Sonderheft Nr. 221
Preis DM 18.–
öS 137.–, sfr 18.–

Funkschau

ZX-81-Kochbuch 2

Für Computer-Fans & Hobby-Elektroniker

Bauanleitungen
Tips & Tricks
Arbeitshilfen

Wenn Sie die Kommunikations- und Unterhaltungselektronik immer im Auge behalten wollen:

Die Funkschau informiert in Fachbeiträgen alle 14 Tage umfassend zu den Themen Kommunikations- und Computertechnik, Audio und Video.

Technische Beiträge

Beschreibung von Geräten, Systemen und ihren Funktionen – einschließlich technischer Details, wenn sie für den Anwender von Bedeutung sind.

Erfahrungsberichte

Geräte der elektronischen Kommunikationstechnik und der Konsumelektronik im praktischen Gebrauch. Die FUNKSCHAU beschreibt den Nutzen für den Anwender.

Reports

Aktuelle Elektronik-Themen aus unterschiedlichen Blickwinkeln betrachtet. Runduminformationen, die neben der Technik auch gesellschaftspolitische Aspekte neuer Entwicklungen beleuchten.

Nachrichten

Was für den Leser zur Beurteilung der Technik des Elektronik-Marktes wichtig ist, wird übersichtlich aufbereitet und mit Stellungnahmen ergänzt.

Produktneuheiten

Das Schaufenster des Elektronik-Marktes. Neue Produkte werden mit den wichtigsten technischen Daten, Anwendungsbeispielen und Bezugsquellenangaben präsentiert.

Service-Beiträge

Arbeitshilfe für Service-Techniker aller Elektronikbereiche: Arbeitsblätter mit Grundschaltungen der Elektronik, Schaltungsdetails, Schaltungsapplikationen, Meßtechnik, Fehlerquellensuche...

Bauanleitungen

Für den anspruchsvollen Hobby-Elektroniker zum sicheren Nachbau. Beispiele: Meßgeräte, Videomischpulte, Heizungsregelung per Computer, Alarmanlagen, Musiksynthesizer...

Wenn Sie sich vom Nutzen der FUNKSCHAU für Ihre Arbeit überzeugen wollen, schicken wir Ihnen gerne im Rahmen unseres Kennenlern-Angebots die neueste Ausgabe kostenlos und unverbindlich ins Haus.
Die vorbereitete Karte finden Sie an der Umschlagklappe.

Funkschau
Zeitschrift für Unterhaltungselektronik und Kommunikationstechnik

Liebe Leser,

„Aller guten Dinge sind drei", lautet ein Sprichwort. Aber diesmal stimmt es nicht, denn das ZX-81-Kochbuch II ist bereits das vierte Sonderheft der FUNKSCHAU für diesen Heimcomputer. Von zwei neuen Beiträgen abgesehen, haben wir hier für Sie die besten ZX-81-Beiträge aus den FUNKSCHAU-Heften von Ende 1984 bis Anfang 1986 zusammengestellt. Und wie gewohnt sind es Beiträge, die sich an aktive Computer-Fans und Hobby-Elektroniker richten – schließlich lädt der ZX 81 noch immer so richtig zum Experimentieren ein.

Einige Bauanleitungen verlangen nach der „Z-80-PIO", die sowohl in der FUNKSCHAU als auch im ZX-81-Kochbuch schon vorgestellt wurde. Wir haben deshalb hier auf eine nochmalige Veröffentlichung verzichtet. Wenn Sie an diese Bauanleitung nicht mehr herankommen sollten, rufen Sie in der Redaktion an (0 89/5117-315), wir helfen Ihnen dann weiter. Und weil wir den Heftpreis wieder knapp kalkuliert haben, wurden die FUNKSCHAU-Beiträge unverändert in dieses Sonderheft übernommen (zwischenzeitlich bekanntgewordene Fehler haben wir selbstverständlich korrigiert). Sie stoßen deshalb häufig auf Querverweise auf FUNKSCHAU-Hefte. Aber nur keine Panik: Bis auf die Querverweise auf die Z-80-PIO („Datendrehscheibe") sind die Beiträge in sich geschlossen, und die Hefthinweise signalisieren lediglich, wo etwas ausführlicher nachzulesen ist. Außerdem beziehen sich die Querverweise oft auf Beiträge, die Sie auch im ZX-81-Kochbuch II finden.

Wir haben die Programme und Bauanleitungen dieses Sonderheftes sorgfältig geprüft, so daß Fehler sehr unwahrscheinlich sind, zumal die meisten Beiträge ihre Bewährungsprobe schon bei den FUNKSCHAU-Lesern machen mußten. Nach dem Gesetz des „gemeinsten Zufalls" ist es dennoch möglich, daß wir eine verborgene Fußangel übersehen haben. Wenn also etwas nicht so läuft, wie es sollte, dann rufen Sie uns an. Aber bitte erst dann, wenn Sie sicher sind, selbst keinen Fehler gemacht zu haben.

In unserem letzten ZX-81-Sonderheft haben wir uns noch mit „Auf Wiedersehen" verabschiedet. Diesmal sagen wir „Ade", denn der ZX 81 wird nicht mehr gebaut und Sir Clive Sinclair hat seine Firma verkauft. Die Ära des ZX 81 ist damit unwiderruflich zu Ende. Doch sollten Sie für den kleinsten aller Heimcomputer später einmal keine Aufgabe mehr haben, werfen Sie ihn nicht gnadenlos weg: Der kleine rabenschwarze Geniestreich hat Geschichte gemacht und wird vielleicht einmal begehrtes Sammelobjekt.

Ihre Funkschau-Redaktion

ZDE

Inhaltsverzeichnis

Fachaufsätze

Software

Hardware

Kurzbeiträge

Software

Hardware

Impressum:

1986:
Franzis-Verlag GmbH, Karlstr. 37–41, 8000 München 2. Bearbeitet von der Redaktion der Zeitschrift FUNKSCHAU.
Für den Text verantwortlich: Ing. (grad.) Stephan Schall. Titelgestaltung: Dipl.-Designer (FH) Volker Hilbel

ISSN 0172-2778

Gesamtherstellung: Franzis-Technik GmbH, München. Printed in Germany. Imprimé en Allemagne.
MK-Artikel Nr. 221011 · F/MK/586/1207/10'

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.

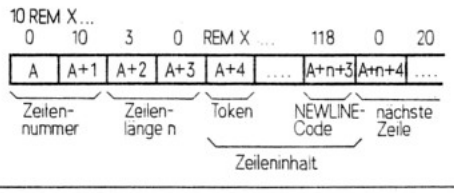
Die LIST-Sperre wird nun folgendermaßen errichtet: Zunächst gibt man vor dem zu schützenden Programm die Zeile 1 REM X ein. Das X muß anschließend durch ein Byte mit Inhalt 118 ersetzt werden, was durch den Befehl POKE 16514,118 möglich ist. Wenn die LIST-Routine auf diese Zeile stößt, „glaubt" sie, 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 "TEST"
 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: 166    16551: 194
16552: 174    16553: 6
16554: 86     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.

```
  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

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-

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: 16  64  35  78  35  35
16550: 237 176 27  235 203 254
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

ZX-81-Softwaretip:

Schloß und Schlüssel II

Tips zum Softwareschutz

Kürzlich wurden in der FUNKSCHAU verschiedene Methoden beschrieben, ZX-81-Programme vor fremdem Zugriff zu schützen und den Programmschutz mit List zu umgehen. Zwei weitere, noch trickreichere Verfahren schließen jetzt dieses Thema für den ZX 81 ab.

Die Weisheit, wonach nichts so gut ist, daß es sich nicht verbessern ließe, ist nur allzu wahr. So ist der in Heft 23/1984 auf Seite 87 beschriebene „Kennwort-Schutz" für ein ZX-81-Programm zwar recht wirksam, aber Könnern in Sachen „knacken" kann er nicht widerstehen.

Der Kennwort-Schutz wird überlistet

Der Nachteil des Kennwort-Schutzes ist, daß das zu schützende Programm im Klartext erhalten bleibt. Und wenn ein falsches Kennwort eingetippt wird, dann stürzt das Programm zwar pflichtgemäß ab, aber das passiert nur dann, wenn es wie üblich im Programmspeicher des Computers untergebracht ist.

Lädt man das Programm jedoch in einen eigens dafür reservierten Speicherbereich und modelt die LIST-Routine etwas um, so ist es um den Kennwort-Schutz geschehen. Dann nämlich kann man das Programm listen und in aller Ruhe nach dem richtigen Kennwort durchforschen.

Das Hilfsprogramm in *Bild 1* ist so ein Bezwinger des Kennwort-Schutzes. Funktionsfähig wird es, wenn die in *Bild 2* gezeigten Dezimalcodes des zugehörigen Maschinenprogramms mit Hilfe eines Dezimal-Laders ihren Platz in Zeile 1 eingenommen haben. Das Maschinenprogramm ist praktisch eine Kopie der LOAD-Routine im ROM – mit einem kleinen Unterschied: Programme werden nicht ab Adresse 16393, sondern ab Adresse 20000 in den Speicher des ZX 81 geladen! Außerdem wird RAMTOP ebenfalls auf Adresse 20000 heruntergesetzt.

Nach dem Start mit RUN lädt das Hilfsprogramm das geschützte Programm also in einen Speicherbereich, auf den das Betriebssystem des ZX 81 keinen Zugriff hat – und prompt ist der Kennwort-Schutz hinfällig.

Soll jetzt das Programm von Anfang an gelistet werden, so ist die Frage am Bildschirm „LISTEN AB ZEILE": mit „0" zu beantworten. Das Hilfsprogramm übergibt dann die passende Programm-Anfangsadresse an die LIST-Routine im ROM. Übliches Listen, z. B. um längere Programme zu betrachten, klappt freilich nicht mehr, da sich das Betriebssystem in diesem Fall an das Hilfsprogramm hält, das ja nach wie vor den Programmspeicher einnimmt.

Um weitere Bildschirmseiten zu betrachten, ist daher das Hilfsprogramm mit GOTO 20 erneut aufzurufen und die gewünschte Zeilennummer einzutippen – oder man wählt gleich die Ausgabe auf einem Drucker.

Selbstverständlich werden auch Variablen und Systemvariablen des geschützten Programms geladen. Aber auch sie sind um 3607 Byte nach „oben" hin verschoben – das muß man beim Auswerten beachten. So hat z. B. die erste Systemvariable jetzt die Adresse 20000, und das Programm beginnt mit Adresse 20116.

Für das Betriebssystem des ZX 81 existiert nur das Hilfsprogramm mit seinen Variablen und den aktuellen Systemvariablen. Deshalb kann der Trick des Kennwort-Schutzes, nämlich die Systemvariablen zu verändern, keinen Programmabsturz mehr bewirken. Für den Computer ist das geschützte Programm

```
   1 REM HIER MINDESTENS 146
BELIEBIGE ZEICHEN EINTIPPEN
  10 RAND USR 16514
  20 PRINT "LISTEN AB ZEILE: ";
  30 INPUT Z
  40 POKE 16434,0
  50 POKE 16435,0
  60 IF Z>=1 THEN RAND Z
  70 PRINT Z,"DRUCKERAUSGABE? (J
/N)"
  80 INPUT D$
  90 CLS
 100 RAND USR (16641-4*(D$="J"))
```

① **Lade-Programm:** Nach dem Programmstart mit RUN lädt dieses Hilfsprogramm ein Anwenderprogramm so, daß es dem Zugriff des Betriebssystems entzogen ist

```
16514: 33   32   78   34
16518: 4    64   249  1
16522: 0    62   197  1
16526: 118  6    197  237
16530: 115  2    64   205
16534: 231  2    22   255
16538: 33   44   78   54
16542: 234  205  164  64
16546: 24   251  14   1
16550: 6    0    62   127
16554: 219  254  211  255
16558: 31   48   45   23
16562: 23   56   43   16
16566: 241  241  186  48
16570: 35   205  164  64
16574: 121  23   48   249
16578: 33   32   78   80
16582: 205  164  64   113
16586: 35   235  42   43
16590: 78   197  1    23
16594: 14   9    193  55
16598: 237  82   235  218
16602: 7    2    24   231
16606: 207  8    213  200
16610: 148  6    26   29
16614: 219  254  23   203
16618: 123  123  56   245
16622: 16   245  209  32
16626: 4    254  86   48
16630: 175  63   203  17
16634: 48   170  201  253
16638: 203  1    205  237
16642: 75   50   64   120
16646: 230  63   71   30
16650: 148  78   84   93
16654: 205  223  9    195
16658: 62   7
```

② **Maschinencode:** Beginnend mit Adresse 16514 sind diese Dezimalcodes Zeile für Zeile der Reihe nach per POKE-Befehl in Zeile 1 des Ladeprogramms unterzubringen

```
   1 REM TEST
   2 SLOW
  10 REM TEST
  20 PRINT "ALLES O.K ";
  30 GOTO 20
9500 PRINT "VERSCHLUESSELUNGS-PA
SSWORD:"
9510 INPUT P$
9520 IF LEN P$<>32 THEN GOTO 951
0
9530 FAST
9540 CLS
9550 IF PEEK 16440<>188 THEN LPR
INT
9560 LPRINT P$;
9570 LET P$=""
9580 RAND USR 16524
9590 SLOW
9600 PRINT "ENTSCHLUESSELUNGS-PA
SSWORD:"
9610 INPUT P$
9620 LPRINT P$;
9630 RAND USR 16545
9640 GOTO 3
```

③ **Programm-Verschlüßler:** Dieses Hilfsprogramm schützt ein Anwenderprogramm nicht nur mit einem Kennwort, sondern es verschlüsselt auch noch die Programmbytes

samt aller Variablen nichts anderes als ein Datensatz.

Sollte es beim Laden des geschützten Programms zu einem Fehler kommen, wird dies mit der Meldung „9" quittiert. Mit GOTO 20 darf man dann zumindest den Programmteil in Augenschein nehmen, der noch geladen wurde. So kann das Hilfsprogramm auch Retter in der Not sein, wenn sich ein Programm, z. B. wegen eines Bandfehlers, sonst nicht mehr laden läßt.

Die Krone des Programmschutzes

Die Mängel des bisherigen Kennwort-Schutzes lassen sich beseitigen, wenn zum einen das Programm nicht im Klartext auf Band gespeichert wird, und zum anderen das Kennwort nicht aus dem Programm hervorgeht. Beide Forderungen erfüllt das Hilfsprogramm von *Bild 3* mit dem zugehörigen Maschinencode gemäß *Bild 4*.

Das Hilfsprogramm nutzt ein Verfahren der Kryptographie, verschlüsselt also das zu schützende Programm. Und damit der Programmschutz endlich einmal unangefochten bleibt, werden Sie in der FUNKSCHAU garantiert keine Anleitung mehr lesen, wie auch dieser Schutz zu knacken ist!

Das zu schützende Programm ist im Zeilennummernbereich von 3 bis 9499 ins Hilfsprogramm einzufügen. In Bild 3 ist in diesem Bereich ein nur dreizeiliges Demonstrationsprogramm untergebracht. Zum Speichern auf Band ist dann mit GOTO 9500 der Einsprung in das Hilfsprogramm vorzunehmen.

Jetzt verlangt das Programm die Eingabe eines Kennworts, das genau 32 Zeichen lang sein muß. Dieses Kennwort wird zur Übergabe an das Maschinenprogramm in den Drucker-Puffer des ZX 81 geschrieben. Zeile 9570 löscht anschließend die Basic-Variable des Kennworts (P$), damit sie nicht mit auf Band aufgezeichnet wird.

Mit dem Aufruf des Maschinenprogramms werden dann gleich mehrere Aktivitäten eingeleitet: Das zu schützende Programm wird verschlüsselt, durch Aufruf der SAVE-Routine im ROM auf Band gespeichert, dann sofort wieder entschlüsselt und automatisch ab Programmanfang ausgeführt.

Nunmehr befindet sich also das Programm verschlüsselt auf Band, und zwar unter dem Namen, der in Zeile 1 (Bild 3) unmittelbar der REM-Anweisung folgt. Hier ist es der Name TEST. Erlaubt sind Namen mit maximal zehn Zeichen, wobei jedoch das letzte Zeichen unbedingt invers sein muß. Dann ist nämlich Bit 7 dieses Bytes gesetzt, und die SAVE-Routine erkennt hieran das Ende des Namens.

```
16514:  57  42  56 185
16518:  27  27  27  27
16522:  27  27 205 155
16526:  64 253  54  56
16530: 188  33 130  64
16534: 205 252   2  33
16538: 125  64  34  41
16542:  64  24  20 253
16546:  54  56 188  33
16550:  60  64 237  91
16554:  28  64   6  32
16558: 126  18 112  35
16562:  19  16 249 217
16566: 229  42  28  64
16570:  84  93   1  17
16574:   0   9   6  15
16578: 217 253 203  56
16582: 126  32   9  33
16586:  28  37 205 216
16590:   9  34 123  64
16594:  17  20  65  42
16598: 123  64 167 237
16602:  82 235 205 232
16606:  64  35  27 122
16610: 179  32 247 225
16614: 217 201 126 217
16618:  19  13  32  10
16622:  14  17 197   1
16626: 239 255 235   9
16630: 235 193  35   5
16634:  32   8   6  15
16638: 197   1 241 255
16642:   9 193 174 235
16646: 174 235 217 119
16650: 201
```

④ **Maschinencode:** Diese Dezimalcodes sind wie gewohnt in Zeile 1 des Verschlüßler-Programms unterzubringen

Andererseits können Könner die Verschlüsselung auch auf den Variablenbereich anwenden, wenn es gilt, wichtige Daten zu schützen. Dafür sind bei unserem Franzis-Software-Service knapp kommentierte Assembler-Listings der hier vorgestellten Maschinenprogramme erhältlich. Legen Sie der Bestellung unter dem Stichwort „Schloß und Schlüssel II" bitte 4 DM in Briefmarken bei.

Michael Schramm/-ll

Da haben Knacker keine Chance

Das Verschlüsselungsverfahren verknüpft Kennwort- und Programmbytes mit Exklusiv-ODER-Operationen (EXOR). Dabei werden die Programmbytes so heillos durcheinandergebracht, daß ein Knacken der Verschlüsselung praktisch ausgeschlossen ist.

Verschlüsselt wird nur das zu schützende Programm. Das hat zwei Gründe: Erstens darf der Maschinencode in Zeile 1 nicht verändert werden, weil er auch zum Entschlüsseln benötigt wird, und zweitens sollte man nichts verschlüsseln, was ohnehin bekannt – weil hier abgedruckt – ist. Wären also auch die Zeilen ab Nummer 9500 verschlüsselt, dann könnten Könner aus der Kenntnis des verschlüsselten und unverschlüsselten Programmblocks durch EXOR-Verknüpfung beider Blöcke das Kennwort gewinnen!

ZX-81-Softwaretip:

Bildhauer

18 Befehle zur Bildmanipulation

Normalerweise kann der ZX 81 Texte oder Grafiken nur niet- und nagelfest am Bildschirm plazieren; allein der SCROLL-Befehl kann da Bewegung hineinbringen. Geradezu ärmlich wirkt der SCROLL-Befehl jedoch, wenn die hier vorgestellten Maschinenroutinen zum Zuge kommen.

Einen Gesamtumfang von etwa 0,75 KByte haben die Maschinenroutinen, die dem ZX 81 die Bildmanipulation leichtmachen. Sie ermöglichen z. B. das Scrollen in alle vier Richtungen und den Zugriff auf einen zweiten Bildspeicher. Besonders eindrucksvoll ist, daß sämtliche Manipulationen nicht zwangsweise auf den gesamten Bildschirm wirken müssen, sondern auch nur für Bildausschnitte gelten können. Das ermöglicht sehr reizvolle Bildgestaltungen.

So wird das Bildfenster abgesteckt

Der Umgang mit dem Programm ist einfach. Jede der 18 Maschinenroutinen, die stets einen Befehl simulieren, muß lediglich mit einem eigenen RAND-USR-Aufruf gestartet werden. Der jeweilige Befehl wird dann in dem zuvor festgelegten Bildfenster ausgeführt. Auf Bildanteile außerhalb des Fensters hat er keine Wirkung.

Zur Definition des Bildfensters sind dessen Bildschirmkoordinaten für die obere linke und untere rechte Ecke in vier Speicherzellen unterzubringen:

POKE 16514	Oben	(0 bis 23)
POKE 16515	Links	(0 bis 31)
POKE 16516	Unten	(0 bis 23)
POKE 16517	Rechts	(0 bis 31)

Selbstverständlich können die Fensterkoordinaten im Verlauf eines Programmes beliebig verändert werden, man muß sich nur klar darüber sein, daß alle Zusatzbefehle nur innerhalb des gerade gültigen Bildfensters Wirkung zeigen. Dabei wirken die Befehle auf sämtliche innerhalb des Bildfensters angetroffene Zeichen (Text, Grafik). Soll also z. B. lediglich eine bestimmte Textpassage wie mit einer Schneeschaufel nach rechts aus dem Bild geschoben werden, dann ist vor dem Aufruf des zugehörigen Befehls das Bildfenster „um diese Textpassage zu legen". Dabei die richtigen Fensterkoordinaten aufzuspüren ist kein Problem, da diese Werte die gleichen Zeilen- und Spaltennummern sind, die auch dem PRINT-AT-Befehl zugeordnet werden (einschließlich der Zeilen 23 und 24).

```
16564: 58 130 64 71 62 23
16570: 144 218 92 65 175 103
16576: 111 87 184 40 6 17
16582: 33 0 25 16 253 58
16588: 131 64 95 62 31 147
16594: 218 92 65 25 35 34
16600: 135 64 58 130 64 71
16606: 58 132 64 95 62 23
16612: 147 218 92 65 123 60
16618: 144 218 92 65 202 92
16624: 65 50 137 64 58 131
16630: 64 71 58 133 64 95
16636: 62 31 147 218 92 65
16642: 123 60 144 218 92 65
16648: 202 92 65 50 138 64
16654: 71 62 33 144 50 139
16660: 64 42 12 64 237 91
16666: 135 64 25 34 140 64
16672: 58 137 64 71 5 175
16678: 184 40 6 17 33 0
16684: 25 16 253 34 142 64
16690: 58 138 64 79 6 0
16696: 13 9 34 146 64 42
16702: 140 64 9 34 144 64
16708: 0 58 134 64 71 62
16714: 33 144 56 1 201 62
16720: 127 144 210 154 13 62
16726: 191 144 218 154 13 201
16732: 207 19 205 180 64 42
16738: 140 64 58 137 64 71
16744: 58 138 64 95 75 135
16750: 198 128 119 35 13 32
16756: 248 58 139 64 35 61
16762: 32 252 16 238 201 205
16768: 180 64 42 140 64 58
16774: 137 64 71 58 138 64
16780: 95 75 58 134 64 119
16786: 35 13 32 251 58 139
16792: 64 35 61 32 252 16
16798: 238 201 205 180 64 33
16804: 146 64 58 134 64 6
16810: 32 119 35 16 252 205
16816: 70 66 205 86 66 205
16822: 91 66 195 113 66 0
16828: 205 180 64 205 12 66
16834: 205 121 66 195 86 66
16840: 205 180 64 205 59 66
16846: 24 242 205 180 64 205
16852: 27 66 205 161 66 195
16858: 70 66 205 180 64 205
16864: 59 66 24 242 205 180
16870: 64 205 32 66 205 200
16876: 66 195 113 66 205 180
16882: 64 205 59 66 24 242
16888: 205 180 64 205 54 66
16894: 205 237 66 195 91 66
16900: 205 180 64 205 59 66
16906: 24 242 42 140 64 17
16912: 148 64 6 0 58 138
16918: 64 79 237 176 201 42
16924: 142 64 24 239 42 140
16930: 64 17 148 64 58 137
16936: 64 71 197 126 18 1
16942: 33 0 9 19 193 16
16948: 245 201 42 144 64 24
16954: 232 33 148 64 6 32
16960: 175 119 35 16 252 201
16966: 42 140 64 235 33 148
16972: 64 58 138 64 79 6
16978: 0 237 176 201 42 142
16984: 64 24 238 42 140 64
16990: 17 148 64 58 137 64
16996: 71 197 26 119 1 33
17002: 0 9 19 193 16 245
17008: 201 42 144 64 24 232
17014: 205 180 64 42 140 64
17020: 84 93 1 33 0 9
17026: 58 137 64 71 5 58
17032: 138 64 79 197 3 237
17038: 160 13 32 250 58 139
17044: 64 71 35 19 16 252
17050: 193 16 238 201 205 180
17056: 64 42 146 64 84 93
17062: 1 33 0 237 66 58
17068: 137 64 71 5 58 138
17074: 64 79 197 6 0 237
17080: 184 58 139 64 71 43
17086: 27 16 252 193 16 240
17092: 201 205 180 64 42 140
17098: 64 84 93 35 58 138
17104: 64 79 13 58 137 64
17110: 71 197 6 0 237 176
17116: 58 139 64 71 35 19
17122: 35 19 16 252 193 16
17128: 238 201 205 180 64 42
17134: 146 64 84 93 43 58
17140: 138 64 79 13 58 137
17146: 64 71 197 6 0 237
17152: 184 58 139 64 71 43
17158: 27 43 27 16 252 193
17164: 16 238 201 0 0 0
17170: 237 91 140 64 42 135
17176: 64 1 103 57 9 58
17182: 137 64 71 58 138 64
17188: 79 201 205 180 64 205
17194: 18 67 235 197 6 0
17200: 237 176 58 139 64 71
17206: 35 19 16 252 193 16
17212: 240 201 0 205 180 64
17218: 205 18 67 24 230 0
17224: 0 205 180 64 205 18
17230: 67 197 65 3 26 237
17236: 160 43 119 35 16 247
17242: 58 139 64 71 35 19
17248: 16 252 193 16 234 201
```

① **Programmbytes:** Diese Dezimalwerte sind in einer mindestens 1534 Zeichen langen REM-Zeile unterzubringen

Tabelle. Befehlsvorrat des „Bildhauers" und Startadressen der zugehörigen Maschinenroutinen

INVERT	16734	LEFTSCROLL	17093
FILL	16767	mit Clear	16880
FRAME	16800	LEFTROT	16868
UPSCROLL	17014	RIGHTSCROLL	17130
mit Clear	16840	mit Clear	16900
UPROT	16828	RIGHTROT	16888
DOWNSCR.	17054	PUSH	17190
mit Clear	16860	POP	17215
DOWNROT	16848	EXCHANGE	17225

Eine Liste aller neuen Befehle und die Startadressen der zugehörigen Maschinenroutinen zeigt die *Tabelle*. Scroll- und Rotationsbefehle gibt es für vier Richtungen. Da die Wirkungen sinngemäß gleich bleiben, begnügt sich die folgende Beschreibung mit der Befehlsgruppe für eine Richtung:

○ INVERT: Der Inhalt des Bildfensters wird invertiert dargestellt.
○ FILL: Ein Bildfenster wird vollständig mit dem unter Adresse 16518 abgelegten Zeichen ausgefüllt. Mit dem Leerzeichen (Code = 0) ist dieser Befehl auch zum Löschen eines Bildfensters geeignet.
○ FRAME: Entlang der Ränder des Bildfensters wird ein Rahmen gezogen, mit dem Zeichen, daß unter Adresse 16518 abgelegt wurde.
○ UPSCROLL: Verschieben des Fensterinhalts um eine Zeile nach oben, ohne Löschen der untersten Zeile. Durch mehrmaliges Aufrufen wird somit das Fenster nach und nach mit der letzten Fensterzeile gefüllt.
○ UPSCROLL und CLEAR: Wirkung wie UPSCROLL, jedoch mit Löschen der untersten Fensterzeile.
○ UPROT: Wirkung wie UPSCROLL, jedoch Ersatz der untersten Zeile durch die oben entfallende Zeile – also ein Rotieren des Bildinhalts.
○ PUSH: Abspeichern des Fensterinhalts in der REM-Zeile des Maschinenprogramms.
○ POP: Zurückholen eines abgespeicherten Fensterinhalts.
○ EXCHANGE: Austauschen des aktuellen und des abgespeicherten Fensterinhalts innerhalb der gerade gültigen Fensterkoordinaten.

Solange es nicht zu einer Überlappung der Bildfenster kommt, ist das Abspeichern und Austauschen mehrerer (kleinerer) Fenster zulässig. So wäre es z. B. möglich, mit EXCHANGE ein zuvor abgespeichertes Bildfenster mit Erklärungen ins aktuelle Bildfenster einzublenden und nach kurzer Zeit wieder gegen den Originalausschnitt auszutauschen. Ebenso dürfen die Zeilen 23 und 24 in ein Fenster einbezogen oder als selbständiges Fenster genutzt werden.

Für die Eingabe des Maschinenprogramms gibt es drei Möglichkeiten: Entweder man tippt eine 1534 Zeichen lange REM-Zeile übers Wochenende selber ein, oder man überläßt diese Arbeit einem Hilfsprogramm (siehe Heft 21/1984, Seite 74). Wer mag, kann das Programm samt Demonstrationsbeispiel auf Kassette auch gegen 10 DM beim Autor erwerben.

Wie gewohnt beginnt die REM-Zeile mit Adresse 16514, wobei aber die ersten 50 Byte für die Aufnahme von Hilfsvariablen und der Fensterkoordinaten vorerst frei bleiben. Die Dezimalcodes des eigentlichen Programms (*Bild 1*) sind erst ab Adresse 16564 bis 17253 (Eckwerte für die Eingabe-FOR-NEXT-Schleife) einzutippen. Die noch verbleibenden 792 Byte der REM-Zeile sind Platzhalter für einen mit PUSH abzuspeichernden Bildinhalt. Da sich der Inhalt der REM-Zeile problemlos mit SAVE auf Band speichern läßt, wird somit auch der abgespeicherte Bildinhalt aufgezeichnet.

```
 10 LET A=3
 20 GOSUB 500
 30 POKE 16518,23
 40 RAND USR 16800
 50 LET A=4
 60 GOSUB 500
 70 PRINT AT 10,15;"*"
 80 PRINT AT 11,14;"**"
 90 PRINT AT 12,15;"*"
100 GOSUB 600
110 RAND USR 16734
120 GOSUB 600
130 FOR I=1 TO 24
140 RAND USR 16868
150 NEXT I
160 FOR I=1 TO 14
170 RAND USR 16828
175 NEXT I
180 FOR B=3 TO 10
190 POKE 16518,B
200 RAND USR 16767
210 GOSUB 600
220 NEXT B
230 PRINT AT 11,13;;"E N D E"
240 LET M=16848
250 GOSUB 300
260 LET M=16868
270 GOSUB 300
280 LET M=16880
290 GOSUB 300
295 GOTO 70
300 FOR I=1 TO 59
310 RAND USR M
320 NEXT I
330 RETURN
500 POKE 16514,A
510 POKE 16515,A
520 POKE 16516,21-A
530 POKE 16517,31-A
540 RETURN
600 FOR I=1 TO 15
610 NEXT I
620 RETURN
```

② **Demonstrationsprogramm:** Das Programm ruft einzelne Maschinenroutinen auf und zeigt eindrucksvolle Bilder

Für einen ersten Test sind zunächst die vier Speicherzellen 16514 bis 16517 per POKE-Befehlen mit Fensterkoordinaten zu füllen. Speicherzelle 16518 erhält z. B. den Code 23 (Stern) zugewiesen. Jetzt kann man bereits die Befehle FILL und FRAME ausprobieren (RAND USR 16800 bzw. 16767). Für einen Test der übrigen Befehle ist (per Programm) erst einmal mit PRINT AT ein beliebiges Zeichen innerhalb des definierten Bildfensters zu plazieren, bevor ein Aufruf der Befehle Wirkung zeigt. Eindrucksvoller ist freilich das Demonstrationsprogramm gemäß *Bild 2*.

Um die normalerweise nicht mit PRINT erreichbaren Zeilen 23 und 24 verwenden zu können, gibt es zwei Möglichkeiten. Zunächst ist der dort gewünschte Text in den Zeilen 21 und 22 unterzubringen:

```
PRINT AT 20,0;"Text"
PRINT AT 21,0;"Text"
```

Anschließend wird mit den Fensterkoordinaten 20, 0, 23, 31 zweimal DOWNSCROLL aufgerufen und das Bildfenster mit PUSH abgespeichert. Diese Befehlsfolge ist als Programm einzugeben, da bei Direkteingabe der Bildschirm zwischendurch gelöscht würde. Der POP-Befehl holt dann jederzeit den Text in die beiden untersten Zeilen, ohne daß vom aktuellen Bildinhalt (Zeile 0 bis 22) etwas verlorengeht.

Ebenso führt es zum Ziel, der Systemvariablen DF-SZ unter Adresse 16418 den Wert 0 zuzuweisen. Dann sind die Zeilen 23 und 24 auch mit PRINT-Befehlen erreichbar. Allerdings muß vor jedem INKEY- und INPUT-Befehl die Systemvariable mit POKE 16418,2 wieder zurückgesetzt werden – sonst streikt der ZX 81. Auf jeden Fall sollte man auch auf den Original SCROLL-Befehl verzichten, denn er birgt jetzt das Risiko unerwünschter Fensterverschiebungen.

Wer das Maschinenprogramm nicht nur nutzen, sondern es auch durchleuchten will, der kann gegen Einsenden von 4 DM in Briefmarken ein kommentiertes Assembler-Listing beim Franzis-Software-Service erwerben. Vergessen Sie aber bitte nicht das Stichwort „Bildhauer" anzugeben. Günther Bless/-ll

ZX 81 à la carte

Horchposten

Decoder und Coder für Selektivruf

Und noch ein attraktiver Anwendungsvorschlag für den ZX 81: Ein komplettes Selektivruf-System, das fürs erste ohne zusätzliche Hardware auskommt. Greifen Sie den Faden auf; es hängt von Ihnen ab, wo er hinführt!

Nomen est omen: Der Selektivruf macht's möglich, in einem Kommunikationsnetz einzelne Teilnehmer gezielt anzusprechen. Funkamateure z. B. nutzen den Selektivruf, wenn sie mit einem ganz bestimmten Funkpartner reden möchten. Sie strahlen dazu eine vereinbarte Tonfolge aus.

Identifiziert ein Selektivruf-Decoder diese Tonfolge, gibt er den Empfang frei oder meldet lauthals, daß jemand gezielt Funkkontakt sucht. Auch Eurosignal, das flächendeckende Personen-Rufsystem, fußt in gleicher Weise auf der Ausstrahlung vereinbarter Tonfolgen.

Es lebe der Werbefunk

Das „Tüteltüt" unzähliger verschiedener Tonfolgen dringt jedem ins Ohr, der seinen UKW-Empfänger auf 87,5 MHz einstellt: Auf dieser Frequenz sendet Eurosignal seine Lockrufe aus. Jede 6stellige Personen-Rufnummer ist hierbei als charakteristische Tonfolge codiert. Und jeder einzelne Ton der Tonfolge repräsentiert eine Ziffer (0 bis 9), wobei es zwischen Tonhöhe und Ziffer einen festen Zusammenhang gibt.

Wollte jetzt ein Hobby-Elektroniker aus purer Abenteuerlust Eurosignale decodieren, dann ist das nicht nur von der Post untersagt, sondern auch noch sündhaft teuer, wenn er dafür Hardware-Tondecoder wie das IC NE 567 verwendet. Um damit nur 100 Tonfolgen decodieren zu können, wären für weit über 500 DM Bauteile anzuschaffen.

Da ist ein Software-Tondecoder per Heimcomputer klar im Vorteil. Das hier vorgeschlagene Programm für den ZX 81 identifiziert überdies schon nach zwei Signalperioden die Tonhöhe und das auf 0,5 % genau! Kritisch wird's nur, wenn Oberwellen das Signal verzerren.

Wer Phantasie hat, wird anstelle der verbotenen Eurosignal-Decodierung gewiß andere Anwendungen für den Software-Tondecoder finden – auch wenn er kein Funkamateur ist. So sollte es Tüftlern gelingen, den Tondecoder so zu sensibilisieren, daß er z. B. im Rundfunk ausgestrahlte Tonfolgen identifiziert. Dann wäre es endlich möglich, ein Rundfunkgerät immer dann pünktlich einzuschalten, wenn die vor Werbesendungen üblichen Tonfolgen eintreffen. Keine Werbesendung mehr zu versäumen – das lohnt jeden Aufwand.

Das Programm kann jedoch nicht nur 1 Mio. Tonfolgen decodieren, es kann solche Tonfolgen auch erzeugen. Mehr darüber später, denn jetzt ist erst einmal der Tondecoder an der Reihe.

Das Ohr des ZX 81

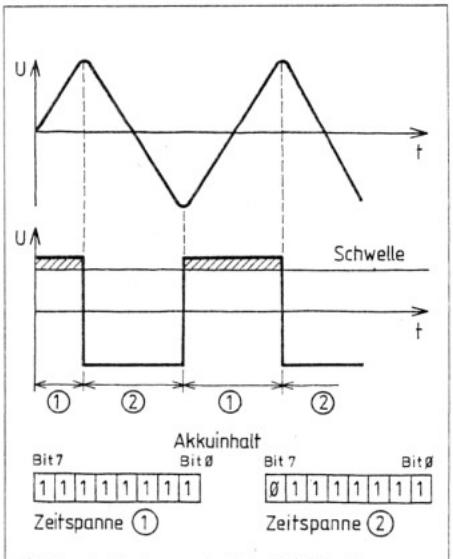
Grundsätzlich läßt sich jeder Heimcomputer, der ein Kassetten-Interface hat, als Tondecoder verwenden. In diesem Kassetten-Interface (beim ZX 81 steckt es im Logik-Chip) ist nämlich ein A/D-Wandler einfachster Bauart untergebracht, der auch für unsere Zwecke brauchbar ist.

Beim ZX 81 wird das an der EAR-Buchse (Eingang) anstehende Signal erst differenziert und dann der Pegel mit einer festen Diskriminator-Schwelle verglichen. Ist der Pegel größer als der Schwellenwert, hat der Abfragebefehl für den EAR-Eingang *in a,254* im Akku der Z-80-CPU den Wert 255 zur Folge. Im Akku sind dann also alle Bits gesetzt.

Liegt der Eingangspegel beim Abfragen der EAR-Buchse jedoch unterhalb der Schaltschwelle, so wird der Akku mit dem Wert 127 geladen, d. h. im Akku sind nur noch die Bits 0 bis 6 gesetzt (*Bild*). Damit ist bereits eine 1-Bit-A/D-Wandlung vollzogen, die freilich nur zwei Analogpegelwerte auseinanderhalten kann. Doch das genügt.

Wie aber bekommt man jetzt möglichst einfach heraus, welche Zahl im Akku steht? Dazu wird der Akkuinhalt um eine Stelle nach links geschoben, das maßgebende Bit 7 gelangt damit ins Carry-Flag. Stand z. B. 255 im Akku, wird das von dem nun gesetzten Carry-Flag signalisiert. *Bild 2* zeigt, wie sich der Inhalt des Carry-Flags ändert, wenn an der EAR-Buchse des ZX 81 ein Rechteck-Signal eingespeist wird.

Um die Periodendauer dieses Signals zu erhalten, ist nur noch die Zeitdauer

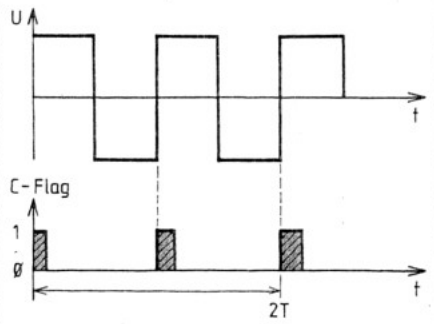


① **Port-Abfrage beim ZX 81:** Das am EAR-Eingang eingespeiste Signal (oben) wird im Computer differenziert und der Pegel mit einer Schaltschwelle verglichen (Mitte). Der Akkuinhalt meldet mit Bit 7, ob die Schwelle über- oder unterschritten ist

zu messen, die zwischen zwei gesetzten Carry-Flags verstreicht. Und genau auf dieser Messung beruht das Maschinenprogramm in *Bild 3*. Bis zur Adresse 4085h prüft es fortwährend, ob das Carry-Flag gesetzt ist. Erst ab dem Moment, da ein Signal (mit ausreichend hohem Pegel) in die EAR-Buchse eingespeist wird, beginnt die eigentliche Zeitmessung mit dem Rücksetzen des bc-Registerpaares.

Da wir der Meßgenauigkeit wegen zwei Signalperioden erfaßen wollen, wird das bc-Registerpaar so lange um 1 erhöht, bis das Carry-Flag zum dritten Mal gesetzt wird (Programmblock zwischen Adresse 408Ah und 409Fh). Sobald dies der Fall ist, hat das bc-Registerpaar den für zwei Signalperioden charakteristischen Zählerstand erreicht. Diesen Wert rettet das Programm in die Speicherzellen 16616/16617 (siehe 40A2h).

Sicherheitshalber wird anschließend der gesamte Zählvorgang bis Adresse 40C4h wiederholt, der zuerst ermittelte



② **Carry-Flag als Melder:** Jede positive Signalflanke setzt das Carry-Flag, mit dessen Hilfe die Periodendauer des Signals zu ermitteln ist

Zählerstand ins hl-Registerpaar geladen und zu dem im zweiten Anlauf erreichten Zählerstand im bc-Registerpaar addiert. Der Rücksprungbefehl bei Adresse 40CEh reicht diesen endgültigen Zählerstand im bc-Registerpaar automatisch an ein Basic-Programm weiter.

Wichtig ist, daß das Maschinenprogramm, dessen verbleibender Teil die Tonerzeugung vornimmt, unbedingt im FAST-Modus des ZX 81 ablaufen muß. Im SLOW-Modus würde es alle 20 ms durch die Bildaufbau-Routine zwangsweise unterbrochen und die ins Basic zurückgemeldeten Zählerstände wären nur „Hausnummern". Während des Bildaufbaus kümmert sich der Computer nämlich nicht darum, was sich an seiner EAR-Buchse abspielt.

Auf dem Weg zum idealen Filter

Wird das Maschinenprogramm z. B. mit der Zeile LET X=USR 16514 vom Basic aus aufgerufen, so erhält die Variable X beim Rücksprung ins Basic den Zählerstand zugewiesen. Dieser Zählerstand ist ein unmittelbares Maß für die Frequenz des eingespeisten Signals; je größer der Wert ist, desto niedriger ist die Frequenz. Und abhängig von der Taktfrequenz bzw. von der Struktur des Abfrageprogramms ergibt sich ein Umrechnungsfaktor, der für das gezeigte Maschinenprogramm beim ZX 81 $3{,}9 \times 10^5$ lautet. Zwischen Zählerstand Z und Frequenz f gilt also der Zusammenhang: $Z = 1/f \times 3{,}9 \times 10^5$

Dabei ist der Frequenzwert in Hertz einzusetzen. Ein 1000-Hz-Tonsignal hat also den Zählerstand 390 zur Folge, ein 400-Hz-Signal den Wert 975 und ein 100-Hz-Signal den Wert 3900. Mit der Programmzeile

```
IF X=975 THEN PRINT „400 HZ"
```

ließe sich der Zählerstand bereits auswerten. Es entstünde damit ein extrem steilflankiges Software-Filter für 400 Hz. Zur Selektivruf-Decodierung sollte man indes nicht gnadenlos z. B. auf den Wert 975 pochen, sondern durch Zulassen eines ganzen Wertbereichs etwas an Bandbreite „spendieren"; das kommt der Betriebssicherheit zugute.

Außerdem ist nicht ein einzelner Vergleich zu treffen, sondern man muß zehn Vergleiche vornehmen (Ziffer 0 bis 9), ehe eindeutig klar ist, welche Frequenz das eingespeiste Tonsignal zum Abfragezeitpunkt hatte. Und dann ist schleunigst wieder das Maschinenprogramm aufzurufen, damit der nächste Ton der Tonfolge decodiert werden kann. Dies programm-technisch zu lösen ist kein Problem und reizt zum Experimentieren. Anhaltspunkte gibt der Lösungsvorschlag zwischen Zeile 70 und 195 in dem fertigen Basic-Programm (*Bild 4*).

