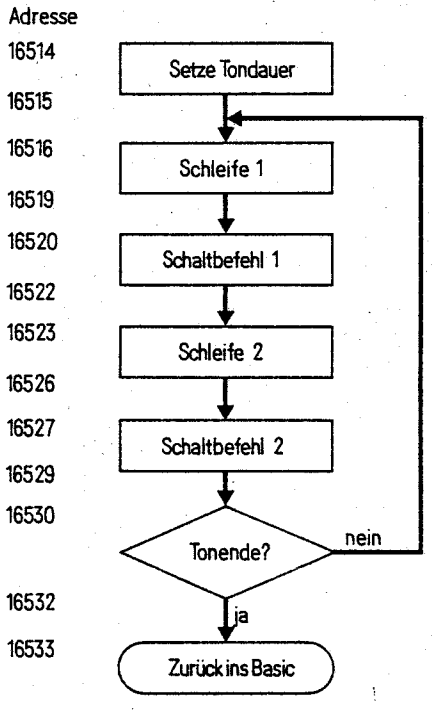
Dies läßt sich erreichen, indem man den Wert der Systemvariablen RAMTOP herabsetzt, der Computer „vergißt" dadurch gewissermaßen den oberhalb von RAMTOP verbliebenen Teil seines Speichers. In diesem vergessenen Speicherteil wird anschließend das Maschinenprogramm untergebracht. Wenn die 6-KByte-Erweiterung voll bestückt ist, hat RAMTOP den Wert 23552. Durch

POKE 16388, 204 und
POKE 16389, 91

wird RAMTOP auf 23500 herabgesetzt. Anschließend muß man NEW eingeben und danach das Tonerzeugungsprogramm vom Band laden. Ihm wird dann noch folgendes Programm angefügt:

```
400 FOR N=16514 T0 16533
410 POKE (N+6986), PEEK N
420 NEXT N
```

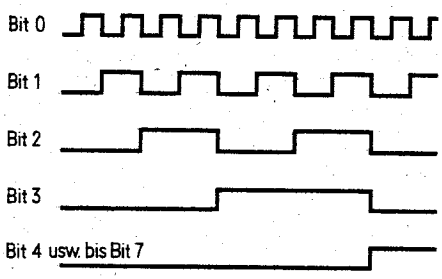
Startet man diesen Programmteil mit RUN 400, so kopiert er das Maschinenprogramm in den Speicherbereich oberhalb von RAMTOP. Das Maschinenprogramm ist dort ohne Probleme lauffähig, da es nur relative Sprünge verwendet und deshalb keine Sprungadressen geändert werden müssen.



④ **Flußdiagramm:** Um Töne im Nf-Bereich zu erzeugen, muß das Maschinenprogramm mit zwei Schleifen gebremst werden

Nach dem Eingeben von NEW ist der Computer zur Aufnahme eines Basic-Programms bereit. Im Rahmen dieses Programms kann die Tonmarke einfach mit LET A = USR 23500 aufgerufen werden.

Soll die Tonmarke im SLOW-Modus verwendet werden, empfiehlt es sich, die Werte für die Tonhöhe zu ändern (Inhalt der Speicherstellen 23503 und 23510: jeweils 50). Zu beachten ist, daß jetzt beim Abspeichern des Programms auf Band das Programm zur Tonerzeugung nicht mit abgespeichert wird, da der ZX 81 den oberhalb von RAMTOP liegenden Speicherbereich hierbei nicht berücksichtigt. Das Tonprogramm muß also jeweils vor dem Basic-Programm in der beschriebenen Weise geladen werden.



⑤ **Signale am Datenausgang:** Auch der OUT-Port (Bit 0 bis Bit 7) der Grundplatine kann zum Tonausgang werden. Dabei hat Bit 0 des Datenworts immer die höchste Frequenz

Aktivierte Leitung	Bitmuster	Hex	Dez
A15	0111 1111	7F	127
A14	1011 1111	BF	191
A13	1101 1111	DF	223
A12	1110 1111	EF	239
A11	1111 0111	F7	247
A10	1111 1011	FB	251
A9	1111 1101	FD	253
A8	1111 1110	FE	254

Aktivierte Leitung	Bitmuster	Hex	Dez
keine	0111 1111	7F	127 (31)
KBD 0	0111 1110	7E	126 (30)
KBD 1	0111 1101	7D	125 (29)
KBD 2	0111 1011	7B	123 (27)
KBD 3	0111 0111	77	119 (23)
KBD 4	0110 1111	6F	111 (15)

⑥ **Tastaturabfrage:** Oben die ausgegebenen, unten die eingelesenen Bitmuster*). Bei der Ausgabe ist das Bitmuster spaltenförmig maßgebend (A8...A15)

*) Bild 6 unten. Die ersten drei Bits der Bitmuster werden vom Sinclair-Logik-Chip bestimmt und können unter Umständen nicht die Werte 0 1 1 haben. Bei Maschinenprogrammen hilft dann das Maskieren dieser Bits. So liefert der Befehl *and 31* den reinen Tastencode. Die zugehörigen Zahlenwerte sind im Bild in Klammern gesetzt.

Durch Setzen/Rücksetzen einzelner Bits des Out-Ports (Adresse 23558, siehe Heft 13/83) läßt sich ebenfalls eine tonfrequente Rechteckspannung erzeugen. Besonders elegant gelingt dies durch fortlaufendes Dekrementieren (um 1 verringern) eines CPU-Registers und Ausgeben des Registerinhalts über den Out-Port.

Tonerzeugung über den Out-Port

Bild 5 zeigt, daß an der Bitposition 0 des ausgegebenen Bytes die höchste Frequenz abgenommen werden kann, während die nächsten Bitpositionen eine jeweils um den Faktor 2 niedrigere Frequenz führen. Damit eignet sich diese Methode sehr gut, um mit dem ZX 81 ein Musikinstrument zu realisieren, da aufeinander folgende Oktaven im Frequenzverhältnis 1:2 stehen.

Ein 50-Ω-Lautsprecher bzw. eine 32-Ω-Postkapsel kann über einen Vorwiderstand von 220...470 Ω (Widerstandswert je nach gewünschter Lautstärke) direkt zwischen die entsprechende Bitposition des Out-Ports (DIL 2) und Masse geschaltet werden.

Will man auf der Tastatur des ZX 81 spielen, dann darf der Ton nur erklingen, solange die entsprechende Taste gedrückt ist. Darum muß im Maschinenprogramm zur Tonerzeugung noch eine Tastaturabfrage vorhanden sein.

Wie „liest" der ZX 81 die Tastatur?

Die Tastatur des ZX 81 ist in einer 8×5-Matrix angeordnet (siehe auch Heft 18/1983, Seite 85; bei der dort abgebildeten Tastaturmatrix muß die Verbindung zwischen den Außenkreisen der Tasten V und B schwarz sein, nicht blau!). Dabei sind die acht Zeilen über Dioden mit den Adreßleitungen A8...A15 der CPU direkt verbunden. Die fünf Spalten-(Keyboard)leitungen sind über Widerstände an 5 V gelegt und werden den zugeordneten Eingängen (KBD 0 bis KBD 4) des Sinclair-Logik-Chips zugeführt.

Die Abfrage erfolgt über einen in-Befehl (Maschinensprache). Dabei wird die Tatsache genutzt, daß bei der Ausführung des Befehls *in a,(N)* die untere Adreßbushälfte (A0...A7) durch die CPU mit dem Operanden n belegt wird (der Operand n wählt die Adresse des gewünschten Eingabekanals an), während auf der oberen Adreßbushälfte (A8...A15) der Inhalt des Akkus erscheint.

Ein Beispiel soll das Ganze verdeutlichen: Mit dem Maschinenbefehl *ld a, 127* wird in den Akku das Bitmuster 0111 1111 geladen. Anschließend erfolgt der Befehl *in a,(254)*. Dadurch erscheint auf der Adreßleitung A15 eine logische 0.

Wird nun gleichzeitig eine der A15 zugeordneten Tasten gedrückt, so erhält auch die entsprechende Keyboardleitung (KBD 4...KBD 0) L-Pegel. Das Bitmuster auf den Keyboardleitungen stellt also eine Binärzahl dar, deren Zahlenwert die gedrückte Taste bestimmt. Der Sinclair-Logik-Chip registriert diesen Zahlenwert, führt ihn dem Datenbus zu und lädt den Wert in den Akku.
In *Bild 6* sind die über A8...A15 ausgegebenen Bitmuster (Zeilenauswahl) und die über den Datenbus eingelesenen Bitmuster (Spaltenauswahl) zusammengestellt. Damit kann jede einzelne Taste der Tastaturmatrix angesprochen werden.

Adresse	Mnemonik	Eingabe (dezimal)
16514	nop	0
16515	nop	0
.		0
.		0
16521	nop	0
16522	ld hl,23558	33
16523		6
16524		92
16525	ld b,0	6
16526		0
16527	ld a,(16518)	58
16528		134
16529		64
16530	ld c,a	79
16531	ld a,60	62
16532		60
16533	in a,(254)	219
16534		254
16535	sub 127	214
16536		127
16537	ret z	200
16538	dec c	13
16539	jrnz,−3	32
16540		253
16541	ld (hl),b	112
16542	inc b	4
16543	jr, −18	24
16544		238

⑦ **Orgelprogramm:** Hier ist neben der Tonerzeugung auch gleich die Tastaturabfrage mit enthalten

Jetzt wird der ZX 81 zur Orgel

Kommt der ZX 81 als Musikinstrument zu Ehren, dann sollte man auf der untersten Tastaturreihe die Stammtöne, in der darüberliegenden Reihe die entsprechenden Halbtöne greifen können (Klaviertastatur). Das bedeutet, daß für die Tastaturabfrage die Adreßleitungen A15, A14, A9 und A8 kontrolliert werden müssen. Hierzu gehört das Ausgabe-Bitmuster 0011 1100, d. h., 60 (dezimal) bzw. 3Ch. Falls eine der gewünschten Tasten gedrückt wird, ist der Zahlenwert des eingelesenen Bitmusters ungleich 127. Diese Tatsache nutzt das Maschinenprogramm von *Bild 7*. Eingegeben wird es, wie bereits beschrieben, in eine REM-Zeile ab Adresse 16514.

In Adresse 16518 wird abschließend der für die Tonhöhe maßgebliche Zahlenwert vor Aufruf des Maschinenprogramms eingegeben. Das eigentliche Programm beginnt bei Adresse 16522 (Aufruf: LET A = USR 16522). Ein Tastendruck in der unteren Hälfte der Tastatur erzeugt dann einen Ton.

In den Zeilen 16530...16537 erfolgt die Überprüfung, ob eine der gewünschten Tasten gedrückt ist. Falls dies nicht der Fall ist, bekommt der Akku durch Subtrahieren von 127 den Wert 0, so daß der Rücksprung aus dem Maschinenprogramm erfolgt (ret z).

Für die Tonerzeugung sind zwei Schleifen wichtig: Die Befehle unter den Adressen 16527...16530 laden in das Register c den für die Tonhöhe wichtigen Zahlenwert aus der Speicherstelle 16518. Nachdem der Inhalt von Register c auf 0 heruntergezählt ist (Schleife 1, 16538...16540), wird der Inhalt des b-Registers (zu Programmbeginn 0) über den Port (Adresse 23558) ausgegeben. Anschließend wird das b-Register inkrementiert (um 1 erhöht). Dann wiederholt sich der gesamte Vorgang (Schleife 2, 16527...16544).

Als Hilfestellung und Anregung für eigene Experimente sind in *Bild 8* einige Zahlenwerte (Inhalt der Speicherstelle 16518) und die zugehörigen Frequenzwerte (Tonabnahme an Bit 2 des Ports) angegeben. Besonders hervorgehoben sind die zu den Tönen der zwischen c' und c'' liegenden Oktave (näherungsweise) gehörenden Frequenzen.

Um den ZX 81 als elektronische Miniorgel zu benutzen, muß das Maschinenprogramm in ein kleines Basic-Programm eingebunden werden (*Bild 9*): Das Maschinenprogramm ist wie üblich in einer REM-Zeile (Zeile 100) untergebracht. Wegen Zeile 200 wartet der Computer solange, bis eine Taste gedrückt wird.

Der Variablen A wird der Code-Wert der entsprechenden Taste zugewiesen (Zeile 300) und in Abhängigkeit von diesem Wert die der gewünschten Frequenz entsprechende Zahl in die Speicherstelle 16518 eingeschrieben.

In *Bild 10* ist die „Orgeltastatur" gezeigt. Für einen ersten Probelauf eignet sich der nachfolgende Code, eines sicher allgemein bekannten Liedchens. Ein fetter Buchstabe bedeutet, daß der Ton etwas länger zu halten ist: Z X C V **B B** N N N N **B** N N N N **B** V V V V **C C** X X X X **Z**

Oskar Merker

Stichworte zum Inhalt

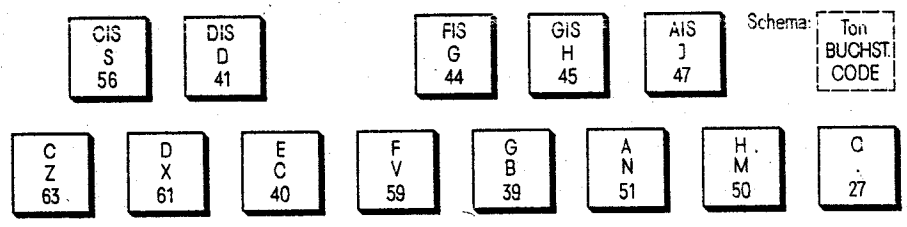
Software-Schalter, SR-Flipflop, Tonerzeugung, ZX 81, Maschinenprogramm, akustische Marken, RAMTOP, Tastverhältnis, Tastaturabfrage, elektronische Orgel.

Zahlenwert für Schleife	Frequenz (Hz)	Ton
255	195	
250	199	
200	247	
190	259	
187	264	c'
180	274	
176	280	cis'
170	290	
166	296	d'
160	307	
156	315	dis'
150	327	
149	329	e'
140	350	
139	352	f'
131	373	fis'
130	376	
123	397	g'
120	406	
116	420	gis'
111	438	a'
110	442	
104	467	ais'
100	485	
98	494	h'
91	530	c''

⑧ **Frequenz am OUT-Port:** Die Werte gelten für die Signalabnahme an Bit 2 des Ports und für den FAST-Modus

```
100 REM 000000000000000000000000
        000000000000000000000000
200 IF INKEY$="" THEN GOTO 200
300 LET A=CODE INKEY$
310 LET X=16518
400 IF A=63 THEN POKE X,187
410 IF A=61 THEN POKE X,166
420 IF A=40 THEN POKE X,149
430 IF A=59 THEN POKE X,139
440 IF A=39 THEN POKE X,123
450 IF A=51 THEN POKE X,111
460 IF A=50 THEN POKE X,98
470 IF A=27 THEN POKE X,91
480 IF A=56 THEN POKE X,176
490 IF A=41 THEN POKE X,156
500 IF A=44 THEN POKE X,131
510 IF A=45 THEN POKE X,116
520 IF A=47 THEN POKE X,104
600 LET A=USR 16522
700 GOTO 200
```

⑨ **Frequenzzuweisung:** Dieses Basic-Programm weist den vom Orgelprogramm abgefragten Tasten die gewünschte Frequenz zu



⑩ **Orgeltastatur:** Zwischen der Folientastatur und dem erzeugten Ton besteht der im Schema gezeigte Zusammenhang

ZX 81 à la carte (4):

Schrittmacher

Mit der hier vorgestellten einfachen automatischen Wiederholfunktion (Auto-Repeat) reagieren die „Tasten" des ZX-81-Tableaus auf Dauerdruck.

Auto-Repeat-Schaltungen in Computern sorgen dafür, daß die den Tasten zugeordneten Zeichen oder Funktionen so lange ausgegeben bzw. ausgeführt werden, wie eine Taste gedrückt bleibt. Das bringt offenkundige Vorteile, wenn man z. B. mit dem Cursor eine bestimmte Stelle am Bildschirm erreichen möchte, wenn das Löschen mehrerer Zeichen notwendig ist oder wenn man in einem REM-Kommentar lauter gleiche Zeichen unterbringen will (Platzreservierung für Maschinenprogramm). Die Zahl der Tastenbetätigungen wird dann drastisch reduziert.

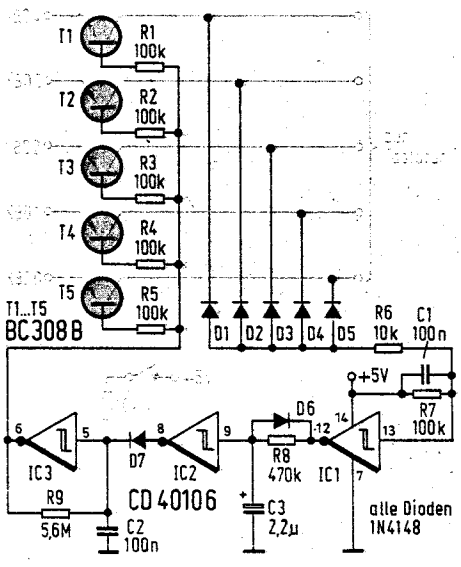
Transistoren kontra Fingertrommeln

Das Schaltbild der Auto-Repeat-Baugruppe zeigt *Bild 1*. Normalerweise sind die fünf von der Tastatur kommenden Leitungen unmittelbar mit den Anschlüssen KBD 0 bis KBD 4 verbunden (Keyboard-Leitungen). Eine Tastenbetätigung hat dann ein charakteristisches Bitmuster auf den Keyboard-Leitungen zur Folge, das vom Sinclair-Logik-Chip ausgewertet wird (siehe auch Heft 18/83 und Heft 24/83).

Aufgabe der Auto-Repeat-Baugruppe ist es, wiederholtes Drücken einer Taste einfach durch rythmisches Unterbrechen der Keyboard-Leitungen zu ersetzen. Die Transistoren T1 bis T5 wirken deshalb als Schalter. Angesteuert werden sie von dem als Oszillator arbeitenden Schmitt-Trigger IC 3. Die Frequenz des Steuersignals hängt von R 9 und C 2 ab; sie ist zugleich die Wiederholrate der Auto-Repeat-Funktion.

Solange keine Taste gedrückt ist (alle Keyboard-Leitungen führen dann H-Pegel), liegt über R 7, IC 1 und IC 2 H-Pegel am Eingang des Oszillators. Der Oszilator kann deswegen nicht schwingen und führt konstant L-Pegel am Ausgang, so daß alle Transistoren leitend sind.

Das ändert sich schlagartig, wenn eine Taste gedrückt wird. Dann liegt auf einer der Keyboard-Leitungen L-Pegel, der über eine der Dioden D 1 bis D 5 auch den Eingang von IC 1 auf L-Pegel zieht. Dadurch führt der Ausgang von IC 1 H-Pegel, und der Kondensator C 3 wird über R 8 langsam aufgeladen. Von der Zeitkonstante dieses Verzögerungsgliedes hängt es ab, mit welcher Zeitverzögerung die Auto-Repeat-Funktion einsetzt. Die Verzögerungszeit ist erforderlich, um überhaupt noch einzelne Buchstaben eingeben zu können. Mit der angegebenen Dimensionierung beträgt die Verzögerungszeit ca. 1 s.

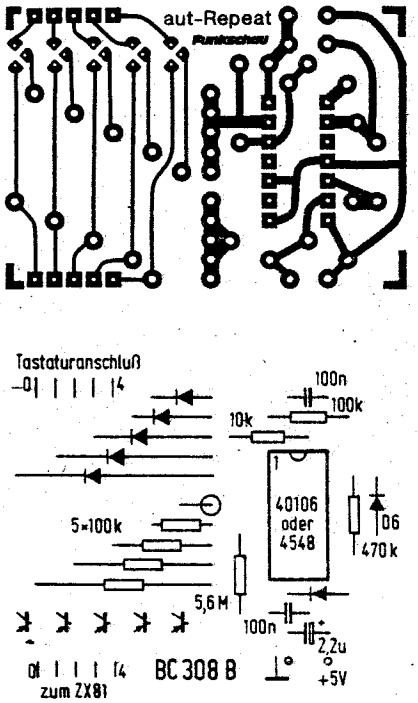


① **Auto-Repeat-Schaltung:** Dieser kleine Zusatz sorgt dafür, daß beim Drücken einer Taste am ZX 81 die jeweils zugeordnete Funktion wiederholt ausgeführt wird

Sobald die Spannung an C 3 die Schaltschwelle des Schmitt-Triggers IC 2 erreicht hat, wird der Kondensator C 2 über D 7 nicht mehr zwangsweise geladen, und der Oszillator IC 3 beginnt zu schwingen, d. h. am Ausgang tritt rhythmisch H-Pegel auf, der die fünf PNP-Transistoren sperrt. Damit ist das Ziel erreicht, die Keyboard-Leitungen kurzzeitig zu unterbrechen.

Nach der Tastenbetätigung sorgt die Diode D 6 für ein Entladen von C 3, damit beim nächsten Tastendruck die Auto-Repeat-Funktion nicht sofort einsetzt. Höhere Werte für R 8 oder C 3 verlängern die Zeitverzögerung, wogegen höhere Werte für R9 und C 2 die Wiederholrate heruntersetzen. Mit der im Schaltbild gestrichelt eingetragenen Ergänzung (auf der Platine *Bild 2* nicht berücksichtigt) läßt sich die Auto-Repeat-Funktion stillegen, wenn z. B. bei Spielen eine softwaremäßige Wiederholfunktion der Cursor-Tasten gegeben ist.

In der gezeigten Ausführung wirkt die Auto-Repeat-Schaltung auf sämtliche Tasten des ZX 81. Wird dagegen nur für die Cursor-Tasten des Computers eine Wiederholfunktion verlangt, so können die Transistoren T 4 und T 5 sowie die



② **Platine und Bestückungsplan:** Die Repeat-Platine paßt im Gehäuse des ZX 81 neben das Kühlblech des Spannungsreglers

ZX 81 à la carte (5):

Daten-Drehscheibe

Teil 1: I/O-Schnittstelle mit Z-80-PIO

Der Z-80-PIO-Baustein (PIO: Parallel In Out) verhilft allen Computern mit (Z-80-Mikroprozessor zu einer ungemein vielseitigen Ein-/Ausgabe-Schnittstelle.

In den Heften 12 und 13/1983 wurde für den ZX 81 schon einmal eine I/O-Schnittstelle (I/O-Port) vorgestellt. Deren Platine trägt zusätzlich eine 6-KByte-RAM-Erweiterung und ein Flipflop zur Tonerzeugung – sie ist deshalb nicht gerade einfach selbst herzustellen.

Die hier vorgestellte I/O-Schnittstelle bietet mehr Ein-/Ausgabe-Funktionen und beim Aufbau kommt man mit einer einseitig beschichteten Platine aus, die keine Herstellungsprobleme birgt. Auf eine RAM-Erweiterung wurde diesmal verzichtet, da käufliche RAM-Module inzwischen sehr preisgünstig sind. Schnittstelle und RAM-Modul dürfen jederzeit gemeinsam angeschlossen werden, sofern das RAM-Modul das zuläßt (durchgeschleifte Steckerleiste). Die neue Schnittstelle hat jedoch noch einen weiteren Vorteil: Sie läßt sich im Prinzip an jeden Computer mit Z-80-CPU anschließen (ZX 81, ZX Spectrum, VZ 200, Laser 110/210, Jupiter Ace, Colour Genie, MZ 80). Jedoch kann das Betriebssystem des Computers Störungen verursachen, wie Teil 2 am Beispiel des ZX 81 zeigen wird.

Auf die „alte" Schnittstelle konnte man mit den Basic-Befehlen POKE (Daten ausgeben) und PEEK (Daten einlesen) zugreifen. Der PIO-Baustein fordert dagegen Z-80-Maschinenbefehle, die ihn in den gewünschten Betriebszustand (Ein-/Ausgabe) bringen. Die PIO-Schnittstelle muß also programmiert werden.