```
4082  IN A,254              DBFE
4084  RLA                   17
4085  JR NC,$4082           30FB
4087  LD BC,0000            010000
408A  INC BC                03
408B  IN A,254              DBFE
408D  RLA                   17
408E  JR C,$408A            38FA
4090  IN A,254              DBFE
4092  RLA                   17
4093  INC BC                03
4094  JR NC,$4090           30FA
4096  INC BC                03
4097  IN A,254              DBFE
4099  RLA                   17
409A  JR C,$4096            38FA
409C  IN A,254              DBFE
409E  RLA                   17
409F  INC BC                03
40A0  JR NC,$409C           30FA
40A2  LD (16616),BC         ED43E840
40A6  IN A,254              DBFE
40A8  RLA                   17
40A9  JR NC,$40A6           30FB
40AB  LD BC,0000            010000
40AE  INC BC                03
40AF  IN A,254              DBFE
40B1  RLA                   17
40B2  JR C,$40AE            38FA
40B4  IN A,254              DBFE
40B6  RLA                   17
40B7  INC BC                03
40B8  JR NC,$40B4           30FA
40BA  INC BC                03
40BB  IN A,254              DBFE
40BD  RLA                   17
40BE  JR C,$40BA            38FA
40C0  IN A,254              DBFE
40C2  RLA                   17
40C3  INC BC                03
40C4  JR NC,$40C0           30FA
40C6  LD HL,(16616)         ED6BE840
40CA  ADD HL,BC             09
40CB  LD B,H                44
40CC  LD C,L                4D
40CD  NOP                   00
40CE  RET                   C9
40CF  NOP                   00
40D0  LD C,255              0EFF
40D2  OUT 255,A             D3FF
40D4  LD B,03E              0623
40D6  DJNZ $40D6            10FE
40D8  CALL 03910            CD460F
40DB  LD B,030              061E
40DD  DJNZ $40DD            10FE
40DF  DEC C                 0D
40E0  JR NZ,$40D2           20F0
40E2  RET                   C9
```

③ **Assembler-Listing:** Die im Text erklärten Routinen verhelfen zum Decodieren und Codieren von Tonfolgen

Töne per Software

Ebenso einfach wie das Decodieren einer Tonfolge per Programm ist auch das Erzeugen einer Tonfolge. Erteilt man der Z-80-CPU den Befehl *out 255,a*, dann sorgt sie dafür, daß der Logik-Chip des ZX 81 einen kurzen Impuls zur MIC-Buchse (Ausgang) schickt. Mehrere nacheinander erteilte out-Befehle bewirken also am MIC-Ausgang ein Signal, dessen Frequenz von der zeitlichen Aufeinanderfolge der out-Befehle abhängt.

Gemäß diesem Prinzip erzeugt das Maschinenprogramm (Bild 3) ab Adresse 40 D2h bei jedem Aufruf einen Ton. Zwischen Adresse 40D4h und 40D6h wird nach dem ersten Impuls eine Warteschleife ausgeführt, dann die BREAK-

Routine im ROM aufgerufen (um eine Programmunterbrechung zuzulassen) und zwischen Adresse 40D8h und 40D0h mit einem Wert geladen, der die Tondauer bestimmt. Solange das c-Register nicht den Wert 0 erreicht hat, erfolgt jetzt ein Rücksprung zum out-Befehl, der den nächsten Impuls auslöst. Die Werte in den Speicherzellen 40D5h und 40DCh bestimmen die Tonhöhe. Ein Basic-Programm muß jetzt nur per POKE-Befehle die richtigen Werte für Tondauer und -höhe im Maschinenprogramm unterbringen. Dies übernimmt der Teil Toncoder ab Zeile 400 im Basic-Programm von Bild 4.

Hier wird zunächst die zu codierende Ziffernfolge in der Zeichenkette L$ untergebracht. Dann beginnt eine Schleife, die den Wert der einzelnen Ziffern festgestellt und mit den zugehörigen X-Werten (für die Tonhöhe) ab Adresse 16621 eine Tabelle anlegt. Zeile 420 bewirkt dabei den bei Selektivruf-Verfahren üblichen „Repitorton": Taucht im String L$ eine Ziffer zweimal hintereinander auf, so wird für die zweite Ziffer der Repitorton gewählt. Er signalisiert dem Empfänger, daß die zuvor „eingetroffene" Ziffer zu wiederholen ist.

Mit POKE-Befehlen zum richtigen Ton

Zeile 440 legt dann fest, wie oft nun die Tonfolge abgesetzt werden soll. Die nächsten beiden Zeilen geben für die Tondauer und die erste Tonhöhe-Warteschleife (nur sie wird hier verändert) feste Werte vor. Der folgende Maschinenprogramm-Aufruf bewirkt damit einen Synchronisationston. Der Decodierteil des Programms erkennt an diesem Ton den Beginn einer Tonfolge. Anschließend bekommt die Tondauer einen anderen Wert zugewiesen (paßt zum Decodierteil) und mit einer Schleife (Zeile 445) werden nun die Töne erzeugt, deren Frequenz die zuvor angelegte Tabelle vorschreibt. Am Schluß der Tonfolge wird nochmals der Synchronisationston abgesetzt.

Im gezeigten Basic-Programm sind die vom Coder erzeugten Töne auf den Decoder abgestimmt. Außerdem ist ein Programmteil enthalten, der lediglich die Aufgabe hat, bei einem Programmabbruch die zuvor decodierten Ziffern anzuzeigen. Dabei meldet das Zeichen ■ den Beginn einer Ziffernfolge (erkennbar am Synchronisationston). Beim Decodieren ist eine Programmunterbrechung aber nur dann möglich, wenn auch ein Signal eingespeist wird. Ansonsten findet das Maschinenprogramm aus der Warteschleife zu Beginn nicht heraus.

Für einen ersten Test kann man mit dem Coderteil eine Tonfolge erstellen und auf Band aufzeichnen. Dieses Signal ist dann dem Decoderteil „vorzusetzen". Bis die gewünschte Meldung am Bildschirm auftaucht, die Tonfolge also richtig decodiert wurde, muß man freilich etwas Geduld aufbringen und die richtige Pegeleinstellung finden. Für eine „echte" Anwendung wäre deshalb ein Begrenzer-Verstärker vor dem EAR-Eingang ratsam.

Außerdem klappt die Decodierung um so sicherer, je weniger unharmonische Obertöne dem eingespeisten Grundton überlagert sind. Bei miserabler Übertragungsstrecke sollte die Tonfolge mehrmals hintereinander übertragen werden, indem der obere Grenzwert der Schleife in Zeile 440 höhergesetzt wird. Und noch ein Tip für Funkamateure: Strahlt der ZX 81 in den Empfänger ein, so ist er abzuschirmen, und mit einem Tiefpaß in der MIC-Leitung wird bei AM- oder SSB-Sendern die Bandbreite nicht unnötig erhöht. Wilfried Ehrensperger/-ll

```
   1 REM ANSTELLE DIESES TEXTES
MINDESTENS 114 BELIEBIGE ZEICHEN
EINTIPPEN
   2 LET A$="DBFE1730FB010000003D
BFE1738FADBFE170330FA03DBFE1733F
ADBFE170330FAED43E840DBFE1730FB0
1000003DBFE1738FADBFE170330FA03D
BFE1738FADBFE170330FAED6BE840094
44D00C9000EFFD3FF062310FECD460F0
61E10FE0D20F0C9"
   3 LET L=16514
   4 FOR N=0 TO LEN A$-2 STEP 2
   5 POKE L+INT (N/2),16*(CODE A
$(N+1)-28)+CODE A$(N+2)-28
   6 NEXT N
  10 FAST
  15 LET B$=""
  16 LET S$=""
  17 LET G$=""
  43 PRINT "GEBEN SIE EINE MAX.
6STELLIGE   ZIFFERNFOLGE EIN, DI
E DECODIERT WERDEN SOLL :"
  45 INPUT T$
  46 PRINT AT 5,5;T$
  47 PAUSE 100
  48 CLS
  49 PRINT "WELCHER TEXT SOLL BE
IM DECODIE- REN DIESER ZIFFERNFO
LGE AM BILD-SCHIRM AUFTAUCHEN?"
  50 INPUT W$
  51 CLS
  52 PRINT AT 7,0;"WENN SIE IN D
EN PROGRAMMTEIL   ""TONCODER""
WOLLEN, DRUECKEN SIE  INNERHALB
VON 5 SEKUNDEN ""T"""
  53 PAUSE 500
  54 IF INKEY$="T" THEN GOSUB 39
9
  55 CLS
  56 PRINT AT 5,1;"DAS PROGRAMM
STARTET GLEICH VON SELBST.";AT 8
,1;"WENN SIE MEHRERE ZIFFERNFOLG
EN   DECODIEREN WOLLEN,SO KOENNE
N    SIE DIESE IN DIE STRINGS G$
 IN  DEN ZEILEN 187 BIS 189 SCHR
EI-  BEN."
  58 PAUSE 600
  60 CLS
  61 PRINT AT 8,0;"DURCH DRUECKE
N VON ""BREAK"" KOEN-NEN DIE LET
ZTEN DECODIERTEN ZIF-FERN BETRAC
HTET WERDEN"
  62 PAUSE 600
  63 CLS
  70 LET X=USR 16514
  72 IF X<=275 THEN GOTO 70
  81 FOR N=0 TO 6
  90 IF X<=275 THEN GOTO 196
  93 IF X>275 THEN LET B$=""
  95 IF X>290 THEN LET B$="0"
 100 IF X>310 THEN LET B$="1"
 110 IF X>335 THEN LET B$="2"
 120 IF X>350 THEN LET B$="3"
 133 IF X>370 THEN LET B$="4"
 140 IF X>385 THEN LET B$="5"
 150 IF X>400 THEN LET B$="6"
 160 IF X>415 THEN LET B$="7"
 170 IF X>430 THEN LET B$="8"
 175 IF X>450 THEN LET B$="9"
 177 IF X>470 THEN GOTO 70
 178 IF S$=B$ THEN GOTO 192
 179 IF B$="" THEN PRINT S$;
 180 IF B$="" THEN LET G$=G$+S$
 183 PRINT B$;
 184 LET G$=G$+B$
 185 IF G$=T$ THEN GOSUB 300
 187 IF G$="172282" THEN PRINT "
1"
 188 IF G$="229717" THEN PRINT "
2"
 189 IF G$="217716" THEN PRINT "
3"
 192 LET X=USR 16514
 194 LET S$=B$
 195 NEXT N
 196 PRINT "■";
 198 LET G$=""
 199 LET S$=""
 200 IF PEEK 16398+256*PEEK 1639
9>20558 THEN CLS
 205 GOTO 70
 300 CLS
 301 PRINT AT 6,0;W$
 310 STOP
 399 CLS
 400 PRINT AT 8,7;"T O N C O D E
R";AT 11,0;"GEBEN SIE EINE MAX.
6STELLIGE   ZIFFERNFOLGE EIN"
 401 LET Y=0
 402 INPUT L$
 403 CLS
 405 FOR N=0 TO LEN L$-1
 406 LET H=N+1
 410 IF L$(H)="0" THEN LET X=150
 411 IF L$(H)="1" THEN LET X=170
 412 IF L$(H)="2" THEN LET X=180
 413 IF L$(H)="3" THEN LET X=190
 414 IF L$(H)="4" THEN LET X=200
 415 IF L$(H)="5" THEN LET X=210
 416 IF L$(H)="6" THEN LET X=220
 417 IF L$(H)="7" THEN LET X=230
 418 IF L$(H)="8" THEN LET X=240
 419 IF L$(H)="9" THEN LET X=254
 420 IF Y=X THEN LET X=140
 425 LET Y=X
 430 POKE 16621+N,X
 435 NEXT N
 440 FOR I=0 TO 1
 441 POKE 16593,240
 442 POKE 16597,130
 443 LET L=USR 16591
 444 POKE 16593,107
 445 FOR N=0 TO LEN L$-1
 450 POKE 16597,PEEK (16621+N)
 455 LET L=USR 16591
 460 NEXT N
 465 POKE 16593,250
 470 POKE 16597,130
 475 LET L=USR 16591
 480 NEXT I
 490 GOTO 52
 900 SAVE "PP■"
 920 GOTO 1
```

④ **Coder-/Decoder-Programm:** Ein kompletter Lösungsvorschlag mit aufeinander abgestimmten Programmteilen zum Codieren und Decodieren von Tonfolgen

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.

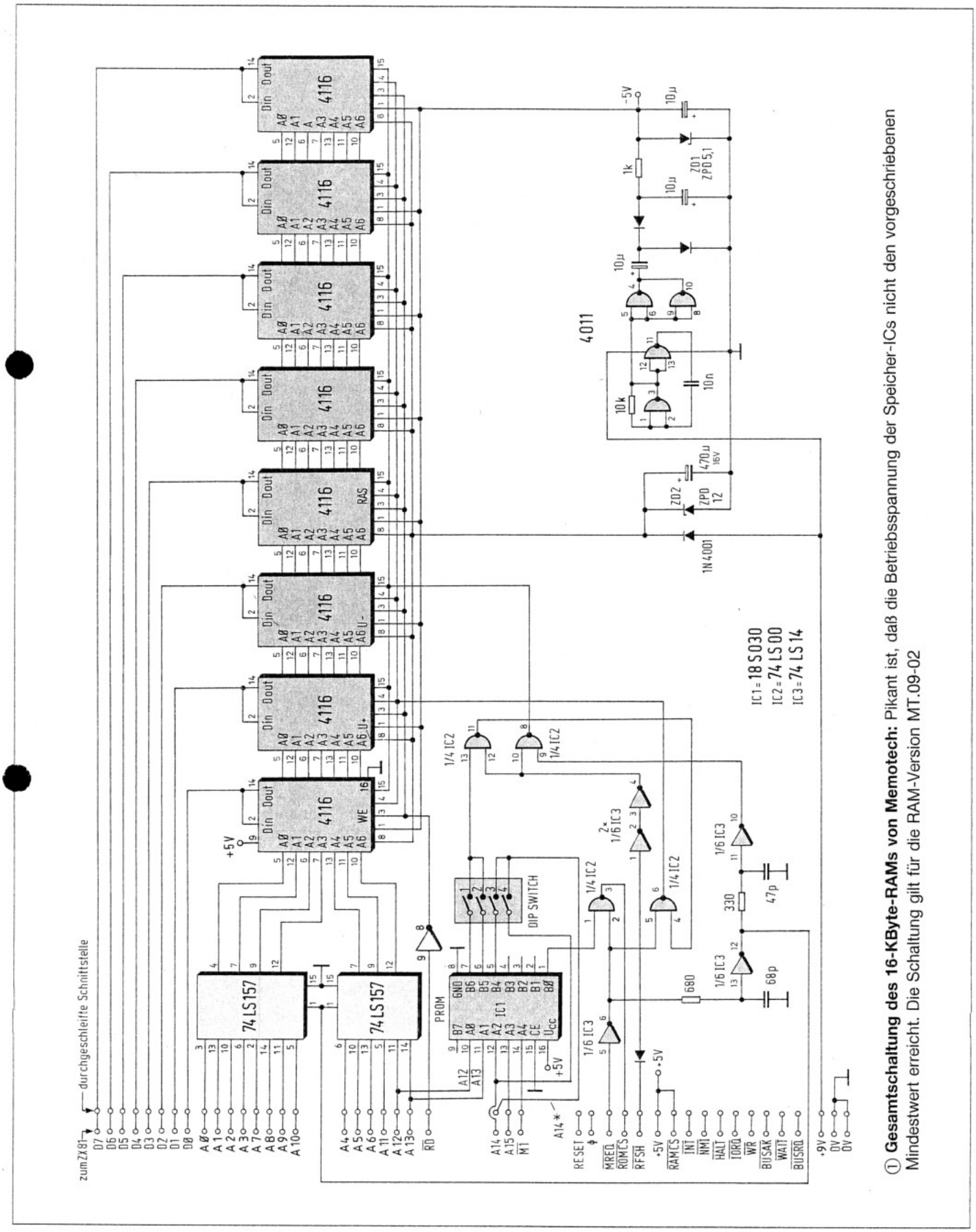
Jedes Speicher-IC hat eine Kapazität von 16 384 Bit (16 K × 1 Bit). Um ein einzelnes Bit adressieren zu können, müßte also jedes IC allein 14 Adreßleitungen haben. Dazu wäre ein großes IC-Gehäuse erforderlich. Um Platz zu sparen, werden die Adreßeingänge der ICs im Multiplex-Verfahren betrieben. Dann 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 $\overline{RAMCS}$ (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	A	RAM	RAM	nicht verwendet				zugehörige Adressen
	$\overline{M1}$	A15	A14	A13	A12	$\overline{CS}$ B0	14* B4	$\overline{CS}$ B5	$\overline{CS}$ 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	32768–36843
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$(704)
   5 POKE 16574,198
   6 REM AB HIER HAUPTPROGRAMM
BIS BIS ZEILE 9994 EINFUEGEN
9995 REM HEX-LADER
9996 LET A$="2A14400110400A7ED424
44D2A10403ABE40EDB120253EC3BE3AB
E4020F43E001105001 9EB2A0C40230312
000EDB0233CFE1620F521BE407E3C77C
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

Messen mit dem ZX 81 (1):

Der Zeit auf den Fersen

Wer seinem ZX 81 eine 8-Bit-Parallel-Schnittstelle spendiert hat, der kann den Computer ohne große Mühe zum digitalen Meßgerät hochpäppeln. Wir zeigen mit einer neuen fünfteiligen Serie, wie das geht, und beginnen hier mit der Zeitmessung.

Hobby-Elektroniker sind Meister im Improvisieren: Oft werden Meßgeräte erfolgreich durch Peilen über den Daumen ersetzt. Zuweilen aber hilft selbst diese Kunst nicht weiter und auch kein einfaches Analog-Multimeter. Wer dann der Versuchung widersteht, sich einen zwar schönen, aber wenig genutzten Meßgerätepark zuzulegen, der kommt auch mit dem ZX 81 ein gutes Stück weiter. Der Vorteil: Nur bei Bedarf muß der ZX 81 den Meßknecht spielen, ansonsten kann er anderen Beschäftigungen nachgehen, z. B. Morsezeichen entschlüsseln oder Zeitzeichen-Signale decodieren.

Die Software fordert die Z-80-PIO

Im Ernstfall ist der ZX 81 in Minutenschnelle zum Meßgerät erweitert. Es genügt, aus den verschiedenen Meßwandlern, die in der Serie beschrieben werden, den passenden auszuwählen (sofern das nötig ist), ihn an ein I/O-Interface anzuschließen und dieses mit dem Computer zu verbinden. Dann heißt es, die passende Software zu laden, und schon kann die Messung losgehen.

Grundsätzlich eignet sich jede 8-Bit-Ein-/Ausgabe-Schnittstelle als I/O-Interface. Die in der Serie beschriebene Software gilt jedoch für den PIO-Port (PIO: Parallel In Out), den die FUNKSCHAU in den Heften 3 bis 5/1984 vorgestellt hat. Dabei wurden die Adressen der PIO gemäß Heft 14/1984, Seite 61, so geändert, daß der Sinclair-Logik-Chip (SLC) keinen Unfug mehr treiben kann. Mehr darüber im Kapitel über die Software zur Zeitmessung.

Die weiteren Folgen dieser Serie befassen sich mit der Frequenz-, Kapazitäts- und Widerstandsmessung. Prinzipiell sind jedoch alle physikalisch-technischen Größen meßbar, die sich mit Wandlerschaltungen auf die Größen Zeit oder Frequenz zurückführen lassen. Die eigentliche Messung dieser Größen geschieht mit einem Programm, wobei z. B. der Zeitmessung selbst Tausendstel Sekunden nicht entgehen dürfen!

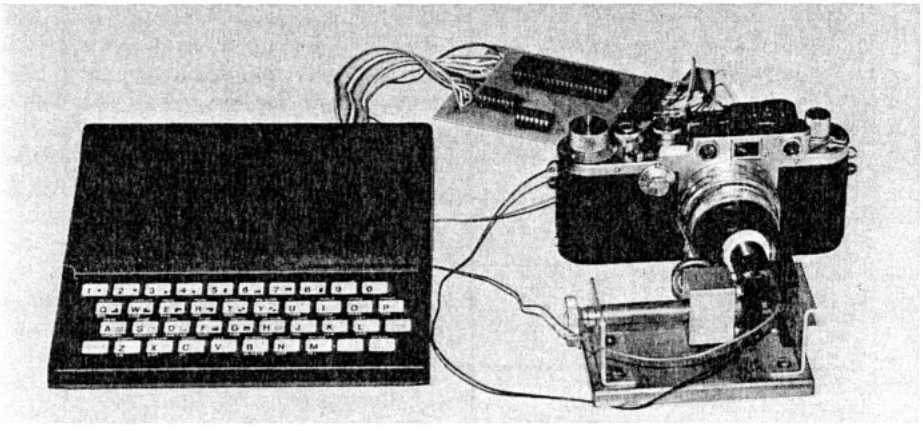
Damit ist klar, daß die Programme in Maschinensprache geschrieben sein müssen. Basic wäre viel zu langsam. Wer die FUNKSCHAU-Serie „Klartext für den ZX 81" verfolgt hat, der kann jetzt seine Kenntnisse nutzen. Die Programme sind jedoch so ausführlich erklärt, daß sie auch jemand versteht, der in der Z-80-Maschinensprache noch nicht ganz sattelfest ist.

Die Messung von Zeiten mit Hilfe des Computers beruht auf dem sehr schnellen und präzisen Takt des Mikroprozessors. Deshalb ist auch kein zusätzlicher Zähler erforderlich, so daß an externer Hardware im Grunde nur der PIO-Port nötig ist. Dieser Baustein hat zwei 8 Bit breite Kanäle (Port A und Port B), von denen bei der Zeitmessung lediglich eine einzige Leitung, nämlich B0 (Bit 0 von Port B), verwendet wird.

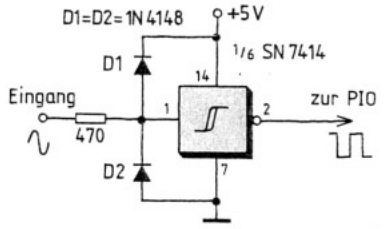
Nur Rechtecksignale sind erlaubt

Da sowohl der Computer als auch der Port-Eingang nur digitale Signale akzeptieren, deren Pegel entweder L (kleiner 1,5 V) oder H (von 3,5 V bis 5 V) ist, muß man die zu messenden Größen, hier also die Zeit, in eine solche digitale Form bringen. Wenn die Meßgröße, z. B. die Impulsdauer eines Signals, bereits in Rechteckform mit TTL-Pegel vorliegt, ist eine besondere Wandlerschaltung nicht erforderlich. Handelt es sich jedoch um ein Dreieck- oder Sinussignal, so kann man mit Hilfe eines Schmitt-Triggers (*Bild 1*) eine Rechteck-Impulsform erreichen.

Die Schaltung kann auch als Schutzschaltung für das teurere PIO-IC betrachtet werden. Deshalb sollte man sie in jedem Fall verwenden. Die Dioden D 1 und D 2 in Verbindung mit dem Widerstand R = 470 Ω sorgen dafür, daß der Eingangsspannungspegel für den Schmitt-Trigger im erlaubten Bereich



Zeitmessung: Dem Einfallsreichtum sind keine Grenzen gesetzt. Mit einer Lichtschranke läßt sich z. B. die Verschlußzeit von Fotoapparaten überprüfen Bild: -II



① **Impulsformer:** Die Schaltung verträgt Eingangspegel bis 10 V und schützt den PIO-Baustein

von 0 V bis 5 V bleibt. Das gilt für von außen angelegte Spannungen bis zu 10 V (Effektivwert).

Das wichtigste an der Funktionsweise eines Schmitt-Triggers ist die „Hysterese". Sie besagt, daß die beiden Umschaltschwellen des Schmitt-Triggers unterschiedlich groß sind. Das garantiert bei jedem noch so zögernd zu- oder abnehmenden Eingangssignal ein prellfreies Ausgangssignal (*Bild 2*). Auf diese Weise entsteht aus einem Sinussignal das für den Computer erforderliche Rechtecksignal.

Die Phasenlage dieses Rechtecksignals ist jedoch entgegengesetzt zum ursprünglichen Signal. Wenn das bei einer Messung stört, schaltet man einfach einen zweiten Schmitt-Trigger des ICs SN 7414 in Serie zum ersten.

Ein Anwendungsbeispiel für die Zeitmessung wäre die Messung der Verschlußzeit eines Fotoapparates. Als lichtempfindliche Sonde dient dann ein Fototransistor BPY 62, dessen Kollektorspannung unmittelbar dem Schmitt-Trigger zugeführt wird. (Kollektorwiderstand 47 kΩ an 5 V). Solange der Verschluß der Kamera geöffnet ist, erhält der Fototransistor Licht und wird niederohmig. Das bedeutet, daß der Schmitt-Trigger ein L-Signal wahrnimmt. Somit erhält der Port-Eingang B0 des Computers während der Verschlußöffnungszeit H-Pegel. Dieser H-Pegel ist die Informationsbasis für das Maschinenprogramm.

Um eine Darstellung der Meßergebnisse am Bildschirm zu erhalten, ist neben dem eigentlichen Zeit-Meßprogramm, das in Maschinensprache geschrieben ist, ein kurzes Basic-Programm erforderlich, das gewissermaßen den Rahmen für das Maschinenprogramm bildet (*Bild 3*).

Das Programm beginnt mit einer REM-Zeile, die im Programmspeicher Platz für das Maschinenprogramm reserviert, das später in den Speicher eingegeben wird. Zeile 19 schaltet den Computer in den FAST-Modus. Das ist wichtig, weil der ZX 81 im SLOW-Modus 50mal in jeder Sekunde ein Bild aufbauen würde. Da aber jeder Bildaufbau den Ablauf des Maschinenprogramms unterbricht, wäre in diesem Fall eine Zeitmessung höchst ungenau.

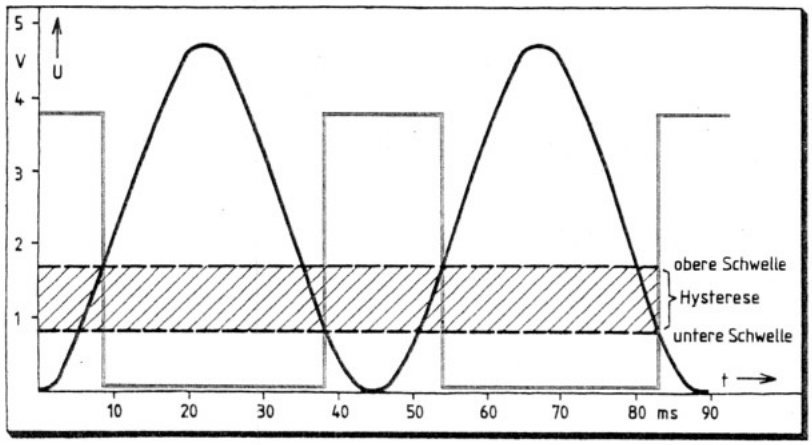
Nachteilig am FAST-Modus ist, daß der Computer während der Messung keine Zeit hat, den Bildschirm zu „bedienen". Das bedeutet, daß man den Zählvorgang leider nicht sichtbar machen kann.

In Zeile 20 wird das Maschinenprogramm für die Zeitmessung mit USR 16514 aufgerufen. Der Meßwert der Zeit, der sich nach erfolgreicher Beendigung des Maschinenprogramms ergibt, wird der Variablen Q zugeordnet.

In Zeile 21 erfolgt die dezimale Umrechnung von Q in Sekunden; dabei gilt, daß das Maschinenprogramm die Zeit in Millisekunden registriert hat. Der Befehl PRINT schreibt den Wert von Q/1000 auf den Bildschirm. In Zeile 22 wird in den SLOW-Modus zurückgeschaltet, worauf die gemessenen Werte solange angezeigt werden, bis die Tastaturabfrage in Zeile 23 eine weitere Messung einleitet (Zeile 24). Dies geschieht durch Drücken der Taste C (Continue).

Mit dem Programmblock ab Zeile 2000 wird das Maschinenprogramm in den Programmspeicher eingegeben. Es beginnt bei Adresse 16514 und endet bei Adresse 16552. 16514 ist die Adresse der Speicherzelle, die das erste Zeichen der REM-Zeile enthält, 16552 enthält das letzte Zeichen. Deshalb ist in der REM-Zeile Platz für 39 Zeichen zu reservieren.

Das Eingabeprogramm ist als erstes nach dem Eintippen durch RUN 2000 zu starten. Es meldet dann die Adresse der jeweiligen Speicherzelle und wartet auf die Eingabe der in *Bild 4* den Adressen zugeordneten Werte. Ist die Programm-Eingabe beendet, stoppt das Programm bei Zeile 2060. Jetzt darf man getrost die Zeilen 2000 bis 2060 löschen, da sie nicht mehr benötigt werden. Dann ist das verbliebene Programm auf Band zu sichern und eine Messung mit RUN einzuleiten.



② **Sinus-Rechteck-Umformung:** Das sinusförmige Eingangssignal wird von einem Schmitt-Trigger in ein Rechtecksignal umgeformt, dessen Tastverhältnis von den Schaltschwellen abhängt

Eine Warteschleife prägt die Zeitmessung

Zum besseren Verständnis des Zeit-Meßprogramms dient ein Flußdiagramm (*Bild 5*): Block 1 enthält die „Initialisierung". Darunter sind alle Maßnahmen zu verstehen, die der Vorbereitung des Prozessors und der PIO auf die folgende Zeitmessung dienen. Für die PIO ist das die Festlegung der Betriebsart. Beim Prozessor besteht die Vorbereitung darin, daß man ein Zählregister auf den Wert Null setzt.

In Block 2 beginnt die Messung. Der augenblickliche logische Pegel auf der als Eingang geschalteten Port-Leitung (B0), wird zum ersten Mal abgefragt. Block 2 dient dazu, daß das Programm einen bereits begonnenen Impuls nicht mißt, sondern sein Ende abwartet.

Block 3 enthält eine Warteschleife, die die Pause zwischen zwei Impulsen überbrückt und in Block 4 beginnt schließlich, nachdem ein neuer Impuls erkannt worden ist, der Zählvorgang für den Zeitzähler.

Damit man die Zeiteinheit, die einem Digit im Zeitzähler entspricht, im Programm festlegen kann, enthält Block 5 eine Warteschleife zum Festlegen der jeweils gewünschten Zeiteinheit. Ohne diese Warteschleife könnte der Computer bis zu etwa 90 000 Impulse je Sekunde zählen. Damit der Zähler in Millisekunden hochzählt, muß die Warteschleife in Block 5 genau 123mal durchlaufen werden. Jedesmal, wenn diese Schleifenzahl erreicht ist, wird der logische Pegel auf der Leitung B0 erneut abgefragt. Ist der Impuls noch vorhanden, wird der Zeitzähler inkrementiert (+1), ist der Impuls inzwischen beendet, erfolgt der Rücksprung (Block 6) in das Basic-Programm, wobei der Stand des Zeitzählers in die Variable Q geladen wird.

③ **Basic-Rahmen-Programm:** Nach dem Eintippen ist mit RUN 2000 zuerst die Eingaberoutine für den Maschinencode aufzurufen

```
   1 REM HIER MINDESTENS 39 BE-
LIEBIGE ZEICHEN EINTIPPEN
  19 FAST
  20 LET Q=USR 16514
  21 PRINT "ZEIT = ";Q/1000;" SE
KUNDEN"
  22 SLOW
  23 IF INKEY$<>"C" THEN GOTO 23
  24 GOTO 19
2000 FOR N=16514 TO 16552
2010 PRINT N;" ";
2020 INPUT D
2030 POKE N,D
2040 PRINT D,
2050 NEXT N
2060 STOP
```

④ **Maschinencode:** Die Eingaberoutine in Bild 3 verlangt für jede Adresse die Eingabe des hier dezimal zugeordneten Maschinencode-Bytes

16514	1	16515	0
16516	0	16517	62
16518	207	16519	211
16520	255	16521	62
16522	1	16523	211
16524	255	16525	219
16526	239	16527	230
16528	1	16529	32
16530	250	16531	219
16532	239	16533	230
16534	1	16535	40
16536	250	16537	3
16538	17	16539	123
16540	0	16541	27
16542	122	16543	179
16544	32	16545	251
16546	219	16547	239
16548	230	16549	1
16550	32	16551	241
16552	201		

Byte für Byte durchs Maschinenprogramm

Nach den Vorbemerkungen ist es nicht mehr schwer, das Maschinenprogramm (*Bild 6*) im Detail zu verstehen. Gemäß Heft 14/1984 wurde die Adressierung der PIO gegenüber Heft 3/1984 geändert und Ausgang Y7 des Adreßdekoders benutzt. Damit haben die PIO-Adressen nicht mehr die Werte F8h, F9h, FAh, FBh, sondern E7h, EFh, F7h und FFh.

Das Programm beginnt mit der Initialisierung der Z-80-CPU. Dazu wird mit *ld bc,00* das bc-Registerpaar, das als Zählregister dient, auf Null gesetzt (Adresse 4082h). Ab Adresse 4085h beginnt die Initialisierung der PIO, indem der Akku mit dem ersten Steuerwort für Betriebsart 3 (Bit-Ein-/Ausgabe) geladen wird. Der Befehl dafür lautet: *ld a,207*. Dieses Steuerwort wird bei Adresse 4087h per out-Befehl in das Steuerwortregister von Port B der PIO (Adresse 255) geschrieben.

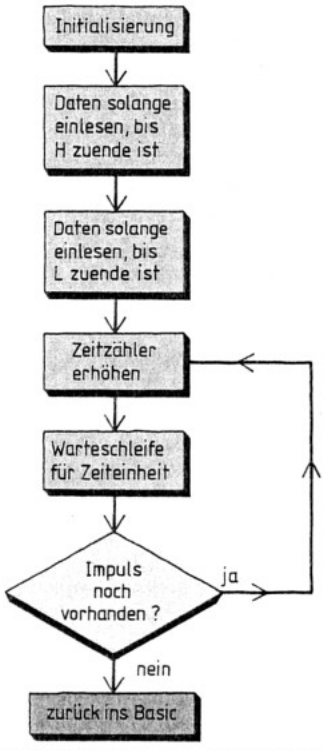
Bei Betriebsart 3 ist nach dem ersten Steuerwort noch ein zweites Steuerwort erforderlich, das festlegt, welche Leitungen von Port B Eingang sind und welche Ausgang. Für die Zeit- und Frequenzmessungen wird Bit 0 als Eingang deklariert und die übrigen sieben Leitungen, die nicht verwendet werden, gelten als Ausgang. Das Steuerwort dafür lautet binär 0000 0001 (dezimal 1). Dieser Wert wird in den Akku und anschließend ebenfalls ins Steuerwortregister von Port B der PIO geladen (Adressen 4089h bis 408Ch). Damit ist die PIO für Messungen an B0 vorbereitet.

Nun ist Block 2 von Bild 5 an der Reihe. Zuerst wird der logische Pegel an B0 in den Akku geladen (408Bh). Der Befehl dafür lautet: *in a,239*. Bei Adresse 408Fh wird dann mit dem Befehl *and 1* geprüft, ob der an B0 anliegende Pegel den Wert HIGH hat. Ist das Ergebnis dieser UND-Verknüpfung ungleich Null, dann liegt bereits ein Impuls an B0 an. Solange das der Fall ist, wird durch einen relativen Sprung (4091h) zur Adresse 408Dh die Pegel-Abfrage wiederholt.

Ist der bereits begonnene Impuls beendet, so gilt die Bedingung (not zero, nz) für den relativen Sprung nicht mehr, und das Programm fährt bei Adresse 4093h fort. Damit geginnt die Arbeit an Block 3 von Bild 5, nämlich solange zu warten, bis der zu messende Impuls eintrifft. Das geschieht ähnlich wie in Block 2: Einlesen des Pegelwertes und Überprüfung auf LOW mit Hilfe eines UND-Befehls (4095h).

Dieser UND-Befehl ergibt nur dann das Ergebnis Null, wenn der Pegel von Bit 0 im Akku und der Pegel an B0 LOW ist. Solange dies der Fall ist, erfolgt bei Adresse 4097h ein relativer Sprung (Jump relative if zero) zurück zur Adresse 4093h, um den Pegel erneut einzulesen. Wenn jedoch an B0 ein neuer HIGH-Impuls registriert wird, beginnt die Zeitmessung.

Das bc-Registerpaar wird dann mit *inc bc* um 1 erhöht. Anschließend läßt sich gemäß Block 5 die Zeiteinheit des



⑤ **Flußdiagramm:** Die Warteschleife für die Zeiteinheit bestimmt, wie genau das Meßergebnis angezeigt wird

```
4082 LD BC,00000        010000
4085 LD A,207           3ECF
4087 OUT 255,A          D3FF
4089 LD A,001           3E01
408B OUT 255,A          D3FF
408D IN A,239           DBEF
408F AND 001            E601
4091 JR NZ,$408D        20FA
4093 IN A,239           DBEF
4095 AND 001            E601
4097 JR Z,$4093         28FA
4099 INC BC             03
409A LD DE,00123        117B00
409D DEC DE             1B
409E LD A,D             7A
409F OR E               B3
40A0 JR NZ,$409D        20FB
40A2 IN A,239           DBEF
40A4 AND 001            E601
40A6 JR NZ,$4099        20F1
40A8 RET                C9
```

⑥ **Assembler-Listing:** Die einzelnen Programmschritte werden im Text ausführlich erklärt

Meßergebnisses festlegen. Registerpaar de erhält dazu die Anzahl der Schleifendurchläufe zugewiesen (409Ah). In Adresse 409Bh steht das niederwertige Byte, in Adresse 409Ch das höherwertige Byte der Schleifenzahl.

Beide Bytes können einen Wert von 0 bis 255 annehmen. Das höherwertige Byte hat dabei aufgrund der hexadezimalen Festlegung eine Wertigkeit von 256 (siehe Teil 2 der Serie „Klartext für den ZX 81"). Für Millisekunden als Zeiteinheit beträgt die Schleifenzahl 123. Für Hundertstelsekunden wäre die Schleifenzahl 1243 erforderlich. Das ergäbe für das höherwertige Byte die Zahl 4, für das niederwertige Byte die Zahl 219 ($1243 = 4 \times 256 + 219$).

Ab Adresse 409Dh beginnt die Zeitschleife. Der Inhalt des Registerpaars de wird um 1 verringert. Bei Adresse 409Eh wird der Inhalt von d in den Akku geladen und anschließend mit dem Inhalt von e ODER-verknüpft. Das hat den Sinn, festzustellen, ob der Inhalt des Registerpaars de Null ist. Solange dies nicht der Fall ist, erfolgt ein bedingter Sprung zurück zur Adresse 409Dh.

Hat der Inhalt von de den Wert 0 erreicht, dann wird der Sprungbefehl bei Adresse 40A0h nicht mehr ausgeführt, da die vorgesehene Zeit z. B. für eine Millisekunde abgelaufen ist. Stattdessen wird der Pegel an B0 erneut eingelesen und mit *and 1* auf HIGH überprüft. Verläuft die Überprüfung auf HIGH positiv, so erfolgt bei Adresse 40A6h ein relativer Sprung zurück zu 4099h.

Dieser Ablauf wiederholt sich solange, bis die Überprüfung des Pegels an B0 LOW ergibt, d. h. bis der Impuls beendet ist. Der ret-Befehl (40A8h) bewirkt dann den Rücksprung ins Basic-Programm.

Hubert Joas/-ll

ZX-81-Ersatzteile:

Eine Quelle für den Logik-Chip

Der ZX 81 fordert es geradezu heraus, daß ihm Hobby-Elektroniker dicht auf den Pelz rücken. Oft läßt sich der Computer nur so gefügig machen, wenn selbst ausgefuchste Software allein nicht zum Ziel führt. Diese Herausforderung hat in der FUNKSCHAU-Redaktion freilich schon drei ZX-81-Computern das „Leben" gekostet: Sie überstanden die Testphase einer Bauanleitung nicht. Und jedesmal zeigte sich, daß ausgerechnet der geheimnisvolle Logik-Chip in die Knie gegangen war.

Wer die gleiche leidvolle Erfahrung sammeln mußte weiß, daß die Beschaffung des Logik-Chips bislang ein schier auswegloses Unterfangen war. Aribert Deckers in Stuttgart ist es jedoch gelungen, einen Posten dieser ICs zu ergattern und er verkauft sie jetzt zum Stückpreis von 40 DM (zuzüglich Versandkosten). Anschrift und Telefonnummer der Firma sind bei der Redaktion abzurufen. Sie gleich hier preiszugeben verbieten uns leider die Bestimmungen des Post-Zeitungsdienstes.

ZX-81-Hardwaretip:

Monitorausgang für scharfe Bilder

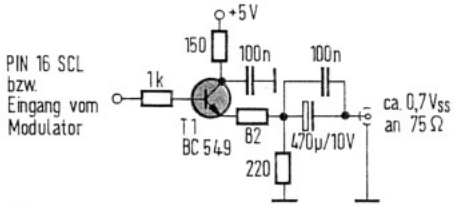
Ein Monitorausgang für das Videosignal des ZX 81 bringt gleich zweifachen Nutzen: Neben dem Plus an Bildschärfe entfällt auch das lästige Nachstimmen am Fernsehgerät, wenn der Modulator im ZX 81 mit zunehmender Erwärmung ebenfalls auf „Wanderschaft" geht.

Bei einigen Fernsehgeräten genügt es bereits, lediglich das Videosignal an Pin 16 des Sinclair-Logik-Chips (SLC) abzugreifen und in die AV-Buchse einzuspeisen (Koaxkabel, Schirm auf Masse legen). Sollte das nicht zum Erfolg führen, so ist ein Emitterfolger *(Bild 1)* erforderlich. Er liefert ein gleichspannungsfreies positives Videosignal von ca. 0,7 V_{ss} an 75 Ω. Die Schaltung kann auf einer kleinen Lochrasterplatte aufgebaut und in den Computer eingebaut werden.

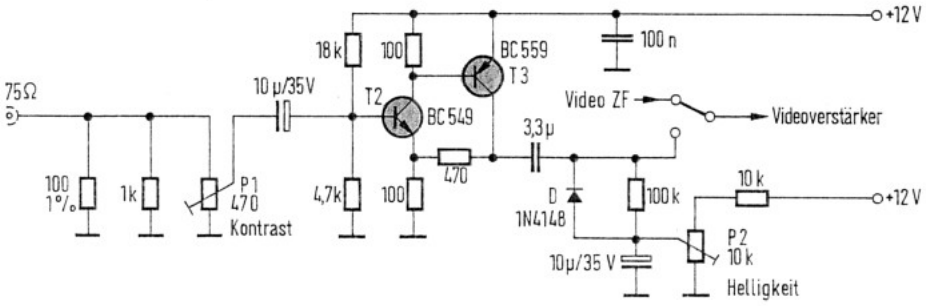
Vor allem ältere Fernsehgeräte benötigen jedoch für den Videoverstärker Eingangsspannungen zwischen 2 und 3 V_{ss}. In diesem Fall ist zusätzlich noch der Vorverstärker gemäß *Bild 2* notwendig, der eine Spannungsverstärkung um den Faktor 5 bietet. Die Klemmschaltung mit der Diode D stellt den Gleichspannungspegel des Videosignals wieder her. Der Abgleich ist einfach: Mit den beiden Potentiometern wird das ZX-81-Bild so eingestellt, daß sich Kontrast und Helligkeit nicht von einem üblichen Fernsehbild unterscheiden.

Leider gibt es kein Patentrezept, wie der Vorverstärker in Fernsehgeräte einzubauen ist. Anhaltspunkte gibt jedoch der FUNKSCHAU-Beitrag „Scharfmacher" in Heft 8/1983. Auf jeden Fall sollte sich hier nur ein Fachmann heranwagen, da bei unsachgemäßem Einbau Lebensgefahr besteht.

Mr. X



① **Monitorausgang:** Die kleine Schaltung paßt noch ohne weiteres ins Gehäuse des ZX 81



② **Vorverstärker:** Diese Schaltung für ältere Fernsehgeräte ohne AV-Eingang sollte nur ein versierter Fachmann einbauen

Messen mit dem ZX 81 (2):

Frequenzmessung mit Präzision

Ohne Vorteiler kann man mit dem ZX 81 Frequenzen bis ungefähr 40 kHz messen. Dabei ist das Ergebnis, abhängig von der Meßdauer, erstaunlich genau. Das gilt übrigens ganz allgemein für jeden Heimcomputer.

Für die Frequenzmessung per Computer ist wieder ein paralleler Ein-/Ausgabe-Port erforderlich, wobei jedoch nur ein einziger Kanal benötigt wird. Einzelheiten dazu sind in Teil 1 (FUNKSCHAU 24/1984, Seite 75) nachzulesen.

Mit der Meßdauer steigt die Genauigkeit

Frequenzmessungen werden digitaltechnisch im allgemeinen so durchgeführt, daß man die Anzahl der Schwingungen innerhalb einer festgelegten Torzeit zählt. Hat diese Torzeit eine Dauer von einer Sekunde, so ist die Zahl der Schwingungen identisch mit der Frequenz in Hertz (Anzahl der Schwingungen pro Sekunde).

Per Programm ist eine Frequenzmessung nach diesem Prinzip denkbar einfach. Sie setzt jedoch voraus, daß der Computer ein Interface mit einem externen Timer hat. Dann übernimmt dieser Timer die Aufgabe, die Torzeit festzulegen. Der Computer muß lediglich die Anzahl der Impulse zählen, die während der Torzeit an seinen Eingang gelangen.

Ohne externen Timer ist eine Frequenzmessung softwaremäßig ein wenig schwieriger, aber trotzdem lösbar. Die Idee besteht darin, daß ein Softwarezähler in einem Maschinenprogramm die Dauer einer Vollschwingung der zu messenden Frequenz ermittelt. Mit einem Basic-Programm wird diese Zeitdauer gemäß f = 1/T anschließend in einen Frequenzwert umgerechnet.

Die Genauigkeit einer Messung nach diesem Prinzip hängt vom Digitalisierungsfehler ab, der durch die Zykluszeit des Softwarezählers zustande kommt. Um den Softwarezähler um 1 zu erhöhen, sind beim ZX 81 mit den beschriebenen Programmen 36 Taktzyklen à 309 ns erforderlich. Das ergibt 11,13 µs. Dieser Wert ist sehr klein, wenn man damit eine Zeit von einer Sekunde (1 Hz) digitalisiert. Bei 1 Hz z. B. ergibt sich ein Zählerstand von etwa 90 000 (exakt: 89 805). Das entspricht einem Digitalisierungsfehler von $^1/_{90000}$ Hz, also einer Auflösung von 0,000011 Hz. In der feinen Auflösung bei niedrigen Frequenzen liegt auch der Vorteil dieser Meßmethode.

Bei 10 Hz ist die Auflösung wegen der kürzeren Periodendauer (Meßzeit) $^{10}/_{9000}$ Hz ≈ 0,0011 Hz, bei 100 Hz immerhin noch 0,1 Hz, bei 1 kHz allerdings nur noch 10 Hz. Man kann jedoch die Auflösung durch Verlängern der Meßzeit erhöhen. Wenn man die Meßzeit um den Faktor 100 verlängert, dann verbessert das die Auflösung auf $^1/_{100}$ ihres ursprünglichen Werts. Bei 1 kHz ergibt z. B. eine bescheidene Erhöhung der Meßzeit von ursprünglich $^1/_{1000}$ s auf 0,1 s eine Verbesserung der Auflösung auf 0,1 Hz.

Nach diesem Verfahren kann man praktisch bei jeder Frequenz durch Erhöhen der Meßzeit die erwünschte Auflösung erreichen. Das heißt jedoch nicht, daß mit diesem Softwarezähler jede noch so hohe Frequenz gemessen werden kann. Die Grenze ist dann erreicht, wenn die Schwingungsdauer weniger als das Doppelte der Zykluszeit des Softwarezählers beträgt. Das ist beim ZX 81 theoretisch ab 45 kHz der Fall, praktisch jedoch schon etwas früher. Höhere Frequenzen kann man dennoch messen, wenn man eine Vorteilung der Frequenz vornimmt.

```
4082 LD BC,00000      010000
4085 LD A,207         3ECF
4087 OUT 255,A        D3FF
4089 LD A,001         3E01
408B OUT 255,A        D3FF
408D IN A,239         DBEF
408F AND 001          E601
4091 JR NZ,$408D      20FA
4093 IN A,239         DBEF
4095 AND 001          E601
4097 JR Z,$4093       28FA
4099 INC BC           03
409A IN A,239         DBEF
409C AND 001          E601
409E JR NZ,$4099      20F9
40A0 INC BC           03
40A1 IN A,239         DBEF
40A3 AND 001          E601
40A5 JR Z,$40A0       28F9
40A7 RET              C9
```

① **50-Hz-Meßprogramm:** Dieses Maschinenprogramm mißt die Netzfrequenz mit einer Auflösung von 3 mHz

Frequenzlupe für die Netzspannung

Als erstes Anwendungsbeispiel ist die Messung der Netzfrequenz (50 Hz) angesagt. Dies soll jedoch eine Präzisionsmessung werden. Deshalb wird die Netzfrequenz durch den Faktor 10 geteilt (größere Meßzeit). Damit läßt sich die Auflösung von 0,03 Hz auf 0,003 Hz verbessern. Die Hardware dazu sieht so aus: Zuerst wird die Netzspannung von 220 V mit einem Trafo auf etwa 5 bis 10 V heruntergeteilt und zur Impulsformung einem Schmitt-Trigger (siehe Teil 1, FUNKSCHAU 24/1984) zugeführt.

Um die Unterteilung mit dem Faktor 10 zu erhalten, wird dieses Signal auf einen BCD-Zähler SN 7490 geschickt. Das unterteilte Signal gelangt anschließend an den Eingang B0 des Z-80-Portbausteins.

Eine Meßreihe beim Autor zeigte, daß die Netzfrequenz nicht konstant ist, sondern innerhalb weniger Sekunden zwischen 49,969 Hz und 49,986 Hz schwankte. Der Digitalisierungsfehler der Messung von durchschnittlich 0,0027 Hz macht sich auf der letzten Stelle mit Sprüngen von 3 bzw. 2 Digits bemerkbar.

Der Mittelwert aus 22 Messungen der Netzfrequenz war 49,980 Hz. Das ist ei-

ne Abweichung von 0,02 Hz von 50 Hz. Wenn man unterstellt, daß diese Abweichung den ganzen Tag über andauern würde, so ergäbe sich daraus für eine netzgesteuerte Uhr eine Gangungenauigkeit von (0,02 Hz/50 Hz) · 86 400 s = 35 s pro Tag. Tatsache ist jedoch, daß die Netzfrequenz von der Belastung des Lichtnetzes abhängt. Tagsüber erhält man häufig Durchschnittswerte unter 50 Hz, nachts erfolgt im allgemeinen die Kompensation, so daß die besagte Uhr, langfristig gesehen, doch „richtig" geht.

```
  10 REM HIER 38 BELIEBIGE ZEICH
EN EINTIPPEN
  15 FAST
  20 LET Q=USR 16514
  25 PRINT "FREQUENZ = ";10/1000
0*INT (898050000/Q+.5);" HZ"
  30 SLOW
  35 IF INKEY$<>"C" THEN GOTO 35
  40 GOTO 15
```

② **Auswerteprogramm I:** Hier wird das 50-Hz-Meßprogramm aufgerufen und das Ergebnis in Hertz umgerechnet

```
4082 CALL 16524      CD8C40
4085 CALL 16536      CD9840
4088 CALL 16549      CDA540
408B RET             C9
408C LD BC,00000     010000
408F LD A,207        3ECF
4091 OUT 255,A       D3FF
4093 LD A,001        3E01
4095 OUT 255,A       D3FF
4097 RET             C9
4098 IN A,239        DBEF
409A AND 001         E601
409C JR NZ,$4098     20FA
409E IN A,239        DBEF
40A0 AND 001         E601
40A2 JR Z,$409E      28FA
40A4 RET             C9
40A5 INC BC          03
40A6 IN A,239        DBEF
40A8 AND 001         E601
40AA JR NZ,$40A5     20F9
40AC INC BC          03
40AD IN A,239        DBEF
40AF AND 001         E601
40B1 JR Z,$40AC      28F9
40B3 RET             C9
40B4 CALL 16524      CD8C40
40B7 CALL 16536      CD9840
40BA LD D,100        1664
40BC CALL 16549      CDA540
40BF DEC D           15
40C0 JR NZ,$40BC     20FA
40C2 RET             C9
```

③ **Universelles f-Meßprogramm:** Die automatische Meßbereichsumschaltung bei 200 Hz erhöht die Meßgenauigkeit

```
  10 REM HIER 65 BELIEBIGE ZEICH
EN EINTIPPEN
  15 FAST
  20 LET Q=USR 16514
  25 IF Q>=449 THEN GOTO 60
  30 LET Q=USR 16564
  35 LET Q=Q+90
  40 PRINT "FREQUENZ = ";INT (89
80500/Q+.5);" HZ"
  45 SLOW
  50 IF INKEY$<>"C" THEN GOTO 50
  55 GOTO 15
  60 PRINT "FREQUENZ = ";INT (89
805/Q+.5);" HZ"
  65 SLOW
  70 IF INKEY$<>"C" THEN GOTO 70
  75 GOTO 15
```

④ **Auswerteprogramm II:** Zeile 25 bewirkt die Meßbereichsumschaltung: Ab 200 Hz wird ein Maschinenprogramm zur Messung von 100 Vollperioden aufgerufen

Eine Vollschwingung wird ausgezählt

Die soeben geschilderte Methode der Frequenzmessung läßt sich mit jedem Mikroprozessor verwirklichen. Mit dem Prozessor Z 80 sieht das Maschinenprogramm so aus, wie in *Bild 1* gezeigt. In den Speicherzellen 4082h bis 408Ch erfolgt die Initialisierung des Prozessors und der PIO. Von Adresse 408Dh bis 4098h wird das Ende eines bereits begonnenen Impulses abgewartet. Diese Prozeduren sind dieselben wie im Zeitmeßprogramm (Teil 1, FUNKSCHAU 24/1984).

Ab Adresse 4099h beginnt die eigentliche Messung der Frequenz. Der Softwarezähler wird hier inkrementiert. Von 409Ah bis 409Dh wird der logische Pegel auf der Portleitung B0 eingelesen und auf HIGH überprüft. Wenn diese Prüfung positiv verläuft, erfolgt bei 409Eh ein Rücksprung nach 4099h. Dieser Vorgang wiederholt sich so lange, bis die HIGH-Phase der Rechteckschwingung beendet ist.

Dann erfolgt ab 40A0h die Zählung der Taktzyklen in der LOW-Phase des Signals. Wenn der bedingte Sprung bei 40A5h nicht mehr ausgeführt wird, ist auch die LOW-Phase des Signals zu Ende. Damit ist ein kompletter HIGH-LOW-Zyklus der Schwingung abgelaufen, und das Programm kehrt mit der Information (Zählerstand des bc-Registers) in ein Basic-Programm zurück, das die Berechnung der Frequenz vornimmt. Dieses Programm zeigt *Bild 2*.

Die einzig bemerkenswerte Zeile in diesem Programm ist Zeile 25. Hier wird die Frequenz auf vier Stellen hinter dem Komma genau berechnet (deshalb der Faktor 10000), wegen der Vorteilung mit dem Faktor 10 multipliziert und dann ausgegeben. Mit Zeile 35 kann dann durch Druck auf die Taste C eine neue Messung gestartet werden. Bei Frequenzen unter 1,37 Hz läuft der Softwarezähler allerdings über. Der Wert von Q ist dann um 65 536 Digits zu klein.

Eingabe der Maschinenprogramme

Bild 1 und Bild 3 zeigen in der rechten Spalte die Hex-Codes der Maschinenprogramme. Diese Codes müssen als erstes einfach der Reihe nach (z. B. CD8C40CD98...) einem Hex-Code-Eingabeprogramm übergeben werden. Einzelheiten dazu sind in den ZX-81-Sonderheften der FUNKSCHAU oder in Heft 12/1983, Seite 76, nachzulesen.

Aufgabe des Eingabeprogramms ist es, das Maschinenprogramm genau in dem Speicherbereich unterzubringen, der in den Basic-Listings (Bild 2 und Bild 4) mit Zeile 10 dafür reserviert wurde. Rufen Sie in der Redaktion an (Tel. 0 89/51 17-3 15), wenn Sie mit der Hex-Code-Eingabe noch nicht zurechtkommen.

Automatische Meßbereichs-Umschaltung

Wie das Anwendungsbeispiel gezeigt hat, ist das Maschinenprogramm hauptsächlich dazu geeignet, sehr kleine Frequenzen sehr genau zu messen. Bei 200 Hz beträgt der Digitalisierungsfehler aber bereits 0,4 Hz. Ab dieser Frequenz ist es daher sinnvoll, die Auflösung zu erhöhen. Das soll jetzt jedoch nicht, wie bei der Messung der Netzfrequenz, durch Vorteilung der Frequenz, sondern softwaremäßig erreicht werden.

Dazu wird im Maschinenprogramm nicht nur eine Schwingung des Meßsignals ausgezählt, sondern ein Vielfaches davon. Um eine spürbare Verbesserung zu erreichen, werden 100 Schwingungen berücksichtigt.

Damit die Vervielfachung der Meßzeit erst da einsetzt, wo sie sinnvoll ist, muß das Programm die vorgesehenen Frequenzbereiche automatisch erkennen und nahtlos aneinanderfügen. Im anderen Falle würden die Meßzeiten für niedrige Frequenzen unerträglich hoch. Am besten probieren wir das einmal mit zwei Frequenzbereichen aus. Im ersten Bereich sollen Frequenzen bis 200 Hz gemessen werden, im zweiten bei 100fach verbesserter Auflösung Fre-

quenzen von 200 Hz bis 2 kHz. Das dafür erforderliche Maschinenprogramm zeigt *Bild 3*.

Es besteht im Prinzip aus zwei Einzelprogrammen für die beiden Frequenzbereiche. Da jedoch in beiden Programmen im wesentlichen dieselben Vorgänge ablaufen, kann man mit Hilfe der Unterprogrammtechnik einiges an Bytes einsparen. Aus diesem Grunde ist das Programm für den Grundfrequenzbereich bis 200 Hz auch nicht identisch mit dem Standardprogramm von Bild 1. Das erkennt man schon an dem unterschiedlichen Speicherplatzbedarf. Es belegt die Speicherzellen von 16514 (4082h) bis 16563 (40B3h) und enthält drei Subroutinen. Diese Subroutinen werden bei den Adressen 4082h, 4085h und 4088h aufgerufen.

Die erste Subroutine erledigt die Initialisierung der PIO und des Prozessors. Sie steht ab 408Ch im Speicher. Die zweite Subroutine sorgt dafür, daß die Messung nicht während einer bereits begonnenen Periode einsetzt, sondern immer am Anfang einer Schwingung. Diese Routine reicht von 4098h bis 40A4h.

Die eigentliche Meßroutine, die eine volle Periode der Schwingung auszählt, belegt die restlichen Zellen von 40A5h bis 40B3h. Nach dem Rücksprung aus dieser Routine kehrt der Computer bei Adresse 408Bh in das Basic-Programm zurück, das die Berechnung der Frequenz durchführt und die automatische Wahl des richtigen Frequenzbereiches vornimmt. Bevor wir dieses Basic-Programm genauer betrachten, noch ein kurzer Blick auf das Maschinenprogramm für den zweiten Frequenzbereich. Es belegt in Bild 3 die Speicherzellen von 40B4h bis 40C2h.

Bei Adresse 40B4h und 40B7h erfolgt der Aufruf der beiden ersten Subroutinen. Bis hierher besteht kein Unterschied zu dem vorherigen Frequenzbereich. Ab 40BAh wird jedoch das d-Register mit 100 geladen. Das ist die Voraussetzung dafür, daß der Prozessor 100 Vollperioden hintereinander mißt. Bei Adresse 40BCh wird dann die Meßroutine für eine Vollperiode aufgerufen. Dieser Aufruf ist Kernstück einer Schleife, die insgesamt 100mal durchlaufen wird.

Nach jedem Durchlauf wird das d-Register dekrementiert (40BFh) und auf Null überprüft (40C0h). Wenn der Wert Null erreicht ist, sind die 100 Vollperioden ausgezählt, und der Prozessor kehrt bei 40C2h in das Basic-Programm (*Bild 4*) zurück.

In Zeile 20 erfolgt dort der Aufruf des Maschinenprogramms für den ersten Frequenzbereich. Das Ergebnis der Auszählung einer Vollperiode wird dabei der Variablen Q zugeordnet. Der Wert von Q dient in Zeile 25 zur Unterscheidung der beiden Frequenzbereiche. Ist $Q \geqq 449$, so handelt es sich um eine Frequenz, die kleiner oder gleich 200 Hz ist. In diesem Fall erfolgt in Zeile 25 ein Sprung zur Zeile 60. Hier wird die Frequenz auf 1 Hz genau berechnet und ausgegeben. Ist Q kleiner als 449, so wird in Zeile 30 das Maschinenprogramm für den zweiten Frequenzbereich aufgerufen.

Die Auszählung von 100 Perioden der Schwingung müßte für Q einen Wert ergeben, der 100mal größer ist als bei einer einzigen Periode. Eine genaue Überprüfung zeigt jedoch, daß dies nicht exakt der Fall ist, sondern daß dieser Wert ein wenig zu klein ist. Deshalb wird Q in Zeile 35 korrigiert.

Dieser kleine Schönheitsfehler, der sich durch die Korrektur im gesamten zweiten Frequenzbereich eliminieren läßt, hat seine Ursache im Maschinenprogramm. Das hängt damit zusammen, daß der Computer nicht zwei Dinge gleichzeitig tun kann. Zwischen je zwei Perioden ist der Prozessor nämlich damit beschäftigt, das d-Register zu dekrementieren, dessen Inhalt auf Null zu überprüfen und anschließend die Subroutine zur Messung einer neuen Periode aufzurufen. Dafür benötigt er 4 + 12 + 17 = 33 Taktzyklen à 309 ns. Das bedeutet einen Zeitverlust von 10,2 µs, der sich bei 100facher Addition am Softwarezähler mit einem Mangel von etwa 90 Digits bemerkbar macht.

Mit der Addition der fehlenden Digits erhält man im gesamten Frequenzbereich von 200 Hz bis 2 kHz exakte Werte der Frequenz. Diese werden in Zeile 40 des Basic-Programms auf 1 Hz genau berechnet und ausgegeben. Hubert Joas

(*Wird fortgesetzt*)

Neues Disketten-Laufwerk:

Gegen alle Zoll-Schranken

Nachdem für Heimcomputer bislang ausschließlich 5¼-Zoll-Floppy-Stationen angeboten wurden, und die kommende Generation der MSX-Heimcomputer auf 3½-Zoll-Laufwerke baut, ist es überraschend, daß sich Sharp jetzt für ein 2,8-Zoll-Laufwerk entschieden hat. Sharp bietet diese Disketten-Station für die Heimcomputer der Serie MZ-700 an, baut sie jedoch nicht selber, so daß damit zu rechnen ist, daß die Station über kurz oder lang auch für andere Heimcomputer angeboten wird.

Die 2,8-Zoll-Disketten bieten je Seite eine Speicherkapazität von 64 KByte und sollen etwa 8 DM kosten. Erstaunlich klein sind die Abmessungen des Laufwerks: Es nimmt bei den MZ-700-Computern den Platz des Kassettenrecorders ein, wobei der Austausch selber vorgenommen werden kann. Das Ergebnis (*Bild*) darf sich sehen lassen.

Mit zum Lieferumfang gehört eine Diskette, die ein (zum Steuern des Laufwerks) erweitertes Betriebssystem bietet. Es umfaßt 32 KByte und ist ruck, zuck in nur 4 s in den RAM-Speicher der Computer zu laden. Wer will da noch vor Kassettenrecordern geduldig verharren? Sharp betont, daß das neue Betriebssystem zu vorhandenen Basic- und Maschinenprogrammen kompatibel ist. Um sie ein letztes Mal von Band laden zu können (vor dem Speichern auf Diskette), läßt sich der Kassettenrecorder auch parallel zur Disketten-Station betreiben. Nicht zuletzt ist die Preisrelation zwischen der Station und den Computern vernünftig: Sharp empfiehlt für das Laufwerk einen Preis von 750 DM. -ll

Zwerg-Disketten-Laufwerk: Nur 15 cm breit, 12,8 cm tief und 6 cm hoch, fügt sich das 2,8-Zoll-Laufwerk fugenlos in die Computer der Serie MZ-700 ein Bild: Sharp

Messen per Computer

Nützliche Anwendungen für einen Heimcomputer zu finden, ist gar nicht so einfach. Unsere Serie, die wir übrigens auch für den C 64 planen, zeigt Hobby-Elektronikern einen Ausweg aus der Misere: Nur bei Bedarf wird der Heimcomputer zum Universal-Meßgerät. Das erspart die Anschaffung teurer Spezialmeßgeräte, die ohnehin nur selten gebraucht werden.

Messen mit dem ZX 81 (3):

Prüfstand für Kapazitäten

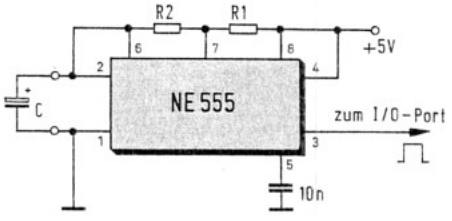
Große Kondensatoren nehmen es mit dem aufgedruckten Kapazitätswert nicht so genau. Besonders deutlich machen das Elektrolytkondensatoren. Computer freilich loten die Kapazität der „dicken Brummer" bis aufs Mikrofarad genau aus.

Das Schlimmste wäre schon geschafft: Wenn Sie das in Teil 1 der Serie beschriebene Zeit-Meßprogramm eingetippt haben, dann fehlt zur Kapazitätsmessung nicht mehr viel. Lediglich eine Wandlerschaltung ist noch erforderlich und selbstverständlich wieder die PIO-Schnittstelle.

Die Ladedauer entlarvt Kondensatoren

Die Aufgabe, den Kapazitätswert eines Kondensators in eine Impuls-(Zeit)Dauer umzusetzen, birgt keinerlei Probleme, denn schließlich ist die Lade- bzw. Entladedauer eines Kondensators proportional zu seiner Kapazität. Und was liegt näher als einen Kondensator ständig zu laden und zu entladen: dazu muß er nur das frequenzbestimmende Element eines Oszillators sein.

Für die Oszillatorschaltung eignet sich z. B. das Timer-IC NE 555 oder dessen CMOS-Version ICM 7555. In der typischen Beschaltung des ICs als astabiler Multivibrator wird das Meßobjekt – ein Kondensator – auf die Grenzwerte ⅔ und ⅓ der Betriebsspannung ge- bzw. entladen (*Bild*). Dabei ist die Frequenz, mit der das geschieht, weitgehend unabhängig vom Wert der Betriebsspannung.



C-Meßschaltung: Die Kapazität des Prüflings C bestimmt maßgeblich die Impulsdauer am Ausgang des ICs

Die beiden Grenzwerte legt ein Spannungsteiler im Timer-IC fest. Wenn die Werte exakt stimmen, gilt für die Ladedauer des Kondensators die Formel:

$t_L = \ln 2 \times (R1 + R2) \times C =$
$0{,}69 \times (R1 + R2) \times C$

Für die Entladedauer gilt:

$t_E = 0{,}69 \times R2 \times C$

Einleuchtend sind die Formeln, wenn man weiß, daß der Kondensator C über die Widerstände R1 und R2 seitens der Betriebsspannung geladen, aber nur über R2 entladen wird. Zum Entladezeitpunkt legt nämlich das IC intern Anschluß 7 auf Masse.

Der Faktor 0,69 fußt auf den beiden Grenzwerten der Spannung am Kondensator. Interne Komparatoren fragen diese Spannung an Anschluß 2 und 6 ab und leiten beim Erreichen der Grenzwerte die Ladung bzw. Entladung des Kondensators ein.

Sowohl die Ladedauer als auch die Entladedauer sind gemäß der Formeln direkt proportional zur Kapazität des Kondensators. Das bedeutet, daß ein doppelt bzw. dreimal so großer Kapazitätswert eine doppelt oder dreifach so lange Lade- bzw. Entladedauer verursacht. Dies bildet die Grundlage für das Meßverfahren.

Der Ausgang des ICs führt während der Aufladung des Kondensators H-Pegel und während der Entladung L-Pegel. Da die Software H-Impulse zur Messung benötigt, muß deshalb die Zeit während der Ladung als Meßgröße dienen. Damit dann ein Kondensator mit einer Kapazität von 1 µF z. B. eine Ladezeit von 1 ms aufweist, muß gemäß der Formel für t_L die Summe der Widerstände R1 und R2 genau 1,44 kΩ betragen. Um diesen Wert zu erreichen, wählt man am besten für R1 einen 1-kΩ-Metallfilmwiderstand und für R2 ein 10-Gang-Spindelpotentiometer (1 kΩ), das auf 440 Ω eingestellt wird.

Das Spindelpotentiometer bietet die Möglichkeit zum Kalibrieren der Schaltung mit Hilfe eines Eichkondensators. Dies ist für exakte Kapazitätsmessungen erforderlich, da der Faktor 0,69 nicht bei jedem Exemplar des Timers 555 genau stimmt (Toleranz ±2 %). Entscheidend für eine präzise Messung der Kapazität ist jedoch die Temperaturkonstanz des Oszillators. Diese ist mit 90 ppm (ppm: parts per million; Faktor 10^{-6}) pro Grad Celsius vergleichsweise gut.

Um die Temperaturkonstanz nicht nennenswert zu verschlechtern, sollten die Temperaturkoeffizienten des Widerstands und des Potentiometers möglichst 100 ppm nicht überschreiten. Die CMOS-Version des Timers hat übrigens eine noch bessere Temperaturkonstanz von 50 ppm.

Wählt man die Summe der Widerstände aus R1 und R2 nicht zu 1,44 kΩ, sondern zu 1,44 MΩ, dann hat ein Kondensator mit der Kapazität 1 nF eine Ladezeit von 1 ms. Mit der Wandlerschaltung lassen sich also ohne weiteres auch Kapazitätswerte im Bereich von Nano-Farad messen. Das Ausgangssignal der Oszillatorschaltung mit dem Timer 555 ist in jedem Fall computergerecht, d. h. es kann unmittelbar dem Interface-Baustein (Bit B0) des Computers zugeführt werden.

Die Software prägt die Genauigkeit

Das Maschinenprogramm zur Kapazitätsmessung ist identisch mit dem Zeit-Meßprogramm aus FUNKSCHAU 24/1984, Seite 77. Dieses Programm mißt die Impulsdauer (Ladephase) mit einer Auflösung von 1 ms. Dimensioniert man dann die Oszillatorschaltung – wie zuvor beschrieben – so, daß ein 1-µF-Kondensator eine Ladezeit von 1 ms aufweist, lassen sich Kondensatoren auf 1 µF genau messen.

Ändert man die Zahl der Schleifendurchläufe im Zeit-Meßprogramm von 123 auf 12, steigt die Auflösung auf 0,1 ms und es lassen sich Kondensatoren auf 0,1 µF genau messen. In beiden Fällen ist der vom Maschinenprogramm an die Basic-Variable Q übergebene Zählerstand gleichbedeutend mit dem Meßwert. Deshalb ist am Basic-Programm (Heft 24/1984, Seite 77, Bild 3), das das Zeit-Meßprogramm aufruft, eine kleine Änderung vorzunehmen. Der PRINT-Befehl zur Meßwertanzeige muß jetzt so lauten:

PRINT "C = ";Q;" MIKRO-FARAD".

Hubert Joas/-ll

(*Wird fortgesetzt*)

ZX-Softwaretip:

Elegant Verzweigen

Normalerweise werden mit dem Basic der Sinclair-Heimcomputer Verzweigungen mit Hilfe der INKEY$-Funktion programmiert, z. B.

```
10 IF INKEY$="A" THEN GOTO 480
20 IF INKEY$="B" THEN GOTO 490
```

usw. bis

```
50 GOTO 10
```

Bei vielen Verzweigungen ist dieses Verfahren umständlich und strapaziert beim Eintippen der Programmzeilen die Geduld des Programmierers. Dann ist es besser, sämtliche IF-INKEY$-Aufrufe durch eine einzige Anweisung zu ersetzen:

```
10 GOTO 100+10*CODE INKEY$
```

Solange keine Taste gedrückt wird (Code = 0) arbeiten die Computer diese Programmzeile in einem Endloskreislauf ab. Ein Tastendruck führt jedoch zum Sprung in eine Programmzeile, deren Zeilennummer maßgebend vom Code der gedrückten Taste abhängt. Wird z.B. Taste A gedrückt (Code = 38 beim ZX81), dann verzweigt das Programm zur Zeile 100+10*38 = 480. Taste B würde zur Zeile 490 verzweigen.

Aus den Codelisten in den Computer-Handbüchern läßt sich schnell ablesen, welche Tastenbetätigungen bei diesem Verfahren sinnvoll sind und welche Zeilennummern damit erreichbar sind. Noch eleganter wird es, wenn die Computer auch melden, daß sie auf eine Eingabe warten, z.B. mit der Zeile

```
99 PRINT "EINGABE?"
```

Diese Meldung am Bildschirm ließe sich nach der Eingabe wieder löschen, wenn der erste Befehl im angesprungenen Programmblock CLS lautet. Hier läßt es sich vorzüglich experimentieren. Eines ist jedoch nicht ratsam, nämlich solche Programme mit einer RENUMBER-Routine neu zu numerieren.

Martin Rötter

ZX-81-Softwaretip:

Große Ziffern

Zu klein sind die Zeichen, die der ZX 81 am Bildschirm abbildet, wenn ein großer Betrachtungsabstand eingenommen wird. Das kann z. B. bei Zeitanzeigen empfindlich stören. Mit einem kleinen Programm (*Bild 1*) lassen sich indes auch große Ziffern am Bildschirm zeigen, und zwar als 5×7-Punktmatrix mit einer Punktgröße, die vom PLOT-Befehl bestimmt wird. Auf einem 31-cm-Bildschirm haben die Ziffern somit eine Höhe von 4,5 cm. Selbstverständlich läßt sich das Programm auch als Unterprogramm nutzen: Übergabevariable ist dann die Zeichenkette A$. Die darin enthaltenen Ziffern werden vom Programm groß am Bildschirm dargestellt.

In der gezeigten Form eignet sich das Programm eher für Demonstrationszwecke. Es zeigt achtstellige Zahlen, die man nach einer INPUT-Aufforderung eingetippt hat. Interessant ist, daß mit Zeile 10 so etwas wie eine DATA-Zeile im Programm enthalten ist. Hierin steckt die Information für die Ziffern 0 bis 9, wie sie der PLOT-Befehl braucht. Als erstes ist daher die Information in Zeile 10 unterzubringen. Dazu wird mit GOTO 200 eine Eingaberoutine gestartet. Jetzt sind die in Bild 1 unten gezeigten Daten einfach Zeile für Zeile der Reihe nach einzeln einzutippen. Anschließend darf man die Eingaberoutine löschen.

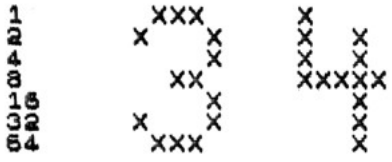
Bild 2 klärt, wie die Ziffern-Daten zustandekommen. Jeder der sieben Punktmatrix-Zeilen ist ein Gewichtungsfaktor zugeordnet. Bei spaltenweiser Betrachtung sind nun lediglich die Gewichtungsfaktoren gesetzter Bildpunkte aufzuaddieren. Der erste Wert für die Ziffer 3 lautet also 2 + 32 = 34, der zweite Wert 1 + 64 = 65 usw.

Nach dem Programmstart mit RUN ist eine achtstellige Zahl einzutippen, die groß am Bildschirm gezeigt wird. Danach verlangt das Programm nach einer neuen Zahl. Maßgebend für die Stellenzahl ist der maximale Wert der Laufvariablen in Zeile 40.

Wolfgang Meissner

```
  10 REM HIER 50 ZEICHEN EINTIPP
EN
  20 INPUT A$
  30 CLS
  40 FOR O=1 TO 8
  50 LET E$=A$(O TO O)
  55 IF CODE E$<28 OR CODE E$>37
 THEN PRINT "FALSCHE EINGABE"
  60 LET E=VAL E$
  70 LET B=E*5+16514
  80 FOR M=0 TO 4
  90 LET C=PEEK (B+M)
100 LET D=128
110 FOR N=1 TO 7
120 LET D=D/2
130 IF C-D<0 THEN GOTO 160
140 LET C=C-D
150 PLOT M+O*7,10+N
160 NEXT N
170 NEXT M
180 NEXT O
190 GOTO 20
200 REM ZIFFERNEINGABE
210 FOR P=0 TO 49
220 INPUT Q
230 POKE (P+16514),Q
240 NEXT P
250 GOTO 20
    DATEN DER ZIFFERN
0         62 65 65 65 62
1         00 00 04 02 127
2         98 81 81 81 78
3         34 65 73 73 54
4         15 08 08 126 08
5         39 69 69 69 57
6         62 73 73 73 48
7         03 01 113 25 07
8         54 73 73 73 54
9         06 73 73 73 62
```

① **Demonstrationsprogramm:** Die PLOT-Daten für die großen Ziffern holt sich das Programm aus Zeile 10



② **Datenschlüssel:** Mit Hilfe der Gewichtungsfaktoren (links) läßt sich jede Ziffer durch fünf Datenwerte beschreiben

Messen mit dem ZX 81 (4):

Widerstände bekennen Farbe

Handelsübliche Analog-Multimeter leisten bei der Widerstandsmessung wertvolle Dienste. Bei hohen Widerstandswerten ist die Meßgenauigkeit jedoch bescheiden: Dann sollten Sie den Computer einspringen lassen.

Der Wert eines ohmschen Widerstands wird im allgemeinen durch eine Strom- bzw. Spannungsmessung bestimmt. Alle analogen und digitalen Vielfachinstrumente messen Widerstände nach diesem Prinzip. Es gilt auch für die Widerstandsmessung mit dem Computer, obgleich es dabei nicht offensichtlich ist. Gemessen wird nämlich die Dauer eines Impulses, die wiederum vom Lade- bzw. Entladestrom eines Kondensators abhängt. Dies geschieht in Anlehnung an die Oszillatorschaltung mit dem Timer 555, die bereits in Teil 3 bei der Kapazitätsmessung gute Dienste leistete; *Bild 1* zeigt noch einmal die Prinzipschaltung.

Die Entladephase führt nicht zum Ziel

Der Kondensator C wird über die Widerstände R1 und R2 geladen sowie über den Widerstand R2 entladen. Wenn man in dieser Schaltung R2 als Meßwiderstand auffaßt, so ist die Entladezeit des Kondensators C proportional zu dessen Widerstandswert.

Diese Überlegung hat jedoch einen Haken: Für R2 = 0 erhält man nicht die Entlade-Zeitdauer t = 0, die man eigentlich erwarten würde, sondern eine Zeitdauer, die einem Widerstandswert von etwa 150 Ω entspricht. Hinter diesem Widerstandswert verbirgt sich der interne Transistor, über den die Entladung des Kondensators erfolgt (siehe Teil 3).

Da der Transistor in Serie zu R2 liegt, verfälscht sein Durchlaßwiderstand nicht nur die Messung bei R2 = 0, sondern bei allen Werten von R2. Das wäre noch nicht so schlimm, wenn der Durchlaßwiderstand des Entladetransistors für jeden Wert von R2 konstant wäre. Da dies jedoch nicht der Fall ist, scheidet die Entladephase des Kondensators für eine Präzisionsmessung des Widerstandes R2 aus.

Richten wir unseren Blick also auf die Ladephase des Kondensators. Die Zeitdauer dafür ist proportional zur Summe aus R1 und R2. Darin steckt auf den ersten Blick schon wieder eine Schwierigkeit, da auf diese Weise immer die Summe zweier Widerstände gemessen wird. Wenn man R2 als Meßwiderstand auffaßt, wird stets R1 mitgemessen und umgekehrt. Wir interessieren uns jedoch für die Messung eines Einzelwiderstandes.

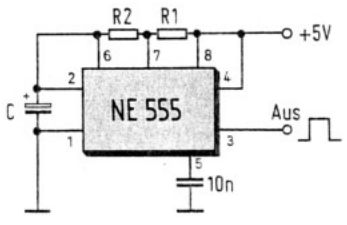
Ohne Software wäre dies tatsächlich ein ernstzunehmendes Problem: Man müßte mit Hilfe eines EXCLUSIV-ODER-Gatters eine digitale Subtraktion zweier Impulse durchführen. Da wir die Messung mit Hilfe des Computers vorhaben, wäre es freilich unklug, diese hardwaremäßige Lösung zu wählen.

Subtrahieren statt Löten

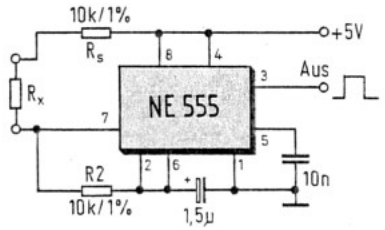
In Basic- oder in Maschinensprache genügt nämlich schon ein simpler Subtraktionsbefehl, um z. B. R1 seiner Wirkung zu berauben. Mit dieser Überlegung spielt es keine Rolle mehr, daß bei der Ladung des Kondensators zwei Widerstände beteiligt sind: Der zweite Widerstand wird einfach als bekannt vorausgesetzt und durch das Programm vom Meßwiderstand subtrahiert.

Es bleibt jetzt nur noch zu überlegen, welcher der beiden Widerstände am besten die Funktion des Meßwiderstands übernimmt. Um die Entladephase, die für den Meßvorgang eine unnütze „Totzeit" ist, kurz zu halten, wird die Schaltung gemäß *Bild 2* vorgeschlagen. Hier hat R2 den Wert 10 kΩ, und eine Serienschaltung aus dem Meßwiderstand R_x und R_s = 10 kΩ ersetzt R1.

Der 10-kΩ-Widerstand in Serie zu R_x bewirkt, daß für R_x = 0 der Entladetransistor nicht ungeschützt mit der Versorgungsspannung konfrontiert wird. Bei dieser Dimensionierung hat der Ladewiderstand den Wert R_x + 20 kΩ. Die Ladezeitdauer, die mit Hilfe des Computers gemessen werden soll, beträgt damit $t_L = 0{,}69 \times C \times (R_x + 20\ k\Omega)$. Der Wert 0,69 × C bestimmt den Umrechnungsfaktor zwischen Meßzeit und Widerstand.



① **Meßwandlerprinzip:** Die Widerstände R1 und R2 bestimmen maßgebend das Tastverhältnis des Rechteck-Signals an Pin 3



② **R-Meßwandler:** Die Widerstände R_s und R2 erzwingen einen Meßfehler, der sich per Software jedoch einfach korrigieren läßt

Eine „glatte" Umrechnung von 1 kΩ in 1 ms ergibt sich für C = 1,44 µF. Dies ist jedoch ein höchst unüblicher Kapazitätswert, der zudem wegen der Exemplarstreuungen des Timers 555 noch experimentell bestätigt werden müßte. Deshalb ist es besser, man verzichtet in diesem Fall auf eine glatte Umrechnung und korrigiert den Fehler, der beim Einsatz eines 1,5-µF-MKC-Folienkondensators entsteht, mit der Software. Bei anderen Kapazitätswerten ist der Korrekturfaktor lediglich so zu wählen, daß bei R_x = 0 auch der Wert 0 angezeigt wird.

Genauigkeit ist Trumpf

Die Taktfrequenz des ZX 81 bestimmt maßgebend die Meßgenauigkeit der Zeit- und Frequenzmessung. Unmittelbar betroffen sind davon bei der Zeitmessung die Zahl der Schleifendurchläufe (siehe Teil 1) und bei der Frequenzmessung der Faktor 89805 (siehe Teil 2).

Den Wert der 3,23-MHz-Taktfrequenz prägt beim ZX 81 ein 6,5-MHz-Keramikschwinger. Doch der unterliegt Exemplarstreuungen und einer Temperaturdrift, die erst nach Erreichen der Betriebstemperatur (nach etwa 60 min) verschwindet. Während der Erwärmungsphase kann sich die Taktfrequenz um 1000 ppm (Parts per Million) ändern, also um 6,5 kHz. Das entspricht zwar nur einem Meßfehler von 1‰, sollte aber bei Präzisionsmessungen nicht vernachlässigt werden. Am besten mißt man deshalb immer mit betriebswarmem Computer. Die im Zeit- und Frequenz-Meßprogramm gewählten Werte gehen ebenfalls davon aus.

Wer höchste Genauigkeit anstrebt, muß noch die Exemplarstreuungen des Keramikschwingers berücksichtigen. Am einfachsten geschieht dies, indem man ein Signal mit genau bekannter Impulsdauer bzw. Frequenz mißt. Weicht das Ergebnis vom tatsächlichen Wert ab, ist die Zahl der Schleifendurchläufe bei der Zeitmessung bzw. der Faktor bei der Frequenzmessung so zu ändern, daß der richtige Wert angezeigt wird. Für die meisten Meßaufgaben dürfte dieser Feinabgleich jedoch nicht nötig sein.

```
  10 REM HIER MUSS DAS ZEIT-MESS
PROGRAMM AUS TEIL 1 STEHEN (MASC
HINENCODE)
  18 FAST
  19 LET Q=USR 16514
  20 LET Q=INT (0.966*Q+0.5)-20
  21 PRINT "WIDERSTANDSWERT = ";
Q;" KILO-OHM"
  22 SLOW
  23 IF INKEY$<>"C" THEN GOTO 23
  24 GOTO 18
```

③ **Meßprogramm:** Hier ist die Dimensionierung gemäß Bild 2 berücksichtigt, so daß die Auflösung 1 kΩ beträgt

Korrektur per Software

Die komplette Software besteht aus dem Maschinenprogramm für die Messung von Zeiten (siehe Teil 1) sowie einem Basic-Programm. Dieses Basic-Programm führt die Korrekturen durch und bewirkt die Anzeige der Meßwerte auf dem Bildschirm (*Bild 3*). Zeile 20 übernimmt die Korrektur für den Umrechnungsfaktor und die „Nullpunktverschiebung".

Selbstverständlich läßt sich die Schaltung auch anders dimensionieren, um die Auflösung bzw. den Meßbereich zu ändern. Der Wert von R_s darf dabei bis auf 1 kΩ verringert werden und R2 sollte ebenfalls mindestens einen Wert von 1 kΩ haben. Für den Kondensator C sind Werte von 10 nF aufwärts zulässig.

Hubert Joas

ZX-81-Software:

Miniorgel

Machen Sie Ihren ZX 81 ohne Zusatz-Hardware zu einer Miniorgel. Ohne Zusatz-Hardware soll heißen, daß keinerlei Interfaceschaltungen zur Tonerzeugung nötig sind, wohl aber ein empfindlicher NF-Verstärker, der das Tonsignal kräftig verstärkt. Quelle der Orgeltöne ist dabei der MIC-Ausgang des ZX 81.

Mit List und Tücke wurde das Programm (*Bild*) so zurechtgestutzt, daß es gerade noch in das RAM der ZX-81-Grundversion hineinpaßt. Würden z. B. die Programmzeilen 3 und 6 der Variablen A die Werte direkt zuweisen (ohne die speicherplatzsparende VAL-Funktion), wäre die RAM-Kapazität bereits überschritten. Im Grunde verbirgt sich hinter dem Listing jedoch ein übliches Eingabeprogramm für die Hexcodes in den Zeilen 1 und 2; Kenner des ZX 81 werden das rasch herausfinden.

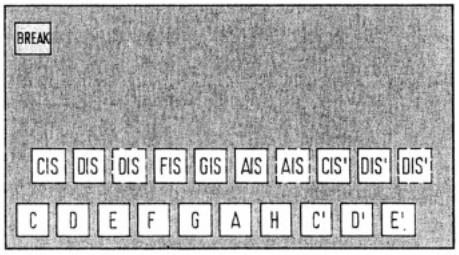
Das Programm wird mit RUN gestartet und muß sich dann nach einigen Sekunden mit der Meldung 9/8 zurückmelden. Der Maschinencode befindet sich jetzt ab Adresse 16518 im RAM und die Orgelroutine darf mit RAND USR 16518 aufgerufen werden. Wenn alles in Ordnung ist, muß nun ein Tastendruck in der untersten Tastenreihe einen Stammton erzeugen, ein Tastendruck in der darüberliegenden Tastenreihe einen Halbton (*Bild 2*). Ein Drücken der Taste 1 bricht das Programm ab. Auch am Bildschirm machen sich die Aktivitäten des Programms bemerkbar.

Für Leser unserer Serie „Klartext für den ZX 81" noch einige Tips zum Programmaufbau: Zwischen den Adressen 16518 und 16593 wird die Tastatur abgefragt und der gedrückten Taste ein Wert aus einer Code-Tabelle zugewiesen. Abhängig von diesem Wert erzeugt das Programm von Adresse 16594 bis 16609 den gewünschten Ton. Die Code-Tabelle ist zwischen Adresse 16610 und 16629 abgelegt. Ein Ändern dieser Codes hat Einfluß auf die Tonhöhe.

Juraj Matus/-ll

```
   1 LET A$="CD230F3EF7D8FEFE7EC
A280F3EFECDC04021E140C4C6403E7FC
DC04021E640C4C6403EFDCDC04021EB4
0C4C6403EBFCDC04021F040C4C640"
   2 LET B$="18C957D8FEFE7FC9CB3
F2338FB7E5F7ACDD240C9D8FEFE7FC84
310FED3FF4310FE7A18F0ECD28AB09C5
C68757C8DDEC6C6A69462826E8383"
   3 LET A=VAL "16518"
   4 GOSUB 9
   5 LET A$=B$
   6 LET A=VAL "16574"
   7 GOSUB 9
   8 STOP
   9 FOR N=0 TO 110 STEP 2
  10 POKE A+INT (N/2),16*CODE A$
(N+1)+CODE A$(N+2)-476
  11 NEXT N
  12 RETURN
```

Orgelprogramm: Mit Ach und Krach paßt es in die 1-KByte-Version des ZX 81



Tastenbelegung: Unten die Stammtöne, oben die Halbtöne. Taste 1 ermöglicht den Programmabbruch

ZX-81/ZX-Spectrum-Hardwaretip:

Joystick für Zwei

Genauer gesagt geht es in diesem Beitrag um ein Interface, an das sich zwei Joysticks anschließen lassen. Damit können dann Spielfiguren am Bildschirm, je nach Joystick, in vier oder acht Himmelsrichtungen gelenkt werden.