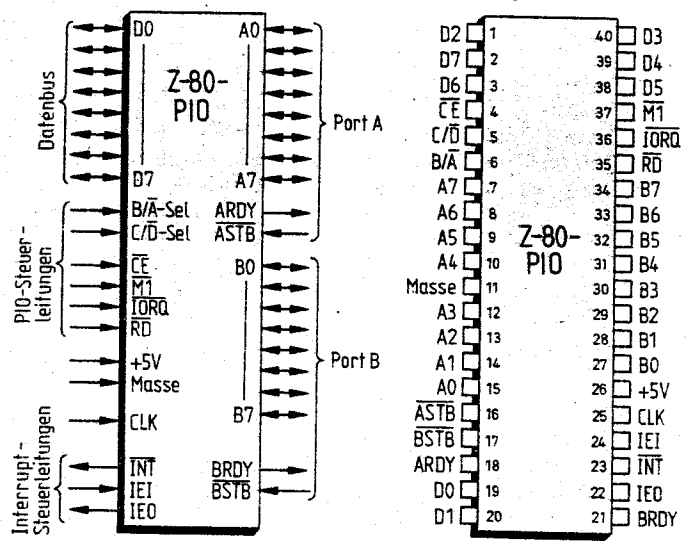
Im dritten Teil werden dafür vielseitige Ein-/Ausgabe-Programme vorgestellt, so daß man die Schnittstelle auch ohne Kenntniss der Z-80-Maschinensprache nutzen kann. Nutzen bedeutet das Steuern externer Geräte oder Anlagen (Ein-/Ausschalten, Lichtorgel, Modelleisenbahn), das Überwachen externer Abläufe oder die Übernahme von Daten (z. B. Eingabe von Morsezeichen).

Z-80-PIO: Tausendsassa im 40poligen Gehäuse

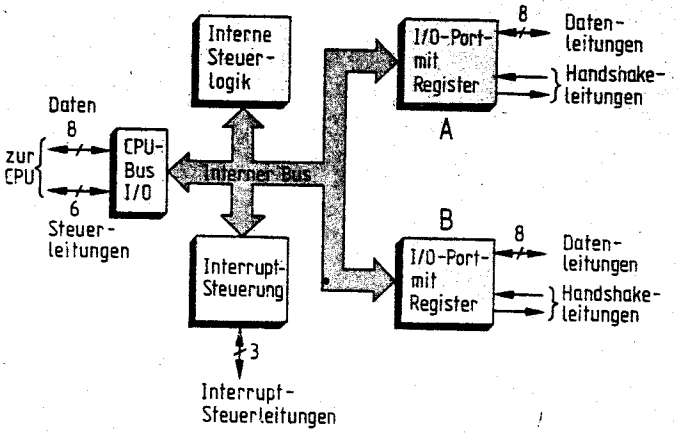
Der PIO-Baustein (*Bild 1*) kann über seine zwei Kanäle (Ports) vom Anwender ausgewählte 8-Bit-Datenworte ausgeben oder an den Ports anliegende 8-Bit-Datenworte zur weiteren Verarbeitung in den Computer holen. Welche der Betriebsarten und welcher Port genutzt wird, muß zuvor mit einem kurzen Maschinenprogramm festgelegt werden.

Darüber hinaus bietet der Baustein eine „Interrupt-($\overline{\text{INT}}$-)Steuerung" des Computers. Das heißt, daß der Computer das laufende Programm unterbricht, wenn an einem der Ports ein bestimmter Zustand herrscht (z. B. wenn ein Bit seinen Wert wechselt). Damit wird die CPU entlastet, denn sie muß sich erst dann um den Port kümmern, wenn dort ein vorgesehener Zustand eintritt (beim ZX 81 leider nicht möglich).

Die Betriebsart-Anweisungen müssen dem PIO-Baustein über den Datenbus mitgeteilt werden. Der Baustein „merkt" sich die jeweiligen Anweisungen, indem er sie in Registern (8-Bit-Speicherstellen) zwischenspeichert. Beim Datenver-



① **Z-80-PIO-Baustein:** Das IC enthält sämtliche Bauelemente für eine Parallel-Schnittstelle mit zwei 8-Bit-Ports



② **Blockschaltung des PIO-Bausteines:** Nach Kommandos der CPU an die interne Steuerlogik lenkt diese den Datenfluß selbsttätig

ehr über die Ports greift eine interne Steuerlogik auf diese Register zu und rkennt am Inhalt, wie mit den Daten zu erfahren ist (*Bild 2*).

Der PIO-Baustein enthält bereits alle ür die Schnittstelle notwendigen Bauelemente. Damit die CPU das IC gezielt nsprechen kann (adressieren), ist lediglich noch ein Adreßdecoder notwendig *Bild 3*). Er sorgt dafür, daß der PIO-Baustein immer nur dann auf den Datenus zugreifen kann, wenn auf dem dreßbus ganz bestimmte Adressen lieen. Dezimal sind das mit der gezeigten eschaltung die vier Adressen 248 bis 51 (hexadezimal: F8h, F9h, FAh und Bh). Liegt eine dieser Adressen auf em Adreßbus, wird der PIO-Baustein ber den Freigabeeingang ($\overline{CE}$: Chip Enale) aktiviert.

Zwei Maschinenbefehle lenken den Datenfluß

Da das Betriebssystem das ZX 81 im dreßbereich zwischen 0 und 8192 unergebracht ist, liegen die Adressen 248 is 251 eigentlich in einem verbotenen ereich, denn woher soll der Computer n Ernstfall wissen, ob er die unter diesen Adressen im ROM gespeicherten Befehle ausführen oder den PIO-Baustein ktivieren soll? Mit POKE und PEEK ommt man hier nicht weiter. Erst die -80-Maschinenbefehle *in* und *out* helen aus der Klemme. Dazu ein Beispiel: Der Befehl *out (F8),a* legt auf den dreßbus die Adresse F8h und auf den atenbus den Inhalt des Akkus (gewünschtes Ausgabe-Datenwort). Umgehrt wird mit *in (F8),a* ein auf dem atenbus liegendes Eingabe-Datenwort den Akku geladen. Zusätzlich bewirken beide Befehle, daß der $\overline{IORQ}$-Ausang der CPU (IORQ: Input/Output-Request – Ein-/Ausgabe-Aufforderung) auf gisch 0 geht.

Abhängig von diesem Signal sperrt im ZX 81 der Sinclair-Logik-Chip das OM, so daß es nicht zu einer Doppeldressierung des ROMs und des PIOausteins kommt. Damit auch das PIO erfährt „was los ist", erhält es ebenlls das $\overline{IORQ}$-Signal (direkt von der PU).

Unmittelbar mit der CPU verbunden nd noch die Datenleitungen D0 bis D7, ie Taktleitung CLK sowie die Steuerleiungen $\overline{M1}$ und $\overline{RD}$. Über $\overline{M1}$ und $\overline{RD}$ ilt die CPU dem PIO-IC mit, ob es aten vom Datenbus übernehmen kann oder ob es die Daten an einem der Ports auf den Datenbus legen darf (siehe auch Heft 19/1983, Seite 72). Ein Verbinden der $\overline{INT}$-Anschlüsse von CPU und PIO ist allein in der Betriebsart „Interrupt-Steuerung" nötig.

Zwei Nachzügler machen die Adressierung komplett

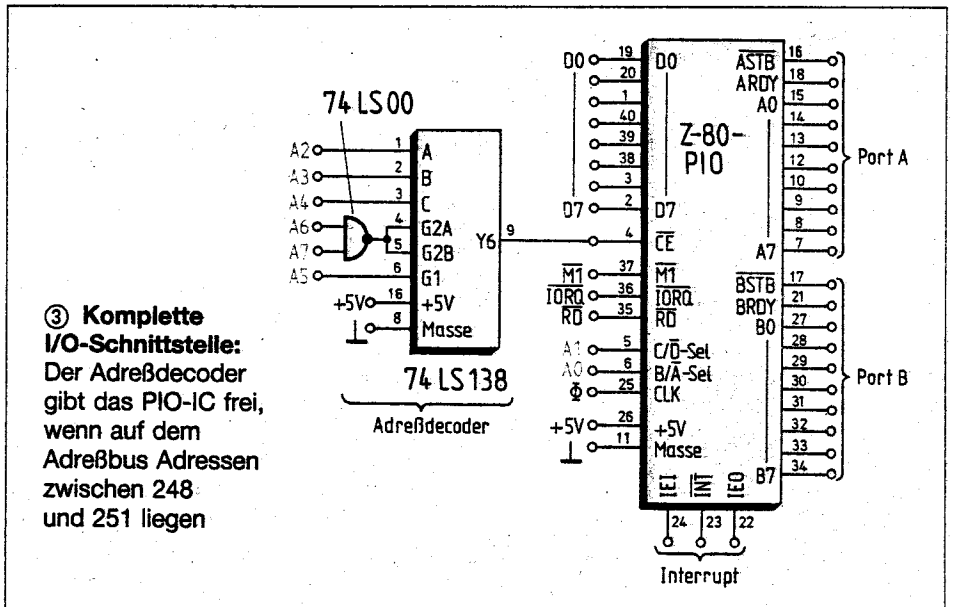
Jetzt fehlen nur noch zwei wichtige Leitungen zur CPU: Die eine hat die Bezeichnung C/$\overline{D}$ SEL und wird mit der Adreßleitung A1 verbunden. Die andere (B/$\overline{A}$ SEL) muß man mit Adreßleitung A0 verbinden.

Erst damit ist die Adressierung des PIO-Bausteins mit den Adreßsignalen A0 bis A7 komplett. Welche der vier PIO-Adressen nun angesprochen wird, entscheidet darüber, ob Port A oder Port B aktiviert wird und ob Daten oder Steuerworte übertragen werden (*Bild 4*). Maßgebend dafür sind die beiden Adreßbits A0 und A1; die restlichen Adreßbits A2 bis A7 dienen lediglich zum Gewinnen des Freigabesignals $\overline{CE}$ über den Adreßdecoder. Hier ließen sich auch andere Adressen als 248 bis 251 wählen, solange man von den 256 möglichen Adressen (0 bis 255) nicht solche verwendet, die auch das Betriebssystem des Computers nutzt.

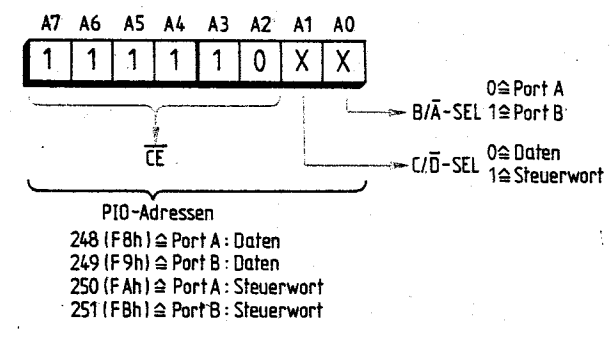
Die Adressierung des PIO-Bausteins ist damit abgeschlossen. Der nächste Schritt ist die Wahl der gewünschten Betriebsart. Dazu muß man dem Baustein unter den Adressen 250 bzw. 251 (FAh bzw. FBh) ein passendes Steuerwort über den Datenbus zuführen. Vier Steuerworte entscheiden dann über vier Betriebsarten.

Vier Betriebsarten erfüllen jeden Wunsch

● In der Betriebsart 0 laßt sich sowohl Port A als auch Port B als Ausgabekanal schalten. Die Daten – jeweils ein Byte – werden dann von der CPU in ein Ausgaberegister des PIO-ICs geschrieben.



③ Komplette I/O-Schnittstelle: Der Adreßdecoder gibt das PIO-IC frei, wenn auf dem Adreßbus Adressen zwischen 248 und 251 liegen



④ Bedeutung der Adreßbits: Von der jeweiligen PIO-Adresse hängt es ab, ob man Daten/Steuerworte für Port A/B auf den Datenbus legen muß

Da nun ein am Port ausgeschlossenes Peripheriegerät nicht genau weiß, wann es die Daten abholen darf, und ebenso die CPU nicht weiß, wann die Daten ausgegeben sind und das Ausgaberegister wieder frei ist, wird der Datenfluß über zwei „Handshake"-Leitungen (Quittungsbetrieb) gesteuert. Jeder Port hat deshalb die Anschlüsse Ready (ARDY, BRDY) und Strobe ($\overline{\text{ASTB}}$, $\overline{\text{BSTB}}$).

Wenn ein Daten-Byte vollständig im Ausgaberegister ist, zeigt das Ready-Signal durch logisch 1 an, daß von außen auf die Daten zugegriffen werden kann. Wenn dagegen das Peripheriegerät die Daten übernommen hat, meldet es dies dem PIO-Baustein durch einen Strobe-Impuls (logisch 0). Dieser löst im PIO-IC ein Interrupt-Signal aus, das man an die CPU weiterleiten kann. War die CPU inzwischen mit anderen Aufgaben beschäftigt, bringt das Interrupt-Signal sie dazu, das nächste Datenbyte auszugeben.

Der Handshake-Betrieb ist für zeitkritische Programme und bei der Bedienung mehrerer Ein-/Ausgabe-Schnittstellen wichtig, da Peripheriegeräte oft doch wesentlich langsamer arbeiten als die CPU und diese somit übermäßig lange in Anspruch nehmen würden.

Im zeitunkritischen Betrieb darf man die Handshake-Leitungen außer Acht lassen. Auf ein Daten-Byte im Ausgaberegister läßt sich auch ohne Ready-Meldung zugreifen, und der Zeitpunkt des nächsten Datentransports wird ohne Strobe-Impuls (Interrupt) eben von der CPU bzw. vom steuernden Programm bestimmt. Wenn es trotzdem zu Störungen kommen sollte, muß man sich mit Warteschleifen bzw. PAUSE-Befehlen helfen, die allerdings die CPU voll beschäftigen.

● In der Betriebsart 1 kann entweder Port A oder Port B als Eingabekanal geschaltet werden. Ein Eingaberegister auf dem PIO-IC übernimmt dabei die Daten vom Peripheriegerät und die CPU liest sie aus diesem Register.

Auch hier kann man wieder mit den Handshake-Leitungen arbeiten: Ist das Eingaberegister leer – hat also die CPU die Daten übernommen –, wird die Ready-Meldung (logisch 1) ausgegeben. Das Peripheriegerät erkennt das und lädt die nächsten Daten ins Eingaberegister. Ein anschließender Strobe-Impuls kann wieder einen Interrupt bei der CPU bewirken, worauf diese das momentan laufende Programm unterbricht, um die neuen Daten „abzuholen".

Ebenso ist ein Betrieb ohne Handshake-Leitungen erlaubt. Man muß nur beachten, daß die CPU, wenn sie den Befehl zum Auslesen des Eingaberegisters erhält, das liest, was gerade darin steht; egal ob die Übertragung der Daten vom Peripheriegerät abgeschlossen ist oder nicht.

● In der Betriebsart 2 wird Port A für bidirektionalen Datenverkehr – also sowohl für Ein- als auch Ausgabe – benutzt. Port B ist dabei unwirksam und muß auf Betriebsart 3 (siehe unten) eingestellt werden. Zum Steuern bzw. Umschalten zwischen Ein- und Ausgabe werden die Handshake-Leitungen beider Ports verwendet.

Die Ausgabe läuft dann wie in Betriebsart 0 ab, nur muß der Eingang $\overline{\text{ASTB}}$ auf logisch 0 sein. Die Eingabe entspricht der in Betriebsart 1 mit der Ausnahme, daß zum Steuern der Dateneingabe über Port A die Handshake-Leitungen von Port B (BRDY und $\overline{\text{BSTB}}$) verwendet werden.

● Beide Ports lassen sich in der Betriebsart 3 benutzen. Hierbei kann jedes Bit eines Ports getrennt als Ein- oder Ausgabekanal geschaltet werden. Die Handshake-Leitungen spielen in dieser Betriebsart keine Rolle. Dafür wird eine Interrupt-Möglichkeit geboten: Ein Interrupt wird ausgelöst, wenn sich der Signal-Zustand an einem Eingabekanal ändert, also ein Bit von 1 nach 0 wechselt oder umgekehrt.

Ebenso kann man vorher festlegen, daß erst alle Bits (oder nur bestimmte, durch eine Maske definierte Bits) ihren Wert ändern müssen, bevor ein Interrupt ausgelöst wird. Das Interrupt-Signal kann die CPU selbstverständlich nur dann auswerten, wenn die $\overline{\text{INT}}$-Anschlüsse von PIO und CPU verbunden sind.

Dieser Interrupt ermöglicht es, z. B. externe Geräte auf ihren Betriebszustand hin zu überprüfen, ohne dabei die CPU zu belasten. Die Geräte melden sich sozusagen selbst über den PIO-Baustein, wenn z. B. irgend etwas nicht stimmt. Erst dann wird die CPU aktiv, indem sie in die für das entsprechende Gerät (durch einen Interrupt-Vektor) festgelegte Interrupt-Routine springt (mehr darüber später).

Man kann aber auch einen Port in Betriebsart 3 ohne Interrupt-Steuerung benutzen, wenn man z. B. nur ein Bit als Eingabekanal verwenden will; die anderen Bits werden dann als Ausgabekanal geschaltet und erscheinen beim Abfragen des Eingaberegisters immer mit dem logischen Wert 0.

So wird der PIO-Baustein programmiert

Mit Steuerworten unter den Adressen 250 bzw. FAh (für Port A) oder 251 bzw. FBh (für Port B) bringt man das PIO-IC in die gewünschte Betriebsart. Die Steuerworte werden dabei in den Registern des PIO-Bausteins zwischengespeichert, und von der Steuerlogik ausgewertet. In *Bild 5* sind alle zulässigen Steuerworte zusammengefaßt.

Da es außer für die Betriebsart noch andere Steuerworte gibt, wird eine Unterscheidung zwischen ihnen mit Hilfe der niederwertigen vier Bits erreicht. Es gibt aber auch Steuerworte, die unmittelbar auf ein anderes folgen müssen; dabei können alle acht Bits zur Geltung kommen.

Für die Betriebsarten 0, 1 und 2 benötigt man nur das entsprechende Betriebsarten-Steuerwort und, falls Interrupts erlaubt sind, einen Interrupt-Vektor. In diesen Vektor (Register) wird der niederwertige Adreßteil einer Interrupt-Routine geladen, die man bereits im Speicher des Computers abgelegt hat. Der höherwertige Teil wird vorher in das i-Register der CPU geladen.

Bei Auftreten eines Interrupts werden die beiden Adreßteile zusammengefügt und zeigen auf die Anfangsadresse der Interrupt-Routine, bei der das Programm dann fortfährt. Der Rücksprung aus dieser Routine wird mit dem Befehl *reti* erreicht. Danach ist über den PIO-Baustein wieder ein neuer Interrupt möglich.

Dank diesem Interrupt können mehrere PIO-Bausteine parallelgeschaltet sein; abhängig von den Interrupt-Vektoren wird dann bei einem Interrupt die Interrupt-Routine ausgeführt, die dem jeweiligen Baustein zugeordnet ist.

Für die Betriebsart 3 benötigt man neben dem Betriebsarten-Steuerwort und dem Interrupt-Vektor (falls Interrupts erlaubt sind) noch drei weitere Steuerworte:

○ Ein Ein-/Ausgaberegister-Steuerwort, mit dem festgelegt wird, welche Datenleitungen Eingang (logisch 1) und welche Ausgang (logisch 0) sind.

○ Da in der Betriebsart 3 die Handshake-Leitungen unbenutzt bleiben, und Interrupts durch Signaländerungen an den Ports erzeugt werden, ist ein Interrupt-Steuerwort nötig. Es besagt u. a., bei welchem Pegel (0 oder 1) ein Interrupt ausgelöst wird.

○ Sollen einige Bits vom Auslösen eines Interrupts ausgeschlossen werden, so muß man diese in einem folgenden Masken-Steuerwort angeben.

○ Soll ein Interrupt während des Programmablaufes gesperrt oder freigegeben werden, so kann dies mit dem Interrupt-Sperrungswort geschehen. Der Rest des Interrupt-Steuerwortes wird davon nicht verändert.

Die vollständige Programmierung des PIO-Bausteins ist nochmals im Zusammenhang in *Bild 6* dargestellt. Man wählt die gewünschte Betriebsart aus und erkennt, welche Steuerworte dann notwendig sind.

Wenn ein PIO-Baustein nicht genügt

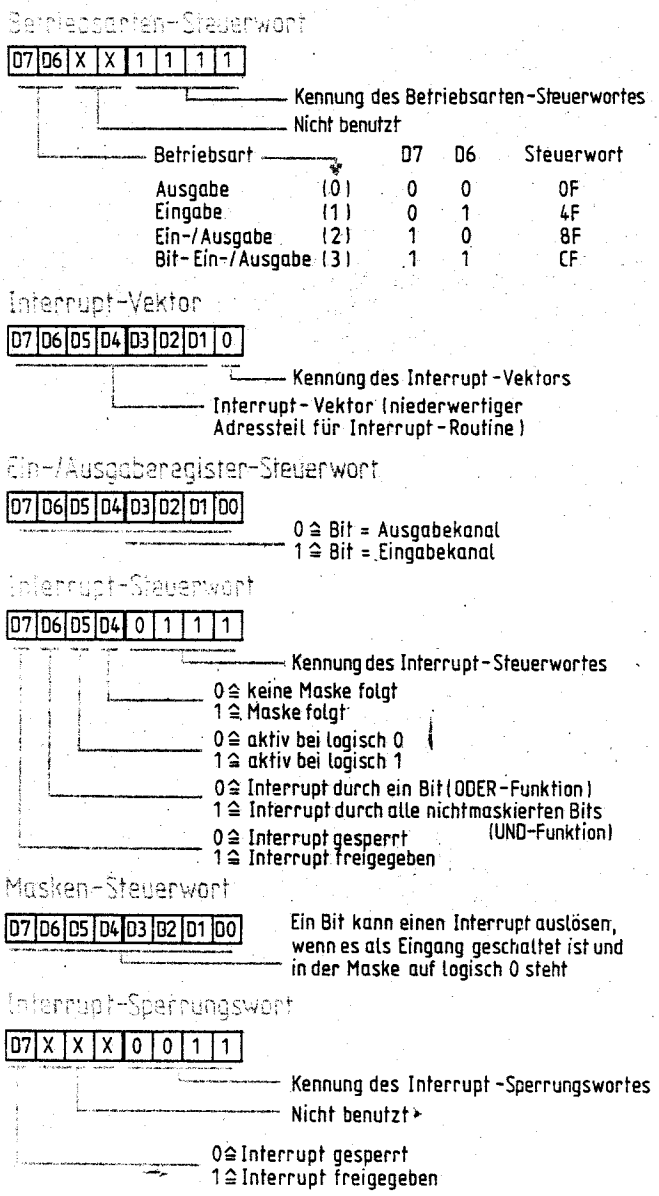
Mit den Signalen IEI (Interruptfreigabe-Eingang) und IEO (Interruptfreigabe-Ausgang) wird ein Interrupt überhaupt erst freigegeben, wenn IEI auf logisch 1 liegt. Beim Verwenden mehrerer PIO-Bausteine wird damit eine Rangfolge (Priorität) der Interrupts erreicht.

Dazu sind die Bausteine in einer „Daisy-chain"-Kette so zusammenzuschalten, daß jeweils ein IEO-Ausgang mit dem IEI-Eingang des nächsten Bausteins verbunden ist. Außerdem muß man alle $\overline{\text{INT}}$-Anschlüsse mit dem der CPU verbinden. Der erste IEI-Eingang liegt fest auf logisch 1 (+5 V). Dieser Baustein hat die höchste Priorität.