```
   1 REM 00000000000000000000000000
000
  10 FOR F=16522 TO 16539
  20 INPUT A
  30 POKE F,A
  40 NEXT F
```

② **ZX-81-Eingabeprogramm:** Die im Text genannten 26 Codes des Maschinenprogramms werden mit diesem Programm in der REM-Zeile untergebracht

```
  10 CLEAR 32573
  20 DATA 33,62,127,219,63,6,8,3
0,0,203,39,203,19,115,35,16,246,
201
  30 FOR f=32582 TO 32599: READ
a: POKE f,a: NEXT f
```

③ **Spectrum-Eingabeprogramm:** Hier sind die Maschinencodes bereits in der DATA-Zeile enthalten. Das Programm speichert sie dann ab Adresse 32582

Hinter einem Joystick-Interface verbirgt sich nichts anderes als ein Eingabe-Port, der, wenn er adressiert wird, seine Eingangsdaten auf den Datenbus legt. Adressieren heißt, wie immer, daß ein Baustein nur dann freigegeben wird, wenn eine ganz bestimmte Adresse auf dem Adreßbus liegt.

Die Hardware ist schnell durchschaut

In unserem Fall wird ein „bidirektionaler Bustreiber" 74LS245 freigegeben (*Bild 1*). L-Pegel an Pin 1 dieses ICs legt die Datenflußrichtung fest: hier von den Joysticks zum Datenbus.

Freigegeben wird der Bustreiber mit L-Pegel an seinem Enable-Eingang (Pin 19). Sobald das der Fall ist, reicht das IC, bildlich gesprochen, die Eingangsdaten zum Datenbus weiter. Sind beide Joysticks in Ruhestellung, dann führen alle acht Leitungen L-Pegel, d. h. auf dem Datenbus liegt das Datenwort 0. Dafür sorgen die 4,7-kΩ-Widerstände.

Die Joysticks sind sogenannte digitale Modelle, die im Versandhandel preisgünstig angeboten werden. Im Fuß dieser Joysticks sind je vier Tast-Schalter (Schließer) untergebracht. Kippt man nun den Steuerknüppel in eine der vier Haupt-Himmelsrichtungen, dann wird jeweils einer dieser Schalter geschlossen.

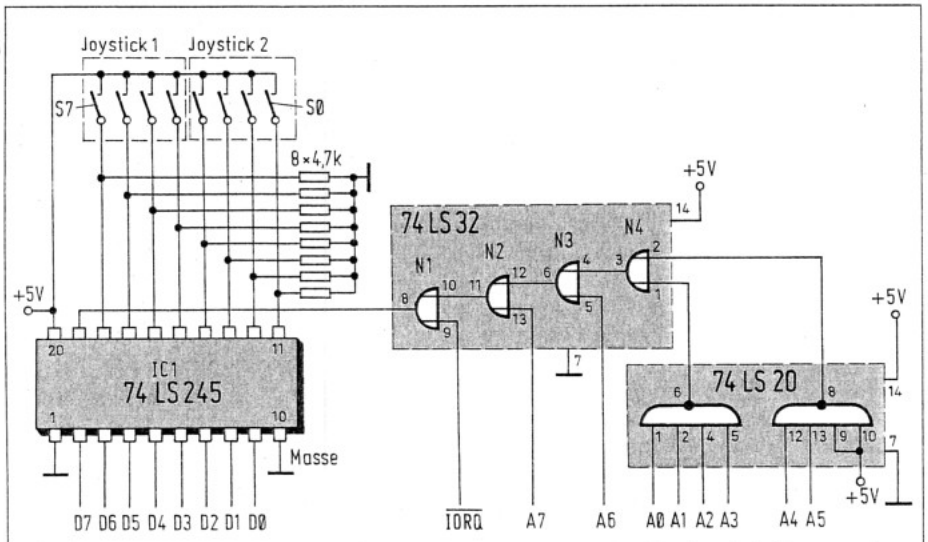
Einige Joysticks sind so konstruiert, daß zusätzlich noch vier Zwischenwerte möglich sind. Dabei werden dann jeweils zwei Schalter geschlossen. Auch diese Modelle sind geeignet, da das Interface die Zwischenwerte an den Computer weitermeldet. Das ist ein großer Vorteil gegenüber der „Tastatureingabe", die normalerweise keine Zwischenwerte (gleichzeitiges Betätigen zweier Tasten) registriert, wenn nicht die Systemvariable LAST-K abgefragt wird.

Gemäß dem Schaltbild sind die Joysticks so an das Interface angeschlossen, daß ein geschlossener Schalter H-Pegel auf die betroffene Eingangsleitung legt. Mit einem L-Impuls an Pin 19 (IC 1) wird die momentan gültige Pegelverteilung auf den acht Eingangsleitungen auch auf den Datenbus übernommen.

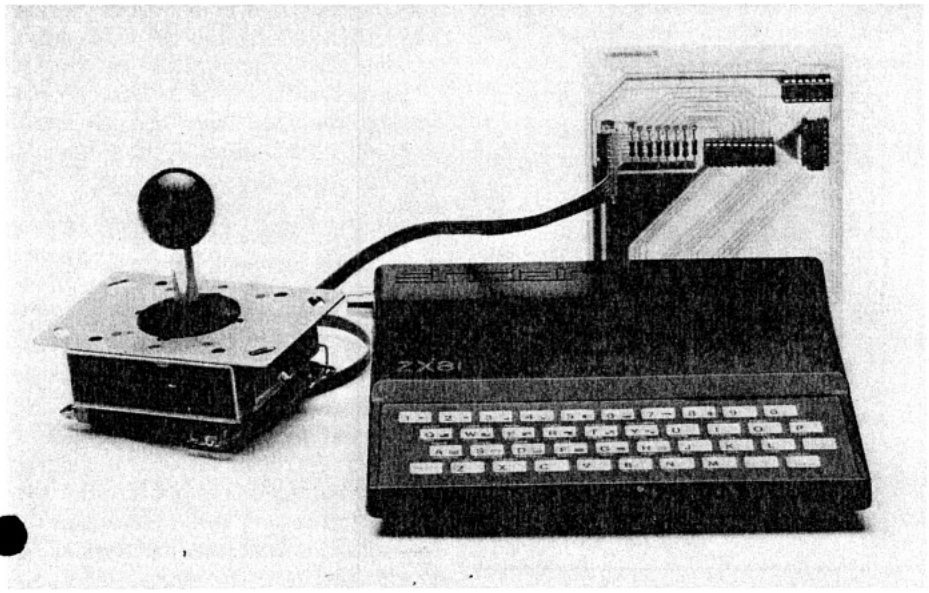
Wie kommt nun dieser L-Impuls zustande? Da das Interface mit dem Eingabe-Befehl *in a, (x)* der Z-80-CPU abgefragt wird, kann das $\overline{\text{IORQ}}$-Signal, das einen solchen Befehl immer begleitet, die Freigabe von IC 1 übernehmen. Ein Adreßdecoder muß lediglich dafür sorgen, daß der $\overline{\text{IORQ}}$-Impuls IC 1 nur dann erreicht, wenn die Adresse des Interfaces auf dem Adreßbus liegt.

In der gezeigten Beschaltung des Adreßdecoders hat die Interface-Adresse den Wert 63. Bis auf A6 und A7 führen dabei alle niederwertigeren Adreßleitungen H-Pegel, so daß beim ZX 81 keine Störungen zu befürchten sind (siehe Heft 14/1984, Seite 61).

Da ein ODER-Gatter nur dann L-Pegel am Ausgang führt, wenn beide Eingänge gleichzeitig auf L-Pegel liegen, ist auch



① **Joystick-Interface:** Bei einem L-Impuls an Pin 19 (IC 1) wird die von den Joysticks bestimmte Pegelverteilung auf den acht Eingangsleitungen zum Datenbus weitergegeben. Der L-Impuls ($\overline{\text{IORQ}}$) wird vom Adreßdecoder (rechter Schaltungsteil) bei Adresse 63 über N1 durchgelassen



Joysticks bringen Bewegung auf den Bildschirm. Je nach Joystick lassen sich Figuren in vier oder in acht Richtungen über den Schirm jagen Foto: Neu

mit PEEK abgefragt und ausgewertet werden.

Beim ZX 81 wird das Maschinenprogramm in einer 26 Zeichen langen REM-Zeile am Programmanfang untergebracht. Da die ersten acht Bytes für die Schalter reserviert sind, müssen die folgenden Dezimalcodes ab Adresse 16522 mit dem Eingabeprogramm (*Bild 2*) geladen werden:

33, 130, 64, 219, 63, 6, 8, 30, 0, 203, 39, 203, 19, 115, 35, 16, 246, 201

Bild 3 zeigt die Eingabe für den 16-KByte-Spectrum.

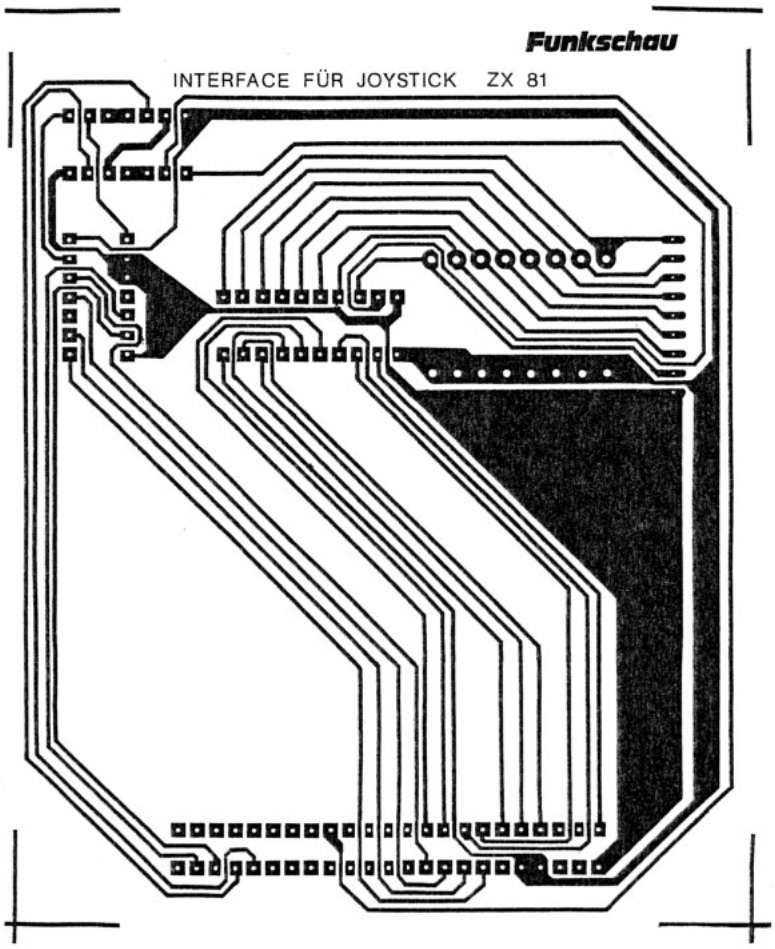
Das Demonstrationsprogramm in *Bild 4* ruft das Maschinenprogramm auf und meldet fortwährend den aktuellen Inhalt der acht „Schalter-Speicherplätze". Da die Auswertung in Basic erfolgt ist die Reaktionsgeschwindigkeit nicht besonders hoch. Wird nur ein Joystick abgefragt (FOR F = 0 TO 3) steigt das Tempo merklich.

die Bedeutung des Gatters N1 klar: Es läßt den $\overline{\text{IORQ}}$-Impuls erst passieren, wenn die richtige Adresse anliegt, der Adreßdecoder also ebenfalls einen L-Impuls liefert.

Soweit die Hardware, die in dieser Form prinzipiell für jeden Computer mit einer Z-80-CPU geeignet ist. Abweichungen kann es schlimmstenfalls bei der Interface-Adresse geben, sofern das Betriebssystem des Computers bereits die Adresse 63 selber verwendet. Selbstverständlich ist es auch zulässig, nur einen Joystick anzuschließen und die übrigen Eingänge mit Tast-Schaltern, z. B. einem „Feuerknopf" zu belegen. Und wer bereits eine 8-Bit-Parallelschnittstelle hat, der braucht lediglich die Dateneingänge über 4,7-kΩ-Widerstände auf Massepotential zu ziehen.

Ohne Software geht es nicht

Da beim ZX 81 kein Basicbefehl zur Abfrage des Interfaces bereitsteht, muß ein Maschinenprogramm diese Aufgabe übernehmen. Nach dem Start des Programms ordnet es jedem Tast-Schalter der Joysticks einen Speicherplatz (Adresse) im RAM zu. In den jeweiligen Speicherplatz wird anschließend der Wert 1 geladen, wenn der zugeordnete Schalter bei der Abfrage geschlossen war oder der Wert 0, wenn der Schalter offen war. Der Inhalt der Speicherplätze kann dann z. B. von einem Basic-Programm



Platinenlayout: In der gewählten Lage kommen die ICs keinem RAM-Modul „mechanisch" in die Quere

```
10 RAND USR 16522
20 PRINT AT 0,0;
30 FOR F=0 TO 7
40 PRINT PEEK (16514+F)
50 NEXT F
60 GO TO 10
```

```
10 RANDOMIZE USR 32582: PRINT
AT 0,0;
20 FOR f=0 TO 7: PRINT PEEK (3
2574+f): NEXT f: GO TO 10
```

④ **Demonstrationsprogramm:** Das Programm meldet für jeden Joystick-Schalter, ob er geschlossen (1) oder offen ist (0). Oben die ZX-81-, unten die Spectrum-Version

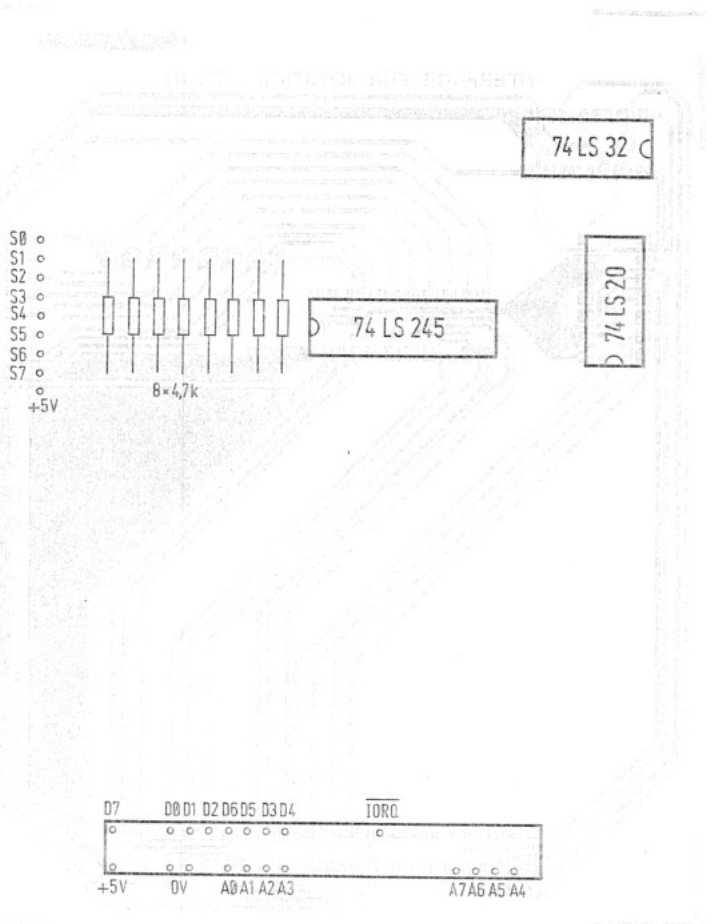
```
Adresse   Code   Assembler
16522     33
16523     130
16524     64     ld hl,16514
16525     219
16526     63     in a,(63)
16527     6
16528     8      ld b,8
16529     30
16530     0      ld e,0
16531     203
16532     39     sla a
16533     203
16534     19     rl e
16535     115    ld (hl),e
16536     35     inc hl
16537     16
16538     246    djnz DIS
16539     201    ret
```

⑤ **Abfrageprogramm:** Den Joystick-Schaltern zugeordnete Speicherplätze werden mit 1 oder 0 geladen, je nachdem, ob der betreffende Schalter geschlossen oder offen war

Vom Maschinenprogramm (*Bild 5*) wird zuerst die Adresse des ersten der acht Schalter-Speicherplätze mit *ld hl, 16514* bzw. beim Spectrum mit *ld hl, 32574* ins hl-Registerpaar gebracht. Dieser Speicherplatz soll mit dem höchstwertigen Bit (Schalter S7) des vom Interface gelieferten Datenwortes geladen werden. Dazu wird das Datenwort mit *in a, (63)* erst einmal in den Akku gebracht und anschließend mit *ld b, 8* das b-Register als Schleifenzähler gestartet.

Der nächste Befehl *ld e, 0* löscht das e-Register. Dann wird mit *sla a* das höchstwertige Bit des Akkus in das Carry-Flag geschoben und mit *rl e* gleich weiter ins e-Register verfrachtet. *ld (hl), e* lädt daraufhin den Inhalt des e-Registers in den ersten Schalter-Speicherplatz mit der Adresse 16514.

Nun wird mit *inc hl* der nächste Schalter-Speicherplatz angepeilt, der das nächstniedrigere Bit (Schalter S6) aufnehmen soll. Die Sprunganweisung *djnz DIS* verzweigt daher zum Befehl *ld e,0*, solange der Schleifenzähler (b-Register) nicht auf 0 steht. Erst wenn das der Fall ist, bewirkt der letzte Befehl *ret* den Rücksprung ins Basic-Programm, wo jetzt eine Auswertung stattfinden kann. Martin Müller/-ll



Bestückungsplan: Für beide Computer läßt sich eine 46polige Buchsenleiste direkt einlöten, da die benutzten Pins übereinstimmen

ZX-81-Softwaretip:

Wiedergewinnen von Bytes

Wenn der Speicherplatz knapp wird, ist jedes Mittel recht, Bytes wiederzugewinnen. So einfach, daß man es gerne übersieht, ist dabei folgendes Verfahren: Merkt man, daß der Speicherplatz knapp wird, ist das (möglicherweise noch unvollständige) Programm erst einmal zu starten.

Dabei werden allen erreichten LET-Anweisungen die definierten Werte zugewiesen und in den Variablenspeicher übernommen. Daraufhin sind die LET-Anweisungen aber überflüssig. Man darf sie löschen, was Platz im Programmspeicher schafft und das Weiterarbeiten ermöglicht. Grundsätzlich darf das Programm aber jetzt nur noch mit GOTO gestartet werden, weil sonst die Variablenzuweisung verlorengeht. Ein Beispiel mag das veranschaulichen:

```
10 LET A$="FUNKSCHAU"
20 PRINT A$
```

Wird das Programm gestartet, danach Zeile 10 gelöscht und das Programm nochmals, aber mit GOTO 20 aufgerufen, so wird wieder der Text ausgegeben. Der Gewinn an freiem Speicherplatz beträgt schon bei diesem kurzen Beispiel immerhin 20 Byte. Paul Webranitz

ZX-81-Hardwaretip:

Lichtgriffel

Die Entwickler des ZX 81 dachten nicht im Traum daran, dem Computer-Winzling einen Lichtgriffel zu spendieren. Wir holen hier das Versäumte – so gut es geht – nach.

Die Z-80-CPU im ZX 81 muß ganz schön schuften, denn außer den üblichen Aufgaben einer CPU muß sie noch den Bildaufbau wahrnehmen. Einen Videoprozessor für den Bildaufbau – fast alle anderen Heimcomputer haben so ein Spezial-IC – kennt der ZX 81 nicht. Das macht es problematisch, an dem kleinen Sinclair-Computer einen Lichtgriffel zu betreiben.

Argusauge für den ZX 81

Für einen Computer ist ein Lichtgriffel gewissermassen ein Auge, das unentwegt auf den Bildschrim starrt. Dabei ist aber stets nur der kleine Bildpunkt im Blickfeld, auf den der Anwender den Lichtgriffel gerade richtet. Eine trickreiche Elektronik gepaart mit Software sorgt dann dafür, daß der Computer die Position (Bildkoordinaten), auf die der Lichtgriffel deutet, erkennt. Damit kann der Computer z. B. an dieser Position einen Bildpunkt setzen oder sogar eine Kurve ziehen, die brav der Bewegung des Lichtgriffel folgt.

Andererseits kann ein Computer an genau definierten Stellen des Bildschirms auch Befehlsworte plazieren. Tippt man so ein Wort mit den Lichtgriffel an, „weiß" der Computer genau, welches Wort gemeint ist und er kann den Befehl ausführen. Bei den Heimcomputern von Thomson sind auf diese Weise Befehlseingaben ohne einen einzigen Tastendruck möglich!

Der ZX-81-Lichtgriffel ist freilich eine Notlösung, denn sein Leistungsvermögen orientiert sich an dem, was beim ZX 81 machbar ist. So muß man sich damit abfinden, daß der Lichtgriffel nur in vertikaler Richtung verschiedene Positionen sicher ausmachen kann. Wird er waagrecht über den Bildschirm geführt, versagt die Positionserkennung. Um die Ursache dieser schmerzlichen Einschränkung zu verstehen, ist ein kurzer Ausflug in die Fernsehtechnik nötig.

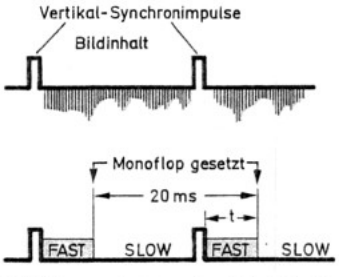
Eine Zeitmessung verrät die Position

In jeder Sekunde baut ein Fernsehgerät 50 sogenannte Halbbilder am Bildschirm auf. Den Beginn eines Halbbildes signalisiert ein Vertikal-Synchronimpuls, der vom ZX 81 stets zum richtigen Zeitpunkt ins Bildsignal eingefügt wird (Bild 1 oben).

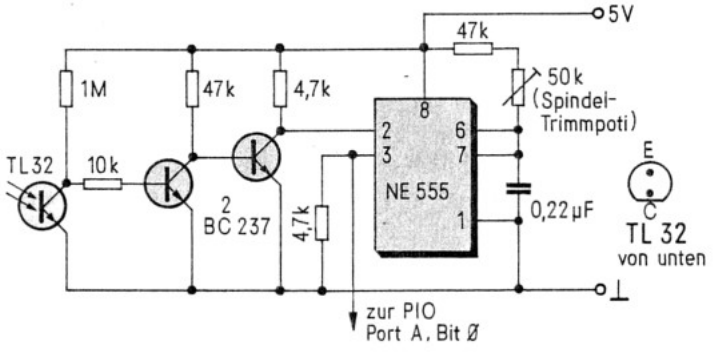
Nach einem Vertikal-Synchronimpuls beginnt also der Elektronenstrahl – genauer der von ihm hervorgerufene Leuchtfleck – von oben nach unten damit, zeilenweise über den Bildschirm zu huschen, bis er das rechte untere Bildeck erreicht hat. Dann leitet ein weiterer Vertikal-Synchronimpuls den nächsten Bildaufbau ein.

Ein auf den Bildschirm gerichteter Lichtgriffel registriert nun, wann der Leuchtfleck am lichtempfindlichen Sensor des Lichtgriffels vorbeikommt. Jetzt muß man nur noch die Zeitdauer bis zum Auslösen des nächsten Vertikal-Synchronimpulses messen. Dieser Wert ist dann ein Maß für die Position des Lichtgriffels am Bildschirm. So weit so gut, nur ist beim ZX 81 die Messung der Zeitdauer mit Komplikationen verbunden.

Immerhin ist der Computer so „freundlich", der Systemvariablen FRAMES (Adressen 16436 und 16437) die Aussendung eines Vertikal-Synchronimpulses zu melden, indem der Wert der Systemvariablen jeweils um 1 verringert wird. Durch Kontrollieren des Wertes läßt sich so der Referenzzeitpunkt für die Messung einfach finden.



① **Fernsehsignale:** Die Vertikal-Synchronimpulse im Videosignal dienen als Bezugspunkt für eine Zeitmessung im FAST-Modus



② **Lichtgriffel-Elektronik:** Die Leitungen zum Kollektor des Fototransistors sollten möglichst kurz sein

```
  10 REM ANSTELLE DIESES TEXTES
HIER 55 BELIEBIGE ZEICHEN Z.B. A
EINTIPPEN
 100 LET A=16514
 110 PRINT AT 21,0;A,
 120 INPUT X
 130 POKE A,X
 140 PRINT X
 150 LET A=A+1
 155 SCROLL
 157 SCROLL
 160 IF A<16565 THEN GOTO 110
 170 STOP
 200 PRINT AT 0,0;"■"
 210 PRINT AT 21,0;USR 16514;" "
 215 REM SCHLEIFE VERRINGERT DAS
BILDFLACKERN BEI FAST-UMSCHALT.
 220 FOR T=0 TO 10
 230 NEXT T
 240 GOTO 200

16514:   62  207 211 247 62   1
16520:   211 247 219 231 254  1
16526:   194 138 64  205 168 64
16532:   205 35  15  219 231 254
16538:   1   194 162 64  3   195
16544:   151 64  197 205 43  15
16550:   193 201 1   0   0   33
16556:   52  64  126 86  166 202
16562:   175 64  201
```

③ **Testprogramm:** Bis Zeile 170 das Eingabeprogramm für die unten gezeigten Maschinencodes; anschließend die eigentliche Testroutine

Für die Zeitmessung muß der ZX 81 allerdings in den FAST-Modus gebracht werden, weil nur dann sichergestellt ist, daß die Messung nicht durch den Bildaufbau unterbrochen wird. Da die CPU des ZX 81 auch den Bildaufbau wahrnehmen muß, würde sie alle 20 ms, möglicherweise gerade während der Zeitmessung, das dazu erforderliche Programm rigoros unterbrechen. Meßfehler wären demnach unabwendbar, wenn der ZX 81 die Messungen im SLOW-Modus vornimmt. Doch im FAST-Modus sendet der Computer keine Vertikal-Synchronimpulse mehr aus und kümmert sich auch nicht um die Systemvariable FRAMES!

Jetzt hilft uns ein Trick weiter: Registriert der Lichtgriffel den vorbeihuschenden Leuchtfleck, setzt er ein Monoflop, das zuvor auf eine Kippzeit von rd. 20 ms eingestellt wurde (Bild 1, unten). Der nächste Vertikal-Synchronimpuls versetzt den ZX 81 in den FAST-Modus (per Programm) und zugleich beginnt die Zeitmessung. Sie dauert so lange, bis das Monoflop in seine Ausgangslage „zurückkippt"; anschließend wird wieder der SLOW-Modus aktiviert.

```
4082 LD A,207          ; PIO intialisieren
4084 OUT 247,A         ;
4086 LD A,001          ;Port A = Eingang
4088 OUT 247,A         ;
408A IN A,231          ;Port-Abfrage
408C CP 001            ;Monoflop gesetzt?
408E JP NZ,16522       ;falls MF nicht gesetzt
4091 CALL 16552        ;
4094 CALL 03875        ;FAST-Modus
4097 IN A,231          ;Port-Abfrage
4099 CP 001            ;Monoflop noch gesetzt?
409B JP NZ,16546       ;falls MF nicht mehr ges.
409E INC BC            ;bc = bc + 1
409F JP 16535          ;neue Port-Abfrage
40A2 PUSH BC           ;bc retten
40A3 CALL 03883        ;SLOW-Modus
40A6 POP BC            ;bc holen
40A7 RET               ;Rücksprung Basic
40A8 LD BC,00000       ;bc zurücksetzen
40AB LD HL,16436       ;FRAMES abfragen
40AE LD A,(HL)         ;aktueller Wert in a
40AF LD D,(HL)         ;
40B0 CP D              ;FRAMES verändert?
40B1 JP Z,16559        ;
40B4 RET               ;Rücksprung
```

④ **Assembler-Listing:** Die Adressierung der PIO orientiert sich an der „entfesselten PIO" aus Heft 14/1984 (Ausgang Y 7 des Adreßdecoders)

Software statt Sehnerv

Die Zeitmessung basiert darauf, daß der Computer über den PIO-Port (siehe Heft 3/1984) feststellt, ob das Monoflop noch gesetzt ist. So lange das der Fall ist, inkrementiert er das bc-Registerpaar (Zähler). Leider dauert bereits einer dieser Schleifendurchläufe zu lange (etwa 20 µs), um horizontale Verschiebungen des Lichtgriffels in definierte Zählerstände umzumünzen. Eindeutig voneinander abweichende Zählerstände gibt es erst bei vertikaler Verschiebung des Lichtgriffels.

Die Schaltung des Lichtgriffels (Bild 2) ist nach eigenem Ermessen in ein Gehäuse zu quetschen, wobei die Öffnung für den Sensor z. B. 5 mm Durchmesser haben darf.Wichtig ist, daß auf den Sensor möglichst kein Fremdlicht fällt. Mit dem Spindel-Potentiometer wird die Kippzeit des Monoflops (NE 555) eingestellt.

Angeschlossen wird der Lichtgriffel an den PIO-Port, der auch die Stromversorgung übernehmen kann. Jetzt darf man das Testprogramm (Bild 3) eintippen und mit RUN starten. Nun sind die in Bild 3 unten gezeigten Maschinencodes einzugeben. Anschließend ist das Programm mit RUN 200 erneut zu starten. Das linke obere Bildeck zeigt jetzt ein schwarzes Quadrat.

Will es nicht auftauchen, ist die Umgebungshelligkeit zu groß. Mit dem Lichtgriffel tippt man nun neben dieses Quadrat und verstellt das Potentiometer so, daß unten am Bildschirm der Wert 1 eingeblendet wird. Dies ist der Zählerstand des bc-Registerpaares, der vom Maschinenprogramm über den USR-Aufruf automatisch ins Basic übernommen wird. Abhängig von diesem Zählerstand, der bei vertikaler Bewegung des Lichtgriffels heftig in Bewegung gerät, kann man dem ZX 81 jetzt irgendwelche Tätigkeiten (z. B. PLOT-Anweisungen) zuteilen. Heinrich Töws/-ll

Preisbrecher:

Akustikkoppler – auch für Btx

Vor gar nicht langer Zeit blühte noch ein „grauer Markt" für Akustikkoppler (Geräte ohne FTZ-Nummer), denn Modelle die den Segen der Post hatten, waren sündhaft teuer. Heute gibt es FTZ-geprüfte Akustikkoppler schon für weniger als 300 DM. Diese Modelle ermöglichen die Datenfernübertragung übers Telefonnetz mit einer Datenrate von 300 Baud.

Das kann auch der knapp 400 DM teure Akustikkoppler AK 2000 S, der von der Firma GVM, Düsseldorf, angeboten wird. Doch dieser bietet noch mehr: Er kann mit 1200/75 Baud sowohl senden als auch empfangen (Semi-Duplex mit Hilfskanal). Und damit läßt sich außer dem sonst üblichen Abfragen von Telefon-Mailboxen jetzt auch der Zugriff auf Datex-P bewerkstelligen.

Die Zulassungsnummer für den AK 2000 S erlaubt es auch, ihn beim Btx-Betrieb anstelle des gemieteten Post-Modems zu verwenden. Dabei kann der Akustikkoppler mit allen Btx-Decodern zusammenarbeiten, die eine serielle RS-232-C-Schnittstelle haben. -ll

Akustikkoppler auch für Btx: Mit einem Preis von knapp 400 DM ist das Gerät erstaunlich preisgünstig

ZX 81 à la carte

Immer richtig temperiert

Komfortable Heizungsregelung spart Kosten

Nicht jedes Zimmer einer Wohnung muß den ganzen Tag über beheizt werden. Wie schön wäre es, wenn das Bad in der Früh und das Wohnzimmer zum Feierabend wohltemperiert wären. Die Bauanleitung, die für solchen Komfort sorgt, gewann den 2. FUNKSCHAU-Preis in der Kategorie „Nachbausichere Schaltung".

Eine der wirklich sinnvollen Anwendungen von Heimcomputern besteht in der selbsttätigen Ausführung von Steuer- und Regelaufgaben, wie z. B. bei der Regelung einer Heizung. Bislang existierende Bauvorschläge beziehen sich jedoch ausschließlich auf eine Optimierung der Brennersteuerung bzw. Regelung der Vorlauftemperatur. Für die Vielzahl von Mietwohnungen, mit den üblichen Zentralheizungen, sind diese Regelungen allerdings uninteressant. Was bisher fehlte, war eine weitere, autonome Regelung der einzelnen Heizkörper nach einem selbst bestimmbaren, individuellen Programm.

Diese Bauanleitung zeigt eine Lösungsmöglichkeit durch konsequente Ausnutzung des ZX 81, der für diese Aufgabe ideal geeignet ist. Die im folgenden beschriebene Regelung eignet sich für alle Heizkörper mit Thermostatventilen. In die Thermostatkörper werden nachträglich „Heizwiderstände" eingebaut, die durch Erwärmung dem Thermostaten eine höhere Temperatur vortäuschen. Auf diese Weise wird jeder Heizkörper auch elektrisch steuerbar, ohne daß man installationsmäßig in die Heizungsanlage eingreifen muß.

Wärme nach Zeitplan

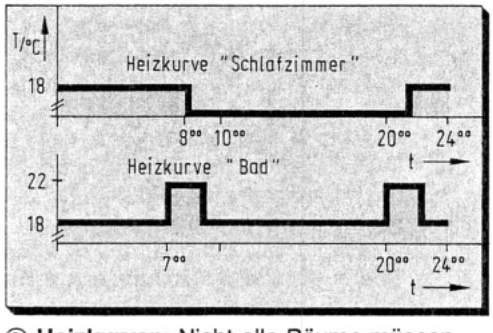
Das Programm liefert eine Temperaturregelung kombiniert mit einer Zeitsteuerung (getrennt für vier Räume), eine Echtzeituhr mit Datum – durch Akkuversorgung bei Netzausfall gesichert – sowie die Möglichkeit einer automatischen Wochenend-Umschaltung. Ein komplettes Listing dazu, bekommen Sie beim Franzis-Software-Service.

Die notwendige Hardware ist minimal: Sie wird auf zwei kleinen Platinen aufgebaut, die zusammen aus einer Europakarte hergestellt werden können.

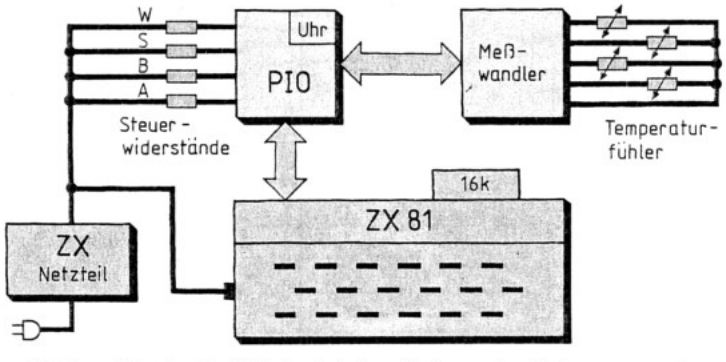
Die Grundidee besteht darin, die Heizzeiten jedes Heizkörpers der Nutzungshäufigkeit des Raumes anzupassen. Da alle Räume unterschiedlich genutzt werden, haben sie auch unterschiedlichen Temperaturbedarf. Dies bedeutet, daß man jedem Zimmer eine „Heizkurve" (*Bild 1*) zuordnen kann. Das Schlafzimmer braucht tagsüber nicht geheizt zu sein, sollte nachts aber auf 18 °C gehalten werden. Das Bad dagegen soll morgens und abends schön warm sein.

Das Heizungsprogramm ermöglicht die Steuerung von vier Räumen (z. B. Schlafzimmer, Wohnzimmer, Bad, Arbeitszimmer) mit bis zu drei Temperaturveränderungen je Tag. Weiterhin ist eine Grundtemperatur einzugeben, die bei nicht programmierten Zeiten als Standardwert gilt. Ein internes Rechenprogramm ermittelt außerdem den momentanen Wochentag und schaltet am Wochenende automatisch auf eine höhere Grundtemperatur um. Weiterhin lassen sich mit einem Befehl sämtliche Heizkörper ein- oder ausschalten. Auf dem Bildschirm werden stets die aktuellen Werte geschrieben: Uhrzeit, Datum, Wochentag, Solltemperaturen, Isttemperaturen, Status der Heizkörper.

Die Basis bildet ein ZX 81 mit 16 K RAM (*Bild 2*). Über den Adreß-, Daten- und Kontrollbus ist eine Ein-/Ausgabeschnittstelle angeschlossen. Diese Platine liefert die Steuerspannungen für die Thermostate, kommuniziert mit dem Uhrenbaustein (serieller I^2C-Bus) und liest die von den Temperatursensoren gelieferten Werte über die Meßwandlerplatine ein.



① **Heizkurven:** Nicht alle Räume müssen den ganzen Tag „voll" beheizt werden



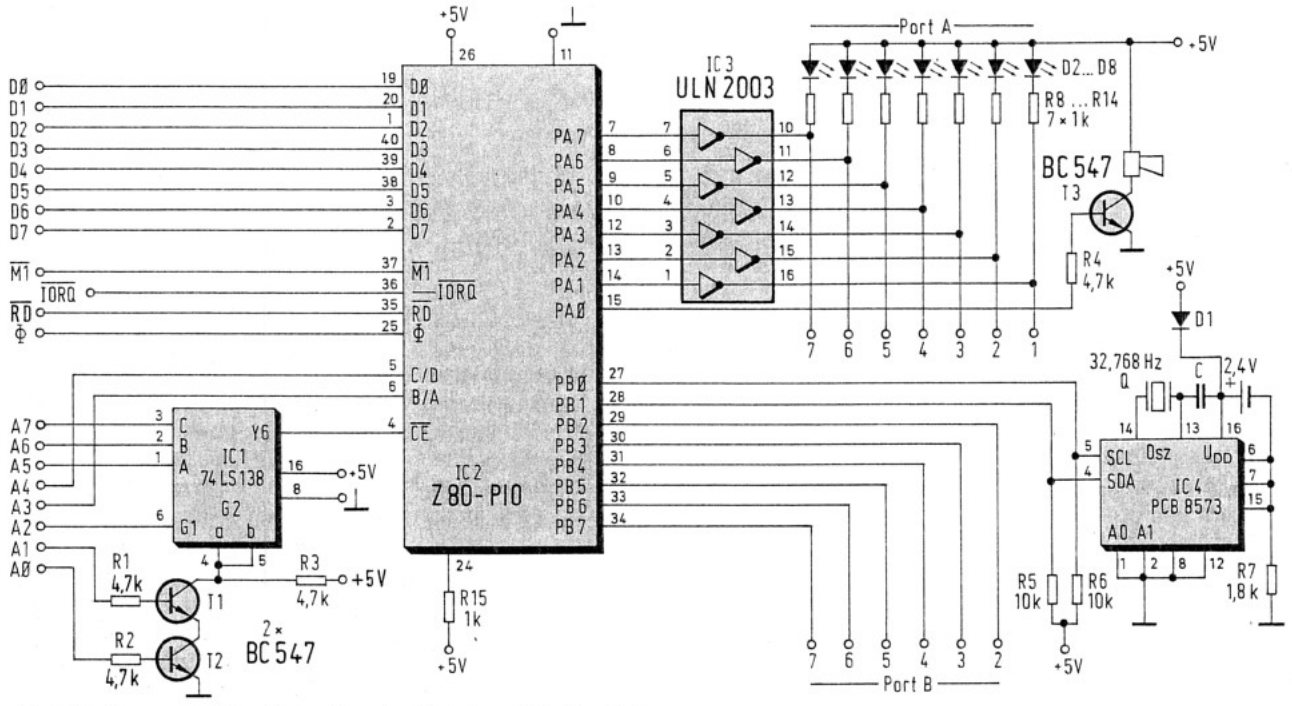
② **Das Blockschaltbild** zeigt den Aufbau der Heizungsregelung

Alle Baugruppen arbeiten mit +5 V, die direkt aus dem ZX 81 entnommen werden. Die Steuerwiderstände für die Thermostate werden aus der unstabilisierten Gleichspannung des ZX-Netzteils versorgt, so daß keine zweite Stromversorgung notwendig ist. Schließt man allerdings noch weitere Zusatzgeräte an, so ist ein größeres Netzteil unumgänglich, da es hier bereits an der Grenze der Belastbarkeit betrieben wird.

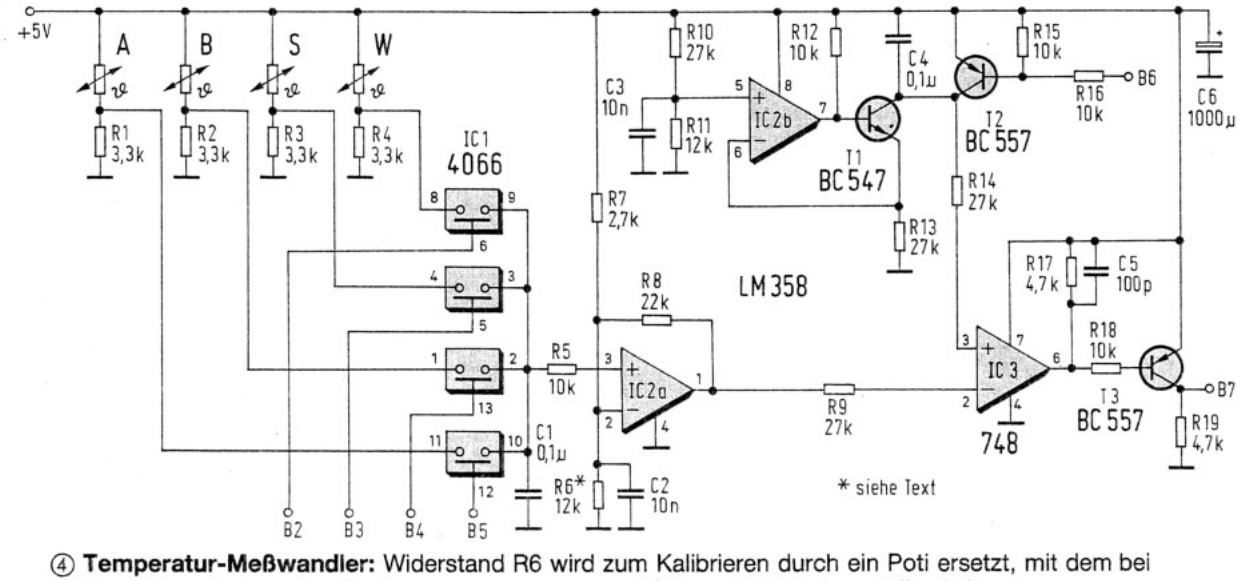
Die Ein-/Ausgabeschnittstelle

Ähnlich wie in dem Artikel „Daten-Drehscheibe" aus FUNKSCHAU 4/84 ist die hier eingesetzte Schnittstelle aufgebaut. Als Hauptbestandteil haben wir wieder eine Z80-PIO mit vorgeschaltetem Adreßdecoder. Die beiden Transistoren T1, T2 bilden ein NAND-Gatter, so daß ein komplettes TTL-IC gespart wird. Des weiteren sind die Adreßleitungen so umverdrahtet, daß es zu keinen Störungen beim Ansprechen der Schnittstelle kommt.

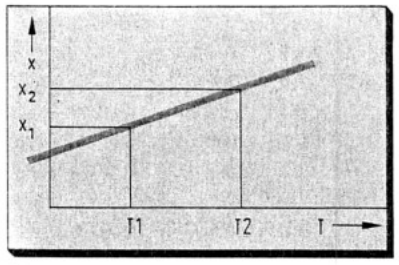
Die PIO (*Bild 3*) stellt zwei 8-Bit-Ports zur Verfügung. Davon wird Port A ausschließlich als Ausgang verwendet. A0 bis A6 werden direkt an den Treiberbaustein ULN 2003 geführt. Es handelt



③ **Schaltung** der Ein-/Ausgabeschnittstelle mit Echtzeituhr



④ **Temperatur-Meßwandler:** Widerstand R6 wird zum Kalibrieren durch ein Poti ersetzt, mit dem bei mittlerer Raumtemperatur die Ausgangsspannung von IC 2a auf 2,5 V eingestellt wird



⑤ **Der Zählerstand** ist eine lineare Funktion zur Temperaturänderung

sich hierbei um 7 Darlington-Treiber, die direkt TTL-kompatibel sind und die „Heizwiderstände" schalten. An jeden Ausgang ist zusätzlich eine LED gelegt, die den jeweiligen Zustand sichtbar macht. Das achte Bit (A7) schaltet über einen Transistor einen Summer, der im Störungsfall angesteuert werden kann.

Die Anschlüsse von Port B sind wie folgt belegt: B0 und B1 bilden mit Hilfe eines speziellen Maschinenprogramms den seriellen Datenbus zum Uhrenbaustein PCB 8573. B2 bis B7 gehen zur Temperaturwandler-Platine und steuern das Einlesen der Meßwerte. Der Uhrenchip ist quarzgesteuert und besitzt eine eigene Akkupufferung. Daher liefert er auch bei Systemzusammenbruch stets korrekte Uhrzeit und Datum.

Das Temperatur-Interface

Die zweite Platine wandelt den Meßwert der Temperatursensoren in eine proportionale Impulsbreite um, die der Rechner verarbeiten kann. Es handelt sich dabei um das Prinzip des temperaturgesteuerten Pulsmodulators. Dieses Verfahren ist preiswert, liefert ausreichende Genauigkeit und benötigt keine teuren A/D-Wandler.

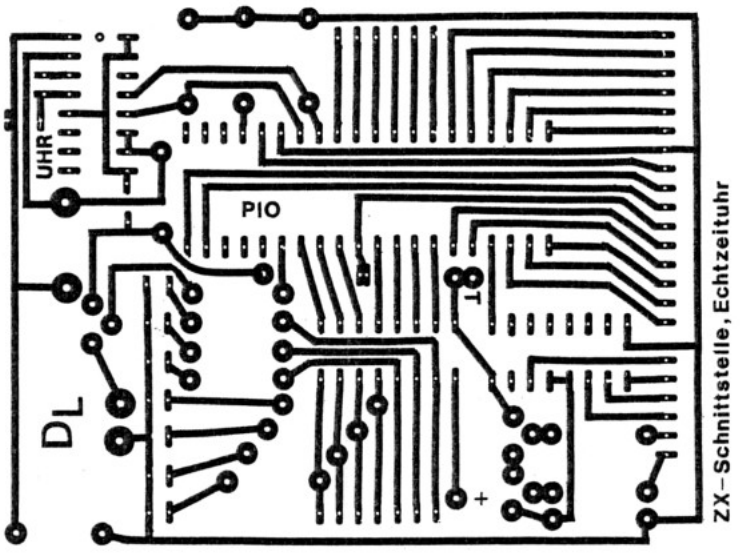
Jeder Temperaturfühler besitzt einen Vorwiderstand, der gleichzeitig zur Strombegrenzung (sonst Eigenerwärmung) und zur Linearisierung dient. Die sich ergebenden Spannungswerte werden auf je einen Eingang des 4fach-Analogschalters 4066 (IC 1) gegeben. Durch eine Binärkombination an B2...B5 wird einer der Raumfühler auf den gemeinsamen Ausgang geschaltet. IC 2a verstärkt den jeweiligen Spannungswert und legt ihn an den invertierenden Eingang des Komparators LM 748. IC 2b bildet mit seiner Beschaltung eine Konstantstromquelle, die den Kondensator C4 speist.

Zu jedem Meßvorgang wird über die Analogschalter ein Raumfühler ausgewählt und die Aufladung von C4 über den Transistor T2 freigegeben. Erreicht die Spannung am Kondensator den Wert der Spannung am Ausgang von IC 2a, so kippt der Komparator. In der Zeit zwischen Freigeben des Kondensators und Umschalten des Komparators wird eine Zählschleife im ZX 81 gestartet. Der Zählerstand stellt ein Maß für die Temperatur dar.

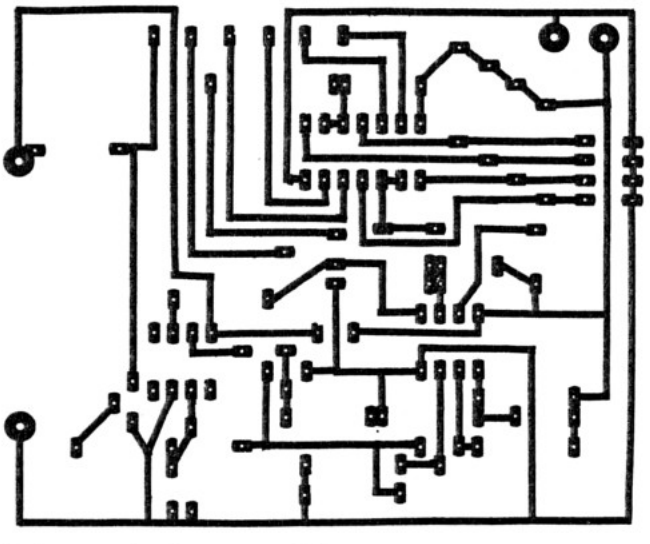
T3 schließlich stellt an seinem Kollektor TTL-Pegel zur Verfügung, da der Komparator nur zwischen 1,5 V und 4 V schalten kann (Aussteuergrenzen). Alle anderen Kondensatoren dienen vorsorglich zum Abblocken von Störimpulsen.

Kalibrierung muß sein

Die Temperatursensoren arbeiten linear; die Spannung an C4 ändert sich aufgrund der Konstantstromquelle ebenfalls linear; infolgedessen ist die Breite des Ausgangsimpulses proportional zur Temperatur. Entsprechend proportional ändert sich auch der Zählerstand in der rechner-internen Zählschleife (*Bild* 5).



⑥ **Layout** der Ein-/Ausgabeschnittstelle



⑦ **Layout** des Temperatur-Umsetzers

Somit ist auch der Zählerstand eine lineare Funktion und es gilt die allgemeine Geradengleichung:

$$T = ax + b$$

mit a und b als Konstanten, T der Temperatur und x dem Zählerstand.

Um eine große Auflösung zu erhalten, läuft IC 2a mit hoher Verstärkung, so daß schon kleine Temperaturänderungen zu einer großen Spannungsänderung führt. Da der nutzbare Spannungshub auf ca. 3 V beschränkt ist (Aussteuergrenzen) wird der nutzbare Temperaturbereich auf etwa 20 K eingeengt (also z. B. 10 °C bis 30 °C).

Um die Geradengleichung zu bestimmen, müssen bei zwei verschiedenen Temperaturen die entsprechenden Zählerstände bestimmt werden. Es gilt:

$$T1 = ax_1 + b \quad (1)$$

$$T2 = ax_2 + b \quad (2)$$

Aus diesen zwei Gleichungen mit zwei Unbekannten können die beiden Konstanten a und b berechnet werden. Setzt man die beiden Gleichungen ineinander ein, erhält man als Ergebnis:

$$a = \frac{\Delta T}{\Delta x} \quad (3)$$

$$b = T1 - \frac{\Delta T}{\Delta x} \cdot x_1 = T1 - a \cdot x_1 \quad (4)$$

mit $\Delta T = T2 - T1$, $\Delta x = x_2 - x_1$

Bevor man nun an den softwaremäßigen Abgleich geht, ist ein Meßvorgang mit Lötkolben und Ohmmeter unumgänglich. Es ist dafür zu sorgen, daß in der Mitte des nutzbaren Temperaturbereiches auch IC 2a in der Mitte des Aussteuerbereiches arbeitet. Dazu wird R6 zunächst durch ein Potentiometer ersetzt und bei mittlerer Raumtemperatur die Ausgangsspannung von IC 2a auf 2,5 V eingestellt. Der sich ergebende Widerstand wird ausgemessen und als Festwiderstand eingelötet. Der angegebene Wert von 12 kΩ ist als Richtwert beim Einsatz von KTY 81 anzusehen. Bei Verwendung des KTY 10 ergibt sich für R6 ein kleinerer Widerstand (Prinzip des Brückenabgleichs).

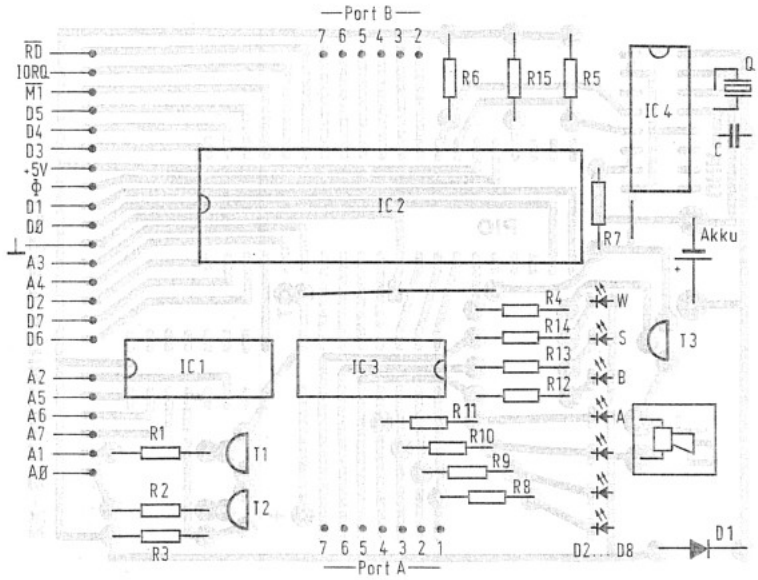
Zum Kalibrieren bringt man die vier Temperaturfühler auf die Temperatur T2 (z. B. in 30 °C warmes Wasser) und registriert die dazugehörigen Zählerstände x_2. Das gleiche Verfahren wiederholt man bei einer niedrigeren Temperatur T1 und erhält die Zählerstände x_1. Dann werden nach obigen Formeln für jeden Sensor die Konstanten a und b errechnet und in das Programm eingesetzt. Hierdurch wird jeder hardwaremäßige Abgleich überflüssig.

Um die Zählerstände zu ermitteln, bietet sich nachstehende Möglichkeit an. Hierbei wird das Unterprogramm „Temperaturmessung" (ab Zeile 8000) wie folgt geändert, wobei die dazwischenliegenden Zeilen unverändert bleiben:

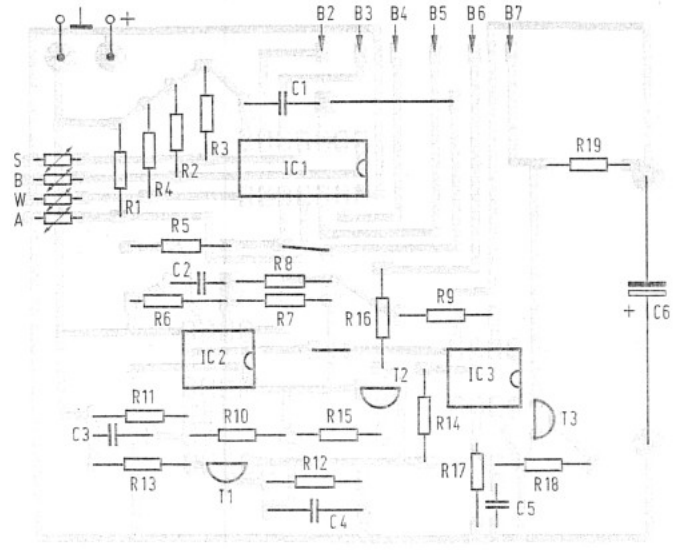
```
8020 LET TW = USR 16952
8040 LET TS  = USR 16952
8060 LET TB  = USR 16952
8080 LET TA  = USR 16952
8086 PRINT AT 0,0;TW,TS,TB,TA
8087 PAUSE 100
8088 GOTO 8000
```

Startet man nun das Programm mit GOTO 8000, so erscheinen die vier Zählerstände der vier Sensoren auf dem Bildschirm. Bringt man jetzt die Sensoren auf +30 °C, so repräsentieren jetzt die Zahlen auf dem Bildschirm die Werte für x_2. Dann bringt man die Fühler auf eine Temperatur von +10 °C und auf dem Bildschirm erhält man die Werte für x_1.

In einem Berechnungsbeispiel liefert der Sensor TW bei T1 = 10C ein x_1 =



⑧ **Bestückungsplan** der Ein-/Ausgabeschnittstelle



⑨ **Bestückungsplan** des Temperatur-Umsetzers

180 und bei T2 = 30C ein $x_2 = 400$. Damit ergibt sich

$$a = \frac{\Delta T}{\Delta x} = \frac{30 - 10}{400 - 180} \approx 0{,}09$$

$$b = T1 - a \cdot x_1 = 10 - 0{,}09 \cdot 180 = -6{,}4$$

Damit ist die Gleichung für TW bestimmt:

$$TW = \frac{1}{11} x - 6{,}4$$

Eingesetzt in Zeile 8020 lautet der endgültige Ausdruck:
8020 LET TW= INT(((USR 16952)/11)−6.4)
Die INT-Funktion schneidet noch zusätzlich die Nachkommastellen ab, damit sich eine übersichtliche Tabelle auf dem Bildschirm ergibt. Nach dem selben Schema erstellt man die Ausdrücke für die anderen Räume und setzt sie entsprechend ein (8040, 8060, 8080). Danach entfernt man wieder das Hilfsprogramm zur Kalibrierung, und der Abgleich ist beendet.

Achtung vor dem Sprungteufel

Ein handelsübliches Thermostatventil setzt sich aus zwei Hauptbestandteilen zusammen:

- das eigentliche Ventil, welches direkt in der Wasserzuführung des Heizkörpers sitzt
- und der Thermostat, an dem sich die gewünschte Temperatur einstellen läßt und der auf das Ventil gesteckt, geklemmt oder geschraubt ist.

Fast alle Thermostate lassen sich nun von ihrem Ventil abmontieren, ohne daß ein Eingriff in die Heizungsanlage notwendig ist. Wird er entfernt, dann ist der Heizkörper lediglich ständig eingeschaltet. Der Thermostat selbst besteht aus folgenden Grundelementen:

- äußeres Thermostatgehäuse mit meist geschlitztem Drehgriff und aufgedruckten Zahlen;
- innere Haltung, in dem der Temperaturfühler sitzt;
- der Fühler selber; meist ein Metallzylinder mit eingesetztem Stift; mit steigender Temperatur wird der Stift weiter herausgedrückt und schließt so das Ventil;
- manchmal auch eine Spiralfeder, die das ganze Gehäuse mechanisch unter Spannung setzt.

Zuerst wird der Drehgriff abgenommen. Alle Thermostate besitzen eine Arretierung, damit er nicht weiter als z. B. Stellung „5" nach links gedreht werden kann. Diese Arretierung wird entfernt und dann der Drehgriff ganz herausgedreht. Die weiteren Innereien sind nun meist innerhalb des Gehäusegriffes montiert. Er wird zerlegt, wobei bei den mit Spiralfedern bestückten Modellen darauf zu achten ist, daß einem die Einzelteile nicht „um die Ohren" fliegen.

Hat man den eigentlichen Temperaturfühler freigelegt, so wird die Oberfläche zunächst mit einem Stück Isolierband überklebt. Über dieses Band verteilt man nun gleichmäßig die „Heizwiderstände". Sie bestehen aus vier, in Reihe geschalteten 47-Ω-Widerständen (¼ W). Anschließend klebt man noch eine Lage Isolierband darüber, damit nichts verrutschen kann.

Die beiden Anschlußdrähte zu den Widerständen werden nun entweder durch einen der Schlitze im Griffteil nach außen geführt oder hinten am Gehäusegriff und durch die Befestiungsmanschette am Ventil ins Freie gebracht. Letztere Methode ist zwar komplizierter, hat jedoch den Vorteil, daß die Kabel unsichtbar bleiben und bei einem Besuch des Hausbesitzers die „Herumpfuscherei" nicht bemerkt wird. Ist das Ganze dann wieder zusammengebaut, so hat man einen Thermostaten, dessen Funktion weiterhin gewährleistet ist, der sich aber jetzt zusätzlich elektrisch ein- und ausschalten läßt.

In den Widerständen wird insgesamt eine Leistung von

$$P = \frac{U^2}{R} = \frac{(9)^2}{200} = 0{,}4 \text{ W}$$

umgesetzt. Dies ist die Mindestleistung, die in Wärme umgesetzt werden muß, damit der Heizkörper vollständig abschaltet. Es ist möglich, daß andere Thermostatfabrikate eine höhere „Heizleistung" benötigen. In diesem Fall sollte man etwas mehr Strom spendieren und von vornherein einen niedrigeren Widerstandswert wählen. Eine Leistung von 1 W dürfte in jedem Fall ausreichen.

Die gesamte Anlage ist seit letzten Winter in Betrieb und läuft ohne Probleme. Allgemeingültige Angaben über die mögliche Energie- bzw. Kosteneinsparung können allerdings nicht gemacht werden, da diese vom jeweiligen Programm und von den Heizgewohnheiten stark abhängen. Die Einsparung dürfte jedoch nicht unerheblich sein, besonders in Wohnungen, die tagsüber nicht genutzt werden.

Im laufenden Betrieb mit der ZX-81-Heizungsregelung werden die Thermostate mindestens eine Stufe höher gestellt, als es die Raumtemperatur erfordert. Die LEDs zeigen auf einen Blick den Status an. Zu beachten ist, daß eine leuchtende LED einen ausgeschalteten Heizkörper darstellt.

Es empfiehlt sich dringend, den gesamten Aufbau fest auf eine Grundplatte zu montieren. Besonders ist auf eine wackelfreie Verbindung des RAMs zu achten, das Sie am besten mit dem ZX 81 auf eine Grundplatte kleben.

Dieter Laues

Stückliste: Ein-/Ausgabeschnittstelle

IC 1	74LS138
IC 2	Z80 PIO
IC 3	ULN 2003 (Valvo)
IC 4	PCF 8573 (Valvo)
T1, T2, T3 BC 547 o. ä.	
D1	1N4148
D2...D8	LEDs 3 mm
Q	Uhrenquarz 32,768 Hz
Akku	NiCd 1,2 V oder 2,4 V
Summer	6 V–
C	10 pF
R1...R4	4,7 kΩ
R5, R6	10 kΩ
R7	1,8 kΩ
R8...R15	1 kΩ

Stückliste: Temperatur-Meßwandler

IC 1	CD 4066
IC 2	LM 358
IC 3	LM 748
T1	BC 547 o. ä.
T2, T3	BC 557 o. ä.
Sensoren	KTY 81 (Nennwert 1 kΩ) oder KTY 10 (Nennwert 2 kΩ)
C1, C4	100 nF
C2, C3	10 nF
C5	100 pF
C6	1000 µF
R1...R4	3,3 kΩ
R5, R12, R15, R16, R18	10 kΩ
R6	12 kΩ (siehe Text)
R7	2,7 kΩ
R8	22 kΩ
R9, R10, R13, R14	27 kΩ
R11	12 kΩ
R17, R19	4,7 kΩ