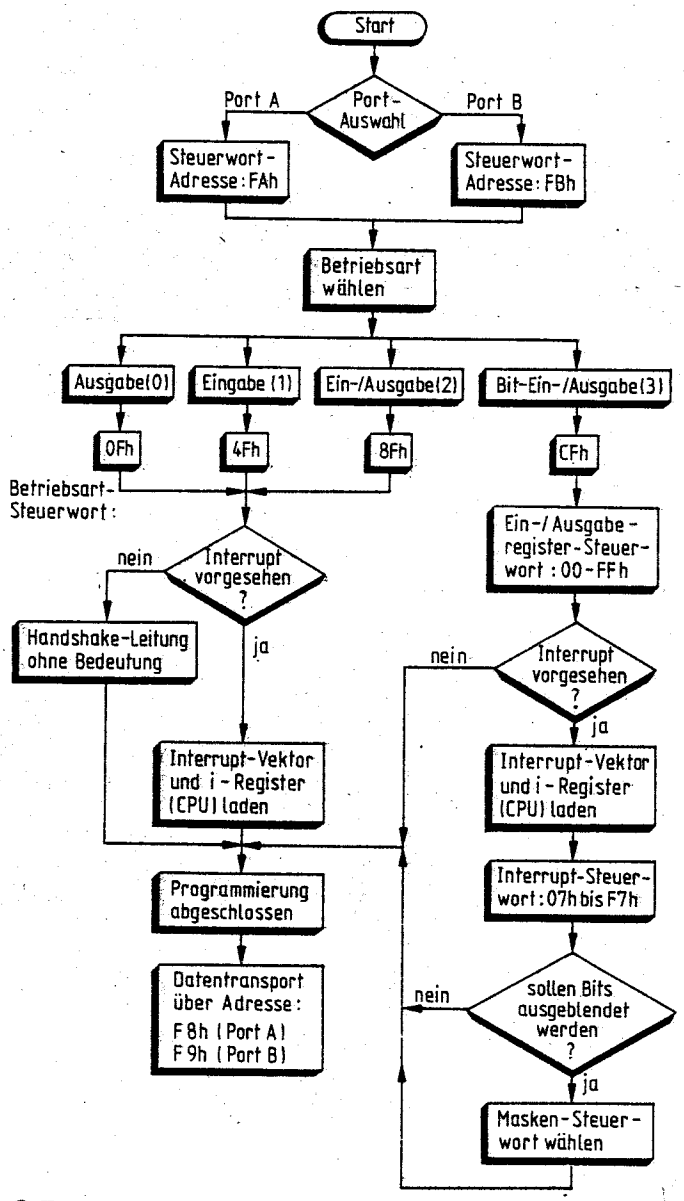
Der Baustein, der einen Interrupt auslöst, unterbricht die Kette und nimmt damit nachfolgenden Bausteinen niedrigerer Priorität die Interrupt-Möglichkeit. Hat man nur einen PIO-Baustein, ist der IEI-Anschluß auf +5 V zu legen, wenn mit der Interrupt-Steuerung gearbeitet werden soll.

Michael Schütz

(Wird fortgesetzt)



⑤ **PIO-Steuerworte:** Immer notwendig ist das Betriebsarten-Steuerwort. Die übrigen Steuerworte sind nur bei bitweiser Ein-/Ausgabe bzw. bei Interrupt von Bedeutung



⑥ **Ermitteln der Steuerworte:** Das Flußdiagramm führt über die Port- und Betriebsart-Anwahl zu den Steuerworten (Wertermittlung im rechten Zweig nach Bild 5)

ZX 81 à la carte (6):

Daten-Drehscheibe

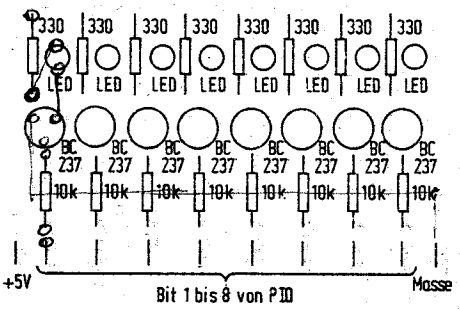
Teil 2: Die komplette Hardware

Beachten Sie den Bestückungsplan: Mehr Bauteile erfordert die Ein-/Ausgabe-Schnittstelle tatsächlich nicht.

Prinzipiell ist die Platine der Ein-/Ausgabe-Schnittstelle (*Bild 1, links*) an jeden Computer mit Z-80-CPU anschließbar, sofern man an die entsprechenden Signalleitungen herankommt (siehe Bild 3 in Teil 1). Außerdem ist darauf zu achten, daß die vier Adressen, unter denen man die Schnittstelle ansprechen kann, nicht auch vom Betriebssystem des Computers verwendet werden (z. B. Bildausgabe, Tastaturabfrage). Ein unkontrolliertes Aktivieren unserer Z-80-PIO wäre die Folge.

Die im Bestückungsplan des I/O-Ports (*Bild 1, rechts*) eingetragene Beschaltung des Adreßdecoders 74 LS 138 bewirkt, daß die Schnittstelle unter den Adressen F8h, F9h, FAh und FBh anzusprechen ist. Leider werden diese Adressen beim ZX 81 offenbar auch vom Sinclair-Logik-Chip ausgewertet, der außerdem noch die Interrupt-Steuerung verbietet. Dem Betrieb der Schnittstelle am ZX 81 sind daher Einschränkungen auferlegt:

○ Port B darf nur zur Eingabe benutzt werden.
○ Port A erlaubt die Ein- und Ausgabe, sollte aber zur Ausgabe benutzt werden.
○ Die Interrupt-Steuerung ist nicht zulässig; im Handshake-Betrieb kann nur die Ready-Leitung benutzt werden.



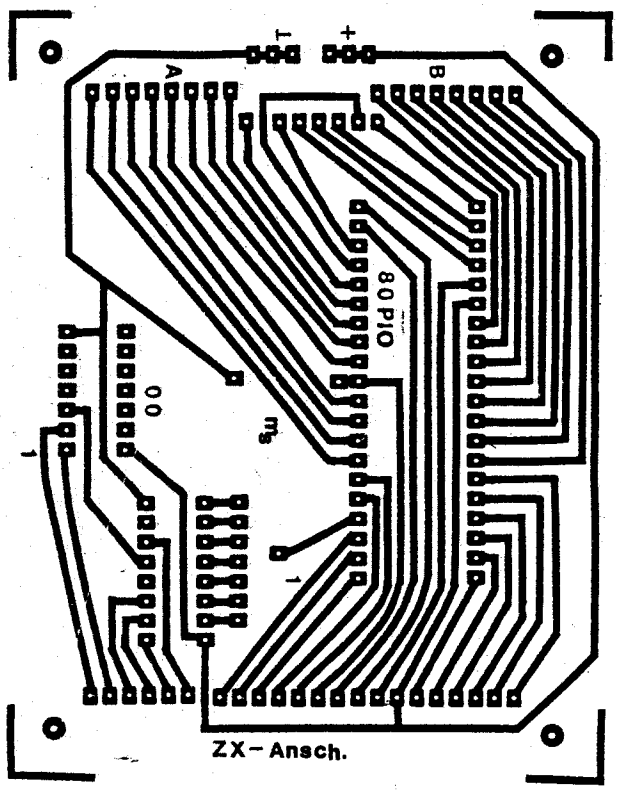
② **Anzeigeplatine:** Da sie so einfach ist, zeigen wir hier nur die Ansicht der Bestückungsseite (die Ätzvorlage wäre spiegelverkehrt dazu)

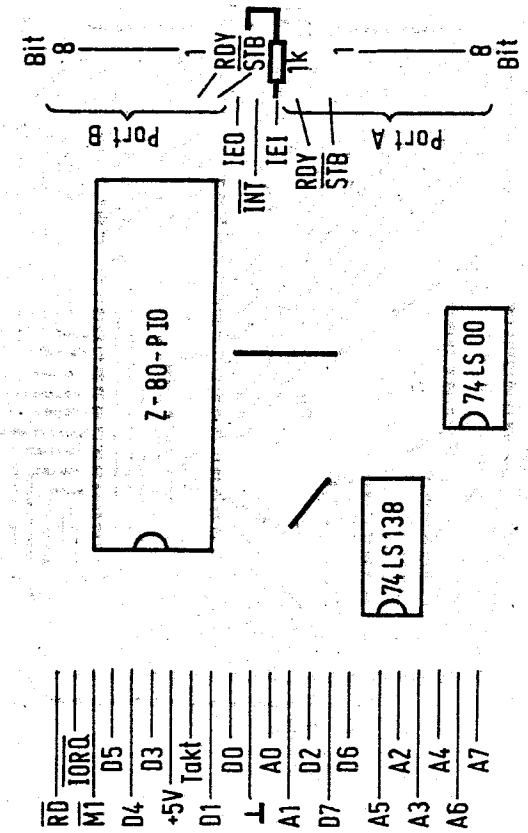
○ Der FAST-Modus ist nicht erlaubt.
○ Ist auch der Sinclair-Drucker angeschlossen, kann es bei der Datenausgabe zu einem Papiervorschub kommen.

Trotz dieser Einschränkungen läßt sich die Schnittstelle am ZX 81 immerhin noch gut verwenden, wie die Programmbeispiele im nächsten Teil zeigen werden. Nützlich ist auf jeden Fall die in *Bild 2* gezeigte Anzeigeplatine: An Port A angeschlossen (z. B. über Faltstecker) zeigt sie ausgegebene Daten optisch an.

Michael Schütz

(Schluß folgt)





① **Ein-/Ausgabe-Schnittstelle:** Die einseitig beschichtete Platine birgt keine Herstellungsprobleme. Rechts: Bestückungsplan und Anschlußanweisung (siehe auch Heft 12/1983, Seite 71)

ZX 81 à la carte (10):

Telefoncomputer

Außer der Basisplatine zur „à-la-carte"-Serie sind nur wenige Bauelemente nötig, um aus dem ZX 81 einen Telefoncomputer mit Kurzwahl und Wahlwiederholung zu machen.

Damit erst gar keine Mißverständnisse aufkommen: Die hier vorgestellte Schaltung darf nur an nicht amtsberechtigten Nebenstellenanlagen betrieben werden. Wer sie ans öffentliche Fernsprechnetz anschließt, der riskiert Ärger mit der Post!

Vieltelefonierern wird der ZX-81-Telefoncomputer gewiß eine Erleichterung bringen, denn er bietet außer der normalen Handwahl noch die Kurzwahl und Wahlwiederholung. Dieses Leistungsvermögen wird allein von der Software bestimmt. Zuerst wollen wir uns jedoch um die Hardware kümmern.

Keine Qual der Wahl

Angeschlossen wird die kleine Schaltung (*Bild 1*) an den Ausgabe-Port der Basisplatine aus Heft 12/13 1983. Selbstverständlich steht auch einem Anschluß an eine andere Parallel-Schnittstelle nichts im Wege (z. B. PIO-Port), doch muß dann im Steuerprogramm (*Bild 2*) die Port-Adresse (Zeile 10) entsprechend geändert werden. Wird der PIO-Port verwendet, ist außerdem ein Maschinenprogramm zur Datenausgabe hinzuzufügen (siehe auch Heft 5/84, Seite 70).

Von den acht Ausgängen des Ports benötigen wir nur die beiden niederwertigsten, nämlich D0 und D1. Über D0 werden die softwaremäßig erzeugten Zählimpulse ausgegeben, die über T1 das Wählrelais aktivieren. Weil dann das Telefon allerdings im Takt der Wählimpulse mitläuten würde, wird es während des Wählens mit Relais 2, das von D1 gesteuert wird, kurzgeschlossen.

Damit das Telefon auch bei abgeschaltetem Computer noch funktioniert, müssen die Relais Modelle mit Ruhekontakt sein. Deshalb sind Reedrelais nicht geeignet, Kammrelais auch nicht, da sie oft zu träge und überdies laut sind. Am besten verwendet man Miniaturrelais für Platinenmontage.

Software prägt unseren Telefoncomputer

Die Betriebsspannung der Relais sollte zwischen 5 V und 9 V liegen. Ausführungen mit geringer Leistungsaufnahme können unmittelbar vom Rechner gespeist werden (5 V oder 9 V), anderenfalls ist eine externe Spannungsversorgung notwendig.

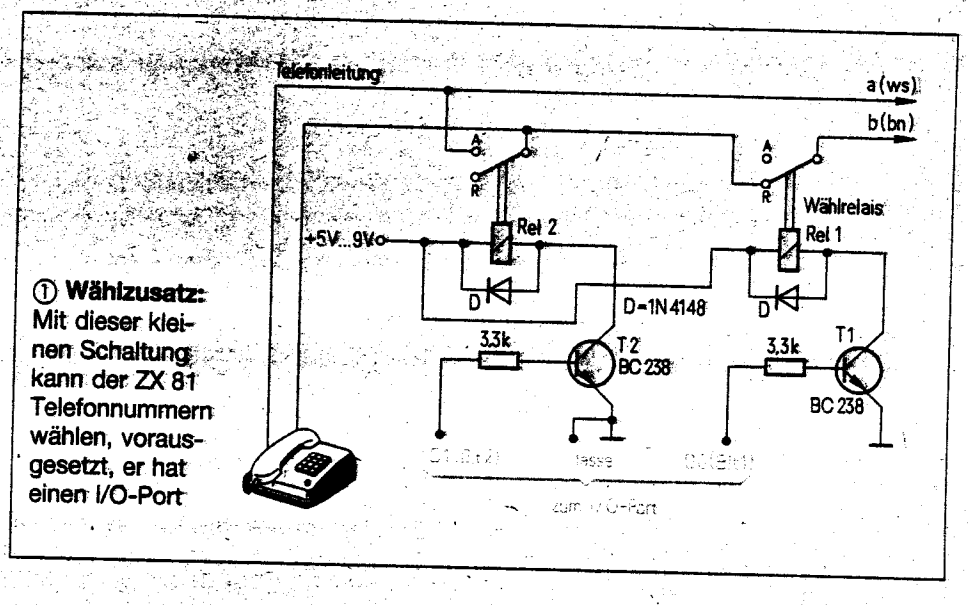
Das Programm, das den ZX 81 zum Telefoncomputer macht, beschert ihm alle die eingangs erwähnten Leistungsmerkmale, die auch kommerzielle Telefoncomputer bieten. Wenn man das Programm eingetippt und mit RUN gestartet hat, erscheint zuerst das von Zeile 100 und 110 vorgegebene „Menü" am Bildschirm.

Entscheidet man sich für die Handwahl, so lassen sich die Ziffern wie bei einem normalen Tastentelefon eintippen. Der Rechner beginnt jedoch erst nach der Bestätigung durch NEWLINE mit dem Wählen, so daß man Tippfehler ausbügeln kann, ohne fremden Leuten auf die Nerven zu gehen.

Die Wahlwiederholung wählt nach Drücken der Taste W die zuletzt gewählte Nummer noch einmal; sie ist also genau das Richtige bei chronisch besetzten Leitungen!

Die Kurzwahl schließlich bietet zugleich ein kleines Telefonregister: Nach Drücken der Taste K werden maximal 40 eingetragene Namen (jeder höchstens 12 Buchstaben) zur Auswahl angezeigt. Gibt man nun eine Nummer von 1 bis 40 ein (durch Eingabe von 0 kann man Einträge ändern oder neue vornehmen), so wird automatisch die zugehörige komplette Rufnummer gewählt.

Den guten Leistungsmerkmalen steht leider ein großer Nachteil gegenüber: Unser Telefoncomputer ist normalerweise auf ein Fernsehgerät angewiesen. Weniger schlimm ist, daß das Aufblättern des Telefonregisters selbst im FAST-Modus ziemlich lange dauert. Hat man jedoch bei der Bedienung des Geräts eine traumwandlerische Sicherheit erlangt, dann darf man durchaus das Fernsehgerät ausschalten, und die Eingaben „blind" vornehmen. In diesem Fall empfiehlt es sich, das Register abzu-



① Wählzusatz: Mit dieser kleinen Schaltung kann der ZX 81 Telefonnummern wählen, vorausgesetzt, er hat einen I/O-Port

schreiben bzw. mit COPY ausdrucken zu lassen. Durch Einfügen der Programmzeile 2005 GOTO 2070 ist es überdies möglich, den zeitraubenden Registeraufbau zu unterdrücken.

Mit den Programmzeilen 2087 bis 2100 werden Leerstellen nach denjenigen Ziffern entfernt, die im Kurzwahlregister mit weniger als 15 Ziffern gespeichert sind. Das Wählprogramm würde sonst mit einer Fehlermeldung reagieren. Die Zeilen 3000 bis 3130 ermöglichen Änderungen bzw. Neueinträge im Register.

Das eigentliche Wählprogramm beginnt mit Zeile 4000. Ab hier (bis Zeile 4300) werden die Wählimpulse mit PAUSE-Befehlen erzeugt und über die POKE-Befehle an den Port ausgegeben. POKE P,3 läßt beide Relais anziehen; Relais 2 bleibt dann während des gesamten Wählvorgangs aktiviert. Nur Relais 1 reagiert auf POKE P,2 indem es abfällt. Der PAUSE-Befehl in Zeile 4120 bestimmt die Dauer der einzelnen Wählimpulse; die Pause zwischen den Wählimpulsen wird mit Zeile 4150 festgelegt. Die von Zeile 4250 bestimmte Wartezeit ist die Pause zwischen den einzelnen Ziffern einer gewählten Nummer.

Die Dauer der Pausen in Zeile 4120 und 4150 wurde experimentell ermittelt. Die Werte sind nicht ganz normgerecht. Als Orientierung für eigene Experimente gilt: Die Öffnungsdauer des Wählrelais sollte je Impuls zwischen 51 ms und 72 ms liegen, die Schließungsdauer zwischen 31 ms und 48 ms. Das zulässige Tastverhältnis reicht damit von 1,3 : 1 bis 1,9 : 1. Eventuell sind die Werte der beiden PAUSE-Befehle geringfügig zu ändern, damit Verbindungen fehlerfrei zustandekommen. Das Flackern des Bildes während der Ausführung der PAUSE-Befehle ist zugleich eine optische Kontrolle für den Programmablauf.

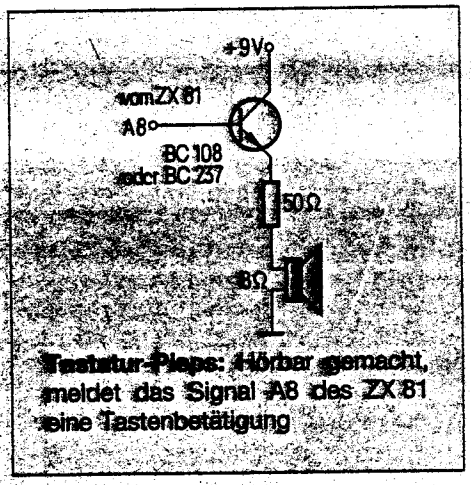
Ist ein Register aufgebaut worden, so läßt es sich durch Betätigen der Taste S auf Kassette speichern; der dazugehörige Programmteil beginnt mit Zeile 9900. Der letzte Befehl dort lautet GOTO 100. Er bewirkt einen automatischen Start des Programms nach dem Laden von Kassette. Wird das Programm versehentlich – oder um Änderungen vorzunehmen – unterbrochen, so darf es nur mit GOTO 100 erneut gestartet werden. Jeder Start mit RUN oder GOTO 10 löscht sämtliche Einträge (Variablen) im Register.

Wolf-Dieter Roth/-ll

ZX-81-Hardwaretip:

Tastatur-Pieps

Die im *Bild* angegebene einfache Schaltung erlaubt eine akustische Kontrolle des ZX 81. Das an der Adreßleitung A8 anliegende Signal wird einfach abgegriffen, verstärkt und einem Kleinlautsprecher zugeführt. Die Adreßleitung kann auch hinter der Diode (im ZX-Schaltplan D6) an der Tastatur angezapft werden, was sich als praktisch beim Einbau in eine Zusatztastatur erweist. Die Schaltung kann vom ZX-Netzteil mitversorgt werden; Masse und + 9 V lassen sich am Spannungsregler abgreifen.



Tastatur-Pieps: Hörbar gemacht, meldet das Signal A8 des ZX 81 eine Tastenbetätigung

Jeder Tastendruck ergibt ein akustisches Echo, sobald der Computer die Eingabe erkannt hat; daher kann man nun auf der Folientastatur etwas eintasten, ohne dabei immer wieder auf den Bildschirm schauen zu müssen. Basic- und Maschinensprache-Programme verursachen typische Geräusche, so daß man auch hören kann, ob ein Programm abgestürzt ist. Der Haken an dieser „Minimallösung" ist, daß durch das Signal auf A8 der Lautsprecher ein ständiges Geräusch erzeugt, aus dem ein Tastendruck lediglich als kurzes „Plop" herauszuhören ist – aber immerhin...

Michael Schramm

```
  10 LET P=23555
  20 DIM V$(40,12)
  30 DIM D$(40,15)
 100 PRINT "H = HANDWAHL";,,,"K
= KURZWAHL",,,,"U = WIEDERHOLUNG
";,,,"B = BEENDEN (OHNE SPEICHERN
);,,,"S = SPEICHERN"
 110 PRINT AT 15,0;"BITTE ENTSPR
ECHENDE TASTE       DRUECKEN"
 120 IF INKEY$="H" THEN GOTO 100
0
 130 IF INKEY$="B" THEN STOP
 140 IF INKEY$="S" THEN GOTO 990
0
 150 IF INKEY$="K" THEN GOTO 200
0
 160 IF INKEY$="U" THEN GOTO 150
0
 200 GOTO 120
1000 CLS
1020 PRINT "GEBEN SIE DIE NUMMER
 OHNE       ZWISCHENRAUM EIN"
1030 INPUT N$
1040 GOSUB 4000
1050 CLS
1060 GOTO 100
1500 CLS
1503 PRINT AT 5,3;"ICH WAEHLE  "
;N$
1505 GOSUB 4000
1510 GOTO 100
2000 CLS
2010 FAST
2020 PRINT " 0 AENDERN"
2030 FOR F=1 TO 40
2040 IF F<10 THEN PRINT " ";F;"
";
2050 IF F>=10 THEN PRINT F;" ";
2060 PRINT V$(F),
2065 NEXT F
2066 SLOW
2070 INPUT F$
2080 IF F$="0" THEN GOTO 3000
2085 CLS
2087 LET N$=" "
2090 LET L$=D$(VAL F$)
2092 LET S=LEN L$
2094 IF L$(S)=" " THEN LET L$=L$
(1 TO S-1)
2096 LET S=S-1
2098 IF L$(S)=" " THEN GOTO 2094
2099 LET N$=L$
2100 PRINT "ICH WAEHLE NR. ";N$;
"TEILNEHMER ";F$;,,,V$(VAL F$)
;
2110 GOSUB 4000
2120 GOTO 100
3000 PRINT AT 0,0;"WELCHEN TEILN
EHMER (NR. 1-40)?"
3010 INPUT H$
3020 CLS
3030 PRINT "TEILNEHMER ";H$,,,,"
NR.";D$(VAL H$),,,V$(VAL H$)
3040 PRINT
3050 PRINT "NEUE NUMMER?"
3060 INPUT J$
3070 LET D$(VAL H$)=J$
3080 PRINT
3090 PRINT "NEUER NAME?"
3100 INPUT K$
3110 LET V$(VAL H$)=K$
3120 CLS
3130 GOTO 100
4000 POKE P,2
4010 PAUSE 25
4020 FOR L=1 TO LEN N$
4030 LET Z$=N$(L)
4040 IF Z$="0" THEN LET Y$="10"
4050 IF NOT Z$="0" THEN LET Y$=Z
$
4100 FOR Y=1 TO VAL (Y$)
4110 POKE P,3
4120 PAUSE 1.9
4140 POKE P,2
4150 PAUSE 1
4200 NEXT Y
4250 PAUSE 25
4300 NEXT L
4310 POKE P,0
4330 PAUSE 25
4400 CLS
4410 RETURN
9900 CLS
9905 PRINT "SCHALTEN SIE DEN REC
ORDER AUF   AUFNAHME"
9910 PAUSE 200
9920 CLS
9930 PRINT "O.K.? DANN DRUECKE "
"NEWLINE""
9935 IF INKEY$="" THEN GOTO 9935
9940 CLS
9950 SAVE "TELEFONWAEHLAUTOMA■"
9960 GOTO 100
```

② **Steuersoftware:** Dieses Programm ist auf die Basisplatine der „à-la-carte-Serie" zugeschnitten

ZX-81-Softwaretip:

Schloß und Schlüssel

Der Programmschutz und dessen Aufhebung

Der Programmschutz ist ein zweischneidiges Schwert: einerseits soll er Programme dem fremden Zugriff entziehen, doch weckt gerade das bei vielen den Ehrgeiz, so geschützte Programme zu knacken. Wir beleuchten für Kenner des ZX 81 beide Aspekte.

Geht es nur darum, die einem Programm zugrunde liegende Idee oder bestimmte Informationen im Programm geheimzuhalten, so genügt es, eine LIST-Sperre zu errichten. Um ihre Wirkungsweise zu verstehen, muß man sich zunächst klarmachen, in welcher Form eine Basic-Programmzeile im Speicher abgelegt wird (*Bild 1*).

Schutz der Programmidee durch LIST-Sperre

Die ersten beiden Bytes enthalten den Wert der Zeilennummer (siehe auch Heft 10/1983, Seite 71). Im Gegensatz zur üblichen 16-Bit-Zahlendarstellung folgt hier das Byte mit niederem Stellenwert (LSB: Last significant Byte) dem Byte mit höherem Stellenwert (MSB: Most significant Byte). Die nächsten beide Bytes geben in üblicher Form an (LSB zuerst), wieviele Zeichen in der Zeile untergebracht sind. Darauf folgt der eigentliche Zeileninhalt.