```
   1 REM 000000000000000000000
   2 REM 000000000
   3 REM 000000000000000000000000000000000000000000000000000000
   4 REM 000000000000000000000000000000000000000000000000000000000000
   5 REM 0000000000000000000000
   6 REM 000000000000000000000000000000000000000000
   7 REM 0000000000000000000000000000000000000000000000000000000
   8 REM 0000000000000000000000000
   9 REM 0000000000000000000000000000000000000000000000000000000000000000000000000
  10 REM 000000000000000000000
  11 REM 000000000000000000000000000000000000000000000000000000000000
  20 REM **INITIALISIERUNG**
  21 DIM T(12)
  22 DIM W(6)
  23 DIM S(6)
  24 DIM B(6)
  25 DIM A(6)
  26 LET G=18
  30 GOTO 500
 100 LET A$="3ECFD3DF3E00D3DF3E03D3CF3E01D3CF3E00D3CFC9"
 102 LET ADR=16514
 103 GOSUB 200
 105 LET A$="3E01D3CF3E03D3CFC9"
 107 LET ADR=16541
 108 GOSUB 200
 110 LET A$="06080707F5CB87D3CFCBC7D3CF0000CB87D3CFF110ED3ECFD3DF3E02D3DFCBC7D3CFDBCF00CB87D3CF3ECFD3DF3E00D3DFC91C1C"
 112 LET ADR=16556
 113 GOSUB 200
 115 LET A$="3ECFD3DF3E02D3DF060817CBC7D3CF00DBCFCB87D3CF10F21FF53ECFD3DF3E00D3DFCB8FD3CFCBC7D3CFCB412806CB87D3CFF1C9CBCFD3CFF1C9"
 117 LET ADR=16614
 118 GOSUB 200
 120 LET A$="0100010105020701011125171314151617181920"
 122 LET ADR=16678
 123 GOSUB 200
 125 LET A$="CD82403ED0CDAC403E00CDAC403A3141CDAC403A3041CDAC403A2F41CDAC403A2E41CDAC40CD9D40C9"
 127 LET ADR=16704
 128 GOSUB 200
 130 LET A$="0E01CD82403ED0CDAC403E00CDAC40CD9D40CD82403ED1CDAC40CDE640323141CDE640323041CDE6403 22F410E00CDE640322E41CDCC41C9"
 132 LET ADR=16751
 133 GOSUB 200
 135 LET A$="F5CB3FCB3FCB3FCB3F772BC61C1213F1E60F77C61C12132BC9"
 137 LET ADR=16813
 138 GOSUB 200
 140 LET A$="212D41ED5B0C40133A3141CDAD413E0E12133A3041CDAD41133E3A12133E2D12133E3712060B1310FD3A2F41CDAD413E1B12133A2E41CDAD413E1B12133E1D12133E2512133E2412133E2112C9"
 142 LET ADR=16844
 143 GOSUB 200
 145 LET A$="203A1F42CBA7321F423E0FD3D73A1F42D3C7C9"
 147 LET ADR=16927
 148 GOSUB 200
 150 LET A$="3ECFD3DF3E83D3DF3E00CBF7CBEFD3CFCBB7D3CF06FF0020FDCBF7D3CF06000E003E10B82808DBCFCB7F0328F4C93E01326C42C900"
 152 LET ADR=16952
 153 GOSUB 200
 160 STOP
 200 FOR N=0 TO LEN A$-2 STEP 2
 205 POKE ADR+INT (N/2),16*(CODE A$(N+1)-28)+CODE A$(N+2)-28
 210 NEXT N
 220 RETURN
 230 CLS
 240 PRINT "EINGABE: UHRZEIT/DATUM"
 250 PRINT "STUNDE/MINUTE/TAG/MONAT"
 260 PRINT "JEWEILS ZWEISTELLIG"
 270 INPUT A$
 280 IF LEN A$<>8 THEN RETURN
 290 LET N=16689
 300 LET C$=A$(1 TO 2)
 310 GOSUB 390
 320 LET C$=A$(3 TO 4)
 330 GOSUB 390
 340 LET C$=A$(5 TO 6)
 350 GOSUB 390
 360 LET C$=A$(7 TO 8)
 370 GOSUB 390
 380 RETURN
 390 LET C=16*(CODE C$(1)-28)+CODE C$(2)-28
 400 POKE N,C
 410 LET N=N-1
 415 RAND USR 16704
 420 RETURN
 500 CLS
 510 PRINT AT 0,0;"ST  = START"
 520 PRINT AT 2,0;"U   = UHR STELLEN"
 525 PRINT AT 4,0;"G   = GRUNDTEMPERATUR"
 530 PRINT AT 6,0;"EIN= ALLE HEIZKOERPER EIN"
 540 PRINT AT 8,0;"AUS= ALLE HEIZKOERPER AUS"
 550 PRINT AT 11,0;"ZEITEN  AENDERN/LESEN:"
 560 PRINT "████████████████████████████████"
 570 PRINT AT 13,0,"W   = WOHNZIMMER"
 580 PRINT "S   = SCHLAFZIMMER"
 590 PRINT "B   = BAD"
 600 PRINT "A   = ARBEITSZIMMER"
 610 INPUT A$
 615 IF A$="G" THEN GOSUB 800
 620 IF A$="U" THEN GOSUB 230
 630 IF A$="W" THEN GOSUB 1000
 640 IF A$="S" THEN GOSUB 2000
 650 IF A$="B" THEN GOSUB 3000
 660 IF A$="A" THEN GOSUB 4000
 670 IF A$="EIN" THEN GOSUB 6100
 680 IF A$="AUS" THEN GOSUB 6000
 690 IF A$="ST" THEN GOTO 4900
 700 GOTO 500
 800 CLS
 810 PRINT AT 20,0;"GRUNDTEMPERATUR: WERKTAGS"
 820 INPUT G1
 830 PRINT AT 20,0;"GRUNDTEMPERATUR: WOCHENENDE"
 840 INPUT G2
 850 CLS
 860 RETURN
1000 CLS
1010 PRINT "WOHNZIMMER"
1020 PRINT "██████████"
1030 PRINT AT 3,0;"PROGRAMMIERTE ZEITEN:"
1040 IF T(1)=0 AND T(2)=0 AND T(3)=0 THEN GOTO 1220
1050 PRINT AT 5,0;"EIN         AUS       TEMPERATUR"
1060 IF T(1)<>0 THEN PRINT AT 7,0;W(1);AT 7,12;W(2);AT 7,22;T(1);" C"
1070 IF T(2)<>0 THEN PRINT AT 9,0;W(3);AT 9,12;W(4);AT 9,22;T(2);" C"
1080 IF T(3)<>0 THEN PRINT AT 11,0;W(5);AT 11,12;W(6);AT 11,22;T(3);" C"
1090 PRINT AT 20,0;"AENDERUNGEN ? (J/N)"
1100 INPUT B$
1110 IF B$="N" THEN RETURN
1120 PRINT AT 20,0;"WELCHER ZEITBEREICH? (1,2,3)"
1130 INPUT A
1132 IF A=1 THEN LET N=1
1134 IF A=2 THEN LET N=3
1136 IF A=3 THEN LET N=5
1140 IF A>3 THEN GOTO 1130
1150 PRINT AT 20,0;"EINSCHALTZEIT ?"
1160 INPUT W(N)
1170 PRINT AT 20,0;"AUSSCHALTZEIT ?"
1180 INPUT W(N+1)
1190 PRINT AT 20,0;"TEMPERATUR ?"
1200 INPUT T(A)
1210 GOTO 1000
1220 PRINT AT 5,0;"KEINE"
1230 GOTO 1090
2000 CLS
2010 PRINT "SCHLAFZIMMER"
2020 PRINT "████████████"
2030 PRINT AT 3,0;"PROGRAMMIERTE ZEITEN:"
2040 IF T(4)=0 AND T(5)=0 AND T(6)=0 THEN GOTO 2220
2050 PRINT AT 5,0;"EIN         AUS       TEMPERATUR"
2060 IF T(4)<>0 THEN PRINT AT 7,0;S(1);AT 7,12;S(2);AT 7,22;T(4);" C"
2070 IF T(5)<>0 THEN PRINT AT 9,0;S(3);AT 9,12;S(4);AT 9,22;T(5);" C"
2080 IF T(6)<>0 THEN PRINT AT 11,0;S(5);AT 11,12;S(6);AT 11,22;T(6);" C"
2090 PRINT AT 20,0;"AENDERUNGEN ? (J/N)"
2100 INPUT B$
2110 IF B$="N" THEN RETURN
2120 PRINT AT 20,0;"WELCHER ZEITBEREICH? (1,2,3)"
2130 INPUT A
2132 IF A=1 THEN LET N=1
2134 IF A=2 THEN LET N=3
2136 IF A=3 THEN LET N=5
2140 IF A>3 THEN GOTO 2130
2150 PRINT AT 20,0;"EINSCHALTZEIT ?"
2160 INPUT S(N)
2170 PRINT AT 20,0;"AUSSCHALTZEIT ?"
2180 INPUT S(N+1)
2190 PRINT AT 20,0;"TEMPERATUR ?"
2200 INPUT T(A+3)
2210 GOTO 2000
2220 PRINT AT 5,0;"KEINE"
2230 GOTO 2090
3000 CLS
3010 PRINT "BAD"
3020 PRINT "███"
3030 PRINT AT 3,0;"PROGRAMMIERTE ZEITEN:"
3040 IF T(7)=0 AND T(8)=0 AND T(9)=0 THEN GOTO 3220
3050 PRINT AT 5,0;"EIN         AUS       TEMPERATUR"
3060 IF T(7)<>0 THEN PRINT AT 7,0;B(1);AT 7,12;B(2);AT 7,22;T(7);" C"
3070 IF T(8)<>0 THEN PRINT AT 9,0;B(3);AT 9,12;B(4);AT 9,22;T(8);" C"
3080 IF T(9)<>0 THEN PRINT AT 11,0;B(5);AT 11,12;B(6);AT 11,22;T(9);" C"
3090 PRINT AT 20,0;"AENDERUNGEN ? (J/N)"
3100 INPUT B$
3110 IF B$="N" THEN RETURN
3120 PRINT AT 20,0;"WELCHER ZEITBEREICH? (1,2,3)"
3130 INPUT A
3132 IF A=1 THEN LET N=1
3134 IF A=2 THEN LET N=3
3136 IF A=3 THEN LET N=5
3140 IF A>3 THEN GOTO 3130
3150 PRINT AT 20,0;"EINSCHALTZEIT ?"
3160 INPUT B(N)
3170 PRINT AT 20,0;"AUSSCHALTZEIT ?"
3180 INPUT B(N+1)
3190 PRINT AT 20,0;"TEMPERATUR ?"
3200 INPUT T(A+6)
3210 GOTO 3000
3220 PRINT AT 5,0;"KEINE"
3230 GOTO 3090
4000 CLS
4010 PRINT "ARBEITSZIMMER"
4020 PRINT "█████████████"
4030 PRINT AT 3,0;"PROGRAMMIERTE ZEITEN:"
4040 IF T(10)=0 AND T(11)=0 AND T(12)=0 THEN GOTO 4220
4050 PRINT AT 5,0;"EIN         AUS       TEMPERATUR"
4060 IF T(10)<>0 THEN PRINT AT 7,0;A(1);AT 7,12;A(2);AT 7,22;T(10);" C"
4070 IF T(11)<>0 THEN PRINT AT 9,0;A(3);AT 9,12;A(4);AT 9,22;T(11);" C"
4080 IF T(12)<>0 THEN PRINT AT 11,0;A(5);AT 11,12;A(6);AT 11,22;T(12);" C"
4090 PRINT AT 20,0;"AENDERUNGEN ? (J/N)"
4100 INPUT B$
4110 IF B$="N" THEN RETURN
4120 PRINT AT 20,0;"WELCHER ZEITBEREICH? (1,2,3)"
4130 INPUT A
4132 IF A=1 THEN LET N=1
4134 IF A=2 THEN LET N=3
4136 IF A=3 THEN LET N=5
4140 IF A>3 THEN GOTO 4130
4150 PRINT AT 20,0;"EINSCHALTZEIT ?"
4160 INPUT A(N)
4170 PRINT AT 20,0;"AUSSCHALTZEIT ?"
4180 INPUT A(N+1)
4190 PRINT AT 20,0;"TEMPERATUR ?"
4200 INPUT T(A+9)
4210 GOTO 4000
4220 PRINT AT 5,0;"KEINE"
4230 GOTO 4090
4900 CLS
4905 PRINT AT 1,0;"████████████████████████████████"
4910 PRINT AT 3,0;"RAUM:        TEMPERATUR  HEIZUNG"
4915 PRINT AT 4,14,"IST  SOLL"
4920 PRINT AT 6,0;"WOHNZIMMER"
4930 PRINT AT 8,0;"SCHLAFZIMMER"
4940 PRINT AT 10,0;"BAD"
4950 PRINT AT 12,0;"ARBEITSZIMMER"
4960 GOSUB 8000
4970 IF INKEY$="H" THEN GOTO 500
5000 REM **UHR LESEN**
5010 RAND USR 16751
5020 LET Z=(PEEK 16685)*10+PEEK 16684+(PEEK 16683)/10+(PEEK 16682)/100
5030 IF Z>=W(1) AND Z<W(2) THEN GOTO 7000
5040 IF Z>=W(3) AND Z<W(4) THEN GOTO 7050
```

```
5050 IF Z>=W(5) AND Z<W(6) THEN
GOTO 7100
5055 GOSUB 6200
5060 IF Z>=S(1) AND Z<S(2) THEN
GOTO 7150
5070 IF Z>=S(3) AND Z<S(4) THEN
GOTO 7200
5080 IF Z>=S(5) AND Z<S(6) THEN
GOTO 7250
5085 GOSUB 6250
5090 IF Z>=B(1) AND Z<B(2) THEN
GOTO 7300
5100 IF Z>=B(3) AND Z<B(4) THEN
GOTO 7350
5110 IF Z>=B(5) AND Z<B(6) THEN
GOTO 7400
5115 GOSUB 6300
5120 IF Z>=A(1) AND Z<A(2) THEN
GOTO 7450
5130 IF Z>=A(3) AND Z<A(4) THEN
GOTO 7500
5140 IF Z>=A(5) AND Z<A(6) THEN
GOTO 7550
5145 GOSUB 6350
5150 LET TAG=(PEEK 16681)*10+PEE
K 16680
5160 LET MONAT=(PEEK 16679)*10+P
EEK 16678
5170 LET J=1986
5180 LET J1=J-1
5190 IF MONAT>2 THEN LET J1=J
5200 LET M1=MONAT+13
5210 IF MONAT>2 THEN LET M1=MONA
T+1
5220 LET TZ=INT (365.25*J1)+INT
(30.6*M1)+TAG
5230 LET WT=7*(((TZ+5)/7)-INT ((
TZ+5)/7))
5240 LET E=INT ((WT*100+.5)/100)
5250 IF E=0 THEN PRINT AT 0,15;"
SO"
5260 IF E=1 THEN PRINT AT 0,15;"
MO"
5270 IF E=2 THEN PRINT AT 0,15;"
DI"
5280 IF E=3 THEN PRINT AT 0,15;"
MI"
5290 IF E=4 THEN PRINT AT 0,15;"
DO"
5300 IF E=5 THEN PRINT AT 0,15;"
FR"
5310 IF E=6 THEN PRINT AT 0,15;"
SA"
5320 IF E=0 OR E=6 THEN LET G=G2
5330 IF E>0 AND E<6 THEN LET G=G
1
5340 GOTO 4905
6000 REM **ALLE HEIZUNGEN EIN**
6010 POKE 16927,254
6020 RAND USR 16936
6030 RETURN
6100 REM **ALLE HEIZUNGEN AUS**
6110 POKE 16927,0
6120 RAND USR 16936
6130 RETURN
6200 IF TW<G THEN POKE 16932,191
6210 IF TW<G THEN PRINT AT 6,15;
TW;AT 6,20;G;AT 6,25;"EIN"
6220 IF TW>=G THEN POKE 16932,25
5
6230 IF TW>=G THEN PRINT AT 6,15
;TW;AT 6,20;G;AT 6,25;"AUS"
6235 RAND USR 16928
6240 RETURN
6250 IF TS<G THEN POKE 16932,183
6260 IF TS<G THEN PRINT AT 8,15;
TS;AT 8,20;G;AT 8,25;"EIN"
6270 IF TS>=G THEN POKE 16932,24
7
6280 IF TS>=G THEN PRINT AT 8,15
;TS;AT 8,20;G;AT 8,25;"AUS"
6285 RAND USR 16928
6290 RETURN
6300 IF TB<G THEN POKE 16932,175
6310 IF TB<G THEN PRINT AT 10,15
;TA;AT 10,20;G;AT 10,25;"EIN"
6320 IF TB>=G THEN POKE 16932,23
9
6330 IF TB>=G THEN PRINT AT 10,1
5;TB;AT 10,20;G;AT 10,25;"AUS"
6335 RAND USR 16928
6340 RETURN
6350 IF TA<G THEN POKE 16932,167
6360 IF TA<G THEN PRINT AT 12,15
;TA;AT 12,20;G;AT 12,25;"EIN"
6370 IF TA>=G THEN POKE 16932,23
1
6380 IF TA>=G THEN PRINT AT 12,1
5;TA;AT 12,20;G;AT 12,25;"AUS"
6385 RAND USR 16928
6390 RETURN
7000 IF TW<T(1) THEN POKE 16932,
191
7005 IF TW<T(1) THEN PRINT AT 6,
15;TW;TAB 20;T(1);TAB 25;"EIN"
7010 IF TW>=T(1) THEN POKE 16932
,255
7015 IF TW>=T(1) THEN PRINT AT 6
,15;TW;TAB 20;T(1);TAB 25;"AUS"
7020 RAND USR 16928
7030 GOTO 5060
7050 IF TW<T(2) THEN POKE 16932,
191
7055 IF TW<T(2) THEN PRINT AT 6,
15;TW;TAB 20;T(2);TAB 25;"EIN"
7060 IF TW>=T(2) THEN POKE 16932
,255
7065 IF TW>=T(2) THEN PRINT AT 6
,15;TW;TAB 20;T(2);TAB 25;"AUS"
7070 RAND USR 16928
7080 GOTO 5060
7100 IF TW<T(3) THEN POKE 16932,
191
7105 IF TW<T(3) THEN PRINT AT 6,
15;TW;TAB 20;T(3);TAB 25;"EIN"
7110 IF TW>=T(3) THEN POKE 16932
,255
7115 IF TW>=T(3) THEN PRINT AT 6
,15;TW;TAB 20;T(3);TAB 25;"AUS"
7120 RAND USR 16928
7130 GOTO 5060
7150 IF TS<T(4) THEN POKE 16932,
183
7155 IF TS<T(4) THEN PRINT AT 8,
15;TS;TAB 20;T(4);TAB 25;"EIN"
7160 IF TS>=T(4) THEN POKE 16932
,247
7165 IF TS>=T(4) THEN PRINT AT 8
,15;TS;TAB 20;T(4);TAB 25;"AUS"
7170 RAND USR 16928
7180 GOTO 5090
7200 IF TS<T(5) THEN POKE 16932,
183
7205 IF TS<T(5) THEN PRINT AT 8,
15;TS;TAB 20;T(5);TAB 25;"EIN"
7210 IF TS>=T(5) THEN POKE 16932
,247
7215 IF TS>=T(5) THEN PRINT AT 8
,15;TS;TAB 20;T(5);TAB 25;"AUS"
7220 RAND USR 16928
7230 GOTO 5090
7250 IF TS<T(6) THEN POKE 16932,
183
7255 IF TS<T(6) THEN PRINT AT 8,
15;TS;TAB 20;T(6);TAB 25;"EIN"
7260 IF TS>=T(6) THEN POKE 16932
,247
7265 IF TS>=T(6) THEN PRINT AT 8
,15;TS;TAB 20;T(6);TAB 25;"AUS"
7270 RAND USR 16928
7280 GOTO 5090
7300 IF TB<T(7) THEN POKE 16932,
175
7305 IF TB<T(7) THEN PRINT AT 10
,15;TB;TAB 20;T(7);TAB 25;"EIN"
7310 IF TB>=T(7) THEN POKE 16932
,239
7315 IF TB>=T(7) THEN PRINT AT 1
0,15;TB;TAB 20;T(7);TAB 25;"AUS"
7320 RAND USR 16928
7330 GOTO 5120
7350 IF TB<T(8) THEN POKE 16932,
175
7355 IF TB<T(8) THEN PRINT AT 10
,15;TB;TAB 20;T(8);TAB 25;"EIN"
7360 IF TB>=T(8) THEN POKE 16932
,239
7365 IF TB>=T(8) THEN PRINT AT 1
0,15;TB;TAB 20;T(8);TAB 25;"AUS"
7370 RAND USR 16928
7380 GOTO 5120
7400 IF TB<T(9) THEN POKE 16932,
175
7405 IF TB<T(9) THEN PRINT AT 10
,15;TB;TAB 20;T(9);TAB 25;"EIN"
7410 IF TB>=T(9) THEN POKE 16932
,239
7415 IF TB>=T(9) THEN PRINT AT 1
0,15;TB;TAB 20;T(9);TAB 25;"AUS"
7420 RAND USR 16928
7430 GOTO 5120
7450 IF TA<T(10) THEN POKE 16932
,167
7455 IF TA<T(10) THEN PRINT AT 1
2,15;TA;TAB 20;T(10);TAB 25;"EIN
"
7460 IF TA>=T(10) THEN POKE 1693
2,231
7465 IF TA>=T(10) THEN PRINT AT
12,15;TA;TAB 20;T(10);TAB 25;"AU
S"
7470 RAND USR 16928
7480 GOTO 5150
7500 IF TA<T(11) THEN POKE 16932
,167
7505 IF TA<T(11) THEN PRINT AT 1
2,15;TA;TAB 20;T(11);TAB 25;"EIN
"
7510 IF TA>=T(11) THEN POKE 1693
2,231
7515 IF TA>=T(11) THEN PRINT AT
12,15;TA;TAB 20;T(11);TAB 25;"AU
S"
7520 RAND USR 16928
7530 GOTO 5150
7550 IF TA<T(12) THEN POKE 16932
,167
7555 IF TA<T(12) THEN PRINT AT 1
2,15;TA TAB 20;T(12);TAB 25;"EIN
"
7560 IF TA>=T(12) THEN POKE 1693
2,231
7565 IF TA>=T(12) THEN PRINT AT
12,15;TA;TAB 20;T(12);TAB 25;"AU
S"
7570 RAND USR 16928
7580 GOTO 5150
8000 REM **TEMPERATURMESSUNG**
8005 FAST
8010 POKE 16965,215
8020 LET TW=INT (((USR 16952)/13
)-3)
8025 IF TW>99 THEN LET TW=99
8030 POKE 16965,223
8040 LET TS=INT (((USR 16952)/12
)-3)
8045 IF TS>99 THEN LET TS=99
8050 POKE 16965,231
8060 LET TB=INT (((USR 16952)/12
)-6)
8065 IF TB>99 THEN LET TB=99
8070 POKE 16965,239
8080 LET TA=INT (((USR 16952)/11
)-3)
8082 IF TA>99 THEN LET TA=99
8085 SLOW
8090 RETURN
```

Listing „Heizungsregelung“: Die Zahl der Nullen in den Zeilen 1 bis 11 muß genau eingehalten werden. Erster Programmstart mit RUN 100; der ZX 81 lädt dann etwa 1 min. lang das Maschinenprogramm in die REM-Zeilen. Danach „Kaltstart“ mit RUN und „Warmstart“ mit GOTO 500 einleiten. Arbeitet das Programm einwandfrei, darf der Eingabeteil für das Maschinenprogramm (Zeile 100 bis 220) gelöscht werden

Disketten-Versand:

Schlappscheibe auf Reisen

Das Problem ist seit langem bekannt: Wie verpacke ich am besten eine Floppy-Disk, damit die „Schlappscheibe“ den Postweg heil übersteht? Mit einem „Disketten-Mailer“ will die Firma 3M diese Aufgabe gelöst haben. Es handelt sich dabei um einen stabilen Versandkarton-Umschlag für 5¼-Zoll- und 8-Zoll-Disketten. Eine beigefügte Kunststoffhülle soll die Disketten zusätzlich gegen Staub und Schmutz schützen. Auf der Vorderseite des Versandkartons brauchen lediglich noch Adressat und Absender eingetragen werden – und schon kann, nach dem Frankieren, die Post abgehen. -ll

ZX 81 à la carte

Datentransfer mit dem ZX 81

Der Kleinste der Kleinen wird hoffähig

Der ZX 81 als Terminal, wer hätte das gedacht. Mailboxen rücken in greifbare Nähe, aber auch der Drucker oder die Modelleisenbahn lassen sich ansteuern – und das alles zu einem äußerst kleinen Preis.

Der ZX 81 wurde in großen Stückzahlen verkauft und ist im Preis-/Leistungsverhältnis bis heute unübertroffen. Dennoch werden viele ZX-Besitzer ihren Rechner unbenutzt lassen. Mit der hier vorgestellten Hard- und Software können Sie Ihren Rechner wieder zu neuem Leben erwecken.

Selbst in der 1-KByte-Version ergeben sich dank der seriellen Schnittstelle eine Reihe von neuen Möglichkeiten. Wer mit 16-KByte-RAM arbeiten möchte, kann den Speicher, auch teilweise, zum Ablegen von empfangenen Daten benutzen und diese später, z. B. mit einem Basic-Programm, bearbeiten, ebenso ist die Ausgabe von vorbereiteten Texten möglich. Dank der Terminal-Funktion ist es auch kein Problem, den ZX 81 mit anderen Rechnern zu verbinden, um so beispielsweise Einkarten-Rechner zu betreiben.

Um ZX-Eigenheiten braucht man sich bei der Terminal-Benutzung nicht mehr zu kümmern: Es steht der gesamte Bildschirm mit 32 × 24 Zeichen auch in der 1-KByte-Version zur Verfügung. Sogar der Cursor hat das Blinken gelernt. Besitzer von Uralt-Versionen können aufatmen, denn die Terminal- und Druckersoftware berücksichtigt ROM-Unterschiede automatisch.

Im Terminal-Betrieb wird die Tastatur zur Erzeugung von ASCII-Zeichen benutzt. Alle Tasten haben jetzt durch die Software eine Autorepeat-Funktion erhalten, d. h. bei gedrückter Taste wird das jeweilige Zeichen wiederholt ausgegeben. Eine wesentliche Anwendung des Terminals kann auch der Modembetrieb sein. Man braucht nur noch den Akustikkoppler, z. B. aus Heft 22/85, ein Telefon und das Rumstöbern in Datenbanken kann losgehen.

Wem Terminal- und Modembetrieb nicht reichen, der kann nun mit dem ZX 81 einen Drucker mit serieller Schnittstelle verbinden: RAND USR 8222 – und das jeweilige Basic-Programm erscheint auf dem Druckerpapier. Natürlich sind auch Stringausgabe, Copy und teilweises Listen eines Basic-Programmes möglich. Über eine Reihe von Routinen ist auch der serielle Schnittstellenbaustein in all seinen Funktionen programmierbar.

Einige der neuesten Entwicklungen, beispielsweise programmierbare Modelleisenbahnen, Miniroboter und sogar eine Armbanduhr, lassen sich ebenfalls über die serielle Schnittstelle ansprechen.

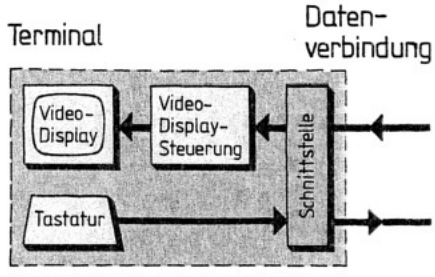
Packen Sie es an: Die Schaltung ist nachbausicher und arbeitet mit normalen Bauteilen. Das EPROM gibt es fertig programmiert über den Franzis-Softwareservice.

Terminal – was ist das?

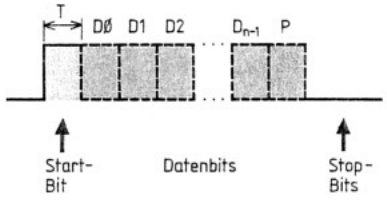
In der Entwicklung der Ein-/Ausgabegeräte für Mikrocomputer stellen Terminals die zweite Stufe dar. Davor gab es noch die den Fernschreibern ähnlichen „Teletypes". Heute ist es für Anwender von Mikrocomputern selbstverständlich, daß Videoausgabe und Tastatur schon Teil des Rechners selbst sind.

Für die Codierung der Zeichen, also der Vorschrift, welcher Nummer des Codevorrates welches Symbol zugeordnet wird, wurden verschiedene, immer noch gültige Vereinbarungen getroffen. Der verwendete ASCII-Code läßt sich in zwei Blöcke teilen, zum einen in die druckbaren Zeichen, die auf dem Bildschirm oder auf dem Papier stehenbleiben, zum anderen gibt es Zeichen, die zur Steuerung der Übertragung und der angeschlossenen Geräte verwendet werden.

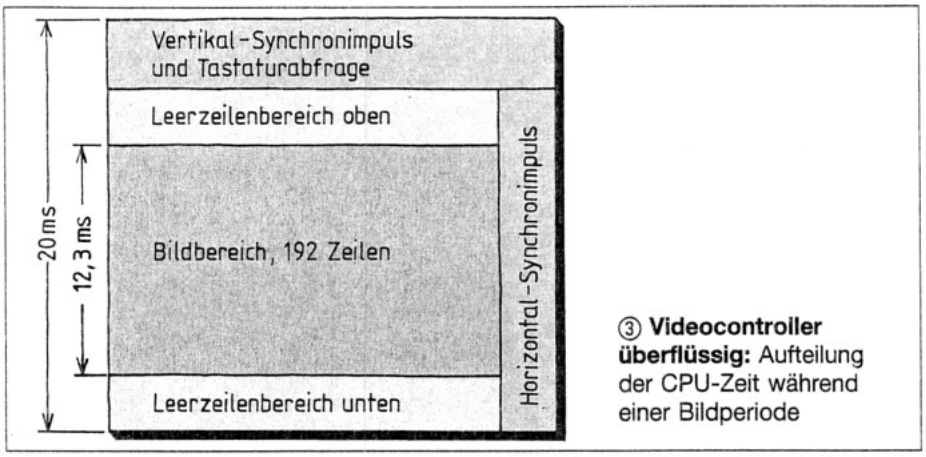
Ein Terminal (*Bild 1*) ist also ein Gerät, das als Empfänger die druckbaren Zeichen darstellt und auf Steuerzeichen reagieren kann. Im Sendebetrieb erzeugt



① **Datenterminal:** Das Blockschaltbild zeigt, was alles dazugehört



② **Serielle Datenübertragung:** Datenwort auf der Schnittstellenverbindung



③ **Videocontroller überflüssig:** Aufteilung der CPU-Zeit während einer Bildperiode

die ASCII-Zeichen aus der Tastaturbetätigung und gibt sie über die Schnittstelle aus. Verbindet man zwei solche Geräte miteinander, dazu können natürlich auch zwei ZX-Terminals verwendet werden, so ist bereits eine Datenkommunikation möglich.

Empfangsseitig gibt es eine Anzahl von weiteren Vereinbarungen. Ist z. B. der letzte Zeichenplatz einer Zeile beschrieben worden, so erscheint das folgende Zeichen am Anfang der nächsten Zeile. Wurde die unterste Zeile schon beschrieben, so wird der gesamte Bildschirminhalt nach oben verschoben, um Platz für eine neue Zeile zu schaffen; diese Verschiebung wird Scrolling genannt. Der Platz auf dem Bildschirm, an dem das nächste Symbol erscheint, wird mit dem Cursor markiert.

Wichtig ist, daß Sende- und Empfangsteil des Terminals unabhängig voneinander arbeiten müssen. Der Empfangsteil reagiert also zunächst überhaupt nicht auf die Tastaturbetätigung und die ausgesendeten Zeichen.

Diese Eigenschaft wird bei Terminals auch „Voll-Duplex-Betrieb" genannt: Informationen können gleichzeitig zwischen Sender und Empfänger übertragen werden. Ist z. B. die Verbindung mit einer Mailbox aufgebaut, so wird ein gesendetes Zeichen gleich zurückgegeben. Der Benutzer hat auf diese Weise eine Eingabekontrolle und weiß auch noch, ob das Zeichen beim Empfänger richtig eingetroffen ist, denn die Übertragungsstrecke kann ja durchaus Tausende von Kilometern betragen.