Wichtig ist, daß beim Wert der Zeilenlänge Basic-Schlüsselwörter nur als ein Zeichen zählen. Tatsächlich speichert der ZX 81 alle Schlüsselworte unabhängig von ihrer wahren Zeichenzahl im Programmspeicher nur mit einem Byte (z. B. würde PRINT normalerweise fünf Byte erfordern). In dieser verkürzten Form gespeicherte Schlüsselworte heißen Token.

Am Ende der Zeile steht grundsätzlich ein NEWLINE-Code (118). Daher beträgt die Zeilenlänge immer mindestens zwei Byte, nämlich ein Byte für das Basic-Schlüsselwort und ein Byte für den Zeilenabschluß.

Wenn man den Befehl LIST eingibt, wird eine ROM-Routine aufgerufen, die folgendermaßen vorgeht: Von jeder Zeile wird zuerst die Zeilennummer am Bildschirm ausgegeben, die Information über die Zeilenlänge ignoriert und dann der eigentliche Zeileninhalt geschrieben. Hierbei werden die einzelnen Zeichen (Bytes) der Reihe nach ausgegeben, bis die Routine auf ein Byte mit dem Inhalt 118 stößt, denn dieses Byte gibt ja das Zeilenende an und taucht (normalerweise) sonst nirgends in der Zeile auf.

Der Adreßzeiger wird dann auf das nächste Byte gesetzt; das sollte der Anfang einer neuen Zeile sein, nämlich das erste Byte der nächsten Zeilennummer. Die LIST-Routine stoppt, sobald der Inhalt dieses Bytes größer als 73 ist. In diesem Fall erkennt die Routine, daß an der erreichten Adresse der unmittelbar auf den Programmspeicher folgende Bildspeicher beginnt (dieser wird durch ein Byte mit dem Wert 118 eingeleitet) und deshalb das Programm vollständig gelistet wurde.

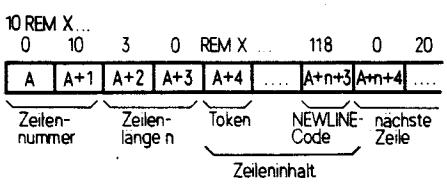
Die LIST-Sperre wird nun folgendermaßen errichtet: Zunächst gibt man vor dem zu schützenden Programm die Zeile 1 REM X ein. Das X muß anschließend durch ein Byte mit Inhalt 118 ersetzt werden, was durch den Befehl POKE 16514,118 möglich ist. Wenn die LIST-Routine auf diese Zeile stößt, „glaubt" sie, die Zeile ende schon mit diesem 118er-Byte.

Das nächste Byte wird dann irrtümlich als Zeilennummer der folgenden Zeile interpretiert. Dieses Byte enthält aber das echte Zeilenende-Zeichen, also ebenfalls den Code 118, und prompt stoppt die LIST-Routine (da 118 größer als 73 ist). Am Bildschirm ist deshalb nur die Zeile 1 zu sehen. Der Programmablauf wird von der LIST-Sperre nicht gestört, denn der Basic-Interpreter errechnet die Anfangsadresse jeder Folgezeile über die gespeicherte Zeilenlänge (siehe Bild 1).

Die Sperre wird höher

Noch ein Stückchen sicherer wird die LIST-Sperre, wenn man sie nicht nur in der ersten Programmzeile, sondern an mehreren Stellen im Programm errichtet. Warum? Das sollten Sie selber herausfinden! Schwierig ist dann allerdings

① **Zeilenorganisation:** So ist z. B. die Basic-Zeile 10 REM X... gespeichert. Ist die Zeile die erste, hat die Anfangsadresse A den Wert 16509



② **Hilfsprogramm:** Stehen irgendwo in einem Programm LIST-Sperren der Form REM X, meldet diese Routine die Adressen der X-Zeichen

```
9900 LET A=16509
9910 IF PEEK A=118 THEN STOP
9920 IF PEEK (A+5)<>61 THEN GOTO
  9940
9930 PRINT 256*PEEK A+PEEK (A+1)
;":";A+5;
9940 LET A=A+PEEK (A+2)+256*PEEK
  (A+3)+4
9950 GOTO 9910
```

```
   10 PRINT "ALLES KLAR"
   20 STOP
 9500 FAST
 9510 LET D=PEEK 16396+117
 9520 LET E=PEEK 16397+34
 9530 POKE PEEK 16396+256*PEEK 16
397,70
 9540 POKE 16396,14
 9550 POKE 16397,71
 9560 POKE 16450,PEEK 16400
 9570 POKE 16454,PEEK 16401
 9580 POKE 16400,15
 9590 POKE 16401,99
 9600 SAVE "TES■"
 9610 POKE 16401,PEEK 16454
 9620 POKE 16400,PEEK 16450
 9630 LET P$=""
 9640 LET A$=INKEY$
 9650 IF A$<>"" THEN GOTO 9640
 9660 LET A$=INKEY$
 9670 IF A$="" THEN GOTO 9660
 9680 IF CODE A$=118 THEN GOTO 97
10
 9690 LET P$=P$+A$
 9700 GOTO 9640
 9710 IF P$<>"FUNKSCHAU" THEN GOT
O 9630
 9720 POKE 16397,E-34
 9730 POKE 16396,D-117
 9740 POKE PEEK 16396+256*PEEK 16
397,118
 9750 SLOW
 9760 GOTO 10
```

③ **Codewort-Schutz:** Nur wer das Codewort FUNKSCHAU kennt, kann dieses Programm nach dem Laden listen

```
   1 REM HIER MINDESTENS 69 BE-
LIEBIGE ZEICHEN EINTIPPEN
 100 RAND USR 16514
 102 DIM A$(19)
 103 LET A$="VERSUCHEN SIE BREAK
"
 104 PRINT AT 5,3;"DAS IST EIN B
ASIC-PROGRAMM"
 105 FOR I=1 TO 19
 107 PRINT AT 10,5+I;A$(I);
 108 FOR T=1 TO 10
 109 NEXT T
 110 NEXT I
 125 CLS
 130 GOTO 104
 200 SAVE "DEMO"
 210 GOTO 100
```

④ **BREAK-Schutz:** Das ist ein Basic-Programm und läßt sich dennoch nicht stoppen!

16514:	237	16515:	123
16516:	2	16517:	64
16518:	225	16519:	253
16520:	203	16521:	1
16522:	142	16523:	42
16524:	20	16525:	64
16526:	34	16527:	26
16528:	64	16529:	34
16530:	28	16531:	64
16532:	253	16533:	203
16534:	45	16535:	174
16536:	42	16537:	10
16538:	64	16539:	253
16540:	203	16541:	0
16542:	126	16543:	40
16544:	30	16545:	42
16546:	41	16547:	64
16548:	62	16549:	192
16550:	156	16551:	194
16552:	174	16553:	6
16554:	66	16555:	35
16556:	94	16557:	237
16558:	83	16559:	7
16560:	64	16561:	35
16562:	94	16563:	35
16564:	86	16565:	35
16566:	235	16567:	25
16568:	34	16569:	41
16570:	64	16571:	235
16572:	34	16573:	10
16574:	64	16575:	205
16576:	77	16577:	0
16578:	205	16579:	193
16580:	12	16581:	24
16582:	192		

⑤ **Maschinencode für BREAK-Schutz:** Das sind die Dezimalcodes des eigentlichen BREAK-Schutzes

das Aufspüren der richtigen POKE-Adressen, denn nur die Adresse 16514 gilt für jedes Programm. Die Adressen für das Zeichen X in den übrigen LIST-Sperren hängen nämlich von der Länge aller vorhergehenden Zeilen ab. Hier hilft das in *Bild 2* gezeigte Basic-Programm, das an das zu bearbeitende Programm angehängt und später wieder gelöscht wird. Es meldet für alle Zeilen der Form REM X die richtige POKE-Adresse.

Wenn das Programm stoppt, weil der Bildschirm voll ist, kann der Programmablauf mit CONT fortgesetzt werden. Die Arbeitsweise des Prgrämmchens wird sofort klar, wenn man den in Bild 1 gezeigten Zeilenaufbau betrachtet.

In dieser Form weist das Verfahren freilich noch einen entscheidenden Nachteil auf: die mühsam erzeugten Sperr-Zeilen können wie jede andere Zeile durch bloßes Eingeben der Zeilennummer wieder gelöscht werden. Das ändert sich sofort, wenn man ihre beiden Zeilennummer-Bytes einfach mit POKE auf den Wert 0 setzt. Das ergibt Zeilen mit der Nummer 0, die sich nicht mehr löschen lassen, den Programmablauf aber nicht beeinträchtigen.

Dennoch läßt sich das Programm immer noch listen, nämlich stückchenweise, indem man den LIST-Befehl mit Zeilennummer benutzt (z. B. LIST 11, wenn Zeile 10 eine LIST-Sperre enthält). Oder indem man das Programm aus Bild 2 geringfügig verändert einsetzt, um die Null-Zeilennummer wieder auf einen „vernünftigen" Wert zu setzen, und dann die Zeilen löscht.

Erfahrungsgemäß kommen jedoch die meisten ZX-81-Benutzer nicht auf diese Idee – erst recht nicht, wenn der LIST-Schutz nicht so offensichtlich ist wie bei scheinbar überflüssigen REM-Zeilen. Man kann jedoch jede Programmzeile so erweitern, daß sie zusätzlich zu ihrer normalen Funktion die Aufgabe einer LIST-Sperre übernimmt!

Hierzu ein Beispiel: Geben Sie die Zeile 110 LET C=3 ein.
Daraus macht man 110 LET C=3+X.
Dann lädt man in die Speicherzellen, die das Pluszeichen und das X enthalten, jeweils die Zahl 118. Achtung: Hinter jeder numerischen Konstanten im Programm, hier also 3, steht die sechs Bytes lange, im Listing unsichtbare interne Zahlendarstellung durch den ZX 81; also bei den Adressen nicht verzählen! Die Zeile wird dann wieder wie ursprünglich interpretiert, wirkt aber gleichzeitig als LIST-Sperre – darauf soll erst einmal jemand kommen!

Unsichtbarer GOTO-Sprung

Es gibt noch eine weitere Möglichkeit, per Eingriff in eine Basic-Zeile Verwirrung zu stiften: Man verändert die Angabe der Zeilenlänge n (siehe Bild 1). Da der Interpreter die Adresse der als nächstes auszuführenden Programmzeile nach der Formel A+n+4 berechnet, kann man dadurch die Reihenfolge verändern, in der die Zeilen abgearbeitet werden.

Wenn A die Anfangsadresse der Zeile ist, deren Zeilenlänge verändert werden soll, und A1 die Anfangsadresse der Zeile ist, die als nächste ausgeführt werden soll, gilt für den neuen Wert der Zeilenlänge die Formel n = A1 − A − 4. Falls das Ergebnis dieser Formel negativ ist, muß 65 536 dazuaddiert werden; das ist dann der Fall, wenn zurückgesprungen werden soll.

Zeilen, in denen die Zeilenlänge verändert worden ist, dürfen nicht mehr verändert werden; sonst werden ganze Blöcke im Programm gelöscht, und es besteht die Gefahr eines Ausstiegs.

Der „Autostart" fällt mit der Tür ins Haus

Viele im Handel erhältliche ZX-81-Programme nutzen eine sehr einfache Methode, um einen Kopierschutz zu erreichen: Gespeichert wird das Programm von der Software-Firma durch ein SAVE-Kommando im Basic-Programm. Nachdem das Programm vom Benutzer geladen worden ist, wird deshalb automatisch die Programmausführung mit dem Befehl fortgesetzt, der dem SAVE-Kommando folgt. Und dort steht ein Befehl, der ein Maschinenprogramm aufruft, das sich nicht stoppen läßt.

BREAK und anschließendes Listen ist dann nicht möglich. Falls keine weiteren Schutzmaßnahmen getroffen wurden, läßt sich der „Autostart" jedoch einfach mit einem kleinen Hilfsprogramm unterbinden (siehe FUNKSCHAU 16/1983, Seite 71, und Heft 20/1983, Seite 6).

Aber auch ohne Kenntnisse in Maschinensprache läßt sich ein Kopierschutz erreichen, der sogar sicherer als der Autostart ist. Man muß sich nur zunutze machen, daß der ZX 81 im FAST-

und im SLOW-Modus arbeiten kann. Der FAST-Modus bewirkt, daß kein Bild aufgebaut wird. In diesem Fall unterläßt der Computer jeden Zugriff auf den Bildspeicher (außer ein PRINT-Befehl verlangt dies), und man kann gefahrlos Manipulationen vornehmen, die mit dem Bildpeicher zusammenhängen.

Solche Manipulationen (Verändern der Systemvariable D-FILE, die die Anfangsadresse des Bildspeichers angibt; Schreiben unzulässiger Werte in den Bildspeicher) bewirken im SLOW-Modus, oder sobald im FAST-Modus ein Bild aufgebaut werden muß, einen Systemabsturz. Darauf beruht das folgende Verfahren, das eine Programmbenutzung nur zuläßt, falls ein geheimes Codewort (ein „Password") bekannt ist.

An das zu schützende Programm hängt man dazu die in *Bild 3* gezeigten Zeilen (9500 bis 9760) an. Auf Band gespeichert wird das komplette Programm dann durch GOTO 9500. Hierbei geschieht folgendes: Zunächst wird der FAST-Modus gewählt und in den Variablen D und E der Inhalt der Systemvariable D-FILE (Adressen 16396 und 16397) gespeichert. Die Zahlen 117 und 34, die addiert werden, dienen nur der weiteren Verschleierung.

In Zeile 9540 und 9550 werden willkürliche Zahlen in D-FILE gebracht. Anschließend wird auch noch die Systemvariable VARS (Adressen 16400 und 16401), die den Beginn des Variablenspeichers angibt, „verdreht", nachdem ihr richtiger Inhalt in zwei Bytes des Drucker-Pufferspeichers abgelegt wurde. Hier sind dafür willkürlich die Adressen 16450 und 16454 gewählt worden, es sind jedoch beliebige Adressen im Bereich von 16444 bis 16476 möglich.

Nun wird das Programm auf Band gespeichert (Zeile 9600). Nach dem Laden mit dem LOAD-Befehl fährt das Programm automatisch in Zeile 9610 fort.

Ein Überlisten dieses Autostarts ist jetzt sinnlos, denn sobald der Computer im gegenwärtigen Zustand versucht, ein Bild aufzubauen, stürzt das System wegen der verdrehten Systemvariablen ab. Erst durch die Zeilen 9610 und 9620 wird die Systemvariable VARS wiederhergestellt (der Inhalt des Druck-Puffers wurde durch den SAVE-Befehl mit aufgezeichnet). Die Zeilen 9630 bis 9710 verlangen jedoch noch die Eingabe eines Codewortes, hier z. B. FUNKSCHAU. Das Wort wird langsam ohne Bildkontrolle eingetippt (nach dem Laden ist der Bildschirm dunkel!) und mit NEWLINE abgeschlossen.

Wenn man sich verschrieben hat, drückt man NEWLINE und wiederholt die Eingabe. Keinesfalls ist SPACE (BREAK) zu drücken, da sonst das System zusammenbricht. D-FILE hat zu diesem Zeitpunkt ja noch einen unzulässigen Wert. Daher darf das Codewort auch kein Leerzeichen (SPACE) enthalten. Erst, nachdem das richtige Codewort erkannt wurde, wird D-FILE wiederhergestellt. Das ist möglich, weil durch SAVE auch die Variablen D und E mit auf Band gelangt sind. Schließlich wählt das Programm den SLOW-Modus und leitet den Sprung zum geschützten Hauptprogramm ein.

Um ein auf diese Weise geschütztes Programm kopieren zu können, muß man entweder das Codewort kennen, direkt von Kassette zu Kassette kopieren oder ein spezielles Kopierprogramm benutzen, das ein Programm lädt und sofort wieder abspeichert, ohne zwischendurch ein Bild aufzubauen. Ein derartiges Kopierprogramm wird später noch vorgestellt. Das bloße Kopieren ist allerdings sinnlos, wenn das Codewort nicht bekannt ist.

Jetzt wird die BREAK-Taste blockiert

Mit einem kleinen Trick ist es zu schaffen, daß sich ein Basic-Programm (!) nicht mehr anhalten läßt. Der Trick besteht darin, daß die Basic-Interpreter-Kontrollroutine im ROM durch eine eigene, veränderte Routine im RAM ersetzt wird. *Bild 4* zeigt hierzu ein Demonstrations-Programm: Hinter dem REM-Schlüsselwort in Zeile 1 müssen 69 beliebige Buchstaben oder Ziffern eingetastet werden, um Platz für den Maschinencode aus *Bild 5* zu reservieren. Dessen Dezimalwerte sind einfach mit dem Hilfsprogramm

```
7 FOR I = 16514 TO 16582
8 INPUT B
9 POKE I, B
10 NEXT I
11 STOP
```

einzugeben. Dann darf das Hilfsprogramm gelöscht und das Hauptprogramm mit GOTO 200 gespeichert werden.

Durch den USR-Aufruf in Zeile 100 wird die neue Interpreter-Routine aktiviert. Von nun an erfolgt die Abarbeitung der Basic-Zeilen über diese Routine. Sie ist fast identisch mit der Original-ROM-Routine, hat aber die folgenden Besonderheiten:

○ Es wird nicht mehr nach jeder Zeileninterpretation abgefragt, ob die BREAK-Taste gedrückt ist. Ein Programmabbruch ist daher nicht möglich.

○ Falls während der Abarbeitung einer Zeile ein Fehler auftritt, wird die Zeile nochmals ausgeführt. Wird also beispielsweise während der Ausführung eines SAVE-Kommandos die BREAK-Taste gedrückt, so wird das Speichern abgebrochen, aber anschließend wiederholt. Wenn im Programm ein Fehler steckt (etwa eine Division durch Null), so wird die betreffende Zeile immer und immer wieder ausgeführt – man erhält eine Endlosschleife. Auch in diesem Fall ist ein unerwünschter Programmabbruch ausgeschlossen. Um jedoch eine derartige Situation von vornherein zu vermeiden, muß sichergestellt sein, daß das Programm absolut fehlerfrei ist.

○ Erst wenn das Programm endet (letzte Programmzeile abgearbeitet, kein STOP-Befehl!), wird wieder der Interpreter im ROM wirksam und damit auch die BREAK-Taste.

Die Routine ist im Speicher frei verschiebbar, das heißt relokatibel. Der Maschinencode kann also auch an eine andere Stelle im Speicher geschrieben werden; im Basic-Programm muß dann nur die Startadresse angepaßt werden. Nachdem die neue Interpreter-Routine aufgerufen worden ist, läßt sich D-FILE wie beschrieben regenerieren und jede weitere Schutzmaßnahme aufheben.

Die eigene Interpreter-Routine hat aber leider einen kleinen Schönheitsfeh-

```
  9 REM ALPHANUMERISCHE EINGABE
 10 PRINT AT PX,PY;"■";
 20 FOR O=1 TO L0
 30 PRINT " ";
 40 NEXT O
 50 LET E$=""
 60 PRINT AT PX,PY;
 70 LET O$=INKEY$
 80 IF O$<>"" THEN GOTO 70
 90 LET O$=INKEY$
100 IF O$="" THEN GOTO 90
110 IF O$=CHR$ 118 THEN RETURN
120 IF O$=CHR$ 119 THEN GOTO 10
130 IF LEN E$=L0 OR O$>CHR$ 63
THEN GOTO 70
140 PRINT O$;
150 LET E$=E$+O$
160 GOTO 70
169 REM **** NUMERISCHE EINGABE
170 GOSUB 10
180 LET O1=0
190 FOR O=1 TO LEN E$
200 LET O$=E$(O)
210 IF O$="." THEN LET O1=O1+1
220 IF O$="-" AND O>1 OR O$<>"-
" AND O$<>"." AND (O$<"0" OR O$>
"9") THEN GOTO 170
230 NEXT O
240 IF O1>1 OR E$="-" THEN GOTO
170
250 LET E=VAL E$
260 RETURN
```

⑥ **INPUT-Ersatz:** Die beiden Unterprogramme (Zeile 9 und 169) machen den BREAK-Schutz noch wirksamer

ler: Der INPUT-Befehl darf nicht mehr verwendet werden, denn dieser wird vom Computer in besonderer Weise gehandhabt und schaltet automatisch die ROM-Interpreter-Routine ein. So kann etwa durch die Eingabe von STOP während eines INPUT's das Programm angehalten werden.

Nachhilfe für die BREAK-Blockade

Der beste Ersatz für INPUT wäre eine Maschinenroutine, die komfortable Eingabemöglichkeiten bietet (Eingabe an beliebiger Bildschirmposition, blinkender Cursor, Repeat-Funktion usw.). Die Beschreibung einer derartigen Routine würde jedoch den Rahmen dieses Beitrags sprengen. Daher wird in *Bild 6* eine Basic-Lösung (Unterprogramm) gezeigt.

Es wird zwischen alphanumerischer (GOSUB 10) und numerischer Eingabe (GOSUB 170) unterschieden. Das Ergebnis der Eingabe erhält man in E$, bei numerischer Eingabe außerdem in E. Vor dem Aufruf eines dieser Unterprogramme müssen drei Parametern Werte zugewiesen werden: PX und PY erhalten die Bildschirmposition, an der die Eingabe beginnt (PX die Zeilen- und PY die Spaltenkomponente); L0 gibt die maximal zulässige Länge der Eingabe an.

Die beiden Unterprogramme stehen am besten ganz am Anfang eines Programms (kleine Zeilennummern), weil sie dann besonders schnell abgearbeitet werden. Das ist wichtig, um eine fließende Eingabe ohne Wartezeiten zu ermöglichen. Die Eingabe-Routine ist damit etwa gleichschnell wie der INPUT-Befehl.

Zur Korrektur einer Eingabe besteht nur die Möglichkeit, durch RUBOUT den gesamten getippten Text zu löschen. Eine als unzulässig erkannte Eingabe wird zurückgewiesen; durch eine Falscheingabe kann kein Programmabbruch bzw. eine Error-Endlosschleife hervorgerufen werden. Man muß nur darauf achten, daß die Parameter PX, PY und L0 so gewählt werden, daß der Bildschirm nicht „überlaufen" kann.

Um jetzt ein mit der neuen Interpreter-Routine versehenes Programm kopieren zu können, benötigt man entweder das bereits erwähnte Kopierprogramm, oder man kopiert direkt von Kassette zu Kassette. Doch das hat seine Grenzen: die Kopie einer kopierten Kopie wird der ZX-81 schwerlich akzeptieren. Liegt zudem ein Codewortschutz vor, so muß selbstverständlich das Codewort bekannt sein.

```
   1 REM HIER MINDESTENS 87 BE-
LIEBIGE ZEICHEN EINTIPPEN
  10 PRINT "KOPIERPROGRAMM VON M
. SCHRAMM"
  20 PRINT ,,"UNTER WELCHEM NAME
N SOLL DAS";"PROGRAMM AUF BAND G
ESPEICHERT";"WERDEN (MAX. 10 ZEI
CHEN)?"
  30 INPUT N$
  40 IF N$="" OR LEN N$>10 THEN
GOTO 30
  50 RUN USR 16514
```

⑦ **Kopierprogramm:** Zeile 1 reserviert Platz für den Maschinencode aus Bild 8

```
16514: 33  202 127 34  4   64
16520: 249 1   0   62  197 1
16526: 118 6   197 237 115 2
16532: 64  17  173 64  1   44
16538: 0   235 213 237 176 42
16544: 15  64  35  78  35  35
16550: 237 176 27  236 203 254
16556: 201 205 231 2   55  205
16562: 67  3   205 187 2   68
16568: 77  44  40  248 205 189
16574: 7   126 254 49  40  233
16580: 254 56  40  6   254 55
16586: 32  232 207 255 205 231
16592: 2   33  246 127 205 252
16598: 2   24  219
```

⑧ **Maschinencode:** Diese 87 Byte ermöglichen das Laden, Speichern und Starten geschützter Programme

Die Kehrseite der Medaille

Jetzt wird das bereits mehrfach erwähnte Kopierprogramm vorgestellt. *Bild 7* zeigt das Basic-Listing: In Zeile 1 wird wieder Speicherplatz für Maschinencode reserviert, nämlich für dessen Dezimalwerte aus *Bild 8* (mittels FOR-NEXT-Schleife eingeben).

Das Programm ist sehr kurz; die eigentliche Kopierroutine in Maschinensprache ist sogar nur 44 Byte lang. Der Basic-Programmteil verlangt lediglich die Eingabe des Namens, unter dem das zu kopierende Programm auf Band gespeichert werden soll. Dieser Name darf aus bis zu zehn Zeichen bestehen. Zeile 50 ruft dann das Maschinenprogramm auf, das die folgenden Operationen durchführt:

○ 54 Byte werden im oberen Bereich des 16-KByte-RAMs (oberhalb von RAMTOP) reserviert. 44 Byte sind für das Kopierprogramm und zehn Bytes für den eingegebenen Namen. Hierzu werden die Systemvariable ERR-SP sowie der Stapelzeiger umgesetzt und an das obere Ende des Z-80-Stapels bestimmte „Kennbytes" geschrieben (in FUNKSCHAU 5/1984, Seite 72, wurde diese Methode, Speicherplatz zu reservieren, bereits näher erläutert).

○ Die Kopierroutine wird in den reservierten Speicherbereich übertragen.

Bit 7 des letzten Bytes des Namens wird gesetzt; hieran erkennt die SAVE-Routine im ROM das Ende des Namens.

○ Es erfolgt ein Sprung in den LOAD-Teil des Kopierprogramms. Dieses geschieht etwas trickreich durch die Befehlskombination *push de – ret* (spart ein Byte gegenüber einem jp-Befehl).

Das Kopierprogramm selbst besteht aus vier Teilen:

MAIN: Der Hauptteil des Programms. Hier erfolgt die Abfrage der Tastatur. Je nach gedrückter Taste wird verzweigt: L → LOAD; S → SAVE; R → RUN. Die übrigen Tasten haben keine Wirkung. Je nachdem, in welchem Zustand (SLOW, FAST) das Programm, das sich gerade im Speicher befindet, ursprünglich mit SAVE gespeichert wurde, ist der Bildschirm weiß oder schwarz während der Abfrage. Falls sich der Computer im SLOW-Modus befindet, kann es sein, daß auf dem Bildschirm etwas angezeigt wird – hieran darf man sich nicht stören.

LOAD: Ein Programm wird geladen (wie durch LOAD "", jedoch immer ohne Autostart). Anschließend Sprung in den MAIN-Teil.

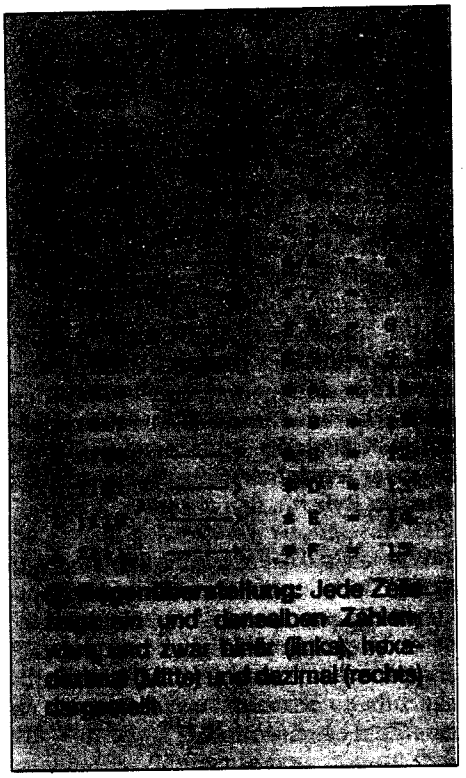
SAVE: Das im Speicher befindliche Programm wird auf Band gespeichert. Anschließend Sprung in den MAIN-Teil.

RUN: Mit dem geladenen Programm wird so verfahren, als sei es durch einen normalen LOAD-Befehl geladen worden: Ein selbststartendes Programm wird gestartet, bei einem nicht selbststartenden wird gestoppt.

Falls man aus der Kopierroutine heraus möchte, ohne daß ein selbststartendes Programm anläuft, drückt man die Taste S und anschließend BREAK. Wurde ein Programm mit BREAK-Schutz geladen, besteht hierbei jedoch die Gefahr eines Systemabsturzes! Durch RUN USR 32721 kann man wieder in das Kopierprogramm hineinspringen; oberhalb von RAMTOP ist es auch sicher vor NEW.

Für Kenner der Z-80-Maschinensprache liegen beim Franzis-Software-Service knapp kommentierte Assembler-Listings der hier vorgestellten Maschinenprogramme bereit. Sie können gegen Einsenden von 4 DM in Briefmarken mit dem Stichwort „Schloß und Schlüssel – FUNKSCHAU 23/1984" angefordert werden.

Michael Schramm



Laut Definition bezeichnet man das untere Halbbyte (Bit 0...3) einer 8-Bit-Binärzahl als niederwertiges oder weniger signifikantes Nibble und das obere Halbbyte (Bit 4...7) als höherwertiges oder mehr signifikantes Nibble. Da ein Nibble nur noch 4 Bit umfaßt, lassen sich mit einem Nibble $2^4 = 16$ verschiedene Zahlen darstellen (z. B. 0...15 im Dezimalsystem oder %0000...%1111 im Binärsystem). Praktisch wäre es, wenn sich der Wert eines Nibbles mit nur einer einzigen Ziffer darstellen ließe. Dies ist beim hexadezimalen Zahlensystem der Fall: Bis zu der Ziffer 9 unterscheidet es sich nicht vom Dezimalsystem, anstelle der Dezimalziffern 10...15 werden jedoch die Buchstaben A...F verwendet (*Bild 3*).

Hexadezimalzahlen sind (wenn erforderlich) durch ein Dollar-Zeichen gekennzeichnet. Jedes Byte läßt sich so mit nur zwei Ziffern darstellen:

38 = %00100110 = $26

95 = %01011111 = $5F

In der Maschinensprache ist das hexadezimale Zahlensystem von größter Wichtigkeit, wie wir später noch sehen werden. Eine Hilfestellung zum Umrechnen von dezimalen, binären und hexadezimalen Zahlen gibt *Bild 4*. Mit etwas Übung läßt sich aus dieser „Matrix" im Nu herausfinden, daß z. B. die Dezimalzahl 194 binär %11000010 und hexadezimal $C2 lautet.

Damit wollen wir unseren kurzen Abstecher in die Zahlensysteme abschließen. Wer mit dieser Gedächtnis-Auffrischung nicht zurechtgekommen ist, sollte sich Band 7 der Franzis-Computer-Bibliothek zulegen. Er trägt den Titel „Zahlen-Umwandlungen" und geht das Thema von der Pike auf an. Im nächsten Teil steigen wir dann endgültig in die Praxis ein und lernen ein Hilfsprogramm zur Eingabe von Maschinenprogrammen kennen.

Klaus Herklotz

(Wird fortgesetzt)

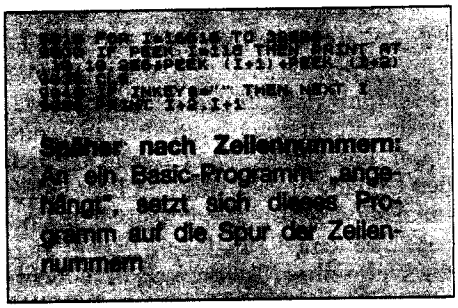
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0001	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	0010	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0011	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	0100	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	0101	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	0110	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	1000	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	1001	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	1010	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	1011	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	1100	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	1101	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	1110	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	1111	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

④ **Umrechnungstabelle:** Ob hexadezimal, binär oder dezimal – bis zum Dezimalwert 255 nimmt einem diese Tabelle das Umrechnen ab. Dabei repräsentiert die senkrechte HEX- bzw. BIN-Spalte die höherwertige Stelle

ZX-81-Softwaretip:

Die Null macht's

Es ist schon lange kein Geheimnis mehr, daß sich Zeilen, denen mit einem POKE-Befehl die Zeilennummer 0 „verordnet" wurde, nicht mehr mit EDIT verändern lassen. Normalerweise wird mit POKE 16510,0 die Nummer der ersten Programmzeile auf Null gesetzt, wenn sich in dieser Programmzeile ein Maschinenprogramm befindet. Komplizierter wird es, so bald eine andere Programmzeilennummer auf Null zu setzen ist. Ein Hilfsprogramm hilft dann beim Aufspüren der richtigen Adressen (*Bild*).



Bis auf die erste Zeilennummer meldet es alle Zeilennummern eines Basic-Programms jeweils kurz am Bildschirm. Taucht die Nummer auf, die man auf Null setzen will, heißt es flugs irgendeine Taste zu drücken. Das Programm schreibt dann die Adressen der Zeilennummer auf den Bildschirm. Bei Zeilennummern unter 256 genügt es dann, den Inhalt der links angezeigten Adresse mit POKE auf Null zu setzen.

Das Programm kann auch dafür genutzt werden, Zeilen in ein zu eng numeriertes Programm einzufügen. Sollen z. B. zwischen den Zeilen 42 und 43 eines Programms noch zwei Zeilen eingefügt werden, so ist zuerst Zeile 42 auf Null zu setzen. Dann ist mit Zeilennummer 42 die erste Einschubzeile einzutippen und das Programm mit CONT weiter auszuführen. Daraufhin wird wieder Nummer 42 angezeigt und auch hier ist zu unterbrechen und die Zeile auf Null zu setzen. Jetzt darf unter Nummer 42 die zweite Einschubzeile eingetippt werden.

Auf diese Weise lassen sich beliebig große Einschübe vornehmen, die jedoch nicht Sprungziele sein dürfen. Und weil das Verfahren zwar interessant aber umständlich ist, sollte der Lerneffekt auf jeden Fall höher eingestuft werden als der Nutzeffekt.

Paul Webranitz

ZX-81-Hardwaretip:

Ladehemmung ade!

Zuweilen zimperlich ist der ZX 81, wenn er auf Band gespeicherte Programme wiedereinlesen soll. Gegen seine Ladehemmungen ist jedoch ein Kraut gewachsen.

Abhängig vom Recorder-Modell, kann es beim Abspeichern und Wiedereinlesen von Programmen zu erheblichen Schwierigkeiten kommen. Der ZX 81 selbst ist daran nicht ganz unschuldig, denn aus nicht einzusehenden Gründen liefert er beim Abspeichern (SAVE) nur ein mickeriges Ausgangssignal von rd. 5 mV (Spitze–Spitze). Das stellt hohe Anforderungen an die Qualität der Aufsprechverstärker in Kassettenrecordern.

Die FUNKSCHAU hat bereits wiederholt Tips veröffentlicht, wie die Ladehemmungen des ZX 81 abzubauen sind. Mit dem hier beschriebenen Gerätchen (*Bild 1*) wird jetzt abschließend eine Komplett-Lösung geboten, die keinerlei Eingriff in den Computer verlangt. Wer auf seinem ZX 81 noch Garantie hat, wird das zu schätzen wissen.

Probleme über Probleme

Sieht man einmal von totalen Fehlschlägen bei Ladeversuchen ab, so hat ein ZX-81-Besitzer mit Kassettenrecordern noch anderweitig seine liebe Not:

○ Beim Laden eines Programms ist keine Mithörkontrolle möglich, weil beim Signalabgriff an der EAR-Buchse des Kassettenrecorders dessen Lautsprecher automatisch abgeschaltet wird. Überbrückt man den Schaltkontakt, ist das Mithören zwar möglich, die Lautstärke richtet sich aber nach dem vom ZX 81 geforderten Pegel und ist deshalb ohrenbetäubend.

○ Bei vielen Recordern muß beim Abspeichern eines Programms der EAR-Stecker gezogen werden, um bei der Aufnahme Rückkopplungen vom Ausgang zum Eingang des Recorders zu vermeiden. Mißachtung wird mit „unleserlichen" Aufzeichnungen bestraft.

○ Zum Speichern ungeeignet waren bislang Recorder mit DIN-Buchse. Sie fordern normalerweise ein viel höheres Eingangssignal, als es der ZX 81 zu bieten vermag, und das Ausgangssignal ist wiederum zu klein, als daß es der ZX 81 auswerten kann. Diese Probleme zwingen zur Anschaffung eines weiteren Recorders mit Klinkenbuchsen.

Insbesondere der letzte Punkt ist höchst ärgerlich, denn schließlich wurde hierzulande jahrelang gefordert, daß auch Recorder aus ausländischer Produktion mit DIN-Buchsen ausgestattet werden. Jetzt, wo diese Forderung halbwegs erfüllt ist, bricht das Computerzeitalter im Heim an und verlangt prompt wieder nach Recordern mit Klinkenbuchsen. Den Recorder-Anbietern mag das guttun, nicht aber den Anwendern.

Das einfache LOAD-/SAVE-Interface (*Bild 2*) ist hauptsächlich für Recorder mit DIN-Buchse gedacht, und es beseitigt alle geschilderten Probleme.

Jetzt sind auch DIN-Pegel erlaubt

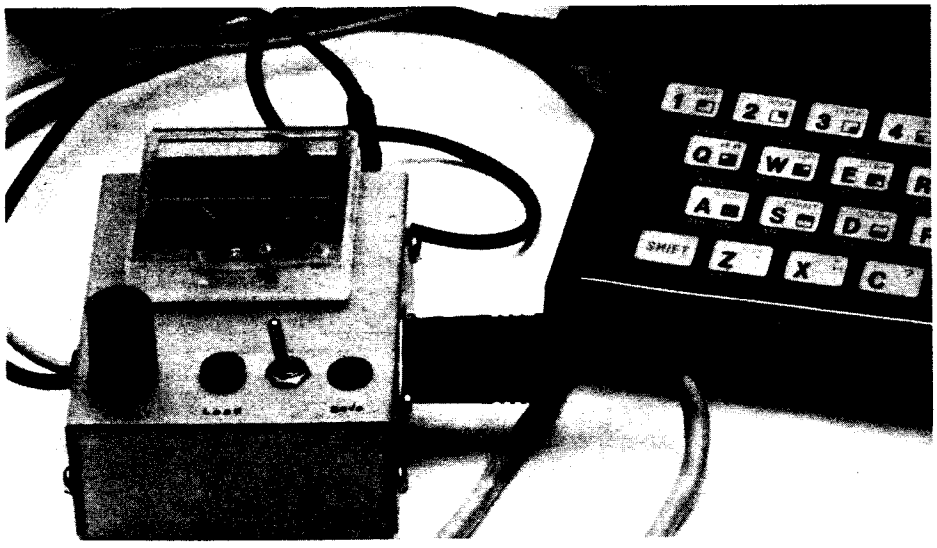
Beim Laden wird das Ausgangssignal des Recorders über eine DIN-Buchse eingespeist und von IC1 auf einen Wert angehoben, den der ZX 81 akzeptiert. Der Aussteuerungsmesser am Ausgang des Verstärkers leistet Hilfestellung zum Justieren des richtigen Pegels mit P1.

Wird das Ausgangssignal an der DIN-Buchse des Recorders abgegriffen, dann ist der Ladevorgang unabhängig von der Lautstärkeeinstellung. Mithören ist jetzt problemlos möglich. Bei Kassettenrecordern mit Klinkenbuchsen sollte der Signalabgriff unmittelbar vor dem Lautstärkesteller erfolgen, wenn man in den Genuß der Vorteile kommen möchte.

Zum Abspeichern eines Programms wird das Signal an der MIC-Ausgangsbuchse des ZX 81 von IC2 soweit verstärkt, daß es bei Einspeisung in die DIN-Buchse des Recorders zur Vollaussteuerung ausreicht. Die Verstärkung bestimmt der Widerstand an Pin 8 des ICs.

Die stabilisierte Betriebsspannung für die Schaltung (5 V) läßt sich vom ZX-81-Netzgerät ableiten. Dabei ist die unstabilisierte Spannung (etwa 9 V) durchzuschleifen, da sie vom ZX 81 in voller Höhe benötigt wird (unweigerlich dann, wenn ein 16-KByte-RAM mit bipolaren Speicher-ICs angeschlossen ist).

Die Rückkopplungsgefahr beseitigt ein Umschalter, der je nach Betriebsart (LOAD oder SAVE) eine der beiden Ver-



① **LOAD-/SAVE-Interface:** Zwischen Kassettenrecorder (mit DIN-Buchse) und Computer eingeschleift, raubt das Interface dem ZX 81 seine „Ladehemmungen" Foto: Neumayr

stärkerstufen einfach stillegt. Welche Betriebsart gerade gewählt ist, zeigen die beiden Leuchtdioden.

Abgleich ohne Meßgeräte

Untergebracht wird die Schaltung am besten in einem kleinen Metallgehäuse (zur Abschirmung), wie in Bild 1 gezeigt. Als Aussteuerungsmesser eignen sich beliebige Ausführungen, die bei etwa 250 µA Vollausschlag zeigen, z. B. einfache Batteriespannungs-Indikatoren. Den Spannungsregler sollte man zur guten Wärmeableitung an einer Gehäusewand montieren.

Schnittstellen der Schaltung sind dann eine DIN-Buchse, eine Klinkenbuchse für die 9-V-Spannung vom ZX-81-Netzteil, zwei Kabel zur EAR- bzw. MIC-Buchse des ZX 81 sowie ein weiteres Kabel, das die Weiterleitung der durchgeschleiften 9-V-Spannung zum Computer übernimmt.

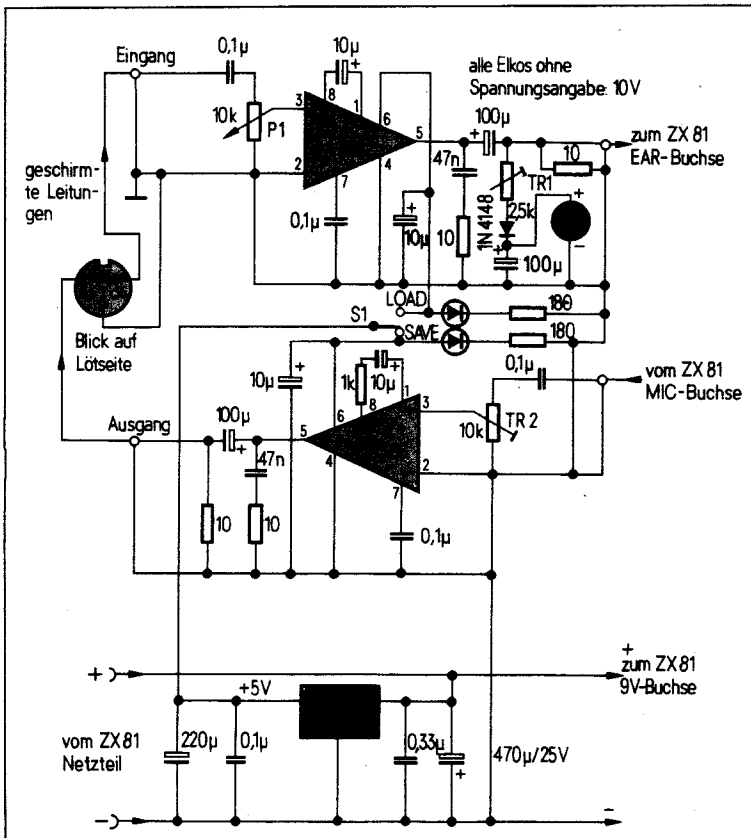
Und so wird die Schaltung justiert: Zunächst ist in den ZX 81 irgendein Programm einzutippen, TR2 in Mittelstellung zu bringen, der Schalter auf SAVE zu stellen und das Programm auf Band abzuspeichern. Hört man das Band dann ab, muß ein hell klingendes kräftiges Signal hörbar sein. Durch wiederholte Versuche ist TR2 so einzustellen, daß das Signal leicht übersteuert aufgezeichnet wird. Hörbar ist das an einem schrill werdenden Klang. Abschließend ist mit dieser Einstellung von TR2 das Programm endgültig abzuspeichern.

Jetzt ist der Schalter auf LOAD zu stellen und das zuletzt gespeicherte Programm in den ZX 81 zu laden. P 1 wird dabei so eingestellt, daß sich das übliche Streifenmuster am Bildschirm ergibt. Ohne die Stellung von P 1 zu verändern, muß man jetzt beim wiederholten Laden desselben Programms TR 1 so einstellen, daß der Zeiger des Aussteuerungsmessers eine markante Stellung einnimmt. Damit ist die Justage abgeschlossen. Beim Laden fremder Programme wird durch Nachstellen von P 1 auf die markante Zeigerstellung die im allgemeinen nötige Pegelanpassung vorgenommen.

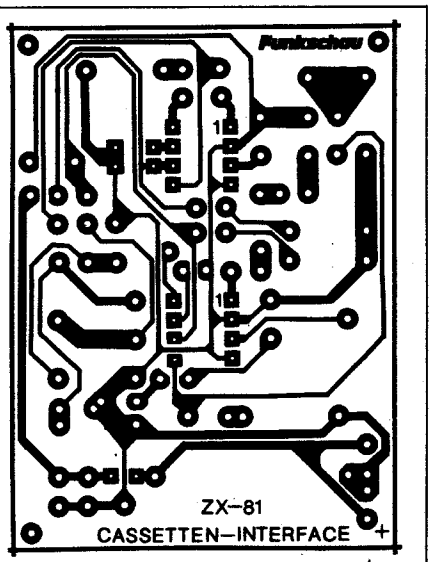
Noch ein Tip für diejenigen, die sich erst einen (Daten-)Recorder anschaffen möchten: Es genügen Billigausführungen ohne HiFi-Prädikat. Wichtig ist jedoch ein Bandzählwerk, das zum raschen Wiederfinden von Programmen auf Kassetten unentbehrlich ist.

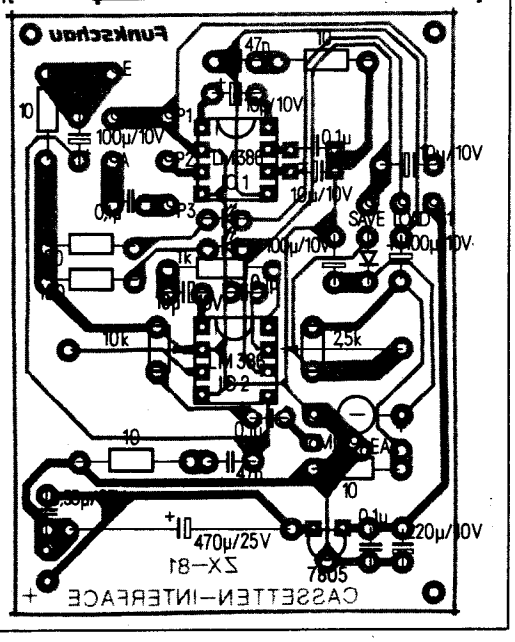
Dann sollte man noch darauf achten, daß sich der Aufnahme-/Wiedergabe-Kopf des Recorders von außen verstellen läßt. Wollen sich nämlich Fremdprogramme trotz Pegelanpassung durch P 1 nicht laden lassen, hilft nur noch ein Verstellen der Kopflage, bis auch das Fremdprogramm den typischen hellen Klang beim Abhören zeigt.

Zum Laden eigener Kassetten muß der Kopf dann wieder in die ursprüngliche Lage gebracht werden. Wenn es nicht unbedingt nötig ist, sollte man jedoch von der Justierschraube am Tonkopf die Finger lassen. Bernd W. Friedrich/-ll



② **Gesamtschaltung:** Zwei Verstärker bringen sowohl das Signal vom ZX 81 als auch das vom Kassettenrecorder auf den jeweils angemessenen Wert





Platine und Bestückungsplan: Wer weniger Perfektion anstrebt, kommt auch mit einer Lochrasterplatte gut zurecht

ZX-81-Softwaretip:

Brückenbauer

Anknüpfen von Basic-Programmen

Der ZX 81 kennt keinen speziellen LOAD-Befehl, der zu einem bereits geladenen Programm ein zweites Programm hinzulädt. Mit zwei Hilfsroutinen läßt sich diese Aufgabe aber dennoch lösen. Dabei rechtfertigt der Nutzen den Aufwand.

Wer selber viel programmiert, wird as Problem kennen: An ein Programm, n dem man gerade arbeitet, ließe sich in schon früher geschriebenes (Unter-) rogramm gut anfügen. Normalerweise leibt dann nichts anderes übrig, als die rgänzung Zeile für Zeile geduldig abzuppen. Das fordert Zeit, die viel besser ır Fertigstellung des eigentlichen auptprogramms spendiert werden ollte.

Rettungsinsel für Programme

Aber wie nur läßt sich die Ergänzung nfach hinzuladen, wo doch der ZX 81 im Ausführen des LOAD-Befehls das gentliche Hauptprogramm erbarungslos aus dem Programmspeicher rauswirft? Hier hilft nur noch ein ick weiter.