```
                        ;       *  ENTRY-POINTS:
                        ;
2000    C3 2072                 JP      INITRM          ;8192 TERMINAL, FESTE INITIALISIERUNG
2003    C3 207A                 JP      INIUTR          ;8195 TERMINAL, FREIE INITIALISIERUNG
2006    C3 2762                 JP      INIRAM          ;8198 RAM-VEREINB. ÜBER RAMTOP SETZEN
                        ;                       ;NUTZUNG DER SCHNITTSTELLE
                        ;                       ;OHNE TERMINAL-FUNKTION:
2009    C3 20C3                 JP      INITSP          ;8201 INITIALISIEREN
200C    C3 2033                 JP      USRCMD          ;8204 CMD SETZEN
200F    C3 2039                 JP      USRSTA          ;8207 STATUS ABFRAGEN
2012    C3 203F                 JP      USRDOU          ;8210 ZEICHEN AUSGEBEN
2015    C3 2045                 JP      USRDIN          ;8213 ZEICHEN LESEN
2018    C3 204B                 JP      USROUW          ;8216 WARTEN UND AUSGABE
201B    C3 2063                 JP      USRINW          ;8219 WARTEN UND EINGABE
                        ;
201E    C3 259D                 JP      BASLIS          ;8222 BASIC-LISTING, VOLLSTÄNDIG
2021    C3 25A4                 JP      BASPLI          ;8225 BASIC-LISTING AB EINER Z.NR.
2024    C3 26C6                 JP      BASSTR          ;8228 AUSGABE VON STRINGS
2027    C3 2708                 JP      BASCOP          ;8231 BILDSCHIRMKOPIE,22 ZEILEN
202A    C3 236A                 JP      DOWNLI          ;8234 AUSGABE DES DOWNL. PUFFERS
202D    C3 271F                 JP      BASAVE          ;8237 BASIC-PROGRAMM SICHERN, BINÄR
2030    C3 2731                 JP      BALOAD          ;8240 BASIC-PROGRAMM LADEN, BINÄR
                        ;
```

④ **Sprungleiste:** Am Anfang des EPROM-Bereichs erfolgt der Einsprung zu den einzelnen Maschinenprogrammen

Serielle Datenübertragung Bit für Bit

Bei der seriellen Übertragung von Daten werden die einzelnen Bits eines Datenwortes zeitlich aufeinanderfolgend über eine Leitung geschickt. Da außer den Daten keine weiteren Signale zur Übermittlung verwendet werden, muß die Synchronisation zwischen Sender und Empfänger ebenfalls über die Datenleitung erfolgen. Dies geschieht durch Festlegung eines bestimmten Datenformates, das dem Sender sowie dem Empfänger vor der Übertragung bekannt sein muß.

Der Aufbau eines Datenwortes läßt sich auch für die Schnittstelle des ZX-Terminals einstellen. Allgemein gilt das in *Bild 2* dargestellte Datenformat. Auf der Schnittstellenverbindung werden die Daten invertiert übertragen. Ein „H"-Pegel entspricht also einer logischen „0".

Den Datenbits ist das Startbit vorangestellt. Je nach Länge des Datenwortes folgen dann die n-Datenbits, wobei das Niederwertigste als erstes Bit gesendet wird. Die höherwertigen Bits schließen sich in aufsteigender Wertigkeit an.

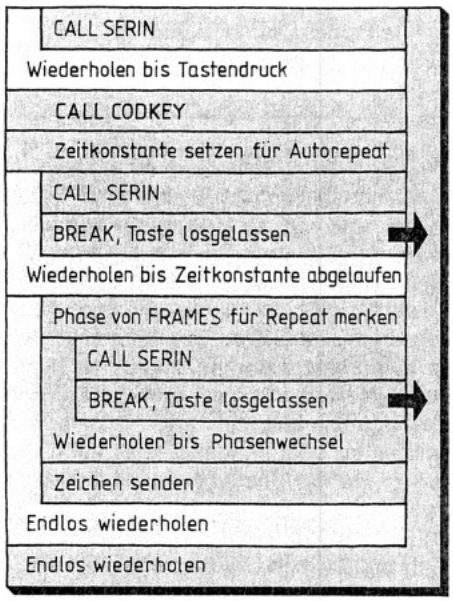
Zur Erkennung von Fehlern, die bei der Datenübertragung aufgetreten sein könnten, kann man ein Parity-Bit (P) benutzen. Je nach Betriebsart setzt der Sender das P-Bit bei gerader oder ungerader Parität, d. h. gerader oder ungerader Anzahl von „1"-Zuständen in den Datenbits.

Die Fehlererkennung geschieht dadurch, daß der Empfänger das Parity-Bit selbst noch einmal für die Datenbits ermittelt und dann mit dem empfangenen Parity-Bit vergleicht. Ein Fehler wird im Statusregister des Schnittstellenbausteines (Port) vermerkt. Das Ende der Übertragung eines Datenwortes bildet das Stop-Bit, in seiner Dauer kann es 1, 1,5 oder 2 Datenbits entsprechen. Der zeitliche Abstand T, mit dem die einzelnen Bits eines Zeichens auf der Übertragungsleitung erscheinen, ist über die Baudrate (Bit/s) festgelegt:

$$T = \frac{1}{\text{Baudrate}}$$

Bei einer Einstellung von z. B. 300 Bd besitzen die Datenbits eine Dauer von 3,33 ms.

MAINLP



⑤ **Terminalhauptschleife:** Das Struktogramm zeigt Ablauf des Hauptprogramms mit seiner Endlosschleife

Datenempfang und Synchronisation

Da die Takte von Datensender und -empfänger nicht starr gekoppelt sind und somit keinen festen zeitlichen Bezug zueinander haben, wird über die Datenleitung die Synchronisierung für jedes einzelne Zeichen neu vorgenommen. Eine wesentliche Funktion hat bei dieser „asynchronen Datenübertragung" das Startbit. Durch die erste Signalflanke wird im Schnittstellenbaustein der Empfangsvorgang für ein Datenwort gestartet. Um sicherzugehen, daß kein Störsignal vorliegt, wird nach der Zeit, die einem halben Datenbit entspricht, überprüft, ob die Datenleitung am Schnittstellenbaustein immer noch logisch „0" ist. Falls das so ist, läuft der Empfangsvorgang für das vorher programmierte Datenformat ab.

Etwa in der Mitte jedes Datenbits wird die Datenleitung abgetastet und der logische Wert übernommen. Das Ende der Übertragung eines Zeichens markiert das Stopbit. Es legt mit seiner Länge den Mindestabstand von aufeinanderfolgenden Zeichen fest. Entsteht während der Datenübertragung eine Pause, bleibt währenddessen der Stopbit-Pegel erhalten, bis nach dem nächsten Startbit wieder ein Zeichen übertragen wird.

So funktioniert der Bildaufbau

Den Konstrukteuren des ZX 81 ist bei seiner Entwicklung ein kleines Kunststück gelungen. Sie brachten es fertig, für die Erzeugung des Fernsehbildes nicht einen Videocontroller oder eine speziell entwickelte Videohardware einzusetzen, sondern die Bilderzeugung im wesentlichen der Z80-CPU aufzubürden. Das hat zur Folge, daß ein nicht unerheblicher Teil der CPU-Zeit dem Benutzer entzogen wird. Aus diesem Grund gibt es die beiden ZX-81-Betriebsarten FAST und SLOW, durch die man selbst entscheiden kann, ob man die volle Rechengeschwindigkeit (FAST) ohne Bilderzeugung oder eine um 75 % (!) reduzierte Geschwindigkeit (SLOW) mit gleichzeitiger Erzeugung eines Fernsehbildes verwenden möchte.

Für das ZX-Terminal ist der SLOW-Modus eingeschaltet, damit die von der seriellen Schnittstelle empfangenen Zeichen auf dem Bildschirm angezeigt werden können.

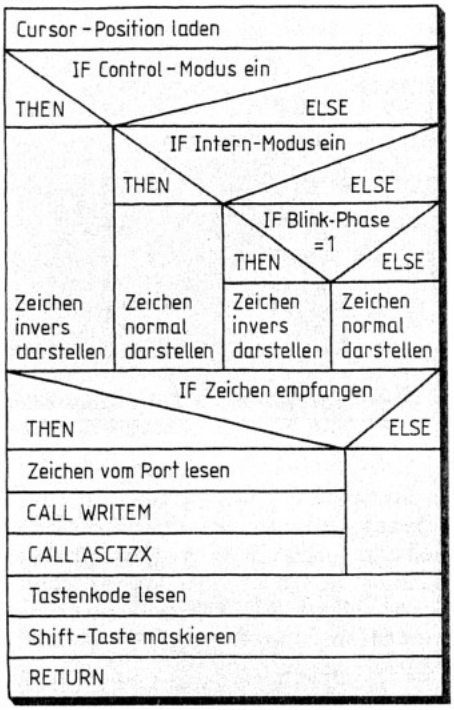
Die Aufgabenverteilung der CPU in einer Bildperiode zeigt das *Bild 3*. Nur in den ober- und unterhalb des Bildbereiches angeordneten Leerzeilenbereichen kann das Benutzerprogramm arbeiten. Neben der eigentlichen Bildinformation werden auch die zum Videosignal gehörenden Synchronimpulse von der CPU zusammen mit dem SCL-Chip erzeugt. Im oberen Bereich wird der Vertikalsynchronimpuls zur Bildsynchronisierung erzeugt und auch die ZX-81-Tastatur nach Benutzereingaben abgefragt. Für die Horizontal-Synchronimpulse wird, wie man erkennen kann, ebenfalls etwas CPU-Zeit benötigt.

Zählt man die von der CPU benötigte Zeit zur Erzeugung des Videosignals zusammen, sieht man, daß für die beiden Leerzeilenbereiche, in denen die Terminalsoftware arbeitet, etwa 5 ms in jeder Bildperiode zur Verfügung stehen. Rechnet man 10 Bit für die Übertragung eines Zeichens und wird die max. auftretende Zeit, in der eine Portabfrage nicht durchgeführt werden kann, berücksichtigt, so ergibt sich die maximale Baudrate des ZX-Terminals:

1/(12,3 ms/10) = 813 Bd

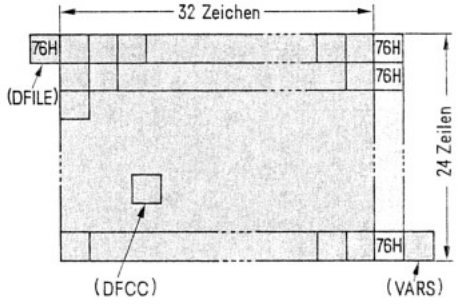
Mit der nächstliegenden genormten Baudrate von 600 Bd kann ein Zeichen dann alle 16,7 ms am Terminal eintref-

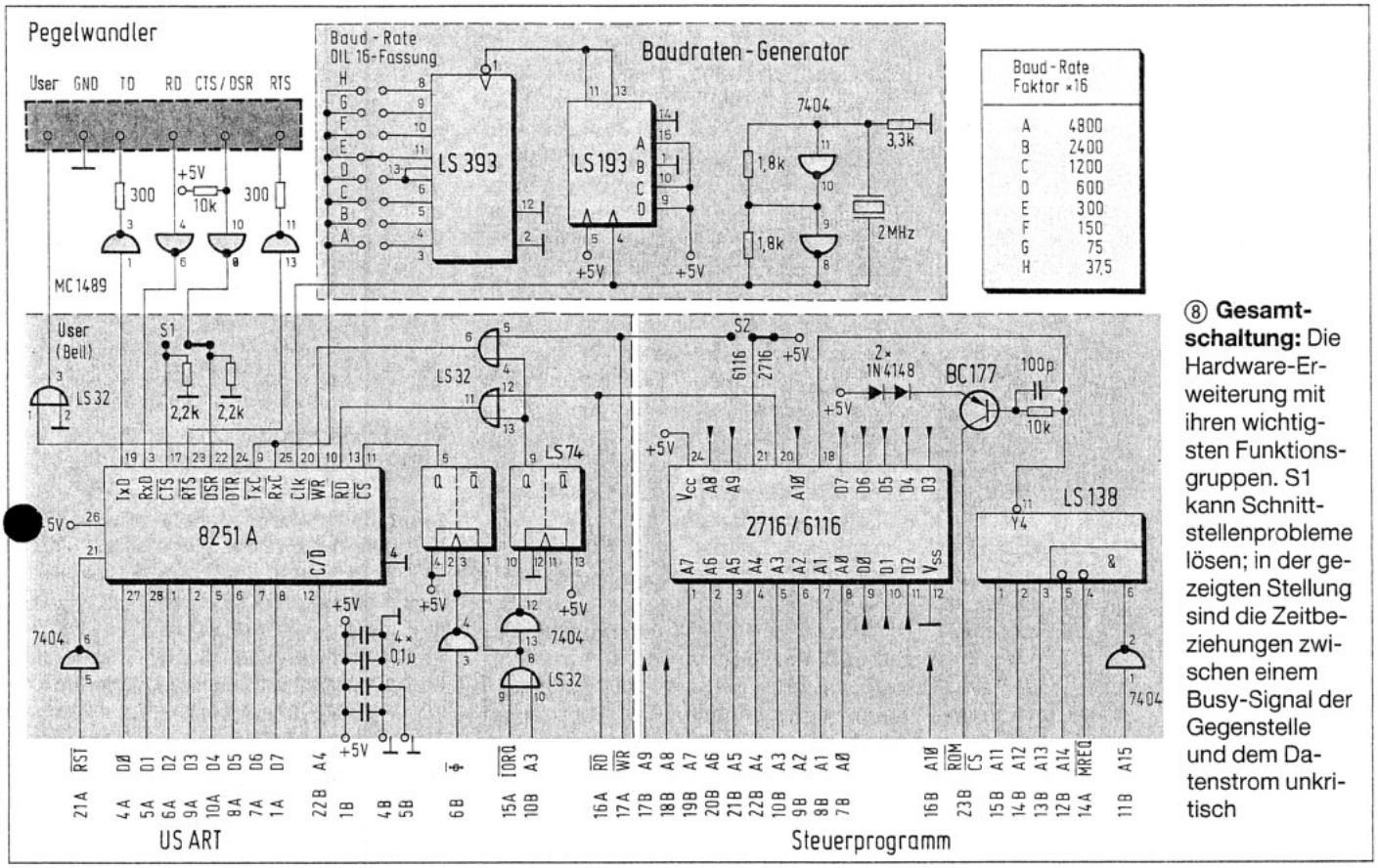
SERIN



⑥ **Unterprogramm SERIN:** Dieses wichtige Unterprogramm fragt die Eingaben ab

⑦ **Bildschirmverwaltung:** Das Terminal-Programm orientiert sich an drei Systemvariablen





⑧ **Gesamtschaltung:** Die Hardware-Erweiterung mit ihren wichtigsten Funktionsgruppen. S1 kann Schnittstellenprobleme lösen; in der gezeigten Stellung sind die Zeitbeziehungen zwischen einem Busy-Signal der Gegenstelle und dem Datenstrom unkritisch

fen. Pro Bildperiode sind damit im Mittel 20 ms/16,7 ms = 1,2 Zeichen zu bearbeiten, dafür stehen die 5 ms zur Verfügung. Im Mittel darf ein Zeichen daher nicht mehr als 5 ms/1,2 = 4,167 ms Bearbeitungszeit beanspruchen. Die mit Abstand längste Verarbeitungszeit t_{max} = 512 ms wird bei der Ausführung eines Bildschirmscrolls erreicht, hier werden 758 Bytes transportiert, zusätzlich wird dann die letzte Zeile gelöscht. Die höchste Verarbeitungsgeschwindigkeit, die auch andauerndes Scrolling einschließt, errechnet sich dann zu:

$$V_{max} = \frac{5\ ms \times Wortformat\ (Bit)}{t_{max} \times Bildperiode}$$

$$V_{max} = \frac{5\ ms \times 10\ Bit}{5{,}2\ ms \times 20\ ms} = 480\ Bd$$

In der Praxis kann man das ZX-Terminal auch mit 600 Bd betreiben. Dies funktioniert, da das Terminal im normalen Betrieb mehr Zeichen empfängt, die nur auf dem Bildschirm darzustellen sind und wesentlich weniger Zeit als das Scrolling benötigen.

Steht der Cursor jedoch in der untersten Zeile und zwingt man das Terminal, mit jedem Zeichen ein Scrolling auszuführen – dies ist nur durch das Steuerzeichen LF (Line Feed, Zeilenverschub) möglich – so geht irgendwann ein Zeichen verloren. Im praktischen Betrieb hat sich gezeigt, daß dies nach etwa 8 LF's der Fall ist, im Normalfall werden jedoch nicht mehrere Bildschirmscrolls direkt hintereinander ausgeführt.

Das Maschinenprogramm

Alle zum Terminal-Betrieb notwendigen Programme wurden modular aufgebaut und in Maschinensprache geschrieben. Die Funktionen werden dabei über eine Sprungleiste erreicht (*Bild* 4), die ganz vorn im EPROM-Bereich liegt. Die Terminal-Funktion wird gestartet mit dem Basic-Aufruf:
RAND USR 8192 (NEWLINE)

Das dann durchlaufene Programm, die Terminal-Hauptschleife (*Bild* 5), ist eine Endlosschleife, d. h. nach Erkennung und Bearbeitung von Ein-/Ausgabeaufgaben wiederholt sich dieses Programm ständig. Passiert gar nichts, so sorgt das Programm lediglich für einen blinkenden Cursor. ZX-Kenner wissen, daß hierfür die Systemvariable FRAMES benutzt wird, die das Interruptprogramm in Abständen von 20 ms dekrementiert.

Wichtig für die Terminal-Funktion und den Voll-Duplex-Betrieb ist, daß die Software an keiner Stelle nur wartet, sondern daß beide möglichen Eingaben, also Tastendruck und Zeichenempfang am Port, ständig geprüft werden.

Die auf eine Eingabe folgende Ausgabe wird über Unterprogramme erledigt. Ein Tastendruck startet das Programm CODKEY (Bild 5). Konnte der Empfang eines Zeichens über die serielle Schnittstelle festgestellt werden, so wird das Pro-

Tabelle 1. Adressenzuordnung des USART

$\overline{IORQ}$	A4	A3	Adresse im I/O-Bereich	USART-Register
0	0	0	E7H	Datenregister
0	0	1	EFH	–
0	1	0	F7H	Crtl-Cmd/Status
0	1	1	FFH	– (ZX-intern)

gramm ASCTZX (*Bild 6*) aufgerufen. Ein Tastendruck hat dann im allgemeinen die Ausgabe eines ASCII-Zeichens zur Folge. Ein empfangenes Zeichen erscheint auf dem Bildschirm, falls es zu den druckbaren gehört. Das ebenfalls aufgerufene Unterprogramm WRITEM legt bei eingeschalteter Download-Funktion das empfangene Zeichen im Speicher ab.

Vom Hauptprogramm wird auch die Autorepeat-Funktion der Tastatur erledigt. Bleibt eine Taste gedrückt, so wird das Zeichen nach einer Verzögerungszeit wiederholt ausgegeben. Je nach Zustand der Autorepeat-Funktion muß stets der Programmteil SERIN (Bild 6) durchlaufen werden. Er ist deshalb als Unterprogramm aufgebaut.

Das Schreiben von Symbolen in den Bildschirmbereich (*Bild 7*) orientiert sich an den Systemvariablen und an der Zeilenendekennung im Bildschirmbereich (76H). Eine vollständige Übersicht bietet das Assemblerlisting des Programms; es ist mit Kommentaren versehen und separat über den Franzis-Softwareservice zu beziehen.

Die Hardware

Die Hardware (*Bild 8*) der seriellen Schnittstelle setzt sich aus zwei unabhängigen Teilen zusammen:

- ○ Das EPROM mit der Adreßdecodierung (wahlweise ist ein RAM einsetzbar)
- ○ Der Schnittstellenbaustein (USART) mit Baudrategenerator, Timinganpassung Z80/8085 und den Treibern/Empfängern für die Schnittstellenverbindung.

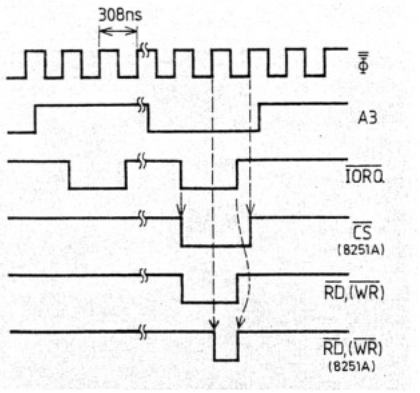
Die Adreßdecodierung für das EPROM/RAM hat die Aufgabe, beim Auftreten von Adressen im Bereich von 2000H(ex) – 27FFH(ex) den Speicherbaustein für den Z80 vorzubereiten. Der Z80 holt sich dann bei aktiver RD-Leitung das von ihm adressierte Byte aus dem Speicher. Gleichzeitig ist – ZX-81-typisch – das ZX-eigene ROM abzuschalten, da dieses ROM zweimal in der unteren Hälfte des Adreßbereichs auftaucht.

Der Baudratengenerator besteht aus dem Oszillator, dem Vorteiler und einem zweifachen 4-Bit-Binärteiler. Der Vorteiler teilt die 2-MHz-Schwingung des Oszillators durch 13 und bietet damit dem Binärteiler die Frequenz von 153 846 Hz an. Für asynchronen Betrieb der Schnittstelle muß der USART die 16- oder 64fache Frequenz der Baudrate erhalten. Bei einem, durch die Initialisierung des USART, wählbaren Teilerfaktor von 16 muß der Binärteiler für 300 Bd die Frequenz 16 × 300 Hz = 4800 Hz liefern. Der Binärteiler muß also auf einen Teilerfaktor von 153 846/4800 ≈ n = 32 eingestellt werden. Der verbleibende Frequenzfehler liegt weit unter dem Wert, bei dem der USART eine Fehlabtastung vornimmt, d. h. falsche Zeichen liefern kann.

Die verwendete Beschaltung der Schnittstelle selbst hat den Vorteil, mit normgerechten Gegenstellen, also etwa einem Modem oder Drucker einwandfrei zusammenzuarbeiten. Es werden keine ±12 V oder weitere Schnittstellentreiber benötigt, und ohne die 300Ω Schutzwiderstände sind die Signale außerdem TTL-kompatibel.

Ebenso wie die Speicherbausteine, müssen die I/O-Bausteine aus einem größeren Adreßbereich einzeln angesprochen werden können. I/O-Bausteine sollten beim ZX 81 nicht in dem Speicherbereich, in dem sich auch ROM und RAM befinden (memory mapped), sondern nur über ein aktives IORQ-Signal angesprochen werden. Im unteren Bereich liegen RAM und ROM. Jede Speicherzelle des oberen 32-KByte-Adreßbereichs wird durch das Interrupt-Programm alle elf Minuten einmal gelesen. Liegen hier Portbausteine, so könnten auf diese Weise Daten gelöscht werden.

Der ZX 81 verwendet zur Ansprache seiner I/O-Adressen eine bitweise Codierung. Es wurden die intern nichtbenutzten Bits A3 und A4 ausgewählt (*Tabelle 1*). Bit A3 wird zur USART-Ansprache benutzt und muß daher für einen Portzugriff immer „0" sein. Bit A4 wird zur Unterscheidung der Register verwendet.



⑨ **Anpassung** des Z80-Timings an den USART

Besondere Aufmerksamkeit mußte der hardwaremäßigen Anpassung zwischen Z80 und 8085-Timing zuteil werden – hier geht es letztlich um Nano-Sekunden. Der USART gehört zur Gruppe der 8085-Bausteine. Den zeitlichen Ablauf der USART-Ansprache zeigt *Bild 9*.

Terminal- und Kommunikationssoftware

Wählt man den Terminal-Einsprung 8192 (RAND USR 8192), so wird die

Tabelle 2. Einige USART-Betriebsarten

Mode-Wort:

Datenbits	Stopbits	Parity	Wert	
7	1	aus	4AH	74
7	2	aus	CAH	202
8	1	aus	4EH	78
8	2	aus	CEH	206
7	1	ein, ungerade	5AH	90
7	1	ein, gerade	7AH	122
7	2	ein, ungerade	DAH	218
7	2	ein, gerade	FAH	250
8	1	ein, ungerade	5EH	94
8	1	ein, gerade	7EH	126
8	2	ein, ungerade	DEH	222
8	2	ein, gerade	FEH	254

Kommando-Wort:

Ausgang $\overline{DTR}$, User	Ausgang RTS	Wert	
1	1	25H	37
1	0	05H	5
0	1	27H	39
0	0	07H	7

Bezeichnung der Ausgänge so, wie für den Benutzer zugänglich

Schnittstellen-Betriebsart mit den TEDAS-Werten (8 Datenbits, 1 Stopbit, Parity aus) eingestellt.

Eine freie Einstellung der Betriebsart kann mit dem zweiten Einsprung 8195 erreicht werden. Vorher muß der gewünschte Wert des Mode-Registers in die RAM-Zelle 16 417, der Wert des Command-Registers in die RAM-Zelle 16 507 gebracht werden. *Tabelle 2* liefert die Werte für die wichtigsten Betriebsarten. Nach Eingabe von

POKE 16417,250
POKE 16507,37
RAND USR 8195

arbeitet der USART z. B. mit 7 Datenbits, 2 Stopbits; Parity ist eingeschaltet und gerade.

Soll nur eine Initialisierung durchgeführt werden, z. B. für den Druckerbetrieb, so ist nach dem Poken der Werte der Aufruf

RAND USR 8201

durchzuführen. Eine ausführliche Beschreibung des Druckerbetriebs folgt im nächsten Heft. Das Kommandowort (KW) kann laufend geändert werden:

POKE 16417,KW
RAND USR 8204

Das Modewort kann nur durch den Aufruf 8201 oder durch den Terminal-Aufruf geändert werden.

Zeichenausgabe und Zeichendarstellung

Die druckbaren ASCII-Zeichen werden nach *Tabelle 3* erreicht. Für einige ASCII-Zeichen bietet der ZX 81 jedoch nicht das entsprechende Symbol. Diese Zeichen werden über die Shift-Taste als Grafik-Symbol erreicht. Soweit möglich wurden ASCII-ähnliche Grafik-Symbole verwendet.

Die Steuerzeichen werden erreicht, indem der Control-Modus durch Shift-Newline (FUNCTION) eingeleitet wird. Dem ZX 81 fehlt eine Control-Taste, daher ist dieser Schritt notwendig. Der Control-Zustand wird dem Benutzer durch den nicht mehr blinkenden Cursor angezeigt. Das nachfolgend gedrückte Zeichen (40H–5FH) wird dann als Control-Zeichen ausgegeben. Durch nochmaliges Shift-Newline kann der Control-Modus ohne Zeichenaussendung verlassen werden. Von der Tastatur können damit alle ASCII-Zeichen von 00H–5FH ausgegeben werden. Die Autorepeat-Funktion arbeitet auch bei den Control-Zeichen. Zwei wichtige Steuerzeichen können zusätzlich direkt erreicht werden:

0DH CR NEWLINE
08H BS Shift 0 (RUBOUT)

Für die Darstellung der empfangenen Zeichen wird das höchste Bit des Zeichens auf Null gesetzt. Zeichen mit den Werten 60H bis 7FH werden durch Rücksetzen des 5. Bits in Großbuchstaben gewandelt.

Von den 32 möglichen Control-Zeichen werden einige als Steuerzeichen nach *Tabelle 4* ausgewertet – unbekann-

```
2000 C3 72 20 C3 7A 20 C3 62 27 C3 C3 20 C3 33 20 C3
2010 39 20 C3 3F 20 C3 45 20 C3 4B 20 C3 63 20 C3 9D
2020 25 C3 A4 25 C3 C6 26 C3 0B 27 C3 6A 23 C3 1F 27
2030 C3 31 27 FD 7E 21 D3 F7 C9 DB F7 FD 77 21 C9 FD
2040 7E 7B D3 E7 C9 DB E7 FD 77 7B C9 CD AB 26 DB F7
2050 CB 47 28 F7 CD AB 26 DB F7 CB 7F 28 F7 FD 7E 7B
2060 D3 E7 C9 CD AB 26 DB F7 CB 4F 28 F7 DB E7 FD 77
2070 7B C9 FD 36 21 4E FD 36 7B 25 E5 CD C3 20 FD 36
2080 21 01 2A 04 40 01 01 44 A7 ED 42 30 06 FD 36 05
2090 80 18 14 2A 04 40 E5 01 06 00 09 7E EE 5A E1 46
20A0 B8 20 04 FD CB 21 F6 E1 CD 2A 0A CD B4 20 CD DF
20B0 20 C3 41 21 3A EE 0E FE 2F 20 04 CD 28 0F C9 CD
20C0 2B 0F C9 3E 00 D3 F7 D3 F7 D3 F7 D3 F7 3E 40 D3
20D0 F7 FD 7E 21 D3 F7 FD 7E 7B FD 77 7C D3 F7 C9 3E
20E0 03 CD FE 20 06 20 21 ED 24 7E C5 E5 CD D7 22 E1
20F0 C1 23 05 20 F4 3E 83 CD FE 20 CD 10 24 C9 ED 5B
2100 0E 40 06 20 12 13 05 C2 04 21 13 ED 53 0E 40 C9
2110 2A 0E 40 FD CB 21 46 28 0C FD CB 21 66 20 0A FD
2120 CB 34 66 28 04 CB FE 18 02 CB BE DB F7 CB 4F 28
2130 08 DB E7 CD 2B 23 CD D7 22 3A 26 40 E6 FE FE FE
2140 C9 CD 10 21 28 FB CD 86 21 3E E0 FD B6 34 FD 77
2150 34 CD 10 21 28 EB FD CB 34 76 20 F5 FD CB 21 9E
2160 FD CB 34 5E 28 04 FD CB 21 DE CD 10 21 28 D2 FD
2170 7E 21 FD AE 34 CB 5F 28 F1 FD CB 21 56 28 DD FD
2180 7E 7B D3 E7 18 D6 FD CB 21 96 ED 4B 25 40 CD BD
2190 07 01 D1 24 09 7E FD CB 21 66 20 3B FE FF C8 FE
21A0 80 28 0D FE 81 20 16 FD CB 21 E6 FD CB 21 C6 C9
21B0 3E 01 FD AE 21 FD 77 21 FD CB 21 A6 C9 FD CB 21
21C0 46 20 0A CB 6F 20 02 E6 1F FD CB 21 C6 D3 E7 FD
21D0 77 7B FD CB 21 D6 C9 FD CB 21 A6 FE 42 CA 04 22
21E0 FE 43 CA 5A 24 FD CB 21 76 C8 FE 52 28 1E FE 45
21F0 28 1F FE 4A 28 20 FE 4E 28 28 FE 4C 28 32 FE 53
2200 CA 89 22 C9 CD 39 21 20 FB C3 B3 26 FD CB 21 EE
2210 C9 FD CB 21 AE C9 2A 04 40 01 0E 00 09 5E 23 56
2220 EB E9 2A 04 40 23 06 00 70 23 70 0E 05 09 70 C9
2230 CD 39 21 20 FB CD F7 23 CD 10 24 CD 10 24 2A 04
2240 40 23 5E 23 56 01 03 00 09 4E 23 46 60 69 7B B2
2250 C8 7E E5 D5 CD D7 22 D1 E1 23 1B CD 39 21 28 EE
2260 CD 7B 22 F5 CD 39 21 20 FB F1 FE 53 CB FE 20 20
2270 DD CD 39 21 28 FB 18 E8 E5 D5 ED 4B 25 40 CD BD
2280 07 01 D1 24 09 7E D1 E1 C9 CD AE 22 20 FB 2A 04
2290 40 01 08 00 09 4E 23 46 23 5E 23 56 EB 79 B0 C8
22A0 CD AE 22 C0 7E D3 E7 CD C4 22 23 0B 18 EF E5 C5
22B0 CD 10 21 C1 E1 C0 DB F7 CB 47 28 F2 DB F7 CB 7F
22C0 28 EC AF C9 FE 0D C0 FD 7E 34 D6 1A FD BE 34 F5
22D0 CD AE 22 F1 20 F6 C9 CB BF 4F E6 60 28 11 CB 71
22E0 28 02 CB A9 AF 47 21 EF 24 09 7E CD AB 23 C9 79
22F0 FE 0D CA F7 23 FE 0A CA 10 24 FE 1C CA 3F 24 FE
2300 09 CA 50 24 FE 08 CA 95 24 FE 0B CA 72 24 FE 0C
2310 CA 5A 24 FE 01 CA B4 24 FE 02 CA B9 24 FE 1B CA
2320 BE 24 FE 07 CA 5D 23 C9 FD CB 21 6E C8 2A 04 40
2330 23 E5 4E 23 46 23 5E 23 56 D5 23 5E 23 56 EB 09
2340 D1 E5 A7 ED 52 E1 37 3F 20 04 01 FF FF 37 77 03
2350 E1 71 23 70 D0 01 05 00 09 06 01 70 C9 3E 02 FD
2360 AE 7C D3 F7 FD 7E 7C D3 F7 C9 2A 04 40 23 5E 23
2370 56 23 4E 23 46 C5 23 4E 23 46 23 7E 69 60 C1 E5
2380 CB 47 28 0C 19 CD A0 23 E5 A7 ED 42 E1 23 20 F5
2390 E1 7B B2 28 07 CD A0 23 23 1B 18 F5 CD B5 26 C9
23A0 7E FD 77 7B CD 4B 20 C9 2A 0E 40 FD CB 21 4E 28
23B0 02 F6 80 77 CD B8 23 C9 23 7E FE 76 20 03 CD C8
23C0 23 22 0E 40 CD CF 24 C9 E5 ED 4B 10 40 0B A7 ED
23D0 42 E1 20 04 CD DA 23 C9 23 C9 2A 0C 40 5D 54 13
23E0 01 22 00 09 01 F6 02 ED B0 2B 16 00 3E 76 BE 28
23F0 04 72 2B 18 F9 23 C9 2A 0E 40 16 00 3E 76 72 23

2400 BE 20 FB 01 20 00 A7 ED 42 CD CF 24 22 0E 40 C9
2410 2A 10 40 01 21 00 A7 ED 42 44 4D 2A 0E 40 CD E0
2420 24 A7 ED 42 F2 35 24 2A 0E 40 01 21 00 09 22 0E
2430 40 CD CF 24 C9 CD DA 23 2A 0E 40 CD CF 24 C9 2A
2440 0E 40 CD E0 24 2A 0C 40 23 22 0E 40 CD CF 24 C9
2450 2A 0E 40 CD E0 24 CD B8 23 C9 2A 10 40 2B 2B 3E
2460 00 0E 18 06 20 77 2B 05 20 FB 2B 0D 20 F5 CD 3F
2470 24 C9 2A 0C 40 01 21 00 09 44 4D 2A 0E 40 A7 ED
2480 42 F8 2A 0E 40 CD E0 24 01 21 00 A7 ED 42 22 0E
2490 40 CD CF 24 C9 ED 4B 0E 40 2A 0C 40 23 A7 ED 42
24A0 C8 60 69 CD E0 24 2B 7E FE 76 20 01 2B 22 0E 40
24B0 CD CF 24 C9 FD CB 21 CE C9 FD CB 21 8E C9 FD CB
24C0 21 76 C8 2A 04 40 01 0C 00 09 5E 23 56 EB E9 CB
24D0 7E 28 06 FD CB 21 FE 18 04 FD CB 21 BE CB FE C9
24E0 FD CB 21 7E 28 04 CB FE 18 02 CB BE C9 3E 3E 20
24F0 5A 58 2D 54 45 52 4D 49 4E 41 4C 20 56 2E 33 2E
2500 30 20 28 43 29 20 31 39 38 34 20 3C 3C 0D 0A 00
2510 05 0B 0C 0D 06 07 01 10 11 17 15 1A 16 1B 18 1C
2520 1D 1E 1F 20 21 22 23 24 25 0E 19 13 14 12 0F 0A
2530 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35
2540 36 37 38 39 3A 3B 3C 3D 3E 3F 04 08 02 03 09 5A
2550 58 43 56 41 53 44 46 47 51 57 45 52 54 31 32 33
2560 34 35 30 39 38 37 36 50 4F 49 55 59 0D 4C 4B 4A
2570 4B 20 2E 4D 4E 42 3A 3B 3F 2F 5C 40 5F 40 5F FF
2580 FF 26 FF 25 27 5D FF 5B 21 08 81 21 5E 5F 22 29
2590 28 24 25 B0 3D 2B 2D 5C 23 2C 3E 3C 2A 21 7D 40
25A0 CD BC 25 C9 2A 0A 40 CD D8 09 20 0F 06 16 C5 CD
25B0 CA 25 C1 7E FE 76 28 03 05 20 F3 C9 7E FE 76 28
25C0 05 CD CA 25 18 F6 CD B5 26 C9 46 23 4E 23 E5 60
25D0 69 CD F8 25 E1 23 23 7E 23 CD B4 07 28 F9 FE 7F
25E0 2B F5 FE 76 28 0E CB 77 28 05 CD 3C 26 18 E8 CD
25F0 35 26 18 E3 CD B5 26 C9 FD CB 21 BE 01 18 FC CD
2600 15 26 01 9C FF CD 15 26 01 F6 FF CD 15 26 7D C6
2610 1C CD 35 26 C9 3E FF A7 09 3C 38 FC ED 42 C6 1C
2620 FE 1C 28 06 FD CB 21 FE 18 07 FD CB 21 7E 20 01
2630 AF CD 66 26 C9 CD 66 26 CD 76 26 C9 F5 CD 75 09
2640 30 04 AF CD 66 26 0A E6 3F CD 66 26 0A 03 87 30
2650 F5 C1 CB 78 28 09 FE 1A 28 04 FE 38 38 04 AF CD
2660 66 26 CD 76 26 C9 E5 F5 21 3D 40 54 5D 2B 7E B3
2670 5F F1 12 34 E1 C9 E5 21 3D 40 54 5D 2B 7E 28 08
2680 1A CD 8A 26 13 35 18 F5 E1 C9 E5 C5 CB 77 20 18
2690 4F CB B9 06 00 21 7F 27 09 CB 7F 28 04 01 40 00
26A0 09 7E FD 77 7B CD 4B 20 C1 E1 C9 3E 7F DB FE 1F
26B0 30 01 C9 CF 0C 3E 0D FD 77 7B CD 4B 20 3E 0A FD
26C0 77 7B CD 4B 20 C9 2A 16 40 23 22 16 40 3A EE 0E
26D0 FE 2F 20 08 CD 52 0F CD F4 13 18 06 CD 55 0F CD
26E0 F8 13 78 B1 28 0B 1A 13 CD 8A 26 0B 18 F4 2A 16
26F0 40 7E FE 76 28 0B FE 00 20 03 23 18 F4 FE 19 2B
2700 03 CD B5 26 22 16 40 C9 06 16 2A 0C 40 23 7E FE
2710 76 28 05 CD 8A 26 18 F5 CD B5 26 05 20 EF C9 CD
2720 53 27 21 09 40 7E FD 77 7B CD 4B 20 CD FC 01 18
2730 F4 CD 53 27 FD 36 14 09 FD 36 15 41 21 09 40 CD
2740 63 20 77 23 EB 2A 14 40 37 ED 52 EB 30 F1 CD B4
2750 20 CF FF 3A EE 0E FE 2F 20 04 CD 20 0F C9 CD 23
2760 0F C9 ED 5B 04 40 21 6F 27 01 10 00 ED B0 C9 3E
2770 00 00 FF 7F 10 64 00 22 00 ED 24 EC 24 3F 24 20
2780 27 5D 5E 5B 21 25 26 5C 5F 40 22 23 24 3A 3F 28
2790 29 3E 3C 3D 2B 2D 2A 2F 3B 2C 2E 30 31 32 33 34
27A0 35 36 37 38 39 41 42 43 44 45 46 47 48 49 4A 4B
27B0 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 20
27C0 60 7D 7E 7B 21 25 26 7C 7F 40 22 23 24 3A 3F 28
27D0 29 3E 3C 3D 2B 2D 2A 2F 3B 2C 2E 30 31 32 33 34
27E0 35 36 37 38 39 61 62 63 64 65 66 67 68 69 6A 6B
27F0 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A
```

⑩ **Hexdump** des Programms; Prüfsumme von 8192 bis 10 238 = 221 546

te Control-Zeichen werden ignoriert. Einheitlich gehandhabt werden bei anderen Terminals nur CR, LF, BS, BELL und vielleicht noch FF (0CH), also Bildschirm löschen.

Von den erwähnten Bitmanipulationen ist die eventuell eingeschaltete Down/Uploadfunktion nicht betroffen. Die Zeichen werden so übernommen, wie sie an der Datenquelle vorgefunden werden. Kleinbuchstaben werden daher auch so abgespeichert. Ebenso können mittels Upload Kleinbuchstaben oder „nicht-ASCII-Zeichen" mit gesetztem höchsten Bit ausgegeben werden.

Intern-Modus und RAM-Vereinbarung

Der Intern-Modus eröffnet eine Reihe von Möglichkeiten, die für die Datenkommunikation wichtig sind. Im wesentlichen lassen sich empfangene Texte im ZX-Speicher ablegen (Download) und vorbereitete Texte über die Schnittstelle ausgeben (Upload). Die meisten der Funktionen im Intern-Modus (IM) lassen sich nur dann benutzen, wenn die RAM-Vereinbarung erfüllt ist. Sie soll daher zunächst erklärt werden.

Erst einmal ist die Reservierung von Speicherplatz notwendig, damit sich Basic-Programme mit den dazugehörigen Variablenfeldern nicht mit den Puffern für Down/Upload überschneiden. Diese Reservierung ist erst bei mehr als 1 KByte Speicher möglich.

Je nachdem, wieviel Speicherplatz man für ein Basic-Programm reservieren möchte, ist also zunächst die Systemvariable RAMTOP auf einen niedrigeren Wert zu setzen. Der Bereich oberhalb von RAMTOP ist dann für Kommunikationszwecke nutzbar. Direkt ab RAMTOP muß die RAM-Vereinbarung angesiedelt werden, die aus 16 Bytes besteht (*Tabelle 5*). Die RAM-Vereinbarung wird beim Start des Terminals immer dann geprüft, wenn der ZX 81 mehr als 1 KByte Speicher besitzt. Die Prüfung, ob die RAM-Vereinbarung eingehalten wurde, erfolgt durch die Verknüpfung zweier Bytes. Passen sie zusammen, wird die gesamte Vereinbarung anerkannt.

Der Benutzer möchte z. B. Speicherplatz ab 5800H, also ab 22528 für Download benutzen. Das höchste Byte dieses Wertes, also 58H, muß in der RAM-Vereinbarung in der Adresse RAMTOP+6 abgelegt werden. Im ersten Byte der RAM-Vereinbarung liegt ein Kontroll-Byte, das das Ergebnis aus der Ex-Or Verknüpfung des variablen Wertes aus RAMTOP+6 und dem konstanten Wert 5AH darstellt. Das Kontroll-Byte errechnet sich zu:

Tabelle 3. Ausgabe und Darstellung der ASCII-Zeichen im Terminalbetrieb

Einige Zeichen wurden durch Grafiksymbole (GS) ersetzt.

Code	Symbol	ZX-81-Tastatur	
20H		SPACE	
21H	!	SHIFT 5, SHIFT 8	GS 5
22H	"	SHIFT P	
23H	£ #	SHIFT SPACE	
24H	$	SHIFT U	
25H	%	SHIFT T, SHIFT Y	GS 6
26H	&	SHIFT E	GS 7
27H	'	SHIFT 1	GS 1
28H	(	SHIFT I	
29H	)	SHIFT 0	
2AH	*	SHIFT B	
2BH	+	SHIFT K	
2CH	,	SHIFT .	
2DH	−	SHIFT J	
2EH	.	.	
2FH	/	SHIFT V	
30H	0	0	
'	'	'	
'	'	'	
39H	9	9	
3AH	:	SHIFT Z	
3BH	;	SHIFT X	
3CH	<	SHIFT N	
3DH	=	SHIFT L	
3EH	>	SHIFT M	
3FH	?	SHIFT C	
40H	§ @	SHIFT S, SHIFT F	GS 10
41H	A	A	
'	'	'	
'	'	'	
5AH	Z	Z	
5BH	[Ä	SHIFT 4	GS 4
5CH	\ Ö	SHIFT A, SHIFT H	GS 8
5DH	] Ü	SHIFT 2	GS 2
5EH	^	SHIFT 7	GS 3
5FH	_	SHIFT D, SHIFT G	GS 9

Tabelle 4. Ausgewertete Steuerzeichen in Empfangsrichtung

0DH CR	∧M	Rest der Zeile löschen, Cursor an den Zeilenanfang
0AH LF	∧J	Cursor eine Zeile nach unten, in der untersten Zeile wird der Bildschirminhalt eine Zeile nach oben geschoben, die unterste Zeile gelöscht
08H BS	∧H	Cursor ein Zeichen nach links, ist der Zeilenanfang erreicht, wird der Cursor auf das nächsthöhere Zeilenende gesetzt. Am Anfang der obersten Zeile bleibt der Cursor stehen
07H BELL	∧G	Am „User"-Ausgang ($\overline{DTR}$) erscheint ein Impuls, je nach Portinitialisierung pos. oder neg., benutzbar zur Ansteuerung eines „Piepers"
1CH	∧Ö*	Der Cursor wird an den Anfang der obersten Zeile gesetzt
0CH	∧L	Der Bildschirm wird vollständig gelöscht; Cursor an Bildanfang
09H	∧I	Der Cursor wird um ein Zeichen nach rechts bewegt, am Zeilenende erscheint der Cursor am Anfang der folgenden unteren Zeile. Am Bildende wird der Bildschirm „gescrollt"
0BH	∧K	Der Cursor wird um eine Zeile nach oben bewegt ohne die Spaltenposition zu verändern. In der obersten Zeile bleibt der Cursor stehen
01H	∧A	Invertierte Darstellung der folgenden Zeichen
02H	∧B	Normale Darstellung der folgenden Zeichen
1BH ESC	∧Ä*	Bei gesetzter RAM-Vereinbarung wird ein Sprung auf eine Benutzer-Routine durchgeführt, ohne RAM-Vereinbarung wird das Zeichen ignoriert
*	siehe auch Tabelle 3, bei allen Control-Zeichen sind die Bits 6 und 5 auf Null gesetzt	