Das Hinzuladen von Programmen – dere Computer haben dafür spezielle debefehle (z. B. MERGE oder ENTER) ist beim ZX 81 allein auf Umwegen glich. Der Systemvariablen RAMTOP lt dabei eine Schlüsselrolle zu: Benntlich sind alle Speicherplätze oberlb von RAMTOP sicher vor dem Zuff des ZX-81-Betriebssystems, also ch sicher vor der Zwangslöschung rch Befehle wie NEW oder LOAD. Ein erhalb von RAMTOP abgelegtes Proımm wird daher nicht durch Hinzuladen eines zweiten Programms gelöscht, wie das normalerweise durch die LOAD-Funktion verursacht wird.

Nunmehr wären zwar die beiden Programme im RAM des Computers, aber sie sind noch nicht aneinander angehängt. Um das zu bewerkstelligen, muß man mit den Programmen etwas jonglieren. Und damit es dabei keine Mißverständnisse gibt, geben wir ihnen die Namen P1 und P2:

P1 sei das Hauptprogramm, an das das Programm P2 angefügt werden soll. P1 ist oberhalb von RAMTOP abgelegt, wogegen P2 durch Hinzuladen seinen Platz im üblichen Programmspeicher gefunden hat. Jetzt gilt es, P2 um genauso viele Bytes im Programmspeicher nach hinten (oben) zu schieben, daß sich P1 nahtlos in die entstandene Lücke hineinkopieren läßt. Dann stehen beide Programme in der richtigen Reihenfolge, erst P1 und dann P2, gemeinsam im Programmspeicher – das Ziel ist erreicht. So kompliziert wie es scheint, ist das alles gar nicht, wenn wir zum Jonglieren Hilfsroutinen verwenden.

Reger Verkehr im RAM

Der Brückenschlag zwischen zwei Programmen erfordert zwei Hilfsroutinen, die einmal RAMTOP herabsetzen (*Bild 1*) und dann das Verschieben und Kopieren erledigen (*Bild 2*). Beide Hilfsroutinen sind zunächst abzutippen und, von einer rd. 5 s dauernden Pause getrennt, auf Band aufzuzeichnen (GOTO 190 bzw. GOTO 240).

Den Umgang mit den Routinen probieren wir jetzt am besten gleich an einem Beispiel aus: An das „Hauptprogramm" (P1)

```
10 PRINT "HAUPTPROGRAMM"
```

soll das „Unterprogramm" (P2)

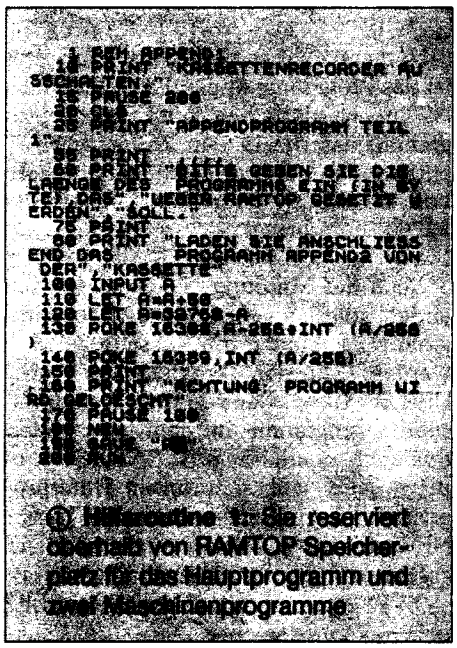
```
20 PRINT "UNTERPROGRAMM"
```

angefügt werden. Tippen wir also das Prgrämmchen P1 ein. Um es später oberhalb von RAMTOP unterzubringen, müssen wir den Speicherbedarf dieses Programms kennen. Dazu verhilft die Direktanweisung:

```
PRINT  PEEK  16396
+  256*PEEK 16397–16509
```

die die Differenz zwischen dem Programmspeicherende (D-FILE) und dem Programmspeicherbeginn (Adresse 16509) bildet. Im Beispiel lautet das Ergebnis 21 (Byte).

Jetzt ist P1 auf Kassette abzuspeichern und danach die Hilfsroutine Append 1 (Bild 1) zu laden. Sie fordert zur Eingabe des soeben festgestellten Speicherbedarfs von P1 auf. Geschieht dies, so wird RAMTOP (Normalwert 32768 bei 16 KByte Speichererweiterung) um diesen



① Hilfsroutine 1: Sie reserviert oberhalb von RAMTOP Speicherplatz für das Hauptprogramm und zwei Maschinenprogramme

```
  1 REM APPEND 2
 10 PRINT "APPENDPROGRAMM TEIL
2"
 20 PRINT
 30 PRINT "DIESES PROGRAMM LAED
T 2 MASCHI- NENPROGRAMME UEBER R
AMTOP."
 40 PRINT
 50 PRINT "STARTEN SIE DEN LADE
VORGANG      DURCH TASTENDRUCK."
100 PAUSE 4E4
110 PRINT
115 PRINT "LADER AKTIVIERT"
120 LET A$="042 012 064 001 125
064 237 065 068 077 237 067 064
127 237 091 004 064 033 125 064
237 176 201 "
125 LET A$=A$+"033 125 064 237
176 254 127 205 158 009 017 125
064 237 075 254 127 042 004 064
237 176 201 "
130 FOR A=32719 TO 32765
140 POKE A,VAL A$( TO 3)
150 LET A$=A$(5 TO )
160 NEXT A
170 CLS
180 PRINT "LADEVORGANG BEENDET"
190 PRINT
195 PRINT
200 PRINT "- WICHTIG - WICHTIG
- WICHTIG -"
210 PRINT
220 PRINT "MASCHINENPROGRAMM 2
(USR 32743) SOLLTE NUR IM ""FAST""
-MODUS AUF- GERUFEN WERDEN"
230 STOP
240 SAVE "A■"
250 RUN
```

② Hilfsroutine 2: Die in den Zeilen 120/125 enthaltenen Maschinenprogramme werden oberhalb von RAMTOP untergebracht. Sie bewirken die Verknüpfung der eigentlichen Anwenderprogramme

Wert zuzüglich 50 Byte automatisch tiefergesetzt. Die 50 Byte Zugabe sind nötig, weil die zweite Hilfsroutine Append 2 (Bild 2) Maschinenprogramme enthält, die ebenfalls oberhalb von RAMTOP untergebracht werden (zum Ablegen von P1 über RAMTOP, zum Verschieben von P2 und Kopieren von P1). Das gezielte Tiefersetzen von RAMTOP ist die einzige Aufgabe von Append 1.

Nun erst geht es richtig los, denn jetzt ist das Programm Append 2 zu laden. Es ist im wesentlichen ein Transferprogramm, das die in den Zeilen 120/125 enthaltenen Maschinencodes oberhalb von RAMTOP (Adressen: 32719 bis 32765) unterbringt.

Nach der Meldung „Ladevorgang beendet" darf unser kleines Hauptprogramm P1 wieder von Kassette geladen werden. Mit der Direkteingabe RAND USR 32719 wird es von dem ersten Maschinenprogramm in den reservierten Speicher oberhalb von RAMTOP kopiert. Nun läßt sich unser Unterprogramm P2 entweder eintippen (zuvor NEW ausführen) oder, wenn es auf Band gespeichert war, mit LOAD laden. Damit steht P1 oberhalb von RAMTOP und P2 im Programmspeicher.

Um beide Programme zu verknüpfen, ist jetzt das zweite Maschinenprogramm mit RAND USR 32743 aufzurufen. Zuvor sollte man den ZX 81 aber unbedingt in den FAST-Modus bringen. Das Maschinenprogramm nutzt nämlich zum Verschieben von P2 eine ROM-Routine (Adresse: 2462d). Dabei zeigt die Systemvariable D-FILE zeitweise nicht auf den Anfang des Bildspeichers, und das kann den ZX 81 beim Bildaufbau durcheinanderbringen (Systemabsturz), sofern er im SLOW-Modus ist.

Da das zweite Maschinenprogramm auch noch P1 vor P2 kopiert, stehen nach Abschluß der Routine und Eingabe von LIST unsere beiden Programmzeilen friedlich untereinander am Bildschirm, Zeile 20 wurde an Zeile 10 angehängt; die Aufgabe ist gelöst.

So richtig nützlich ist die Anknüpfung eines Programms freilich nur bei umfangreicheren Ergänzungen. Dann muß man darauf achten, daß es bei den Zeilennummern keine Überschneidungen gibt. Am besten wird das Programm P2 deshalb vor dem Anknüpfen so umnumeriert, daß es möglichst nahtlos an P1 anschließt.

Wiedergewinnen der vollen RAM-Kapazität

Speicherplatzprobleme kann es geben, wenn man zwei Mammutprogramme miteinander verknüpfen möchte. Dann läßt sich der Bedarf an Speicherplatz am niedrigsten halten, wenn das kürzere der zu verknüpfenden Programme über RAMTOP gesetzt wird. Denn schließlich geht die nach vollzogener Verknüpfung überflüssige Kopie von P1 oberhalb von RAMTOP dem Programmspeicher verloren.

Abhilfe ist möglich, wenn das verknüpfte Gesamtprogramm auf Band gespeichert wird, wir den ZX 81 kurz ausschalten, und dann das Programm wieder laden. RAMTOP hat dann seinen Normalwert.

Da der Umgang mit dem „Brückenbauer" anfangs gewiß gewöhnungsbedürftig ist, gibt die folgende Liste einen zusammenfassenden Überblick auf die nötigen Arbeitsschritte:

- ○ Länge des Hauptprogramms (P1) bestimmen.
- ○ Append 1 laden und Speicherbedarf eingeben.
- ○ Append 2 laden und starten.
- ○ Hauptprogramm (P1) laden.
- ○ Erstes Maschinenprogramm von Append 2 mit RAND USR 32719 starten.
- ○ Unterprogramm (P2) eventuell umnumeriert laden.
- ○ Zweites Maschinenprogramm von Append 2 mit RAND USR 32743 (im FAST-Modus!) starten.

Andreas Brecht/-ll

Berichtigung

Wobbeln bis 30 MHz

FUNKSCHAU 1984, Hefte 2 bis 4

Dieser Wobbelgenerator ist von vielen Lesern nachgebaut worden, die uns von ihren Erfahrungen berichtet haben. Einige dieser Tips wollen wir nachtragen, um den Nachbau zu erleichtern und Schwierigkeiten auszuräumen. Außerdem haben sich in einer Schaltung Fehler eingeschlichen, die trotz mehrfacher Kontrolle leider nicht aufgefallen waren.

In der Schaltung für den Mischer (Bild 5 in Heft 2, Seite 69) ist die Diode falsch gepolt gezeichnet. Der Emitterwiderstand des zweiten BF 311 muß wie beim ersten 12 Ω betragen (nicht 120 Ω). Das Dämpfungsglied am Ausgang IF des Mischers IE 500 ist nicht übereinstimmend mit der Bestückung angegeben. Die beiden Widerstände – am Ausgang IF und vor L1 (beide nach Masse) – müssen 270 Ω haben. Der zweite Widerstand ist im Bestückungsplan (Bild 15 in Heft 3, Seite 72) richtig angegeben, der erste fehlt dagegen ganz. Die freien Lötaugen unterhalb des IE 500 sind dafür vorgesehen.

Ein Tip für diejenigen, die noch Zugriff auf den Hochstrom-FET T8002 haben (wird nicht mehr hergestellt): Wenn der BF 246 A im Mischer durch diesen Typ ersetzt wird, erhält man bei gleichem Drainstrom eine höhere Verstärkung (ca. 20 dBm). Dann muß der Drainwiderstand von 180 Ω durch eine Reihenschaltung aus 220 µH und 28 Ω ersetzt werden.

ZX-81-Softwaretip:

Schreib-Kraft

Platzmacher für Maschinenprogramme

Wer in einer REM-Zeile ein Maschinenprogramm unterbringen will, muß dort zuvor ebenso viele Zeichen eintippen, wie das Programm Bytes hat. Überlassen Sie diese Arbeit lieber einer Schreib-Kraft.

Jede der drei Methoden, ein Maschinenprogramm im Speicher des ZX 81 unterzubringen, hat ihre Vor- und Nachteile. Darüber wurde bereits in Heft 11/1984, Seite 73, berichtet. Das Hauptmanko der Eingabe in eine REM-Zeile läßt sich jedoch beseitigen: Nicht mehr der geplagte Programmierer, sondern der ZX 81 selbst muß dafür sorgen, daß in der REM-Zeile genug Platz für ein Maschinenprogramm reserviert wird.

Die „Tipp-Orgien" beim Platzschaffen für längere Maschinenprogramme sind damit zu Ende. Und überdies gewinnt man Zeit, denn wer schafft es schon mitzuhalten, wenn die „Schreib-Kraft" in rd. 7 s 1000 Zeichen in eine REM-Zeile schreibt.

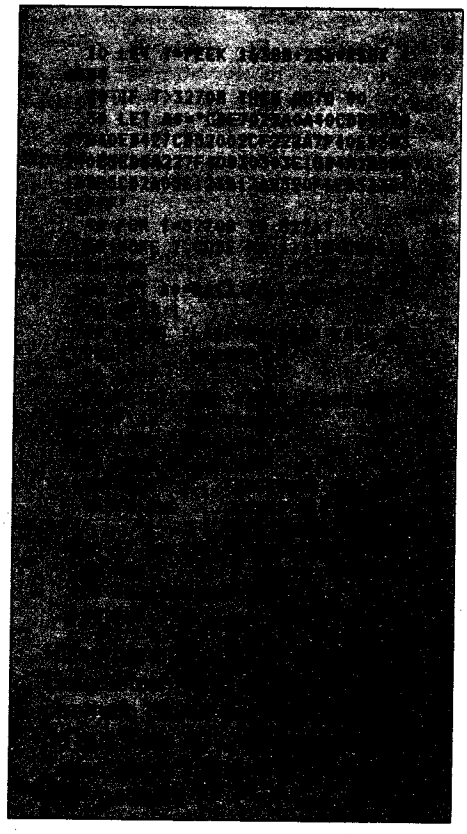
Ein sicheres Plätzchen für den Helfer

Hinter der Schreib-Kraft verbirgt sich ein kurzes Hilfsprogramm (*Bild*), das zunächst abzutippen und mit GOTO 150 auf Kassette zu speichern ist. Wie das Listing zeigt, wird der in Zeile 30 untergebrachte Maschinencode ganz oben im RAM-Speicher (ab Adresse 32708; ZX 81 mit 16 KByte) abgelegt. Deshalb muß hier erst einmal Platz geschaffen werden, indem man RAMTOP um 60 Byte tiefer setzt. Vor dem Wiedereinlesen des Programms sind dazu folgende Direkteingaben nötig:

```
POKE 16388,196
POKE 16389,127
NEW
```

Erst nach dem NEW-Befehl akzeptiert auch das ZX-81-Betriebssystem den neuen Wert für RAMTOP. Jetzt darf das Programm von Band geladen werden. Vergißt man einmal das vorherige Tiefersetzen von RAMTOP, erkennt das Programm dies und bügelt den Fehler selbsttätig aus. Allerdings muß der Ladevorgang dann wiederholt werden.

Taucht die von Zeile 80 vorgegebene Meldung am Bildschirm auf, steht das Programm richtig über RAMTOP und man darf mit NEW unbesorgt den Programmspeicher löschen.



Da das Maschinenprogramm ohne absolute Sprünge arbeitet, ist es in jedem Speicherbereich lauffähig. In der 1-KByte-Version des ZX 81 kann man es z. B. unterbringen, wenn die FOR-NEXT-Schleife in Zeile 40 von Adresse 17348 bis 17407 reicht (RAMTOP anpassen). Mangels Speicherplatz müssen dann aber die nicht unbedingt nötigen Programmzeilen (10, 20, 90 bis 140) gestrichen werden. Selbstverständlich ist die neue Startadresse auch später beim Aufruf des Maschinenprogramms einzusetzen (z. B. 17348 anstelle von 32708).

Nur noch drei Zeilen Handarbeit

Konnte sich der ZX 81 bis jetzt noch vor der Arbeit drücken, so wird es nun ernst für ihn: Er soll z. B. zehn Zeichen in einer REM-Zeile unterbringen. Selber muß man dazu lediglich noch die folgenden Zeilen eintippen:

```
1 REM
2 RAND 10
3 PRINT USR 32708
```

Sekunden nach dem Programmstart (RUN) wird sich der ZX 81 mit der Meldung 0/3 zurückmelden und ein Druck auf die NEWLINE-Taste zeigt, daß er hinter dem REM-Befehl tatsächlich zehn Zeichen eingefügt hat. Außerdem wurde Zeile 1 auf 0 umnumeriert. Ein versehentliches Löschen und zuweilen riskantes Editieren dieser Zeile (s. a. Heft 13/1984, Seite 70) ist damit ausgeschlossen. Maßgebend dafür, daß zehn Zeichen eingefügt wurden, ist die Zahl nach dem RAND-Befehl.

Sollte sich der ZX 81 nicht mit 0/3 sondern mit Z/3 zurückmelden, signalisiert das eine „verbotene" Position des Programm-Cursors. Der darf nämlich nicht in der ersten Zeile stehen. Erst ein wiederholter Programmstart bei verschobenem Cursor führt dann zum Erfolg.

Da die Zeilen 2 und 3 ihren Zweck erfüllt haben, kann man sie jetzt löschen oder mit einem Maschinencode-Eingabeprogramm überschreiben. Aber Vorsicht! Stehen in der REM-Zeile mehr als 694 Zeichen, darf Zeile 2 nicht gelöscht, sondern nur überschrieben werden. Der ZX 81 versucht sonst die komplette REM-Zeile am Bildschirm zu listen, und weil ihm das bei mehr als 694 Zeichen

nicht gelingt (Bildschirm voll), läßt er sich davon nicht mehr abbringen.

Die REM-Zeile wird normalerweise immer gemeinsam mit einem Maschinencode-Eingabeprogramm benötigt. Deshalb ist es zweckmäßig, nach dem Ablegen der Schreib-Kraft oberhalb von RAMTOP ein solches Programm wie üblich zu laden, und dann erst die REM-Zeile schreiben zu lassen. Bei dem Eingabeprogramm aus Heft 12/1983, Seite 76, wäre z. B. Zeile 10 durch die Zeilen 1 bis 3 zu ersetzen. Damit spart man sich das Eintippen des Eingabeprogramms.

Sonderwünsche sind kein Problem

Liegt bereits ein Maschinenprogramm in einer REM-Zeile vor, dann kann auch in einer solchen Zeile zusätzlich Platz geschaffen werden. So läßt sich z. B. an ein bestehendes Programm ein weiteres anhängen. Auch hier ist wieder zuerst die Schreib-Kraft oberhalb von RAMTOP unterzubringen, und dann das bestehende Programm zu laden. Ein Beispiel verdeutlicht das weitere Vorgehen:

```
1 REM FUNKSCHAU
2 RAND 4
3 PRINT USR 32708
```

An die bereits mit den Zeichen FUNKSCHAU versehene REM-Zeile werden in diesem Fall vier weitere Zeichen angehängt, wenn man das Programm startet. Wichtig ist, daß diesmal der Cursor keinesfalls in der ersten Zeile stehen darf.

Zuweilen kommt es auch vor, daß in einem Listing die REM-Zeile nicht die erste Programmzeile ist, sondern beispielsweise die sechste. Oder man muß zwei REM-Zeilen schreiben. Dann gilt es die Sicherungsmaßnahme der Schreib-Kraft (Nullsetzen der ersten Zeilennummer) rückgängig zu machen. Dazu ist nach dem Schreiben der REM-Zeile mit dem Befehl

```
POKE 16510,1
```

die Zeilennummer der REM-Zeile auf 1 zu setzen. Jetzt läßt sie sich mit EDIT aufrufen und wunschgemäß umnumerieren. Das nunmehr überflüssige „Original" in Zeile 1 wird durch Eingabe von 1 (gefolgt von NEWLINE) aus dem Listing gestrichen. Tippt man dann wieder

```
1 REM
```

ein, und weist dem RAND-Befehl eine neue Zahl zu, dann kann eine weitere REM-Zeile ins Listing eingefügt werden.

Kern des Maschinenprogramms ist der Aufruf der ROM-Routine „Einfügen". Sie schiebt den Speicherinhalt bei jedem Aufruf ab der im hl-Registerpaar gespeicherten Adresse bis STKEND um ein Zeichen nach oben. Das Zeichen im Akku belegt daraufhin die freigewordene Speicherzelle.

Zu Beginn des Programms (Adressen 32708 bis 32727) wird jedoch geprüft, ob der Cursor in der ersten Programmzeile steht. Bei mehr als 694 eingefügten Zeichen käme es nämlich auf Anhieb zu dem beschriebenen Endloslisten dieser Zeile. Die Prüfung sorgt dann sicherheitshalber für einen Programmabbruch mit der Meldung Z/3. Peter Bluschke/-ll

Echtzeituhr für ZX 81:

Da schlägt's 13

Wer mit seinem ZX 81 Meß-, Steuer- und Regelaufgaben lösen möchte, für den kann eine Echtzeituhr eine feine Sache sein. Mit ihrer Hilfe lassen sich Ereignisse zu einem vorbestimmten Zeitpunkt auslösen oder anderseits läßt sich der Zeitpunkt von Ereignissen festhalten. Denkbar wäre z. B. auch die Anwendung in einem Langzeit-Terminkalender.

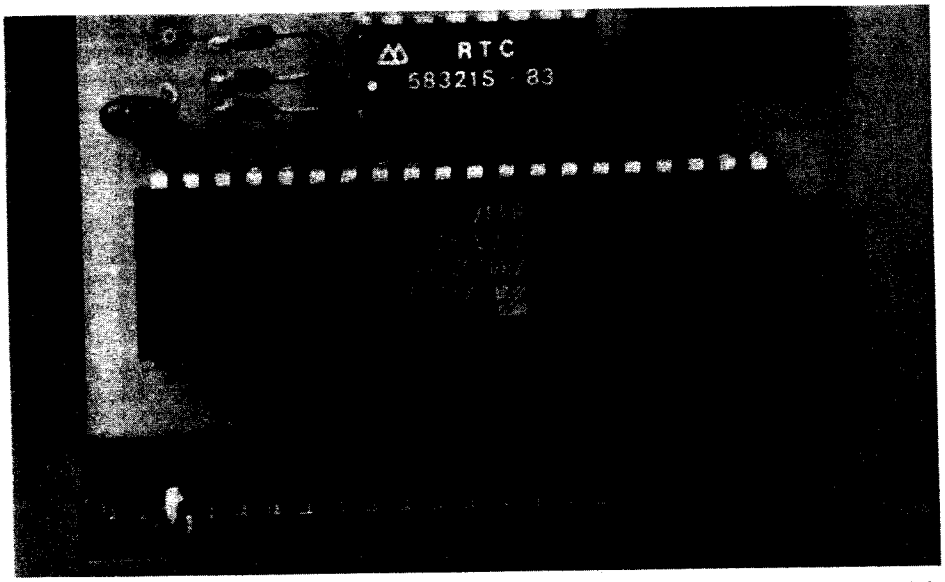
Gegenüber einer per Programm verwirklichten Uhr hat eine Echtzeituhr den Vorteil, daß die Uhrzeit nur abgefragt und nicht „ausgerechnet" werden muß. Wichtig ist dies, wenn zum „echten" Zeitpunkt eines Ereignisses möglichst ohne Verzögerung irgendeine Reaktion auszulösen ist.

Zum Preis von 89 DM gibt es für den ZX 81 eine solche Echtzeituhr (Elektronik Bolling, Pulheim). Sie besteht im wesentlichen aus einem CMOS-Uhren-IC und einer Z-80-PIO.

Der Umgang mit der Uhr ist sehr einfach: Zuerst muß das Modul auf die ZX-81-Schnittstelle gesteckt und dann die mitgelieferte Software von Kassette geladen werden (Ladedauer rd. 40 s). Anschließend wird man dazu aufgefordert, Startdaten einzugeben (Tag, Monat, Jahr, Stunden, Minuten). Damit ist die Uhr gestellt und zeigt sämtliche Werte am Bildschirm an.

Da bei der mitgelieferten Software die Anzeige der Werte von einem Basic-Programm übernommen wird, kommt es bei der Sekundenanzeige zu einem störenden Hüpfen der Werte in 4- bis 5-s-Schritten. Mit einem Maschinenprogramm ließe sich dies vermeiden. Viel wichtiger ist jedoch, daß die einmal gestartete Uhr ganz ohne Software weiterläuft, batteriegepuffert sogar dann, wenn der ZX 81 ausgeschaltet wird!