```
- 5AH = 01011010
  58H = 01011000
        00000010 = 02H
```

Dieser Wert muß also in die Speicherzelle eingesetzt werden, auf die RAMTOP zeigt. Die RAM-Vereinbarung wird sonst nicht anerkannt und die Möglichkeiten des Intern-Modus blieben ungenutzt. Das Kontroll-Byte ist lediglich ein Test, ob eine sinnvolle RAM-Vereinbarung existiert – da falsche Werte zum „Absturz" des Rechners führen. Der kleinste Wert für RAMTOP ist 4E00H. Wichtig ist auch, daß der Bereich RAMTOP–RAMTOP+15 nicht etwa durch Puffer überschrieben werden kann. Die unterste mögliche Pufferadresse darf erst ab RAMTOP+16 beginnen.

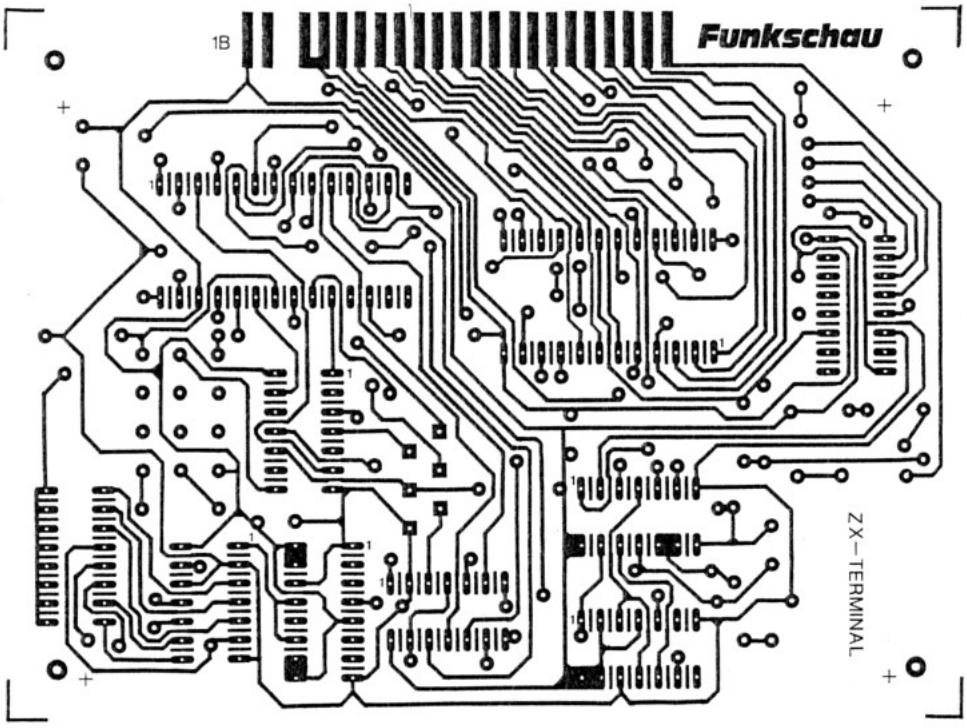
Wem das Setzen der RAM-Vereinbarung zu kompliziert ist, der kann durch den Aufruf RAND USR 8198 eine feste RAM-Vereinbarung (*Tabelle 5*) vorgeben, die nach dem Aufruf natürlich auch veränderbar ist. Beim Ändern der Startadresse des Download-Puffers muß aber immer das Kontroll-Byte nach dem beschriebenen Muster mitgeändert werden.

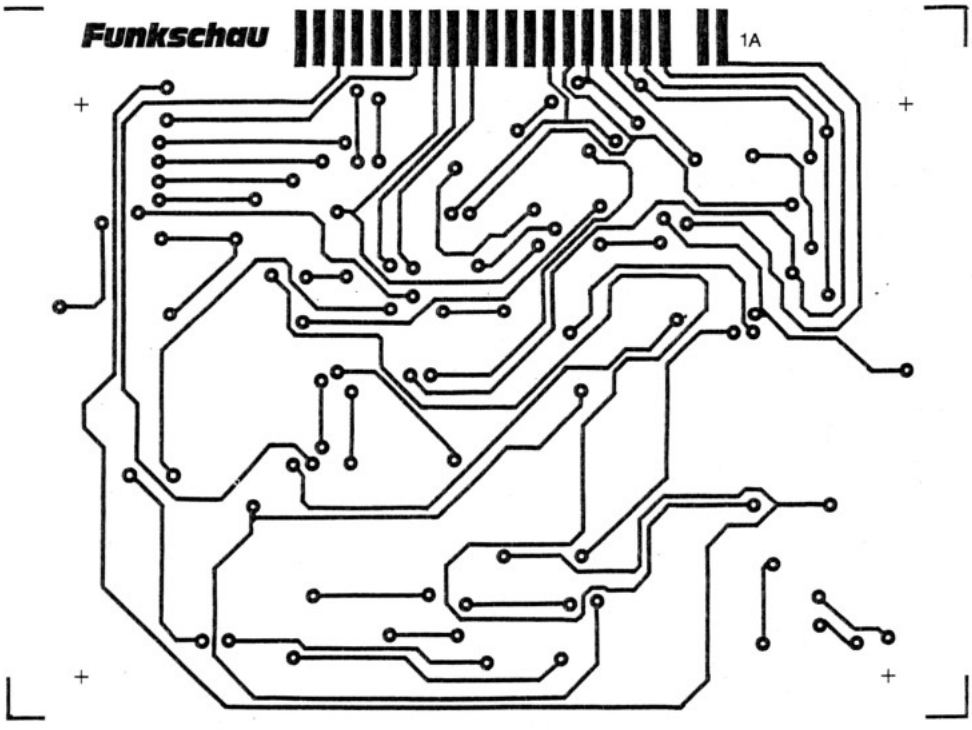
Ein Start des Terminals mit der Reservierung von Speicherplatz sieht dann z. B. so aus:

```
POKE 16388,0      ;RAMTOP auf 25600
                   setzen
POKE 16389,100
NEW
RAND USR 8198     ;feste RAM-Vereinba-
                   rung setzen
RAND USR 8192     ;Terminal aufrufen
```

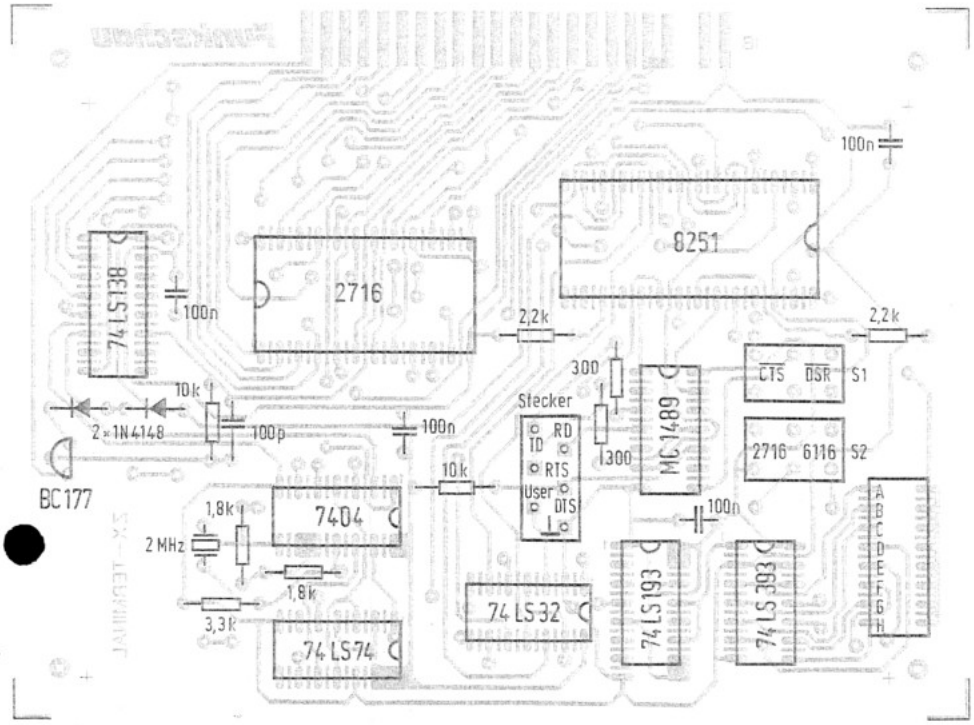
Meldet sich das Terminal, kann der Intern-Modus mit Shift 9 (GRAPHICS) aufgerufen werden. Das danach gedrückte Zeichen wählt die Funktionen nach *Tabelle 6* aus. Der Zustand „Intern-Modus" wird dabei durch den abgeschalteten Cursor angezeigt.

Download ist nach jedem Starten des Terminals ausgeschaltet und muß bei Bedarf durch die IM-Funktion „R" eingeschaltet werden. Der Download-Puffer wird als Ringpuffer verwaltet: Sobald er voll ist, werden die ältesten Daten überschrieben. Mit dem Basic-Aufruf RAND USR 8234 kann der Inhalt des Puffers dann über die serielle Schnittstelle, z. B. an einen Drucker, ausgegeben werden. Die Ausgabe-Routine berücksichtigt dabei einen eventuellen Pufferüberlauf, d. h. die Daten werden in der richtigen Reihenfolge ausgegeben. Auch ohne erfüllte RAM-Vereinbarung sind zwei Funktionen nutzbar, Rückkehr zum Basic (IM-B) und Löschen des Bildschirms (IM-C).





Für den Anschluß eines Modems muß man sich ein Kabel so anfertigen, daß mindestens drei Verbindungen mit dem 25poligen Stecker hergestellt werden: TD an 2, RD an 3 und GND an 7. Ist das Terminal fertig aufgebaut, dann lassen sich alle Möglichkeiten auch ohne ein angeschlossenes Gerät einfach testen. Man macht sich die Eigenschaft „voll-Duplex" zunutze und verbindet die Da-



tensendeleitung TD mit der Empfangsleitung RD, am D-Stecker sind dies die Pins 2 und 3. Getippte Symbole müssen sogleich auf dem Bildschirm erscheinen.

Beim Betrieb des ZX 81 macht sich die instabile mechanische Konstruktion nachteilig bemerkbar; auch die lediglich verzinnte und nicht vergoldete Anschlußleiste spielt eine Rolle. Eine Verbesserung läßt sich dadurch erreichen, daß der ZX 81, zusammen mit Interfaceplatine, RAM-Erweiterung und Netzteil, auf eine Platte montiert wird.

Weitere Probleme, wie Spannungsversorgung der Speichererweiterung und Wärmestau im Gehäuse, lassen sich mit einigem Aufwand ebenfalls lösen.

Friedrich Bollow/Uwe Clausen

Tabelle 5. Aufbau der RAM-Vereinbarung

RAMTOP ist eine Systemvariable, der Wert zeigt auf das erste Byte des oberen Speichers, das vom Basic unberührt bleibt

RAMTOP–)	Byte	;Kontroll-Byte
RAMTOP+1	Byte low	;Länge Download-Puffer, wird vom Terminal-
RAMTOP+2	Byte high	;prog. bei jedem Zeichen inkrementiert
RAMTOP+3	Byte low	;Ende-Adr. des Download-Puffers
RAMTOP+4	Byte high	
RAMTOP+5	Byte low	;Anfangsadr. des Download-Puffers
RAMTOP+6	Byte high	
RAMTOP+7	Byte	;Puffer-voll Flag
RAMTOP+8	Byte low	;Länge des Upload-Puffers
RAMTOP+9	Byte high	
RAMTOP+10	Byte low	;Anfangsadr. des Upload-Puffers
RAMTOP+11	Byte high	
RAMTOP+12	Byte low	;Unterprogrammadresse, wird bei einem empf.
RAMTOP+13	Byte high	;ESC-Zeichen angesprungen
RAMTOP+14	Byte low	;Unterprogrammadresse, wird durch die
RAMTOP+15	Byte high	;IM-J Funktion aufgerufen

Mit dem Aufruf RAND USR 8198 erhält man folgende RAM-Vereinbarung:

RAMTOP	3EH	Kontroll-Byte	
+1	00H	Länge der empf. Zeichen	,0
+2	00H		
+3	FFH	Ende-Adr. Download	,32767
+4	7FH		
+5	10H	Start-Adr. Download	,25616
+6	64H		
+7	00H	Flag	
+8	22H	Länge der Upload-Zeichen	,34
+9	00H		
+10	EDH	Start-Adr. Upload-Puffer	,9453
+11	24H	(String im EPROM)	
+12	ECH	Escape-Routine, zeigt auf ein Return	
+13	24H		
+14	3FH	IM-J Routine, zeigt auf Cursor-Home-Routine im EPROM	
+15	24H		

Tabelle 6.
Funktionen im Intern-Modus (IM)

Der Intern-Modus wird durch Shift 9 (GRAPHICS) eingeleitet

IM -B Basic,	zurück zum Basic, das Basic-Programm wurde ebenso wie die Variablen nicht gelöscht
IM -C Clear,	Bildschirm löschen, Cursor home
IM -R Receive,	empfangene Zeichen werden im Download-Speicher abgelegt
IM -E End,	Zeichen nicht mehr im Download-Speicher ablegen
IM -S Send,	die Zeichen des Upload-Puffers werden ausgegeben, nach einem CR folgt 0,5 s Pause
IM -N New,	Rücksetzen des Download-Speichers, Länge = 0, Puffervoll Flag = 0
IM -J Jump,	Sprung zur eigenen Maschinen-Routine, die Routine muß mit einem Return beendet werden
IM -L List,	Anzeige der empfangenen Zeichen des Download-Puffers, Pufferüberlauf wird nicht berücksichtigt, also nur die vorhandene Länge dargestellt, Abbruch der Ausgabe mit „S", Anhalten und Warten mit „Space", weiter und kurzes Anhalten mit jeder anderen Taste. Während des Listens wird ein evtl. Empfang von Zeichen nicht berücksichtigt.

ZX 81 à la carte

Drucksache

ZX 81 akzeptiert übliche Drucker

Einer der Hauptgründe, die den ZX 81 bei vielen in Mißkredit gebracht haben, ist seine Unfähigkeit, handelsübliche Drucker mit serieller Schnittstelle anzusteuern. Das im vorangegangenen Heft vorgestellte ZX-81-Terminal räumt jetzt auch mit dieser „Macke" auf – ohne zusätzlichen Aufwand.

Der ZX 81 kennt die Druckbefehle LLIST, LPRINT und COPY. Möchte man das ZX-81-ROM unangetastet lassen, so können diese Befehle lediglich mit den speziellen ZX-Druckern angewendet werden. Dennoch sind durch den Aufruf von Maschinenroutinen gleichwertige Druckfunktionen mit einem „normalen" Drucker möglich; die notwendige Codewandlung vom ZX-81-Code in ASCII-Zeichen wird dabei gleich mit erledigt.

Drucker und ZX 81 müssen sich erst anfreunden

Vor dem Aufruf der Routinen muß einmal der Schnittstellenbaustein gemäß Tabelle 2 (siehe Heft 23, Seite 75) initialisiert werden. Anschließend ist der Drucker auf die gleiche Baudrate und das gleiche Datenformat einzustellen.

Anders als bei einer Modemverbindung kann der Drucker den Datenstrom über eine eigene Leitung anhalten, da meist die Druckgeschwindigkeit kleiner als die Übertragungsgeschwindigkeit ist. Der Umgang mit den z. B. auf diese Art entstandenen Datengruppen wird über Vereinbarungen geregelt, die allgemein „Kommunikationsprotokoll" oder einfach „Protokoll" genannt werden. Häufig lassen sich Drucker auf verschiedene Protokolle einstellen, es ist in unserem Fall dann das Ready/Busy- oder RTS/CTS-Protokoll einzustellen.

Ein eigenes Thema ist die Steckerbeschaltung für serielle Datenverbindungen, denn genormt ist nur die Verbindung zwischen Terminal und Modem. Bei Rechnern findet man normalerweise folgende Beschaltung der 25poligen D-Buchse:

Pin 2	TD	(Ausgang)
Pin 3	RD	(Eingang)
Pin 4	RTS	(Ausgang)
Pin 5	CTS	(Eingang)
Pin 7	Gnd	(Masse)

Das Ready/Busy-Signal an der D-Buchse des Druckers kann, wenn die Belegung nicht bekannt ist, durch die meist vorhandene ON-LINE oder SELECT-Taste ausfindig gemacht werden. Für „ON-LINE" muß der Ausgang eine positive Spannung, für „OFF-LINE" eine negative Spannung liefern. Dieser Ausgang muß mit dem CTS-Eingang der Terminal-Schnittstelle verbunden werden. Auf der Platine kann dieses Signal auch auf den $\overline{\text{DSR}}$-Eingang des 8251A gelegt werden; in diesem Fall sind die Zeitbeziehungen zwischen dem Busy-Signal und dem Datenstrom unkritisch. Die Software berücksichtigt beide Beschaltungsmöglichkeiten. Natürlich muß auch die Verbindung vom Datenausgang (TD) der Schnittstelle zum Dateneingang (RD) des Druckers und die Massenverbindung (immer Pin 7) hergestellt werden.

Bei einem Zeilenende gibt der ZX 81 immer CR/LF aus; auch darauf muß der Drucker eingestellt werden. Und noch ein Hinweis: Eine Druckausgabe läßt sich immer durch die BREAK-Taste beenden, eine Überwachung der Zeilenlänge auf einen bestimmten Wert wird von der Treibersoftware jedoch nicht vorgenommen.

So werden Listings und Texte gedruckt

Jetzt wird es spannend. Wenn man glaubt, alle Verbindungen zwischen der Terminal-Schnittstelle richtig gezogen zu haben und das Protokoll stimmt, dann darf man nun irgendein kurzes Programm am ZX 81 eintippen. Wird danach mit

```
RAND USR 8222
```

die Routine aufgerufen, die den LLIST-Befehl simuliert, dann muß der Drucker das Listing des Programms ausgeben. Wenn nicht, dann stimmt bei der Verdrahtung etwas nicht oder es wurden bei der Initialisierung der Schnittstelle Fehler begangen. Kein Fehler ist es jedoch, wenn Programmzeilen, die am Bildschirm mehrere Zeilen in Anspruch nehmen, vom Drucker als eine Zeile ausgegeben werden.

Soll ein Listing nur auszugsweise (höchstens 22 Zeilen) gedruckt werden, dann ist dies mit dem Maschinencode-Aufruf

```
RAND USR 8225
```

möglich. Dabei ist die Zeilennummer der ersten zu druckenden Programmzeile mit LIST n der Ausgabe-Routine zu übergeben (für n ist die Nummer der gewünschten Programmzeile einzugeben). Die Nummer muß tatsächlich vorhanden sein, sonst erfolgt keine Ausgabe.

Die Maschinencode-Routine mit Sprung-Adresse 8228 ist für die Ausgabe von Zeichenketten und Zahlen zuständig, sie simuliert also den LPRINT-Befehl. Zeichenketten können direkt (Text zwischen Anführungszeichen) oder als Stringvariable definiert (z. B. A$) der Ausgabe-Routine übergeben werden. Zahlenvariable lassen sich durch ein vorangestelltes STR$ ausgeben.

Ein Strichpunkt am Ende einer Druckanweisung unterdrückt das sonst immer ausgesendete CR/LF-Zeichen. Eine folgende Druckanweisung setzt in diesem Fall dort fort, wo die vorangegangene

aufhörte (vergleichbar den PRINT-Befehlen). Zur Klärung einige Beispiele:

```
PRINT CHR$ USR 8228, „TEST"
```

bewirkt die Ausgabe der Zeichenkette TEST auf dem Drucker. Wurde die Zeichenkette TEST zuvor der Stringvariablen A$ zugeordnet (A$ = „Test"), dann ist auch die Ausgabe mit dem Befehl

```
PRINT CHR$ USR 8228, A$
```

zulässig. Eine Zahlenvariable (z. B. PI) wird so ausgegeben:

```
PRINT CHR$ USR 8228, STR$ PI
```

Der alleinige Wagenrücklauf mit Zeilenvorschub wird durch Aussenden der CR/LF-Zeichen ausgelöst. Dies bewirkt der Befehl

```
PRINT CHR$ USR 8228,""
```

Der Ausdruck CHR$ vor jedem USR-Aufruf der Routine mit Sprungadresse 8228 ist nötig, um eine für den Syntax-Test des ZX 81 verträgliche Programmzeilenstruktur zu bekommen.

Eine formatierte Ausgabe am Drucker, also die gezielte Positionierung von Texten, Ziffern und Zahlen ist dann möglich, wenn für die Zwischenräume Leerstrings gleicher Länge ausgegeben werden. Man kann aber auch auf Tabulator-Befehle des Druckers zurückgreifen. Invertierte Zeichen in einer Zeichenkette tauchen beim Ausdruck als Kleinbuchstaben auf; die Ausgabe von Sonderzeichen und Ziffern bleibt jedoch von einer Invertierung unbeeinflußt.

Nicht gedruckt werden die Grafikzeichen des ZX 81. Diese Zeichen werden vielmehr in ASCII-Zeichen umgewandelt, die nützlicher sind und im Zeichenvorrat des ZX 81 fehlen. Die Zuordnung zwischen dem jeweiligen ZX-81-Zeichencode (Grafiksymbole) und den stattdessen ausgedruckten ASCII-Zeichen ist der *Tabelle* zu entnehmen. Sind dort einem ZX-81-Code zwei Symbole gegenübergestellt, so sind das die Zeichen die Drucker mit umschaltbarem Zeichensatz bieten.

Bei den Aufrufen der Routine 8228 taucht der Text, der gedruckt wird, nicht am Bildschirm auf. Dennoch registriert der ZX 81 die Ausgaben auf den Drucker so, als ob sie am Bildschirm stünden. Nach 22 Zeilen ist daher für den Computer der Bildschirm „voll"; er reagiert darauf mit dem Abbruch der Routine und mit der Ausgabe des Fehlercodes 5. Mit dem SCROLL-Befehl läßt sich dieser Zwangsabbruch nach 22 Zeilen vermeiden.

Steuerzeichen für den Drucker übermittelt die Routine ab der Sprungadresse 8216. Dazu ist zuerst der Dezimalcode des Steuerzeichens per POKE-Befehl in die Speicherzelle 16507 zu laden und dann die Routine aufzurufen. Um z. B. den Steuercode ESC (Escape) zu senden, sind folgende Schritte nötig:

```
POKE 16507,27
RAND USR 8216
```

Jetzt wird der COPY-Befehl simuliert

Mit dem COPY-Befehl des ZX 81 wird bei den ZX-Druckern der Inhalt des Bildschirms auf Papier gebannt. Bei der EPROM-Software erfüllt die Routine ab Sprungadresse 8231 den gleichen Zweck. Diese Routine läßt sich zwar auch im Direktmodus aufrufen, da der ZX 81 dann aber erst einmal den Bildinhalt löscht, erhält man nur 22 Leerzeilen. Der Aufruf

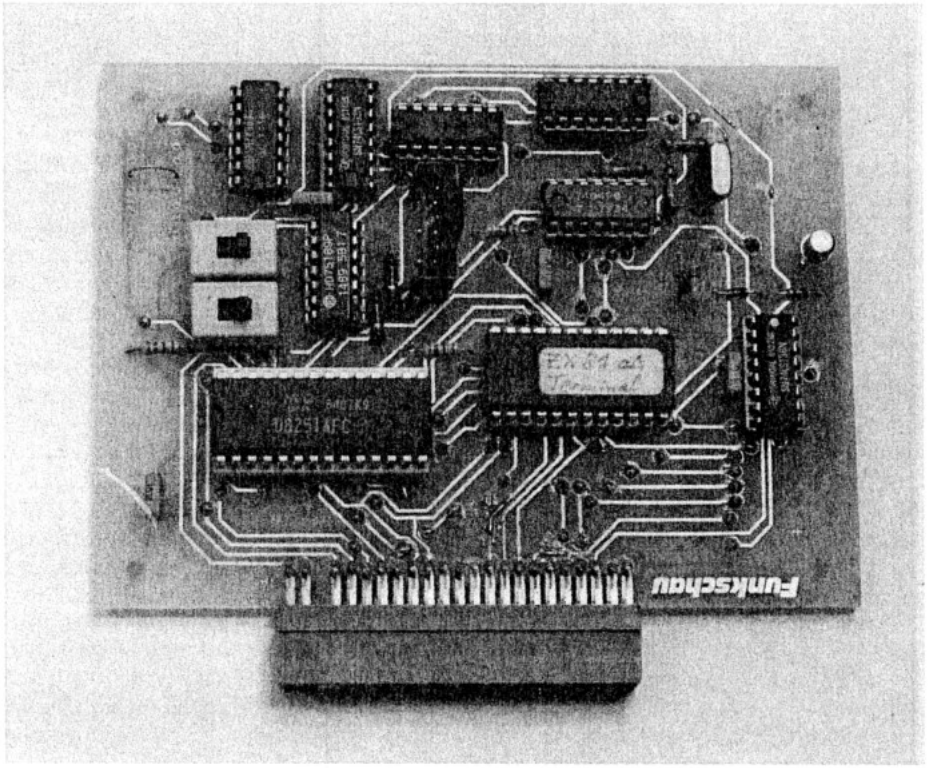
```
RAND USR 8231
```

ist daher immer an den Stellen in ein Programm einzufügen, an denen die Ausgabe einer Bildkopie sinnvoll ist. Dies gilt auch für Maschinenprogramme. Wer z. B. den Disassembler aus dem FUNKSCHAU-Sonderheft 08 (ZX-81-Kochbuch) abgetippt hat, der kann dieses Programm jetzt um die Druckausgabe eines Assembler-Listings bereichern.

Dazu sind nach dem Laden des Disassemblers folgende Dezimalwerte mit POKE-Befehlen ab Adresse 32754 im RAM unterzubringen: 202 72 126 254 53 204 39 32 195 168 126. Durch Drücken der Taste „P" läßt sich nun ein Assembler-Listing Bildseite für Bildseite ausdrucken.

Nützliche Routinen für Tüftler

Wir kennen jetzt die Bedeutung fast aller Routinen aus dem EPROM des ZX-81-Terminals. Wer darüber hinaus noch selber das Leistungsvermögen des Terminals ausloten möchte, der kann dazu auf die restlichen Routinen zugreifen.



Serielle Schnittstelle: Mit dieser Zusatzhardware, die im vorherigen Heft vorgestellt wurde, lassen sich beliebige Drucker ansteuern Foto: Niemeier

Tabelle: Die Software der Terminal-Schnittstelle wandelt ZX-81-Grafikzeichen in ASCII-Zeichen um (Codes 1 bis 10 siehe Tabelle 3 in Heft 23/85, Seite 77)

ZX-81-Code	ASCII-Zeichen
129	'
130	} ü
131	~ ß
132	{ ä
133	!
134	%
135	&
136	| ö
137	DEL
138	@ §

So wartet die Routione mit Sprungadresse 8219 das Eintreffen eines Zeichens an der Terminal-Schnittstelle ab, um dieses Zeichen dann in einer freien RAM-Zelle zur Weiterbearbeitung bereitzuhalten:

```
RAND USR 8219
LET D=PEEK 16507
```

Die Routine mit Sprungadresse 8207 macht dem ZX-81-Kenner den Inhalt des 8251-A-Statusregisters ebenfalls über eine freie RAM-Zelle zugänglich:

```
RAND USR 8207
LET S=PEEK 16417
```

Die Routine mit der Sprungadresse 8210 dient dem Laden des 8251-A-Datenregisters, wogegen die Routine mit der Sprungadresse 8213 den Inhalt des Datenregisters abfragt. Der Inhalt des Statusregisters bleibt dabei unberücksichtigt. Nach der Ausführung dieser kurzen Routinen erfolgt sofort ein Rücksprung.

Schließlich gibt es noch die Routinen mit den Sprungadressen 8237 und 8240: Sie ermöglichen die Ausgabe bzw. das Laden von Basic-Programmen über die Terminal-Schnittstelle. Dabei erfolgt keine Codeumwandlung, so daß stets ein 8-Bit-Datenformat zu wählen ist. Diese Routinen erwiesen sich als zweckmäßig, um Programme schnell auf Diskette zu speichern bzw. von ihr zu laden. Auch zwei ZX-81-Terminals können mit Hilfe dieser Routinen ohne zusätzliche Hardware einen schnellen Programmaustausch bewerkstelligen.

Nun noch zwei Tips für die Kombination von Basic- und Terminal-Software-.Das Terminal-EPROM enthält sämtliche Tabellen zur wechselseitigen Umformung von ASCII- und ZX-81-Zeichen. Diese Tabellen lassen sich gut zur Codeumwandlung verwenden. Dazu werden die Programmzeilen

```
LET D=PEEK(DS+C)
GOSUB....
LET A$=A$+CHR$(PEEK(9487+D-32))
```

in eine FOR-NEXT-Schleife gepackt, die mit der Laufvariablen C den ASCII-Text (oberhalb von RAMTOP) abfragt. Ein Unterprogramm muß das Ende eines Strings am Wert 13 (CR) erkennen und auch für die Behandlung der restlichen Steuerzeichen (‹32) und der Kleinbuchstaben (›95) sorgen. Die Variable DS sowie die Länge lassen sich aus der RAM-Vereinbarung entnehmen, beachtet werden muß die Art der Speicherverwaltung (siehe Beschreibung der RAM-Vereinbarung).

Um einen ZX-81-String auszugeben (Upload), ist vorher wieder die wesentliche Programmzeile in eine Schleife zu legen:

```
POKE UL+C, PEEK(10111+CODE A$(C TO C))
```

In den Textspeicher (Startadresse UL aus RAM-Vereinbarung) ist nach einem String-Ende noch der Wert 13 (CR) und evtl. 10 (LF) zu setzen, außerdem ist die Gesamtlänge in die RAM-Vereinbarung einzutragen.

Friedrich Bollow/Uwe Clausen

(K)ein ZX-81-Hardwaretip:

CMOS-CPU führte zum Tiefschlaf

Die Meldung war höchst vielversprechend: Eine Vertriebsfirma für elektronische Bauelemente informierte uns, daß sie eine CMOS-Version der Z-80-CPU im Programm habe, die zur üblichen Z-80-CPU voll kompatibel sei. Das wäre genau das richtige, um den ZX 81 auf Batteriebetrieb umzurüsten, sagten wir uns und bestellten ein Muster. Die Lieferung traf postwendend ein: Sachkundig in leitendem Schaumstoff verpackt erhielten wir unsere CMOS-CPU vom Typ µPD 70008 für 4 MHz Taktfrequenz (Hersteller: NEC Electronics).

Die Gehäusemaße stimmten ebenso wie die Pinbelegung mit der üblichen CPU überein, nur machte der ZX 81 nach dem Austausch keinen Mucks mehr. Eine Kontrolle der Signale auf dem Adreß- und Datenbus zeigte, warum der Computer nicht Erwachen wollte: Statt munter Bytes hin und her zu transportieren, zeigten beide Busse nur eine statische Pegelverteilung – nichts rührte sich, obwohl der Taktoszillator brav schwang. Schade, denn damit hat der Rückgang der Gesamtstromaufnahme um immerhin 60 mA keinen Wert.

ZX-81-Softwaretip:

Eingangstor für Fremdsignale

Eine Ein-/Ausgabe-Schnittstelle befähigt den ZX 81 zu eindrucksvollen Taten. In der FUNKSCHAU wurde bereits mehrfach darüber berichtet. Allerdings setzt der Selbstbau so einer Schnittstelle Erfahrung voraus. Wer die nicht hat und sich mit der Eingabe von Daten begnügt, der kann die EAR-Buchse des ZX 81 als Eingangstor für Fremdsignale nutzen.

Da der EAR-Eingang vom Kassetteninterface durch einen Kondensator getrennt ist, können allerdings keine statischen Signale (Gleichspannung) abgefragt werden. Es muß ein Pegelsprung vorliegen, der unmittelbar am Eingang des Kassetteninterfaces (Pin 20 des Logik-Chips) zu einem Spannungswert von etwa −1 V führt. Ohne Signal am EAR-Eingang führt Pin 20 dagegen nur etwa +0,7 V. Das Kassetteninterface des ZX 81 reagiert also auf die negative Flanke des Eingangssignals.

Abfragen läßt sich der Pegel am EAR-Eingang mit dem Z-80-Befehl *in a,(FEh);* FEh ist dabei die Adresse des EAR-Eingangs. Das Ergebnis der Abfrage wird im Akku abgelegt: Ohne Signal hat der Akkuinhalt den Wert 127, mit Signal den Wert 255. Das Eingangssignal setzt also lediglich Bit 7 des Akkus. Diese Information kann jetzt nach Belieben weiterverarbeitet werden. Für erste Versuche eignet sich als Signalquelle z. B. der übliche Datenrecorder. Erwin Friese

Softwaretip:

Radiergummi für Basic-Listings

DELETE-Befehl für Zeilenblöcke

Zur Verzweiflung treiben kann einen der ZX 81, wenn ein umfangreicher Zeilenblock aus einem Listing getilgt werden soll: Normalerweise ist dann jede Zeile einzeln zu löschen. Mit einem kurzen Hilfsprogramm kann man sich diese Mühe sparen.

Bessere Heimcomputer kennen einen Befehl, von dem ZX-81-Besitzer bislang nur träumen konnten: den DELETE-Befehl. Er verlangt normalerweise die Eingabe der ersten und letzten Zeilennummer eines zu löschenden Programmzeilenblocks. Dann noch ein Druck auf ENTER oder RETURN (beim ZX 81: NEWLINE) und der überflüssige Zeilenblock ist spurlos aus dem Listing verschwunden. Fast so elegant geht dies mit dem hier vorgestellten Hilfsprogramm jetzt auch beim ZX 81.

Ein USR-Aufruf ersetzt den DELETE-Befehl

Halten wir uns nicht weiter lange mit Vorreden auf, sondern gehen wir gleich in die Praxis. Dazu sollten Sie freilich kein totaler Anfänger auf dem ZX 81 mehr sein, denn es geht um die Eingabe eines Maschinenprogramms, die Sie aber gewiß beherrschen. Erster Schritt ist also das Laden irgendeines Dezimal-Eingabeprogramms und das übliche Reservieren von Speicherplatz in REM-Zeile Eins. Besitzer des ZX-81-Kochbuchs I können zum Erstellen der diesmal 243 Zeichen langen REM-Zeile den „Raumgestalter für Maschinencodes" werkeln lassen. Alle anderen müssen die Zeichen wohl oder übel einzeln von Hand eintippen.

Nächster Schritt ist die Eingabe der in *Bild 1* gezeigten Dezimalcodes des Maschinenprogramms. Startadresse ist wie gewöhnlich die Adresse 16514. Sind die Codes fehlerfrei eingetippt und in die REM-Zeile übertragen worden, sollte noch vor dem ersten Start des Maschinenprogramms, das Eingabeprogramm samt Maschinencodes auf Band gespeichert werden. Sie wissen schon warum!

Jetzt können wir einen Testlauf wagen und probieren, ob sich das nunmehr überflüssige Eingabeprogramm löschen läßt – mit Hilfe des Maschinenprogramms auf einen Schlag. Dazu ist das Maschinenprogramm folgendermaßen aufzurufen:

PRINT STR$ USR 16514="Z1-Z2"

Anstelle von Z1 und Z2 sind dabei die Anfangs- und End-Zeilennummer des Eingabeprogramms einzugeben. Vorsicht: Im Eifer des Gefechts darf selbstverständlich nicht die REM-Zeile mit dem Maschinenprogramm gelöscht werden. Nach Drücken von NEWLINE sollte schlagartig das Eingabeprogramm vom Bildschirm und aus dem Programmspeicher verschwunden sein.

Meldet sich der ZX 81 jedoch mit der Fehlermeldung „F", so wurde die Anfangs- und End-Zeilennummer nicht korrekt eingegeben. Um risikoreiche „Programmabstürze" zu verhindern, prüft das Maschinenprogramm die Zahleneingaben. Buchstaben werden z. B. abgewiesen. Nur eines darf man nicht ungestraft: Das DELETE-Programm innerhalb eines Basic-Programms aufrufen. Erlaubt ist nur der Direkt-Aufruf (ohne vorangestellte Zeilennummer). Wenn soweit alles geklappt hat, können wir darangehen, eine realistische Nutzung des Programms ins Auge zu fassen.

Ein sicherer Platz für den Radiergummi

Hilfsprogramme für den ZX 81 sind zwar gut und schön, sie haben aber den Nachteil, daß sie – wenn man nicht glücklicher Besitzer eines batteriegestützten RAMs oder eine EPROMs ist – nach jedem Einschalten des ZX 81 stets aufs Neue geladen werden müssen. Und dies nur, um irgendeinen Befehl zu simulieren, den andere Heimcomputer so selbstverständlich wie etwa PRINT bereithalten.

Für das DELETE-Programm sollten daher zwei Regeln beachtet werden, um die lästige „Laderei" auf ein Minimum zu begrenzen.

○ Wird ein neues Basic-Programm geschrieben, so ist zuvor das DELETE-Programm (am besten Version mit bereits gelöschtem Eingabeprogramm) zu laden.

```
16514:  231 205 85  15  205 248
16520:  12  33  60  64  235 121
16526:  254 10  48  58  183 40
16532:  55  237 176 62  12  16
16538:  33  60  64  17  0   0
16544:  6   10  126 35  254 22
16550:  32  1   28  254 12  32
16556:  1   20  16  242 123 254
16562:  1   32  23  122 254 1
16568:  32  18  33  60  64  6
16574:  0   126 254 28  56  10
16580:  254 38  48  4   4   35
16586:  24  243 207 14  254 12
16592:  40  9   254 22  62  1
16598:  50  70  64  32  241 120
16604:  183 40  237 254 5   48
16610:  233 126 35  254 22  40
16616:  212 58  70  64  254 1
16622:  32  220 42  22  64  229
16628:  33  60  64  34  22  64
16634:  223 205 72  21  205 138
16640:  21  197 225 205 216 9
16646:  32  196 34  71  64  231
16652:  205 72  21  205 138 21
16658:  197 225 205 216 9   32
16664:  179 35  35  78  35  70
16670:  43  43  43  235 225 34
16676:  22  64  42  71  64  235
16682:  237 82  56  158 237 74
16688:  235 35  35  115 35  114
16694:  42  20  64  1   6   0
16700:  229 205 158 9   209 33
16706:  60  64  126 254 22  40
16712:  5   18  35  19  24  246
16718:  235 237 91  22  64  27
16724:  27  54  0   35  229 237
16730:  82  225 32  247 54  127
16736:  195 12  6   33  31  127
16742:  17  130 64  34  4   64
16748:  235 1   225 0   237 176
16754:  195 229 3
```

① **Maschinencode-Listing:** Mit einem Dezimal-Eingabeprogramm müssen diese Codes in eine mindestens 243 Zeichen lange REM-Zeile ab Startadresse 16514 untergebracht werden

Es wird dann jedesmal mit dem nach und nach entstehenden neuen Basic-Programm automatisch auf Band gespeichert.

○ Soll das DELETE-Programm ohne viele Tipperei einem bereits vorhandenem Basic-Programm vorangestellt werden, so empfiehlt sich das im ZX-81-Kochbuch II beschriebene „Append"-Programm zum Verknüpfen der Programme. Als Alternative dazu hat das DELETE-Programm einen eingebauen „RAMTOP-Lader" ab Adresse 16739. Das Programm nutzt nämlich ausschließlich relative Sprünge und verwendet als Variablenspeicher den stets bei Adresse 16444 liegenden Drucker-Puffer. Im Klartext heißt dies: das DELETE-Programm darf an beliebiger Stelle im RAM Platz nehmen, wenn dieser nicht anderweitig belegt ist. Diesen Umstand macht sich der RAMTOP-Lader zunutze. Wird er mit

PRINT USR 16739

aufgerufen, so setzt er den Wert von RAMTOP automatisch auf 32543 herab (funktioniert damit erst ab 16 KByte RAM), deklariert diesen Wert mit einem simulierten NEW-Befehl für das Betriebssystem und kopiert das DELETE-Programm aus der REM-Zeile in den nunmehr geschützten Speicherbereich an der 16-KByte-RAM-Obergrenze.

Jetzt darf man ein Basic-Programm wie gewohnt laden, ohne daß dadurch das DELETE-Programm verlorengeht. Nur hat der DELETE-Aufruf nun eine neue Startadresse, nämlich 32543; sonst ändert sich am Aufruf nichts.

Unser „Radiergummi" oberhalb von RAMTOP kann jederzeit auf nachgeladene Basic-Programme angewendet werden, nur dann nicht, wenn oberhalb von RAMTOP schon ein anderes Machinenprogramm steht: Der RAMTOP-Lader orientiert sich nämlich nicht am aktuellen RAMTOP-Wert, setzt also einen bereits tiefer herabgesetzten Wert kategorisch auf 32543. Und als ZX-81-Kenner wissen Sie freilich auch, daß das DELETE-Programm über RAMTOP nicht mehr bei SAVE gemeinsam mit einem Basic-Programm gespeichert wird.

Für Kenner der ZX-81-Maschinensprache ist auch das in *Bild 2* gezeigte Assembler-Listing des DELETE-Programm gedacht. Daß es relativ kurz ist hat einen Grund, denn das DELETE-Programm verwendet immerhin sieben ROM-Routinen und drei Restarts.

Hubertus Kehl/-ll

```
ADR. HEX-KODE ASSEMBLER-BEFEHL  ; KOMMENTAR

-------------------------------*** UEBERTRAGE STRING ***-----------
4082 E7       RST  20H          ; NEXT-CHARACTER RST UEBERSPRINGT "="
4083 CD550F   CALL 0F55H        ; SCANNING ROUTINE  -> DE=STRINGADRESSE
4086 CDF813   CALL 13F8H        ; STK-FETCH ROUTINE -> BC=STR.-LAENGE
4089 213C40   LD   HL,403CH     ; HL=ANFANG PRBUFF
408C EB       EX   DE,HL        ; TAUSCHE DE MIT HL
408D 79       LD   A,C          ; TESTEN OB STRING
408E FE0A     CP   0AH          ; ZU LANG (>10 ?)
4090 303A     JR   NC,$+58>40CC ; DANN SPRINGE ZUR FEHLERROUTINE
4092 B7       OR   A            ; LAENGE=0 ?
4093 2837     JR   Z,$+55>40CC  ; -> FEHLER
4095 EDB0     LDIR              ; UEBERTRAGE STRING IN PRBUFF
-------------------------------*** TEST OB 1 $0C UND 1 "-" ***------
4097 3E0C     LD   A,0CH        ; SETZE ABSCHLUSS-
4099 12       LD   (DE),A       ; ZEICHEN (0C H)
409A 213C40   LD   HL,403CH     ; HL=PRBUFF
409D 110000   LD   DE,0000H     ; ZAEHLER=0
40A0 060A     LD   B,0AH        ; SCHLEIFENZAEHLER=10
40A2 7E       LD   A,(HL)       ; HOLE ZEICHEN
40A3 23       INC  HL           ; NAECHSTES ZEICHEN
40A4 FE16     CP   16H          ; = "-" ?
40A6 2001     JR   NZ,$+1>40A9  ; WENN NICHT ->
40A8 1C       INC  E            ; ERHOEHE ZAEHLER 1
40A9 FE0C     CP   0CH          ; = ABSCHLUSSZEICHEN ?
40AB 2001     JR   NZ,$+1>40AE  ; WENN NICHT ->
40AD 14       INC  D            ; ERHOEHE ZAEHLER 2
40AE 10F2     DJNZ $-14>40A2    ; SOLANGE B>0  -> HOLE NAECHSTES Z.
40B0 7B       LD   A,E          ; TEST
40B1 FE01     CP   01H          ;  ZAEHLER 1 >1 ?
40B3 2017     JR   NZ,$+23>40CC ; ->FEHLER
40B5 7A       LD   A,D          ; TEST
40B6 FE01     CP   01H          ;  ZAEHLER 2 >1 ?
40B8 2012     JR   NZ,$+18>40CC ; ->FEHLER
-------------------------------*** TEST AUF EINGABEFEHLER 1. ***---
40BA 213C40   LD   HL,403CH     ; HL=PRBUFF
40BD 0600     LD   B,00H        ; ZEICHENZAEHLER=0
40BF 7E       LD   A,(HL)       ; HOLE ZEICHEN
40C0 FE1C     CP   1CH          ; < "0" ? ** TESTE OB NUR ZAHLEN **
40C2 380A     JR   C,$+10>40CE  ; -> ENDZEICHEN UND "-" TEST
40C4 FE26     CP   26H          ; > "9" ?
40C6 3004     JR   NC,$+4>40CC  ; -> FEHLER
40C8 04       INC  B            ; ERHOEHE ZEICHENZAEHLER
40C9 23       INC  HL           ; NAECHSTES ZEICHEN
40CA 18F3     JR   $-13>40BF    ; -> HOLE ZEICHEN
-------------------------------*** EINGESCHOBENE FEHLERROUTINE ***-
40CC CF       RST  08H          ; STOPPT PROGRAMMABLAUF MIT ERROR-
40CD 0E       DEFB              ; RESTART UND ERZEUGT MELDUNG "F"
-------------------------------*** TEST AUF EINGABEFEHLER 2. ***---
40CE FE0C     CP   0CH          ; = ENDZEICHEN ?
40D0 2809     JR   Z,$+9>40DB   ; ->TEST ZEILENLAENGE
40D2 FE16     CP   16H          ; = "-" ?
40D4 3E01     LD   A,01H        ; WENN ZWEITE ZAHL UEBERPRUEFT
40D6 324640   LD   (4046H),A    ; WIRD, FLAG SETZEN (4046)
40D9 20F1     JR   NZ,$-15>40CC ; WENN UNGLEICH "-" ->FEHLER
40DB 78       LD   A,B          ; A=ZEICHENZAEHLER
40DC B7       OR   A            ; ZAHLENLAENGE=0 ?
40DD 28ED     JR   Z,$-19>40CC  ; -> FEHLER
40DF FE05     CP   05H          ; ZAHLENLAENGE>4 ?
40E1 30E9     JR   NC,$-23>40CC ; -> FEHLER
40E3 7E       LD   A,(HL)       ; WENN ERST EINE
40E4 23       INC  HL           ; ZAHL UEBERPRUEFT
40E5 FE16     CP   16H          ; WURDE DANN
40E7 28D4     JR   Z,$-44>40BD  ; -> AN ANFANG
40E9 3A4640   LD   A,(4046H)    ; FLAG MUSS
40EC FE01     CP   01H          ; GESETZT SEIN
40EE 20DC     JR   NZ,$-36>40CC ; SONST -> FEHLER