Abgefragt wird die Uhr (samt Kalender) von einem rd. 100 Byte langen Maschinenprogramm, das an jeder Stelle im RAM stehen darf. Dieses Programm legt jedesmal, wenn es aufgerufen wird, die aktuellen Werte in zwölf Speicherplätzen ab. Zur Auswertung durch ein beliebiges Basic- oder Maschinencode-Anwenderprogramm genügt daher das Abfragen dieser Speicherplätze. Perfektionisten sei noch gesagt, daß der Kalender sogar Schaltjahre berücksichtigt. -ll



Echtzeituhr: Sie bietet außer quarzgenauer Uhrzeit einen Kalender und läßt sich prinzipiell an jeden Computer mit Z-80-CPU anschließen Foto: -ll

ZX-81-Softwaretip:

Tücken der SCROLL-Funktion

Wer sich für den ZX 81 eine Speichererweiterung zugelegt hat, wird sich wundern: Je mehr Variablen in einem Programm definiert sind, desto träger wirkt ein im Programm vorkommender SCROLL-Befehl. Das kurze Programm in *Bild 1* demonstriert das in aller Deutlichkeit: Im Zeitlupentempo rutschen die Zeilen am Bildschirm hoch. Ist der Bildschirm vollgeschrieben, kann man versuchen, das Programm mit BREAK zu stoppen. Dabei wird Geduld gefordert, denn eventuell klappt der Abbruch erst nach einigen Sekunden Dauerdruck auf die BREAK-Taste.

Von der Botschaft „Ich bin defekt" überzeugt endgültig die anschließende Betätigung von NEWLINE: Das Bild beginnt agonisch zu zucken, der ZX 81 ignoriert jede Eingabe und erst nach Minuten ist der Bildschirm freigeräumt. Dann dauert es noch eine Weile, bevor endlich das Listing wieder erscheint. Aber keine Sorge, der ZX 81 ist selbstverständlich nicht defekt – lediglich das Betriebssystem hat uns einen Streich gespielt.

Der ZX 81 muß erst Platz freischaufeln

Bei jedem Erreichen des SCROLL-Kommandos schiebt der ZX 81 die vorhandenen Textzeilen um eine Zeile nach oben und markiert den Anfang der untersten Zeile mit einem NEWLINE-Code (118). Folgt jetzt, wie im Beispiel, ein PRINT-Befehl, werden die Zeichen hinter den NEW-LINE-Code in diese Zeile geschrieben. Und das ist das Problem.

Nach dem NEWLINE-Code folgen nämlich im Bildspeicher nur noch die beiden für INPUT und Fehlermeldungen reservierten Zeilen, und dann beginnt bereits der Variablenspeicher (siehe Kapitel 27 des Original-Sinclair-Handbuches). Zuerst muß also Platz geschaffen werden, indem der gesamte Speicherinhalt nach dem NEWLINE-Code um ein Byte verschoben wird. Der vom Betriebssystem dafür verwendete *ldir*-Befehl ist zwar sehr leistungsfähig, benötigt in unserem Fall aber zum Verschieben der ca. 15 KByte im Variablenspeicher rd. 2 s. Erst dann kann das erste Zeichen des PRINT-Befehls gesetzt werden. Für jedes weitere Zeichen wiederholt sich der Verschiebevorgang. Da während der Ausführung des PRINT-Befehls keine Tastaturabfrage stattfindet, reagiert der Computer auch nicht sofort auf BREAK.

Gelöscht wird zeichenweise

Normalerweise würde beim ZX 81 mit Zusatz-RAM auch jeder PRINT-Befehl (ohne Kopplung mit SCROLL) so langsam ausgeführt. Um das zu verhindern, haben die Entwickler jedoch zu einem Trick gegriffen. Sobald das Betriebssystem des ZX 81 erkennt, daß mehr als 3,5 KByte RAM zur Verfügung stehen, wird eine Leerzeile von vorneherein mit 32 Leerzeichen aufgefüllt (zuzüglich NEWLINE). Das zeitraubende Verschieben entfällt dann, denn es werden lediglich die Leerzeichen durch die jeweiligen Zeichen des PRINT-Textes überschrieben. So „organisiert" benötigt der Bildspeicher freilich ständig rd. ¾ KByte des RAM (siehe auch Serie „Klartext für den ZX 81", Teil 2).

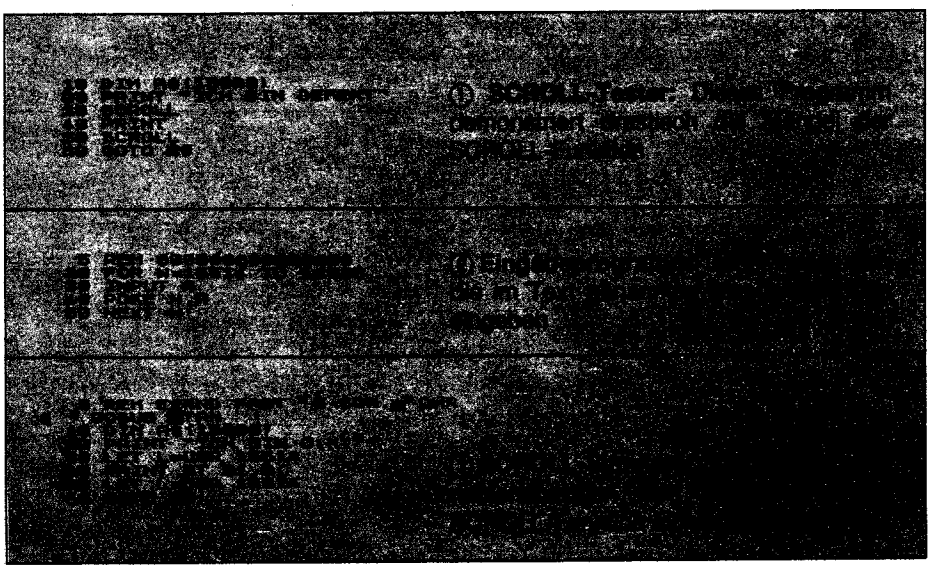
In Verbindung mit dem SCROLL-Befehl wird allerdings nur ein NEWLINE-Code und keine komplette Leerzeile nachgeschoben. Und das führt nicht nur beim PRINT-Befehl, sondern auch beim Löschen des Bildschirms zu der Verzögerung, denn dann werden die Zeilen auf je 32 Leerzeichen aufgefüllt – Zeichen für Zeichen!

Es geht auch schneller

Wer sich mit dem „Schneckentempo"-SCROLL nicht abfinden möchte, der kann mit dem Eingabeprogramm (*Bild 2*) folgenden Maschinencode (Dezimalform) in der REM-Zeile unterbringen (Start mit RUN und Eingabe der 15 Zahlen): 42, 12, 64, 229, 1, 33, 0, 209, 9, 1, 214, 2, 237, 176, 201. Anstelle des SCROLL-Befehls ist dann der Aufruf des Maschinencodes LET L=USR 16514 in ein Programm einzufügen.

Das Programmbeispiel (*Bild 3*) verdeutlicht das: Der Zusatz AT 21,0; bei dem zweiten PRINT-Befehl ist nötig, weil das Maschinenprogramm im Gegensatz zum Original-SCROLL die PRINT-Position nicht automatisch an den unteren Bildrand zwingt. Aber Achtung, das Maschinenprogramm darf nicht gemischt mit dem Original-SCROLL oder in der 1-KByte-Version des ZX 81 verwendet werden – dies würde zum „Ausstieg" des Computers führen.

Wolf-Dieter Roth



ZX-81-/ZX-Spectrum-Zubehör:

Computer im Gepäck

Maßgeschneidert für die Sinclair-Heimcomputer hat die Cambrasound Ltd. (England) einen Transportkoffer herausgebracht. Darin sollen sich fein säuberlich geordnet und vor dem Verrutschen sicher ein ZX Spectrum (oder ein ZX 81), das Original-Netzteil, der ZX Drucker und acht Compactcasetten unterbringen lassen. Hierzulande will die Cambra GmbH, Overath, den Vertrieb des schwarzen Kunststoffkoffers übernehmen (Preis: etwa 35 DM). –ll

ter. Am Bildschirm wird das mit der Meldung START quittiert.

Das Maschinenprogramm besteht diesmal aus zwei Teilprogrammen. Das erste ist das Ausgabeprogramm von Bild 2. Es übernimmt die Ausgabe der Zähler-Variablen N und wird in Zeile 120 aufgerufen. Der erste ret-Befehl (Adresse 408Ah) bewirkt dann den Rücksprung in das Basic-Programm.

Neun Bytes nach dem ersten Teilprogramm beginnt das zweite. Der Aufruf in Zeile 140 lautet deshalb

LET B=USR (X+9)

Mit dem unter Adresse FBh erteilten Steuerwort CFh wird Port B auf Betriebsart 3 eingestellt (Bit-Ein-/Ausgabe). Gemäß Teil 1 ist darüber hinaus noch ein Ein-/Ausgaberegister-Steuerwort erforderlich. Wählen wir dafür den Wert 80h (Ausgabe unter Adresse FBh), dann ist Datenleitung 8 von Port B als Eingang geschaltet. Die übrigen Datenleitungen ermöglichen keine Eingabe, sie sind als Ausgang geschaltet.

Der Rest des Maschinenprogramms stimmt mit dem Eingabeprogramm von Bild 4 überein. Nach dem Rücksprung ins Basic-Programm meldet die Variable B, ob die Taste gedrückt oder nicht gedrückt war.

Bei allen Programmbeispielen wurde das jeweilige Maschinenprogramm mit jedem Aufruf vollständig abgearbeitet und somit die Steuerworte jedesmal neu ausgegeben. Das PIO-IC speichert jedoch die Steuerworte in Registern. Deshalb ist es auch zulässig, bei einem wiederholten Aufruf der Maschinenprogramme die Definition der Betriebsart zu überspringen. Michael Schütz/-ll

ZX-81-Softwaretip:

Reservieren eines geschützten Speicherbereiches

Zuweilen ist es notwendig, einen Teil des RAMs für Maschinencode, Tabellen oder andere Zwecke zu reservieren. Die einfachste Methode dafür ist, per Handeingabe die Systemvariable RAMTOP zu verringern und anschließend mit NEW den Speicher neu zu initialisieren, wodurch erst der Bereich oberhalb von RAMTOP vor dem Zugriff durch das Betriebssystem geschützt wird. Dieses Verfahren hat jedoch einige Nachteile:

○ NEW löscht jedes Basic-Programm und die Variablen.

○ Der Z-80-Stapel, der ganz oben im RAM steht, beansprucht Platz, der nicht belegt werden darf.

○ Falls der Stapel schon zuvor von anderen Programmen verschoben wurde, besteht die Gefahr eines Systemzusammenbruchs beim Beschreiben des vermeintlich freien Speicherplatzes.

Abhilfe bietet ein Maschinenprogramm (*Bild*), das in einer REM-Zeile unterzubringen ist und im Speicher frei verschoben werden darf. Die folgende Programmbeschreibung zeigt Kennern des ZX-81-Betriebssystems, wie's gemacht wird.

Zunächst machen wir von einem Trick Gebrauch, der in der Serie „Klartext für den ZX 81" Teil 18 (Heft 2/1984) angesprochen wurde. Es geht darum, den neuen Wert für RAMTOP dezimal einzugeben, ohne ihn erst ins Hexadezimalsystem umzurechnen. Dabei hilft die Systemvariable SEED.

Ein dem Basic-Befehl RAND folgender Zahlenwert (1 bis 65535) wird nach dem Drücken von NEWLINE automatisch in den zwei Bytes von SEED (Adressen 16434 und 16435) untergebracht. Der Zahlenwert liegt dann genau in der Form vor, wie sie der ZX 81 weiterverarbeiten kann. RAND 20000 z. B. legt unter Adresse 16434 (4032h) den Wert 32 und unter Adresse 16435 den Wert 78 ab (20000 = 32 + 256*78).

Das Maschinenprogramm beginnt deshalb mit dem „Umladen" dieser Werte in die Systemvariable RAMTOP (Adressen 4004h und 4005h).

Zu beachten ist, daß das RAM bei Adresse 4000h beginnt. Anschließend wird der Stapel neu initialisiert, d. h. es werden Daten auf den Stapel gelegt, an denen das Betriebssystem erkennt, daß hier der Stapel endet (wichtig unter anderem für GOSUB-RETURN). Dies bedingt aber, daß auch die zuvor auf dem Stapel liegenden Rücksprungadressen verloren sind (bei einem Aufruf mittels der USR-Funktion sind das immer mehrere Adressen).

Daher kann die Routine nicht einfach durch ret verlassen werden, sondern es wird die Error-Routine (im ROM) zum Rücksprung in das aufrufende Basic-Programm genutzt. Sie wird durch RST 08 aufgerufen. Das auf den Aufruf folgende Byte FFh wird von der Error-Routine gelesen und bedeutet Error 0; es erfolgt also keine Fehlermeldung mit Programmabbruch.

Noch ein Wort zum Verändern von RAMTOP. Wird diese Systemvariable von Hand geändert und anschließend mit PEEK am Bildschirm angezeigt, so ist tatsächlich der aktuelle Wert zu sehen. Aber wirksam ist er nicht! Denn der direkt unterhalb des ursprünglichen Wertes von RAMTOP liegende GOSUB- und Maschinenstapel hat seine Lage nicht verändert. Erst durch Eingabe von NEW erhalten beide Stapel ihren neuen Platz unter dem gewählten Wert von RAMTOP, und der Speicher oberhalb von RAMTOP ist vor dem Betriebssystem geschützt. NEW ist allerdings nicht unmittelbar nach RAMTOP nötig; zwischenzeitlich dürfen sogar Programme laufen.

Selbstverständlich können zwischen der Stapel-Initialisierung und dem rst-Befehl weitere Befehle eingefügt werden, beispielsweise zum Übertragen eines Maschinenprogramms oder von Daten in den nun geschützten Bereich, dessen Anfangsadresse an dieser Stelle noch in hl-Registerpaar zur Verfügung steht. Michael Schramm

Platzreservierung: Dieses Maschinenprogramm setzt den Wert von RAMTOP herunter, ohne daß durch NEW ein im Speicher abgelegtes Basic-Programm gelöscht wird

Bytes	Mnemonik		Kommentar
2A 32 40	ld	hl,(SEED)	Inhalt von SEED nach
22 04 40	ld	(RAMTOP),hl	RAMTOP umladen
F9	ld	sp,hl	Basic-
01 00 3E	ld	bc,3E00h	GOSUB-
C5	push	bc	Stapel
01 78 06	ld	bc,0678h	neu
C5	push	bc	initialisieren
ED 73 02 40	ld	(4002h),sp	SEED korrigieren
CF	rst	08	Abbruch der Routine, nächste
FF	Datenbyte		Basic-Zeile ausführen

ZX-81-Softwaretip:

Listen am laufenden Band

Allen, die sich schon einmal an den Grenzen des LIST-Befehls beim ZX 81 gestoßen haben, zeigt das hier vorgestellte Programm einen Ausweg. Es ermöglicht ein kontinuierliches Auflisten – d. h. das Programmlisting läuft am Bildschirm ohne Abbruch so lange hoch, bis man es stoppt.

Zentraler Teil des LIST-Programms ist ein Maschinenprogramm, mit dem wir dem ZX 81 zu Leibe rücken. Platz dafür ist in Zeile 1 (*Bild*). Mit den Zeilen 2 bis 6 wird die Startnummer, ab der gelistet werden soll, eingegeben, für den Hausgebrauch des ZX 81 um 1 verringert und dann in der Systemvariablen E-PPC (Zeilennummer, in der der Cursor steht) abgespeichert.

Dann wird mit RAND USR 16514 das erste Maschinenprogramm aufgerufen. Es stellt die höchste im Programm auftretende Zeilennummer fest und lädt sie in die Speicherzellen 16507 und 16508, die zwar im Bereich der Systemvariablen liegen, aber vom Computer nicht belegt werden. Im Programm ist diese Information nötig, um nach Ausgabe der letzten Zeile das Programm zu unterbrechen.

Die Zeilen 8 bis 10 dienen dem Zweck, die PRINT-Position auf Zeile 18 zu verschieben. Dazu wird die Zahl der Leerzeilen im unteren Teil des Bildschirms durch Poken von DF-SZ (Zahl der Zeilen im unteren Teil des Bildschirms) von zwei auf fünf erhöht. Anschließend wirkt auf die Zeilen 0 bis 18 der SCROLL-Befehl. Jetzt wird DF-SZ wieder auf 2 gesetzt. Die PRINT-Position bleibt davon unberührt in Zeile 18. Es können also nun vier Zeilen im unteren Teil des Bildschirms ausgegeben werden.

In Zeile 11 wird überprüft, ob bereits die letzte Zeile ausgegeben wurde, oder ob durch Drücken von NEWLINE das Programm gestoppt werden soll. Wird kein Halt gefordert, ruft der ZX 81 das zweite Maschinenprogramm auf, das bei Adresse 16539 beginnt. Es verschiebt den Programm-Cursor zur folgenden Programmzeile und gibt sie auf dem Bildschirm aus. Die abschließenden SCROLL-Befehle sorgen dafür, daß auch längere Programmzeilen richtig ausgegeben werden. Nach einem Abbruch läßt sich das Auflisten durch Eingabe von CONT fortsetzen. Dabei werden allerdings einige Programmzeilen „verschluckt".

In den Zeilen 50 bis 90 ist der Maschinencodelader aus FUNKSCHAU 12/83 untergebracht. Um die Maschinencodezeile übersichtlicher zu machen, wurde er etwas abgeändert, so daß die Hex-Codes nun durch ein Leerzeichen (SPACE) getrennt einzugeben sind!

Zum Gebrauch: Erst das Programm eingeben und dann sicherheitshalber durch GOTO 17 abspeichern. Mit GOTO 50 sind anschließend die Maschinenprogramme zu laden. Nach dem Start durch RUN läßt sich die Startzeile eintippen (NEWLINE nur kurz drücken, sonst stoppt der Computer). Wenn jetzt kein Listing läuft, liegt ein Eingabefehler vor. Ist dagegen alles in Ordnung, können die Zeilen 30 bis 90 gelöscht und darf das Programm endgültig abgespeichert werden.

Im Listing fällt auf, daß jede Zeile mit Programm-Cursor ausgegeben wird, aber nach Stopp und erneutem Drücken von NEWLINE zeigt sich, daß er in der zuletzt ausgegebenen Zeile steht. Für Programmzeilen großer Länge muß statt des Wertes 5 in Zeile 8 ein höherer Wert gewählt werden, und wem es zu schnell geht, der muß eine Pause einfügen. Das fertige Programm sollte vor der Eingabe größerer Programme geladen werden. Da es am Anfang steht, genügt dann zum Aufrufen die Eingabe von RUN. Andreas Brecht

```
   1 REM REMZEILE ZUR AUFNAHME V
ON 2 MASCHINENCODEPROGRAMMEN
   2 PRINT "STARTZEILE EINGEBEN"
,"STOP MIT NEULINE"
   3 INPUT START
   4 LET START=START-1
   5 POKE 16394,START-256*INT (S
TART/256)
   6 POKE 16395,INT (START/256)
   7 RAND USR 16514
   8 POKE 16418,5
   9 SCROLL
  10 POKE 16418,2
  11 IF CODE INKEY$=118 OR (PEEK
 16507<=PEEK 16394 AND PEEK 1650
8<=PEEK 16395) THEN STOP
  12 RAND USR 16539
  13 SCROLL
  14 SCROLL
  15 SCROLL
  16 GOTO 8
  17 SAVE "LIS■"
  30 LIST
  50 LET A$="01 04 04 2A 0C 40 2
B 2B 3E 77 3D ED B9 23 23 7E 32
7C 40 23 7E 32 7B 40 C9 2A 0A 40
 23 22 0A 40 CD D8 09 7E 32 0B 4
0 23 7E 2B 32 0A 40 1E 00 CD 45
07 C9"
  60 LET ADR=16514
  70 FOR N=0 TO LEN A$-2 STEP 3
  80 POKE ADR+INT (N/3),16*(CODE
 A$(N+1)-28)+CODE A$(N+2)-28
  90 NEXT N
```

LIST-Programm: Damit kann man ein Programm vollständig (ohne Zwangsabbruch bei vollem Bildschirm) listen lassen. Hat das Eingabeprogramm ab Zeile 30 seinen Zweck erfüllt, darf es gelöscht werden

ZX-81-Softwaretip:

Gegen hektischen Bildaufbau

Bei langen Programmen in Basic kommt es unweigerlich immer wieder zu der unangenehmen Situation, daß die Zeile, die man gerade neu eingibt, nicht mehr auf den Bildschirm paßt. Der ZX 81 baut das Bild dann in mehreren Anläufen auf, was durchaus schon einige Sekunden dauern kann und recht nervenaufreibend ist. Auch der Befehl LIST xx funktioniert im allgemeinen nicht, weil der Computer schon beim nächsten NEWLINE wieder macht, was er will.

Ein einfaches und wirksames Gegenmittel ist die Anweisung POKE 16420,100. Sie bewirkt, daß die Zeile, in der der Cursor vor der Anweisung stand, jetzt die erste angezeigte Zeile ist – und man hat seine Ruhe.

Horst Kling

Berichtigung

Messer und Gabel

FUNKSCHAU 1983, Heft 12, Seite 74

Im Schaltplan der 6-KByte-RAM-Erweiterung ist uns bei IC 3 die Pin-Numerierung durcheinandergeraten: Nr. 5 ist gegen Nr. 6 und Nr. 14 gegen Nr. 15 zu vertauschen. Auf der Platine ist nichts zu ändern – dort stimmt die Leiterbahnführung.

Ein Demonstrationsprogramm beseitigt letzte Zweifel

Das gezeigte kleine Programm (*Bild*) will lediglich das Prinzip an einem Beispiel verdeutlichen. In diesem Programm sind alle wesentlichen Elemente des Verfahrens enthalten, so daß lediglich Zeile 420 dem vorgesehenen Anwendungszweck angepaßt werden muß, beispielsweise mit einem GOSUB-Befehl.

Es ist nicht nur wichtig, die erste Zeile jedes DATA-Blocks zu finden, sondern auch in irgendeiner Form das Ende der zu lesenden DATA-Zeilen zu erkennen. Hierzu dient die Variable DE. Ihr Wert ist der errechnete Anfang des nächstliegenden DATA-Blocks. Das Programm gibt dann alle zum angewählten DATA-Block gehörenden DATA-Zeilen aus (Zeile 420).

Um mit einem Rechenformalismus die anzuwählenden DATA-Zeilen berechenbar zu machen, sollen die DATA-Blöcke einen gleichbleibenden Abstand haben. Dieser wird mit der Variablen DS festgelegt.

Die Zeilennummer der ersten DATA-Zeile wird der Variablen D1 zugewiesen. Die erste Zeile (DA) jedes DATA-Blocks kann jetzt nach der Regel DA = N*DS+DB berechnet werden. Hierbei ist N die eingegebene Codenummer (hier 1, 2, 5 oder 10) und DB die Differenz D1−DS.

Damit auch nach dem höchsten DATA-Block (Codenummer N = 99 ≙ Zeile 10800) ein Rücksprung sichergestellt ist, muß als letzte Zeile DATA"" angefügt sein. Die zugehörige Zeilennummer errechnet sich aus:

$N_{max} * DS + D1$

Für das Demonstrationsprogramm ergibt das 99*100+1000 = 10900 als Zeilennummer der letzten DATA-Anweisung. In der Regel wird die höchste Codenummer weit unter 99 liegen. Dies hängt auch vom Speicherplatz ab.

Fehlt eine aufgerufene DATA-Zeile, dann wird eine Fehlermeldung ausgegeben. Verlangt man z. B. einen zur Codenummer 3 gehörenden DATA-Block, dann erfolgt die Meldung

3 = DATA 1200 FEHLT.

Verantwortlich dafür ist der Vergleich in Zeile 320.

Selbstverständlich funktioniert die beschriebene Methode auch bei anderen Computer-Modellen. Hierbei ist lediglich die aktuelle DATA-Zeilennummer unter einer anderen Adresse zu finden, die Variablen DL und DH erhalten also einen anderen Wert. Außerdem sind die Zeilen 100 und 200 an den jeweiligen Basic-Dialekt anzupassen.

Alex Pütz

Tabelle: Bedeutung der im Programm verwendeten Variablen

D1 = 1. DATA-Zeilennummer
DS = Abstand der DATA-Blocks
DL = L-Byte der DATA-Zeilennummer
DH = H-Byte der DATA-Zeilennummer
N = Codenummer des DATA-Blocks
DA = 1. Zeile des DATA-Blocks
DE = 1. Zeile des nächsten DATA-Blocks
DZ = aktuelle gelesene Zeilennummer
D$ = in DATA-Zeile gespeicherte Daten
DV = Dummy-Variable für <RETURN>

Stichworte zum Inhalt

DATA-Zeilen, VC 20, Zeilennummer, PEEK, DATA-Block.

Klartext im Detail:

Verwirrung durch Register

Maschinensprache ist eine ausgefeilte Angelegenheit. Das schützt jedoch nicht vor Widersprüchen, wie sie teilweise in der Assemblerschreibweise und in Programmbeschreibungen auftreten. Fragen wir uns zunächst, was ein Register eigentlich ist?

Bisher kennen wir ein Register als basicähnliche Variable. Tatsächlich aber ist ein 8-Bit-Register bloß eine bessere 8-Bit-Speicherzelle. Eine Speicherzelle nämlich, die weder eine 16-Bit-Adresse hat, noch im RAM-Bereich liegt, sondern die kurz mit a, b, c, usw. bezeichnet wird und direkt in der CPU zu finden ist! Der Inhalt eines Registers muß demnach der Wert sein, der sich gerade in dem Register befindet. Dies bedingt folgende, sachlich korrekte aber umständliche Formulierungen:

○ *inc b:* Erhöhe den Inhalt des b-Registers.

○ *inc hl:* Erhöhe den Inhalt des hl-Registerpaares.

○ *inc (hl):* Erhöhe den Inhalt derjenigen Speicherzelle, die durch den Inhalt des hl-Registerpaares festgelegt ist.

Jetzt könnte einer fragen: Warum lautet die Z-80-Assemblerschreibweise dann *ld a, N* und nicht *ld (a), N*? Denn eigentlich wird durch diese Operation das Datenbyte N zum Inhalt des Akkumulators. Weiterhin müßte es dann auch *inc (b)* heißen!

Üblicherweise läßt man jedoch die erste Klammerebene in der Assemblerschreibweise weg. Dies verleitet freilich zu satzbaulicher Bequemlichkeit. Nicht selten findet man deshalb in Handbüchern und dokumentierten Listings folgende Schreibweise:

○ *inc b:* Erhöhe das b-Register.

○ *inc hl:* Erhöhe das hl-Registerpaar.

○ *inc (hl):* Erhöhe den Inhalt des hl-Registerpaares.

Wer nun in einer Erläuterung liest „...wird der Inhalt des hl-Registerpaares erhöht...", der muß auf alles gefaßt sein. Ratsam ist dann ein sofortiger Blick aufs Listing!

Im Rahmen der Serie „Klartext für den ZX-81" wollen wir folgende Formulierungen zulassen:

○ *inc b:* Erhöhe den Inhalt des b-Registers; eventuell in einer kurzen Programmbeschreibung: Erhöhe das b-Register.

○ *inc hl:* Erhöhe den Inhalt des hl-Registerpaares; eventuell: Erhöhe das hl-Registerpaar.

○ *inc (hl):* Erhöhe den Inhalt der Speicherzelle, die das hl-Registerpaar festlegt; auf gar keinen Fall: Erhöhe den Inhalt des hl-Registerpaares!

Klaus Herklotz

Berichtigung

Die Folie bekennt Farbe

FUNKSCHAU 1983, Heft 18, Seite 85

Ausgerechnet bei einem Beitrag mit diesem Titel ist uns ein Fehler bei der Farbgebung unterlaufen: Im Bild von der ZX-81-Tastatur ist die Verbindung zwischen den Tasten V und B blau gedruckt, und das ist falsch; richtig muß diese Verbindung schwarz gezeichnet sein und die äußeren Kontaktflächen verbinden.

ZX-81-Softwaretip:

Laufzeit-Analyse

Ein verblüffendes Ergebnis erbrachte die Laufzeit-Analyse an zwei verschiedenen ZX-81-Computern: Der eine Computer ist fast doppelt so schnell wie der andere!

Durch geschicktes Programmieren ist es möglich, Programme „schneller" zu machen. Eine Voraussetzung dafür ist die Kenntnis über die Ausführungszeit einzelner Funktionen. So dauert z. B. beim ZX 81 das Potenzieren etwa 50mal länger als das Multiplizieren. Mit dem nachfolgend beschriebenen kurzen Programm läßt sich die Laufzeit ganzer Programmteile (z. B. Unterprogramme oder Schleifen) ermitteln.

Zentraler Teil des Programms ist Zeile 60 (*Bild 1*). Hier wird der Variablen X das Ergebnis der zu untersuchenden Funktion bzw. Operation zugewiesen. Um bei der späteren Zeitmessung auch die Dauer der Zuweisung selbst zu berücksichtigen, steht in Zeile 60 zunächst LET X=A. Diese Anweisung wird in einer Schleife 100mal ausgeführt um Abweichungen auszugleichen, die bei einmaliger Ausführung auftreten können.

Zur Zeitmessung wird die Systemvariable FRAMES verwendet (Adressen 16 436 und 16 437). Sie zählt die Anzahl der ausgegebenen TV-Bilder (50 pro Sekunde), indem ihr Wert, ausgehend vom Startwert 65 536, alle 20 ms um 1 verringert wird. Zum Zurücksetzen dieses internen Zeitgebers dient Zeile 40. Nach dem Ausführen der Schleife wird mit Zeile 80 der Zählerstand der Systemvariablen abgefragt. Zeile 90 übernimmt das Umrechnen in Millisekunden und normiert ihn so, daß mit der Zuweisung X = A (Zeile 60) exakt der Wert 0 geliefert wird.

Da der Normierungsfaktor von Rechner zu Rechner abweichen kann, ist es möglich, daß nach dem Starten des Programms ein von 0 verschiedener Wert auftritt! Dann ist das fünffache dieses Wertes vom Normierungsfaktor 65 034 (Zeile 90) abzuziehen. Gibt ein ZX 81 also z. B. −1.2 aus, dann ist 65 034 durch 65 040 zu ersetzen: 65 034 −(−1.2 × 5) = 65 040.

Damit die benötigten Variablen schon vor der Zeitmessung bekannt sind, werden sie in den Zeilen 10 bis 30 angelegt und initialisiert.

Um nun die Rechenzeit z. B. für eine Addition zu ermitteln, ersetzt man Zeile 60 durch 60 LET X=A+A und startet das Programm mit RUN. Das Ergebnis ist diejenige Zeit, welche das Statement LET X=A+A länger dauert als das Statement LET X=A. Die beiden Anweisungen unterscheiden sich aber nur durch die Addition und den zusätzlichen Zugriff auf die Variable A.

Tabelle: Ausführungszeiten einzelner Operationen und Funktionen. Der ZX 81 der FUNKSCHAU (farbig unterlegte Zeiten) ist fast doppelt so flott, wie der des Autors

−A	3,2 ms	1,8 ms
A+A	9,0 ms	5,0 ms
A−A	16,8 ms	9,2 ms
A*A	17,4 ms	9,6 ms
A/A	22,6 ms	12,4 ms
A**A	831,8 ms	456,0 ms
SQR A	835,6 ms	458,0 ms
EXP A	301,0 ms	165,0 ms
LN A	493,8 ms	270,6 ms
PI	4,8 ms	2,6 ms
SIN A	288,6 ms	158,2 ms
COS A	300,2 ms	164,6 ms
TAN A	604,6 ms	382,6 ms
ASN A	1284,4 ms	704,0 ms
ACS A	1296,8 ms	710,8 ms
[illegible]	[illegible]	[illegible]
INT A	8,4 ms	4,6 ms
ABS A	3,6 ms	2,0 ms
SGN A	4,2 ms	2,2 ms
RND	85,8 ms	47,0 ms
PEEK A	8,8 ms	5,2 ms
VAL A$	428,6 ms	235,0 ms
CODE A$	8,8 ms	4,8 ms
LEN A$	8,8 ms	4,8 ms

```
 10 LET A=PI/10
 20 LET A$=STR$ A
 30 LET X=0
 40 PAUSE 0
 50 FOR I=1 TO 100
 60 LET X=A
 70 NEXT I
 80 LET Z=PEEK 16436+256*PEEK 1
6437
 90 PRINT .2*(65034-Z)
100 STOP
```

① **Listing „Laufzeit-Analyse":** Damit läßt sich die Ausführungszeit einzelner ZX-81-Befehle ermitteln

Das Programm berechnet also im wesentlichen die Rechenzeit für eine Addition (die zusätzliche Zugriffszeit kann dabei vernachlässigt werden).

Größere Programme können in die Zeilen 51 bis 69 geschrieben oder als Unterprogramm aufgerufen werden. Dann muß man aber auch die Zeit für GOSUB und RETURN berücksichtigen, und das macht eine erneute Anpassung der Normierungskonstante in der angegebenen Weise notwendig.

Die Resultate verschiedener Berechnungen sind in der *Tabelle* zusammengefaßt, wobei die farbig abgesetzten Werte mit dem ZX 81 der FUNKSCHAU-Redaktion ermittelt wurden. Die genaue Ursache der erheblichen Abweichungen – die sich schon durch einen anderen Normierungsfaktor angekündigt haben – ist unbekannt. Da sich die Werte jedoch auch nach dem Überprüfen mit einer Stoppuhr als korrekt erwiesen haben, ist der Schluß zulässig: Es gibt unterschiedlich schnelle ZX 81! Wer sich nicht auf die modellunabhängige Aussage verlassen möchte, daß z. B. eine Addition nur halb so lange dauert wie eine Multiplikation, sollte deshalb die Laufzeit-Analyse für absolute Zahlenwerte auf seinem ZX 81 nachvollziehen.

Die Resultate variieren etwas nach oben oder unten, wenn man die Variable durch Konstanten ersetzt (z. B. X = 3**3 anstelle von X = A**A). Dies ist bei anderen Rechnern i. a. nicht so und durchaus einen Vergleich wert. Generell gilt aber, daß langsame Funktionen wie EXP, LN oder TAN auch auf anderen Rechnern langsam sind. Das Programm kann nur im SLOW-Modus zur Zeitmessung verwendet werden. Im FAST-Modus wird die Systemvariable FRAMES nicht verändert, da keine Bildausgabe erfolgt.

Abschließend sei noch darauf hingewiesen, daß auch die Zugriffszeiten auf die Variablen variieren. Und zwar werden sie um so größer, je später die Variable im Programm vereinbart wird (das hängt mit der Speicherorganisation zusammen). Michael Redmann

ZX-81-Tastatur:

Die Folie bekennt Farbe

Bevor man darangeht, dem ZX 81 mit einer mechanischen Tastatur Marke Eigenbau auf die Sprünge zu helfen, muß man sich mit der Folientastatur auseinandersetzen.

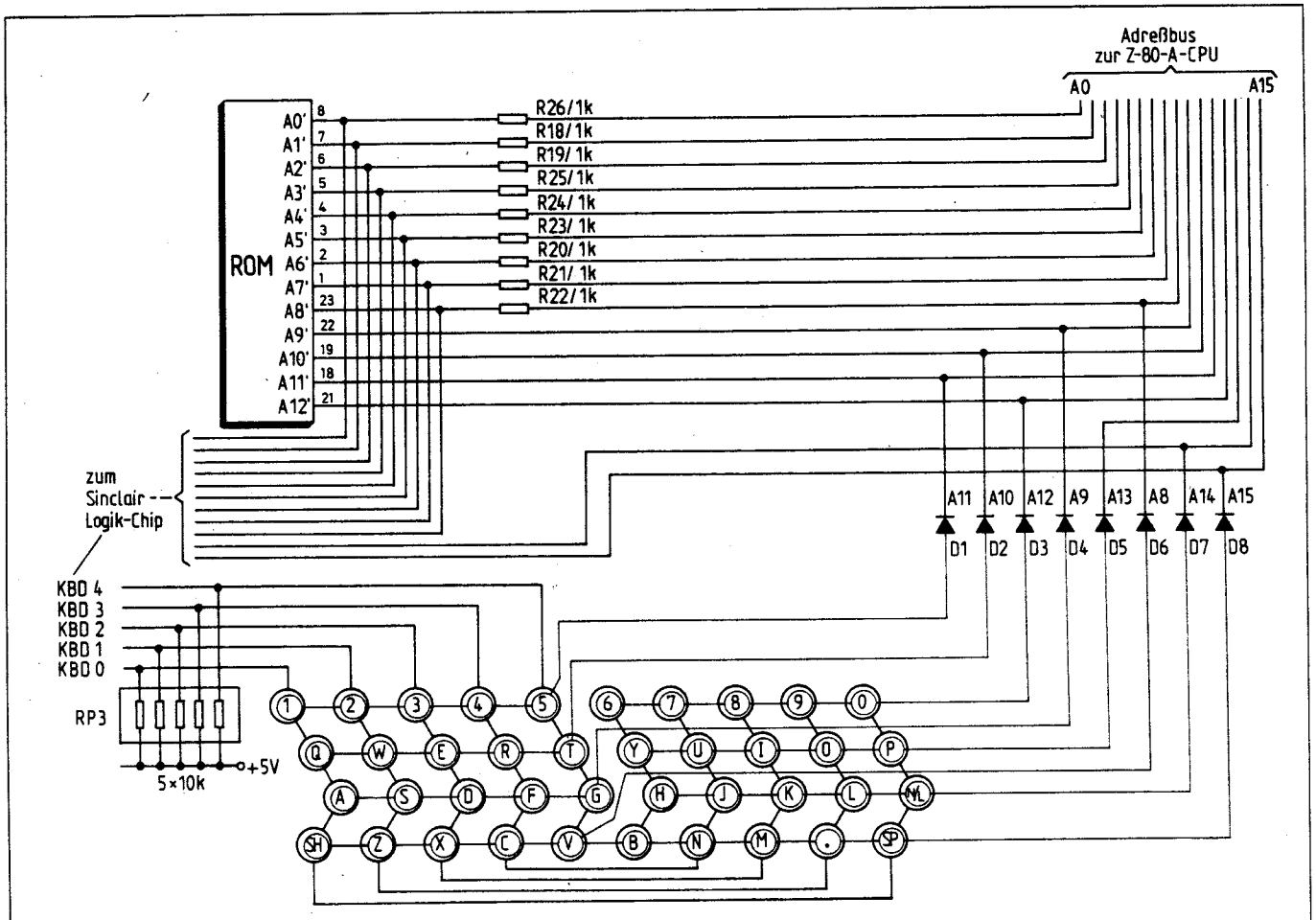
Die Tastatur des ZX 81 besteht aus drei übereinandergeklebten Folien. Auf der oberen und der unteren sind auf den einander zugewandten Seiten Leiterbahnen und Kontaktpunkte aus einer sehr dünnen Metallschicht aufgebracht. Die mittlere, etwas dickere Folie dient als Trennschicht und hat an den Kontaktpunkten Löcher.

Die Verbindung von der Tastatur zur Leiterbahnplatte des ZX 81 erfolgt ebenfalls durch zwei Folienstreifen mit fünf bzw. acht Leiterbahnen. Die genaue Kontaktbelegung der Tastatur und deren Anschluß an die Leiterplatte zeigt das *Bild*.

Zeilenleitungen teilen die Tastatur

Jede Taste hat die Funktion eines Schließers, wobei stets die blaue und die schwarze Kontaktfläche miteinander verbunden werden. Die fünf Leitungen KBD 0 bis KBD 4 sind dabei die Spaltenleitungen der Tastaturmatrix. Fünf Leitungen genügen, da die Tastatur von den Zeilenleitungen in zwei Hälften geteilt wird (A 11, A 10, A 9, A 8 und A 12, A 13, A 14, A 15). Trotz gleicher Spaltenleitungen für beide Hälften läßt sich so jede gedrückte Taste eindeutig lokalisieren. Die Spaltenleitung KBD 3 z. B. ist für die Tasten 4, R, F, C sowie N, J, U und 7 zuständig.

Wird eine Taste betätigt, so wird die isolierende Luftschicht überwunden, und die Kontaktpunkte berühren sich.



ZX-81-Tastatur: Ein Tastendruck verbindet die blaue und die schwarze Kontaktfläche. L-Pegel auf einer der Adreßleitungen A 8 bis A 15 ist dann auch für die betroffene Spaltenleitung KBD 0 bis KBD 4 gültig

L-Pegel meldet Tastendruck

Ist noch kein Kontakt geschlossen, so liegt an den 5 Anschlußpunkten KBD 0 bis KBD 4 eine Spannung von etwa 5 V an, also der Pegel für logisch H. Dafür sind die fünf Pull-up-Widerstände im Widerstandsnetzwerk RP 3 verantwortlich. An den restlichen Anschlußpunkten A 8 bis A 15 liegt beim Abfragen der Tastatur durch die CPU L-Pegel an. Schließt man nun einen Kontakt, so wird der Pegel an einem der Anschlußpunkte KBD 0 bis KBD 4 auf logisch L heruntergezogen, sobald am entsprechenden Anschluß A 8 bis A 15 ebenfalls logisch L liegt.

Ein Beispiel soll den Signalfluß bei einem Tastendruck verdeutlichen. Gedrückt sei die Taste G: Durch sein Betriebsprogramm wird der ZX 81 gezwungen etwa alle 400 ms die Tastatur abzufragen, das heißt, er legt kurzzeitig auf eine der Adreßleitungen A 8 bis A 15 L-Pegel (bei gedrückter SHIFT-Taste wird auf zwei Adreßleitungen L-Pegel gelegt). Sobald die Adreßleitung A 9 L-Pegel führt, wird auch die Spaltenleitung KBD 4 auf L-Potential gezogen, da Taste G gedrückt ist. Der Potentialwechsel der Spaltenleitung von H nach L wird vom Sinclair-Logik-Chip (ein von Ferranti im Kundenauftrag gefertigtes Gate-Array) registriert, das nun zusammen mit dem Adreßsignal dafür sorgt, daß im 8-KByte-ROM der Zeichengenerator aktiviert wird. Die dem Zeichen „G" zugeordneten Bitmuster (8 Byte, da ein Zeichen mit einer 8×8-Punktmatrix dargestellt wird) gelangen damit auf den Datenbus, und das Zeichen wird am Bildschirm geschrieben.

Eine komplette Zusammenstellung aller möglichen Tastenkombinationen und der daraus resultierenden logischen Pegel an den Anschlußpunkten ist in der *Tabelle* wiedergegeben.

Hans-Jürgen Ollech

Stichworte zum Inhalt

Folientastatur, ZX 81, Schließer, Pull-up-Widerstand, Sinclair-Logik-Chip.

ZX-81-Hardwaretip:

Höherer Pegel bei SAVE

Der beim Abspeichern von Programmen am Mic-Ausgang des ZX-81 auftretende Pegel ist für viele Kassettenrecorder und Tonbandgeräte zu gering, um das Magnetband voll aussteuern zu können. Die Folge sind Aufzeichnungen mit geringer Dynamik und verhältnismäßig hohem Störpegel, die bei einem späteren LOAD Einleseprobleme bereiten.

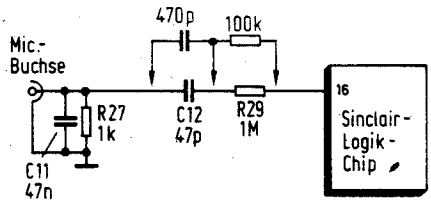
Abhilfe läßt sich schaffen durch einen kleinen Eingriff in den Computer. Es genügt, parallel zu dem RC-Glied zwischen dem Sinclair-Logik-Chip und der Mic-Buchse (im Schaltplan R 29 bzw. C 12) ein weiteres RC-Glied mit kleinerem Wechselstromwiderstand zu schalten (bewährt hat sich die Kombination von 100 kΩ und 470 pF). Das *Bild* macht dieses deutlich; es müssen lediglich an drei Punkten Lötungen vorgenommen werden. Auch, wenn man keinen Schaltplan des ZX-81 besitzt, lassen sich C 12 (47 pF) und R 29 (1 MΩ) schnell finden, indem man die Leiterbahnen verfolgt, die von der Mic-Buchse ausgehen.

Durch die beschriebene Maßnahme wird die Signalform beim Abspeichern nicht verfälscht, der Pegel jedoch deutlich angehoben, so daß zum Beispiel auch ein Recorder mit DIN- oder Cinch-Eingangsbuchse zur Programmaufzeichnung verwendet werden kann.

Michael Schramm

	Eingänge					Ausgänge							
Taste	KBD4	KBD3	KBD2	KBD1	KBD0	A11	A10	A12	A9	A13	A8	A14	A15
1	H H	H H	H H	H H	L L	L L	–	–	–	–	L –	–	–
2	H H	H H	H H	L L	L H	L L	–	–	–	–	L –	–	–
3	H H	H H	L L	H H	L H	L L	–	–	–	–	L –	–	–
4	H H	L L	H H	H H	L H	L L	–	–	–	–	L –	–	–
5	L L	H H	H H	H H	L H	L L	–	–	–	–	L –	–	–
6	L L	H H	H H	H H	L H	–	–	L L	–	–	L –	–	–
7	H H	L L	H H	H H	L H	–	–	L L	–	–	L –	–	–
8	H H	H H	L L	H H	L H	–	–	L L	–	–	L –	–	–
9	H H	H H	H H	L L	L H	–	–	L L	–	–	L –	–	–
0	H H	H H	H H	H H	L L	–	–	L L	–	–	L –	–	–
Q	H H	H H	H H	H H	L L	–	L L	–	–	–	L –	–	–
W	H H	H H	H H	L L	L H	–	L L	–	–	–	L –	–	–
E	H H	H H	L L	H H	L H	–	L L	–	–	–	L –	–	–
R	H H	L L	H H	H H	L H	–	L L	–	–	–	L –	–	–
T	L L	H H	H H	H H	L H	–	L L	–	–	–	L –	–	–
Y	L L	H H	H H	H H	L H	–	–	–	–	L L	L –	–	–
U	H H	L L	H H	H H	L H	–	–	–	–	L L	L –	–	–
I	H H	H H	L L	H H	L H	–	–	–	–	L L	L –	–	–
O	H H	H H	H H	L L	L H	–	–	–	–	L L	L –	–	–
P	H H	H H	H H	H H	L L	–	–	–	–	L L	L –	–	–
A	H H	H H	H H	H H	L L	–	–	–	L L	–	L –	–	–
S	H H	H H	H H	L L	L H	–	–	–	L L	–	L –	–	–
D	H H	H H	L L	H H	L H	–	–	–	L L	–	L –	–	–
F	H H	L L	H H	H H	L H	–	–	–	L L	–	L –	–	–
G	L L	H H	H H	H H	L H	–	–	–	L L	–	L –	–	–
H	L L	H H	H H	H H	L H	–	–	–	–	–	L –	L L	–
J	H H	L L	H H	H H	L H	–	–	–	–	–	L –	L L	–
K	H H	H H	L L	H H	L H	–	–	–	–	–	L –	L L	–
L	H H	H H	H H	L L	L H	–	–	–	–	–	L –	L L	–
NL	H H	H H	H H	H H	L L	–	–	–	–	–	L –	L L	–
SH	H H	H H	H H	H H	L L	–	–	–	–	–	L L	–	–
Z	H H	H H	H H	L L	L H	–	–	–	–	–	L L	–	–
X	H H	H H	L L	H H	L H	–	–	–	–	–	L L	–	–
C	H H	L L	H H	H H	L H	–	–	–	–	–	L L	–	–
V	L L	H H	H H	H H	L H	–	–	–	–	–	L L	–	–
B	L L	H H	H H	H H	L H	–	–	–	–	–	L –	–	L L
N	H H	L L	H H	H H	L H	–	–	–	–	–	L –	–	L L
M	H H	H H	L L	H H	L H	–	–	–	–	–	L –	–	L L
.	H H	H H	H H	L L	L H	–	–	–	–	–	L –	–	L L
SP	H H	H H	H H	H H	L L	–	–	–	–	–	L –	–	L L

Tabelle: Den Tasten zugeordnete Pegel auf den 13 Tastaturzuleitungen. Schwarz: ohne gedrückte SHIFT-Taste (auch gültig für F-Modus und G-Modus ohne SHIFT). Blau: mit gedrückter SHIFT-Taste (auch gültig für G-Modus mit SHIFT)



Höherer Ausgangspegel beim ZX-81: Ein zusätzliches RC-Glied senkt den Wechselstromwiderstand bei gleichbleibender Filtercharakteristik