-------------------------------*** UMWANDLUNG DER Z.NUMMERN ***-----
40F0 2A1640   LD   HL,(4016H)   ; RETTE
40F3 E5       PUSH HL           ; CHADD WERT
40F4 213C40   LD   HL,403CH     ; LADE CHADD,PRBUFF (ANF. DER ZAHLEN)
40F7 221640   LD   (4016H),HL   ; FUER DIE DIE UMWANDLUNG DER ZN.
40FA DF       RST  18H          ; COLLECT CHARACTER RESTART HOLT 1. ZAHL
40FB CD4815   CALL 1548H        ; INTEGER TO FLOATING POINT ROUTINE
40FE CD8A15   CALL 158AH        ; FLOATING POINT TO BC
4101 C5       PUSH BC           ; ZN. IST JETZT BINAER IN BC
4102 E1       POP  HL           ; BC-> HL
4103 CDD809   CALL 09D8H        ; LINE ADDR SUCHT ANF DER Z. IM SPEICHER
4106 20C4     JR   NZ,$-60>40CC ; WENN Z. NICHT VORHANDEN -> FEHLER
4108 224740   LD   (4047H),HL   ; RETTE ADRESSE
410B E7       RST  20H          ; UEBERSPRINGE "-"
410C CD4815   CALL 1548H        ; BESTIMME
410F CD8A15   CALL 158AH        ; GENAUSO
4112 C5       PUSH BC           ; DIE ANF.ADRESSE
4113 E1       POP  HL           ; DER ZEILE MIT
4114 CDD809   CALL 09D8H        ; DER 2. ZEILENNUMMER
4117 20B3     JR   NZ,$-77>40CC ; WENN ZN NICHT VORHANDEN -> FEHLER
4119 23       INC  HL           ; HOLE
411A 23       INC  HL           ; LAENGE
411B 4E       LD   C,(HL)       ; DER
411C 23       INC  HL           ; 2. ZEILE
411D 46       LD   B,(HL)       ; NACH BC
411E 2B       DEC  HL           ; LADE
411F 2B       DEC  HL           ; DE
4120 2B       DEC  HL           ; MIT
4121 EB       EX   DE,HL        ; ANF.ADRESSE Z2
4122 E1       POP  HL           ; LADE CHADD MIT
4123 221640   LD   (4016H),HL   ; SEINEM ALTEN WERT
-------------------------------*** TEST Z1>Z2 ***------------------
4126 2A4740   LD   HL,(4047H)   ; LADE HL MIT ANFANGSADR. Z1
4129 EB       EX   DE,HL        ; STELLE UNTERSCHIED
412A ED52     SBC  HL,DE        ; ZWISCHEN Z1 UND Z2 FEST
412C 389E     JR   C,$-98>40CC  ; WENN Z1>Z2 -> FEHLER
-------------------------------*** LAENGE Z1 WIRD VERAENDERT ***---
412E ED4A     ADC  HL,BC        ; LAENGE DER
4130 EB       EX   DE,HL        ; 1. ZEILE WIRD
4131 23       INC  HL           ; SO VERAENDERT
4132 23       INC  HL           ; DASS Z1 ALLE
4133 73       LD   (HL),E       ; ZU LOESCHENDEN
4134 23       INC  HL           ; ZEICHEN
4135 72       LD   (HL),D       ; UMFASST
-------------------------------*** LADE ARBEITSRAUM MIT Z1,57F ***-
4136 2A1440   LD   HL,(4014H)   ; MACHE PLATZ FUER DIE ZEILENNUMMER-
4139 010600   LD   BC,0006H     ; AUFNAHME IM ARBEITSRAUM
413C E5       PUSH HL           ; (E-LINE - STKBOT)
413D CD9E09   CALL 099EH        ; MAKE ROOM ROUTINE
4140 D1       POP  DE           ; LADE
4141 213C40   LD   HL,403CH     ; SOLANGE
4144 7E       LD   A,(HL)       ; ZAHLEN
4145 FE16     CP   16H          ; IN DEN
4147 2805     JR   Z,$+5>414E   ; ARBEITSRAUM
4149 12       LD   (DE),A       ; BIS.
414A 23       INC  HL           ; DAS "-" -
414B 13       INC  DE           ; ZEICHEN
414C 18F6     JR   $-10>4144    ; AUFTAUCHT
414E EB       EX   DE,HL        ; LADE
414F ED5B1640 LD   DE,(4016H)   ; DEN
4153 1B       DEC  DE           ; REST-
4154 1B       DEC  DE           ; LICHEN
4155 3600     LD   (HL),00H     ; PLATZ
4157 23       INC  HL           ; DES
4158 E5       PUSH HL           ; ARBEITS-
4159 ED52     SBC  HL,DE        ; RAUMES
415B E1       POP  HL           ; MIT
415C 20F7     JR   NZ,$-9>4155  ; 0
415E 367F     LD   (HL),7FH     ; LETZTER ARBEITSPLATZ=KURSOR
-------------------------------*** SIMULIERE DRUCK AUF NEWLINE ***-
4160 C30C06   JP   060CH        ; SPRINGE ZUR NEWLINE-ROUTINE
-----------------------------------------------------------------
```

② **Assembler-Listing:** Das DELETE-Programm nutzt sieben ROM-Routinen und drei Restart-Befehle der Z-80-CPU

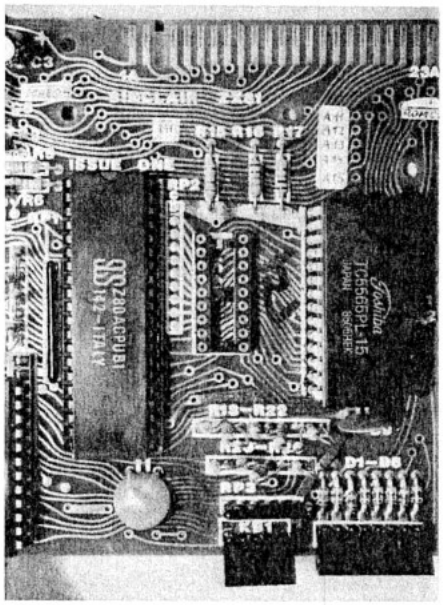
Hardwaretip:

Dreisprung im RAM-Erweitern

Speicherausbau mit 8-KByte-ICs

Mit dem Preisverfall von Speicher-ICs wurden auch die RAM-Module für den ZX 81 billiger. Aber: Wer nicht auf Vorrat kaufen oder nur nicht von den Modul-Herstellern abhängig sein will, der wird eine Bauanleitung schätzen. Unsere Schritt-für-Schritt-Lösung hat da einiges zu bieten.

Zum Redaktionsschluß dieses Sonderheftes gab es noch preisgünstige RAM-Module für den ZX 81, wenngleich das Angebot schon stark dezimiert war. Ob Sie heute im Handel noch fündig werden, ist jedoch fraglich. Wenn Sie mit maximal 24 KByte RAM zufrieden sind – normalerweise ist das mehr als genug – kann Ihnen die Verfügbarkeit von RAM-Modulen indes egal sein. Bauen Sie sich Ihre RAM-Erweiterung für den ZX 81 selbst, wenn Sie einen Lötkolben sicher führen können.



RAM-Sitzplatz: So ist das 8-KByte-RAM im vorbereiteten Steckplatz zu positionieren

Vorsicht ist die Mutter der Porzellankiste

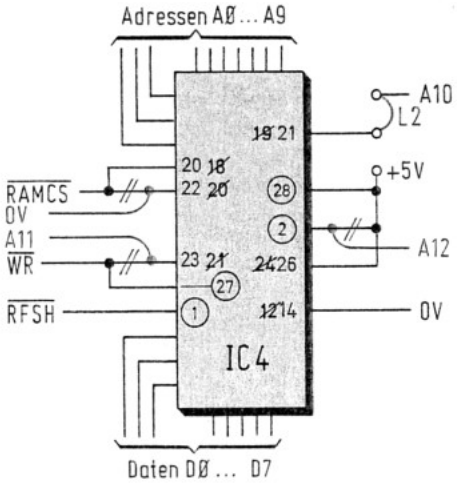
Die hier vorgestellten RAM-Erweiterungen können zumindest bis zu einer Speicherkapazität von 16 KByte problemlos im ZX-81-Gehäuse untergebracht werden. Das hat den Vorteil, daß die ZX-81-Schnittstelle z. B. für den Anschluß selbst gebauter Hardware-Erweiterungen frei bleibt. Das Original-RAM-Modul von Sinclair macht dagegen diese Schnittstelle „dicht". Für die allermeisten ZX-81-Programme sind 16 KByte Basic-Programmspeicher ausreichend. Die restlichen 8 KByte unserer Bauanleitung sind deshalb für Maschinenprogramme vorgesehen. Diese 8 KByte liegen im Adreßbereich von 8 KByte bis 16 KByte. Das zugehörige IC ist akkugepuffert!

Da der ZX 81 Eingriffe in sein Innenleben nicht besonders schätzt, sollte man ihn nicht reizen. Unbedingt empfehlenswert ist daher ein potentialfreier Lötkolben. Wer da unsicher ist, sollte den Lötkolben jedesmal unmittelbar vor dem Löten lieber vom Netz trennen. Und dann sollte man auch keinesfalls „geladen" an den ZX 81 herangehen, sondern beim Arbeiten durch gelegentliches Berühren des Schutzkontaktes an einer Steckdose für Entladung sorgen. Dies ist zwar nur die halbe Weisheit, hilft aber üblicherweise weiter. Zinnbrücken und Verdrahtungsfehler können den ZX 81 ohnehin in die Jagdgründe schicken. Wer sich jetzt noch traut, der darf nun frisch ans Werk gehen.

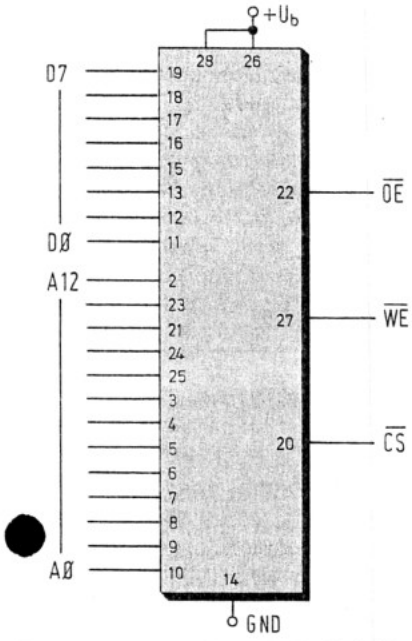
Mit 8 KByte geht's los

Kernstück der Bauanleitung sind 8 K × 8 Bit organisierte RAM-ICs, von denen das Stück derzeit weniger als 15 DM kosten sollte. Wir verwendeten den gängigen Typ 6264 LP 15 (pinkompatibler Ersatztyp: TC 5565-15); bei anderen 8-KByte-RAM-ICs ist eine eventuell abweichende Pinbelegung zu berücksichtigen. In Frage kommen für uns nur statische (keine dynamischen) RAMs.

Zuerst muß man die beiden 2114-RAMs auf der ZX-81-Platine entfernen. Dann ist der vom außenliegenden RAM-IC jetzt freigewordene Steckplatz für das von Sinclair ursprünglich vorgesehene 2-KByte-RAM von Lötzinn zu befreien. Gemäß *Bild 1* (siehe auch ZX-81-Schaltplan im ZX-81-Kochbuch I) muß man anschließend die Versorgungsspannungs-Leiterbahn von Pin ② abtrennen. Dazu wird beiderseits von Pin ② die Leiterbahn aufgetrennt und mit einer Drahtbrücke die Spannungsversorgung



① **Platinen-Änderungen:** Das 8-KByte-RAM 6264 verlangt drei Leiterbahnunterbrechungen sowie nach zusätzlichen Adreßsignalen



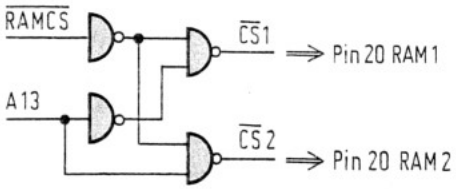
② **Pinbelegung des 6264:** Bei Ersatztypen kann die Pinbelegung anders aussehen

für Pin ㉔ und ㉘ wiederhergestellt. Ebenso ist auf der Bestückungsseite die Verbindung zwischen Pin ⑱ und Pin ⑳ (RAM $\overline{CS}$) zu lösen. Und auf der Lötseite ist Pin ㉑ von der seitlich zu Pin ㉗ weiterführenden Leiterbahn abzutrennen.

Alle bisher genannten Pin-Nummern beziehen sich auf den ZX-81-Schaltplan, der jedoch nicht die Pinbelegung für das 28polige RAM-IC 6264 berücksichtigt (*Bild 2*). Für dieses IC gelten die in *Bild 1* blau eingetragenen Pin-Nummern. Dort, wo Sinclair z. B. Pin ⑱ eines 2-KByte-RAMs vorgesehen hat, ist deshalb jetzt Pin ⑳ des 6264-ICs zu finden.

Nun darf man einen 28poligen Sockel (möglichst flache Form) in die vorhandenen Bohrungen einlöten. Mit der Drahtbrücke L2 ist dann die Adreßleitung A 10 auf Pin ㉑ (neue Zählweise!) des ICs zu führen. Die zuvor abgetrennten Anschlußpunkte sind mit feinem Draht folgendermaßen neu zu verdrahten (Achtung: wieder neue Zählweise):
Pin ㉓ wird mit A 11 an der Katode der Tastatur-Diode D1 verbunden.
Pin ② wird mit A 12 (an der Katode der Tastatur-Diode D3) verbunden.
Pin ㉒ wird mit Pin ⑭, also mit Masse, verbunden.

Wurde alles richtig gemacht und wird jetzt das RAM-IC in den Sockel gesteckt, dann kommandiert Ihr ZX 81 schon über 8 KByte RAM.

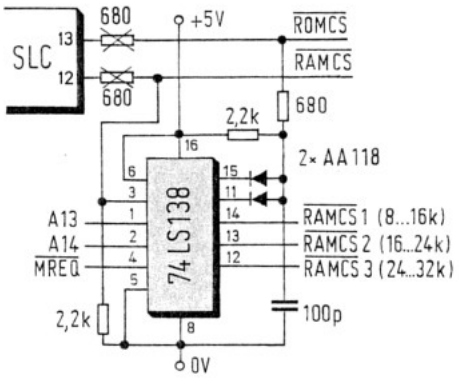


③ **Adreßlogik für zwei RAMs:** Vier NAND-Gatter eines 74LS00-ICs gewinnen die beiden CS-Signale

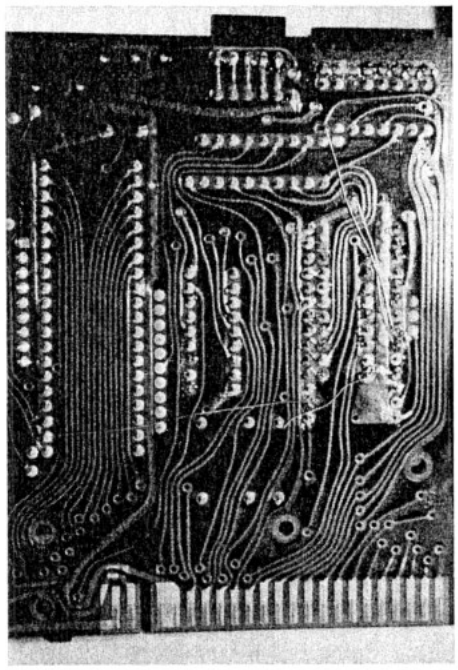
Aufstocken bis 24 KByte

Ein Aufstocken auf 16 KByte kann man im wahrsten Sinne des Wortes vornehmen: Es ist lediglich ein zweites 6264-IC über das erste zu stülpen und alle Beinchen, bis auf die beiden Pins ⑳ sind miteinander zu verlöten. Eine mit RAM CS und A 13 gespeiste höchst einfache Adreßlogik (*Bild 3*), deren Ausgänge zu den abgewinkelten Pins ⑳ führen, hält dann die beiden ICs so auseinander, daß es zu keiner Doppeladressierung kommt. Der ZX 81 bekommt damit 16 KByte RAM im Adreßbereich zwischen 16 KByte und 32 KByte.

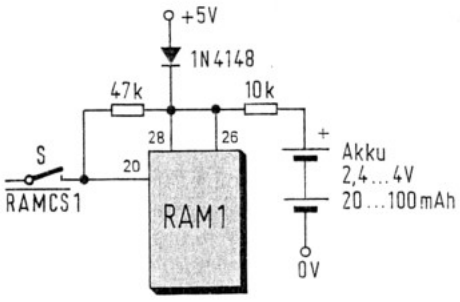
Wer einen Maschinencode-Speicher im Adreßbereich zwischen 8 KByte und 16 KByte zum Ziel der Wünsche gemacht hat, der muß sich jedoch für die etwas aufwendigere Adreßlogik gemäß *Bild 4* entscheiden, die noch einen weiteren Eingriff in den ZX 81 verlangt: Die beiden 680-Ω-Widerstände an Pin ⑫ und ⑬ des Logik-Chips (SLC) sind hierbei ersatzlos (nicht durch Brücken ersetzen) zu entfernen. Anderenfalls kommt es zu Störungen. Über je einen 2,2-kΩ-Widerstand der Adreßlogik werden



④ **Adreßlogik für drei RAMs:** Die Schaltung gibt drei 8-KByte-Blöcke zwischen 8 KByte und 32 KByte Adreßraum frei



Um-Leitungen: Hier führen vier Fädeldrähte Adreßsignale (A 11, A 12), Masse und Betriebsspannung an das 8-KByte-RAM heran



⑤ **Akkupufferung:** Das Maschinencode-RAM zwischen 8 KByte und 16 KByte wird damit „nichtflüchtig"

ROM $\overline{CS}$ auf H-Pegel und RAM $\overline{CS}$ auf L-Pegel gezogen. Externe RAM-Module können dadurch RAM $\overline{CS}$ auf H-Pegel legen und so alle internen RAMs sperren; die Schaltung ist also mit externen RAM-Modulen verträglich. Mit einem externen RAM geht damit allerdings auch der Zugriff auf das Maschinencode-RAM verloren.

Sofern noch genug Platz da ist, kann diese Adreßlogik drei gestapelte 6264-RAM-ICs adressieren, indem die Ausgänge mit den „abgewinkelten" Pins ⑳ verbunden werden. Das zwischen 8 KByte und 16 KByte adressierte RAM läßt sich dann gemäß *Bild 5* über einen Akku puffern. Georg Baumhauer/-ll

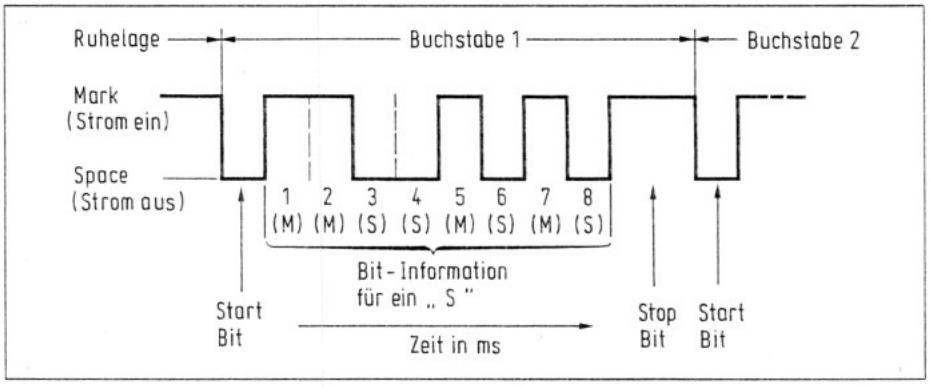
Akustikkoppler fürs Haustelefon:

Zwei Töne machen die Musik

Wer mit dem hier im Heft beschriebenen ZX-Terminal per Telefonleitung in Mailboxen herumstöbern möchte, der braucht dazu einen Akustikkoppler. Die Zeitschrift ELO veröffentlichte für so ein Gerät eine ungewöhnlich preisgünstige Bauanleitung (Materialkosten etwa 40 DM). Mit freundlicher Genehmigung der ELO-Redaktion können wir Ihnen diese Bauanleitung, in gekürzter Form, auch hier präsentieren.

Der Computer kennt zwei Zustände: ein und aus. Seine Daten übermittelt er parallel oder im „Gänsemarsch" hintereinander. Werden die Daten parallel übermittelt, so benötigt man für jedes Bit einen separaten Leiter. Stehen jedoch zur Datenübermittlung nur zwei Drähte zur Verfügung – wie beispielsweise beim Telefon – so muß eine bitserielle Übertragung erfolgen. Da, wo die Bits in Form des Datenstromes den Computer verlassen, befindet sich die Schnittstelle. Schnittstellen sollten genormt sein. Als serielle Schnittstelle ist uns die RS-232-C-Norm bekannt. Eine sehr bekannte Parallel-Schnittstelle heißt Centronics und wird hauptsächlich bei Druckern verwendet.

Werden beispielsweise Daten mit 300 Baud übermittelt, so entspricht dies einer Geschwindigkeit von 300 Bit in jeder Sekunde. Was hier übermittelt wird, ist ein mit 300 Hz geschalteter Gleichstrom. Allerdings nicht dauernd, denn wenn nichts geschrieben wird, dann schweigt auch der Gleichstrom-Ton. In diesem Fall fließt Strom, und da sonst nichts passiert, nennt man diesen Zustand die Ruhelage. Im Englischen heißt die Ruhelage Mark. Sobald jetzt eine Taste der Computer-Tastatur angeschlagen wird, wird das Zeichen im dementsprechend genormten Telegrafen-Alphabet übermittelt. Das Alphabet legt auch fest, aus wievielen Informations-Bits das Zeichen besteht. Beispielsweise besteht ein übermitteltes Zeichen im Alphabet des CCITT-2-Code (CCITT ist die Abkürzung der französischen Bezeichnung für den internationalen, beratenden Ausschuß für den Telegrafendienst des internationalen Nachrichtenvereins, franz. Comité Consultatif International Télégrafique et Téléfonique) aus fünf Zeichenschritten. Heute tauschen drahtgebundene und Funkfernschreiber ihre Daten immer noch mit diesem Übermittlungsverfahren aus. Durch unterschiedliche Belegung lassen sich allerdings mit einem Fünfer-Code nur jeweils $2^5 = 32$ Möglichkeiten darstellen, deshalb 2^n Möglichkeiten, weil man zwei Zustände kennt: ein und aus.



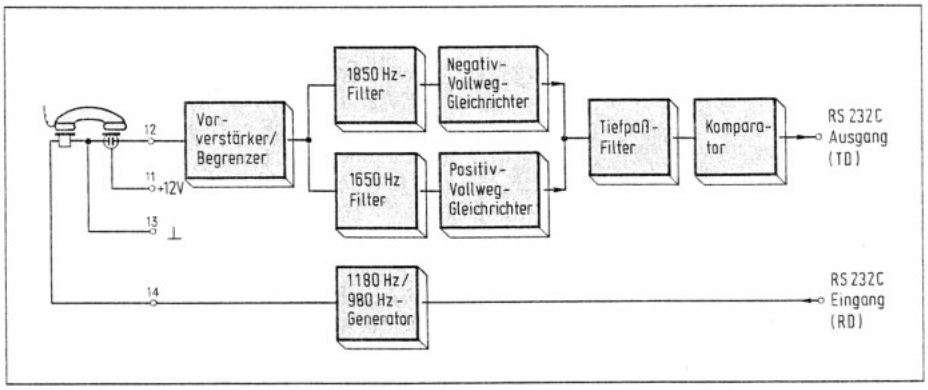
① **ASCII-Code:** Hier wird das „S" dargestellt. Die Geschwindigkeit der Zeichenübermittlung ist in ms (Millisekunden) angegeben

Asynchroner ASCII

Im Jahre 1968 publizierte das ANSI (engl. American National Standard Institute) den ANSI-Standard X3.4-1968, schlicht und ergreifend ASCII-Code (engl. American Standard Code for Information Interchange) genannt, ein serieller 7-Bit-Code mit einem zusätzlichen achten Bit, allgemein auch Prüfbit genannt, um bei Bedarf einen Prüfsummen-Test eines jeweiligen Zeichens zu ermöglichen. Statt der üblichen 32 Befehle lassen sich nun 128 darstellen!

Der Computer übermittelt seine Zeichen im ASCII-Code. Seine Ruhelage heißt Mark, oder auch Stromschritt, oder auch logisch Eins. Wenn jetzt ein komplettes Zeichen gesendet werden soll, so will man ja, daß die Empfangsmaschine genau das gleiche wie die Sendemaschine tut. Also folgt nach der Ruhelage mit der logischen Eins (Stromschritt) erst einmal ein Sprung in die Nichtstrom-Ebene – logisch Null = kein Strom. Diesen Schritt nennt man Startbit. Einem jeden Zeichen geht nach der Ruhelage zuerst immer der Startschritt (Startbit) voraus. Natürlich muß der sendende Computer oder Fernschreiber immer mit der gleichen Geschwindigkeit wie der empfangende auf der Gegenseite laufen, weil sich sonst keine Synchronität ergeben könnte. Für den Ablauf eines jeden Zeichens herrscht nämlich Synchronität vor. Abgeschlossen wird das Zeichen oder auch der Befehl mit einem Stoppschritt. Dann liegt wieder die Ruhelage an. Dann tut sich nichts auf beiden Seiten, nichts ist mehr synchron. Die Syn-

② **Blockschaltbild:** Für das Empfangsteil ist deutlich mehr Aufwand nötig als für das Sendeteil



chronität herrscht also nur für die Dauer eines kompletten Zeichens vor, das mit Start- und Stoppschritt (-bit) eingerahmt wird. Solche Übermittlungsverfahren werden asynchrone genannt.

Vom Gleichstrom mag das Telefon nichts wissen

Abgesehen von der Versorgungsspannung geht sonst mit Gleichstrom absolut gar nichts bei „Postens Telefon". Nur Wechselströme lassen sich bekanntlicherweise mit einem Transformator übertragen. In den meisten Fällen verwendet der „gelbe, große Bruder" den Trafo, um eine galvanische Trennung zu bewirken.

Das ist also der Grund, warum aus den zwei Datenzuständen Mark und Space, High und Low, Eins und Null, Strom und Nichtstrom jeweils ein Wechselstrom werden muß. Aber nicht nur deshalb! Wer sich schon einmal mit dem Problem herumschlagen mußte, über längere Distanzen solche Datenströme zu schicken, kennt den grauen „Leitungsgilb", der über kurz oder lang jede Spannung zusammenschrumpfen läßt. Im Fachjargon heißt er: Ohmscher Widerstand. Sie ahnen sicher alle, wie die physikalischen Gesetze da ihren Einfluß haben. Also, nehmen wir da zwei Frequenzen – eine für Mark und eine für Space. Für Space die höhere mit 1180 Hz, für Mark die mit 980 Hz. Diese Frequenzen sind genormt. Wenn man also mit diesen „Tönen" irgendeine Datenbank aktiviert, so kann man sicher sein, daß die Demodulatoren auch diese Frequenzen wieder in ein Mark und ein Space umwandeln können. Der Mailbox-Computer TEDAS (im Franzis-Verlag) beispielsweise wird sofort diese in der Frequenz umgetasteten Datenzustände in ein Gleichstrom-Mark oder -Space umwandeln, der Computer „frißt" dann die Befehle in seinen Datenranzen und wird dann Ihre Befehle ausführen.

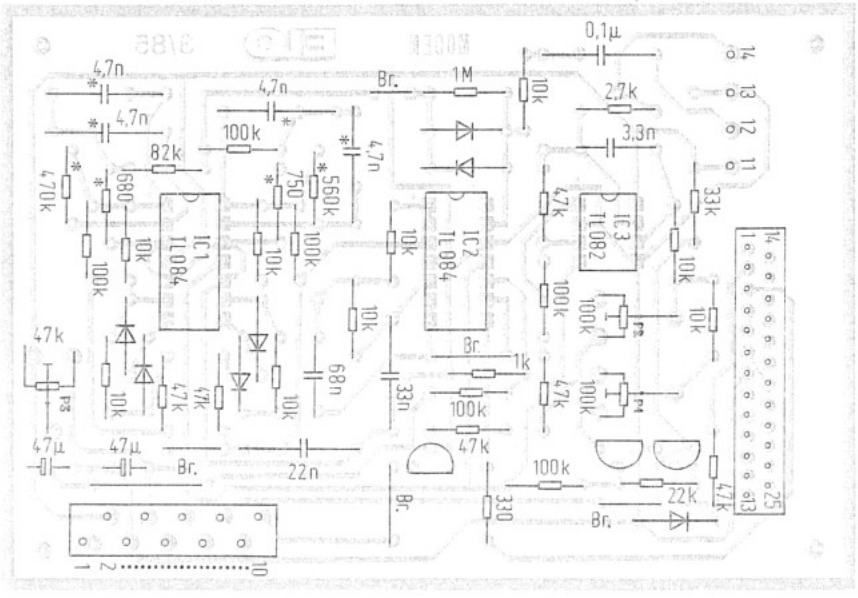
Der TEDAS-Computer sendet natürlich nicht über sein Modem mit 1180 und 980 Hz zurück. Das würde nämlich bedeuten, daß sich irgendwo im System oder auf der Platine diese Frequenzen „sehen oder hören" würden. Dann käme es zu einer Rückkopplung. Der von ihnen angesprochene Computer sendet auch wieder Mark und Space aus, diesmal jedoch 1650 Hz und 1850 Hz. Die Mark-Frequenz, also Strom oder High nimmt immer die tiefere Frequenzlage ein. Und da zwischen zwei Frequenzen hin- und hergeschaltet wird, spricht man auch von einer Frequenzumtastung.

Wagen wir uns jetzt also an den Bau eines Akustikkopplers, der den Telefonhörer als Bindeglied nutzt, um in NF-Signale umgeformte Computer-Daten auf die Reise zu schicken. Und selbstverständlich kann unser Akustikkoppler ankommende „NF-Daten" ebenfalls verarbeiten, daraus also wieder Computer-Daten formen.

Und jetzt geht es endlich los

Für die Schallaufnahme eignet sich am besten ein Elektret-Mikrofon. Selbst bei einer Eingangsspannung von nur 120 µV funktioniert die Schaltung noch. Am Ausgang des invertierenden Begrenzers ist die Eingangsspannung nun mittlerweile auf 10 mV angestiegen. Nun wird das Signal den beiden aktiven NF-Filtern OP 2 und OP 3 zugeführt. OP 2

Bestückungsplan: Sternchen markieren „kritische" Bauteile



wertet die höhere, OP 3 die tiefere der beiden Frequenzen aus. Bei den aktiven Filtern sollten unbedingt Styroflex-Kondensatoren und Metallfilmwiderstände verwendet werden. Die kritischen Bauteile sind mit einem Sternchen im Schaltbild bezeichnet. OP 4 und OP 5 sind als aktive Doppelweg-Gleichrichter beschaltet. OP 4 liefert die negative Spannung, OP 5 die positive. Die Oszillogramme C und D sind Anhaltspunkte. OP 6 (IC 2) stellt einen nicht invertierenden Summenverstärker dar und OP 7 einen nicht invertierenden Tast-Tiefpaß zweiter Ordnung (12 dB), dessen obere Grenzfrequenz bei der maximalen Bitrate liegt – in unserem Fall bei knapp über 300 Hz. Der Sinn des Tiefpaßfilters liegt darin, Rauschen oder Störanteile, die dem gleichgerichteten Signal noch überlagert sind, zu eliminieren. Damit ein „datenreines" Mark-Space-Signal für den Computer bereitgestellt werden kann, erfolgt in OP 8, einem Vergleicher, nochmals eine Regenerierung der Flanken. Solche Schaltungen nennen die Engländer auch Slicer. Wichtig ist der Offsetabgleich, der mit P 3 durchgeführt werden muß. Es kann nämlich vorkommen, daß sich der Gleichspannungsanteil aus sämtlichen Verstärkerstufen bis zum Slicer hin addiert. Diese Tatsache würde dazu führen, daß dieses aktive Bauteil dann nicht exakt schalten könnte. Und genau dies gilt es also tunlichst zu vermeiden. An Pin 8 von OP 8 kann ein astreines Datensignal mit einem Pegel von ± 10 V entnommen werden. Die zusätzliche Auskopplung über den Schalttransistor T 3 ermöglicht eine Anpassung für TTL-Datennetze.

Der FSK-(Frequency-Shift-Keying, engl. Frequenzumtastung)Modulator besteht aus einem Funktionsgenerator, aufgebaut mit OP 9 und OP 10. Für die eigentliche Frequenzumtastung ist der BC 237 (T 2) verantwortlich. T 1 dient zur Pegelanpassung an die RS-232-C-Schnittstelle. Falls der Modulator nur mit TTL-Pegel betrieben werden soll, so kann der BC 237 (T 1) entfallen. Das Dreieckssignal – das Oszillogramm J dient auch hier als Anhaltspunkt – steuert den Schallwandler an. Eine Telefonhörkapsel leistet hier gute Dienste.

Der Abgleich ist denkbar einfach – mit drei Potis sind Sie dabei

Als Stromversorgung können Sie extern ± 12 V zuführen. Wenn die Mimik jedoch netzspannungsungebunden betrieben werden soll, reichen zwei 9-V-Blockbatterien vollauf. Halter und Batterie-Clips führen sämtliche gutsortierten Elektronikläden. Die Schaltung nimmt im Normalfall bei +12 V 19,4 mA, bei −12 V 19,1 mA auf. Bei ±9 V sind es knapp unter 19 mA. Selbst wenn der Abgleich bei ±12 V durchgeführt wurde,

Anschlußbelegung nach der Schaltbildbezeichnung

1	unbelegt
2	+12 V
3	Masse
4	−12 V
5	unbelegt
6	unbelegt
7	unbelegt
8	TD-TTL
9	Masse
10	RD-TTL
11	+12 V für Elektretmikrofon
12	NF-Eingang (Mikrofon)
13	Masse
14	NF-Ausgang (Hörkapsel)

Stückliste

Halbleiter
2 TL 084
1 TL 082
3 BC 237
7 1N 4148

Widerstände (0,125 W/5 %)
1 330 Ω
1 1 kΩ
1 2,7 kΩ
9 10 kΩ
1 22 kΩ
1 33 kΩ
6 47 kΩ
1 82 kΩ
6 100 kΩ
1 1 MΩ

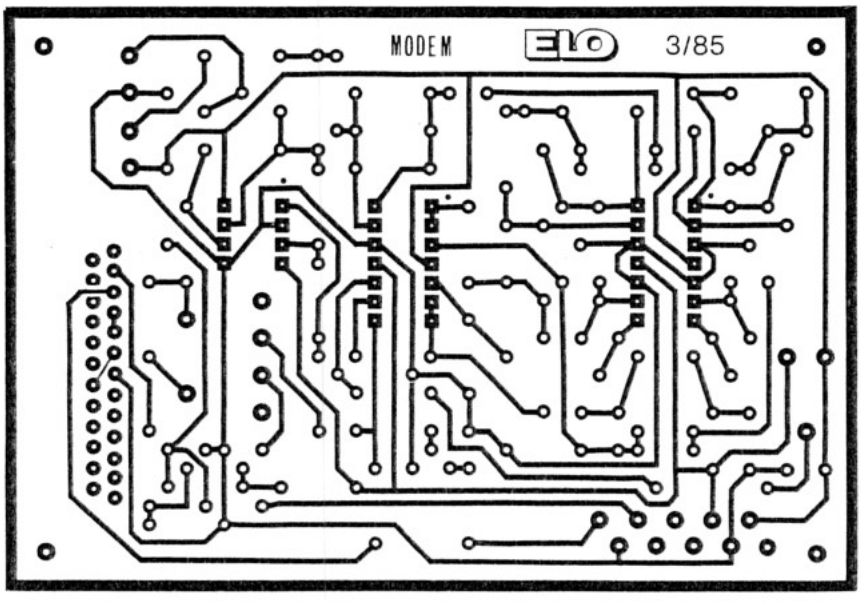
Metallfilmwiderstände
1 680 Ω
1 750 Ω
1 470 kΩ
1 560 kΩ

Potentiometer
1 50 kΩ, linear
2 100 kΩ, linear

Kodensatoren
1 3,3 nF
4 4,7 nF, Styroflex
1 22 nF
1 33 nF
1 68 nF
1 100 nF
2 47 µF/16 V, Elko

Sonstiges
1 Electret-Mikrofon
2 Buchsen
1 Stecker
1 Gehäuse
1 Platine

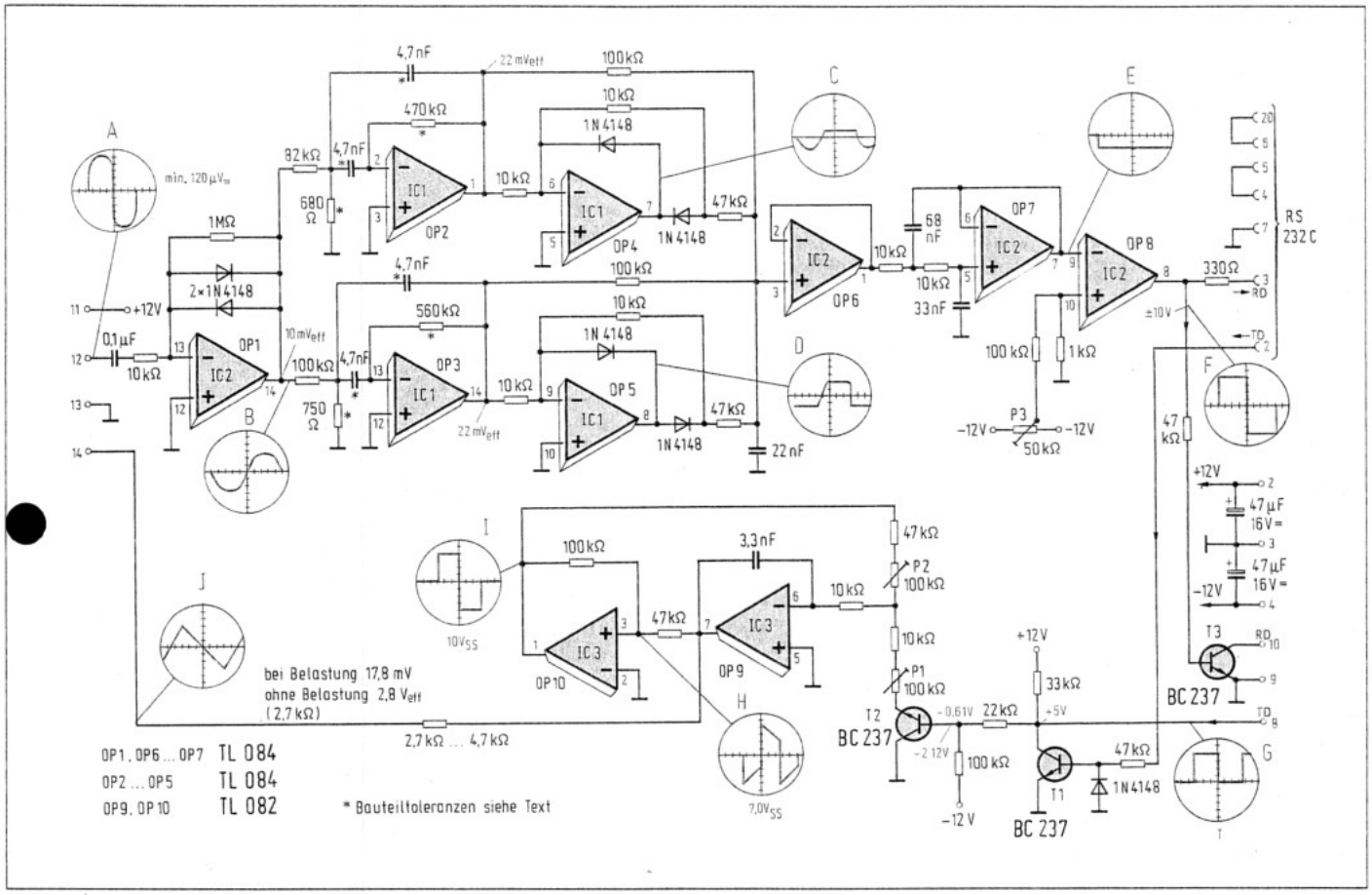
Platinenlayout der Gesamtschaltung



verändert der Generator seine Frequenz bei einer Betriebsspannungsumstellung auf 9 V nicht. Nur am Offset-Potentiometer muß dann ein wenig nachgeglichen werden.

Die beiden Sendefrequenzen-Potentiometer stelle man zuerst auf Linksanschlag. Den TTL-Eingang Pin 8 lassen Sie offen. Mit P 1 stellen Sie 980 Hz ein. Ein Frequenzzähler ist für diese Einstellung erforderlich. Jetzt legen Sie den TTL-Eingang an Masse. Mit P 2 stellen Sie nun 1180 Hz ein. Dann öffnen Sie den TTL-Eingang erneut, und korrigieren mit P 1 die Frequenz von 980 Hz. Zum Abschluß sollten Sie nochmals den TTL-Eingang an Masse legen und die zweite Frequenz kontrollieren. Damit ist der Senderabgleich vorgenommen.

Der Offset-Abgleich des Demodulators ist ebenso denkbar einfach. Der „Emp-



③ **Gesamtschaltung:** Die eingezeichneten Oszillogramme gelten als Anhaltspunkte und nicht als Normkurven

fänger-Eingang" Pin 12 wird kurzgeschlossen. Das Oszilloskop oder Multimeter legen Sie an Pin 3 der RS-232-C-Schnittstelle. Nun wird P 3 langsam vom rechten zum linken Anschlag durchgedreht, bis das Signal von positiv auf negativ wechselt. Da stehen lassen – fertig.

Mechanik ist für manchen Elektroniker ein Greuel – hier ist sie wichtig

Akustikkoppler haben einen gravierenden Nachteil. Sie sind von akustischen Umwelteinflüssen abhängig. Deshalb muß man die Kopplung so dicht wie möglich zwischen dem Telefonhörer und dem akustischen Schallgeber und -nehmer durchführen. Auf eine störungsfreie und gute akustische Kopplung ist unbedingt Wert zu legen, damit nicht Geräusche Übertragungsfehler verursachen. Die gesamte Schaltung wird am besten mit Abstandsbolzen in ein Kunststoffgehäuse eingebaut, in dem auch noch die Batterien oder die stationäre Stromversorgung Platz finden.

Wenngleich voll funktionstüchtig, wird unser Akustikkoppler der Post unangenehm ins Auge stechen. Er wird sich im wahrsten Sinne des Wortes „postialisch" gebärden – der große gelbe Hüter des Fernmeldemonopols, wenn er davon Wind bekommt. Modems, sei es, daß ihre Daten akustisch-gekoppelt oder mittels galvanischer Übertragung in das öffentliche Fernmeldenetz eingespeist werden, bedürfen einer Genehmigung. Eine solche erhalten Sie nur nach einer Einzelprüfung durch das ZZF (Zentralamt für Zulassungen im Fernmeldewesen).

Doch die Prüfung kostet meist ein Mehrfaches des Materialpreises dieses akustisch-gekoppelten Modems. Daher unser guter Rat: Bitte nur zu Hause am eigenen Haustelefon verwenden!

Josef Kleinschwärzer, René Füllmann

ZX-81-Kochbuch:

Berichtigungen

Wir haben es befürchtet, und es ist prompt eingetroffen: Im ZX-81-Kochbuch hat der Fehlerteufel zugeschlagen. Zuerst auf Seite 27. Wählen Sie zum „entfesseln" der PIO nur Lösungsvorschlag 2; der andere tut's in der gezeigten Form nicht. Auf Seite 49 funktioniert die „Wiederbelebung" besser, wenn in die $\overline{WR}$-Leitung ein Schalter eingefügt wird und Pin 21 des ICs über einen 22-kΩ-Widerstand auf +5 V gelegt wird. Der Schalter ist dann nur beim Laden des Zusatz-RAMs zu schließen. Auf Seite 42 verlangt die sechste REM-Zeile 448 Zeichen und der Code bei Adresse 16624 muß 55 lauten. Auf Seite 55 ist der Code bei Adresse 16546 in 88 zu ändern. Das Prüfsummenprogramm von Seite 58 darf die Summe nur zwischen der Adresse 16514 und Adresse 16833 bilden.

ZX-81-Softwaretip:

Die Null kann Platz schaffen

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*).

```
9010 FOR I=16510 TO 32000
9030 IF PEEK I=118 THEN PRINT AT
 10,10;256*PEEK (I+1)+PEEK (I+2)
9035 CLS
9040 IF INKEY$="" THEN NEXT I
9050 PRINT I+2,I+1
```

Späher nach Zeilennummern: An ein Basic-Programm „angehängt", setzt sich dieses Programm auf die Spur der Zeilennummern

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-Softwaretip:

Computer als Dia-Showmaster

Im Schrank stapeln sich die Dias und warten darauf, wieder einmal betrachtet zu werden. Vielleicht gibt das hier vorgestellte Programm den Anstoß dazu, denn es steuert den Ablauf einer regelrechten Dia-Show. Wird noch ein Kassettenrecorder zur Musik oder Sprachuntermalung hinzugenommen, dann ist die Sache noch attraktiver.

Wenn Sie den ZX 81 zum Dia-Showmaster ernennen wollen, dann brauchen Sie dazu eine Schnittstelle zur Datenausgabe. Das Programm (*Bild 1*) verlangt z. B. nach der Basisplatine (I/O-Port) der ZX-81-à-la-carte-Serie. Ebensogut läßt sich aber auch das Spar-Interface aus dem ZX-81-Kochbuch verwenden oder, mit einer Programmergänzung, auch die PIO-Schnittstelle aus Heft 3/1984.

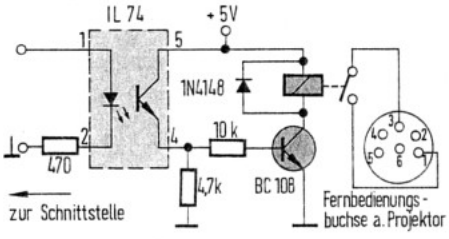
Haben Sie jetzt noch einen Projektor mit Fernbedienungsanschluß, dann kann eigentlich nichts mehr schiefgehen. *Bild 2* zeigt, wie die vom Computer gelieferten Steuerimpulse zum Diawechsel den Projektor erreichen. Quelle der Steuerimpulse ist bei der Basisplatine Pin 1 (Bit 0) der DIL-2-Buchse (Datenausgang). Das Programm ist sogar noch auf der 1-KByte-Grundversion des ZX 81 lauffähig und kann dann 36 Diawechsel steuern (bei einer größeren RAM-Kapazität selbstverständlich mehr).

Und so wird das Programm genutzt: Nach der Eingabe ist es zuerst einmal mit GOTO 60 abzuspeichern. In dieser Ur-Form ist das Programm für jede neue Dia-Show zu laden, worauf man als erstes den Namen bzw. das Motto der Vorführung eintippen darf. Dann wird eine Startfreigabe abgewartet. Drückt man eine Taste, projiziert der Projektor das erste Dia. Passend zur Tonbegleitung kommt es nun mit jedem Drücken der NEWLINE-Taste zu einem Diawechsel.

BREAK beendet die Eingabeprozedur, falls die Vorführung weniger als 36 Dias umfaßt. Im ZX 81 sind jetzt die Bildwechsel-Zeitpunkte gespeichert. Daher sollte man schnellstens das Programm samt Steuervariablen mit GOTO 70 auf Band retten.

```
  2 LET B=0
  3 DIM A(36)
  4 PRINT AT 10,7;"NAME DER DIA
-SHOW?"
  5 INPUT N$
  6 CLS
  8 PRINT AT 4,10;N$
 11 PRINT AT 10,2;"ZUM START EI
NE TASTE DRUECKEN"
 12 IF INKEY$="" THEN GOTO 12
 13 CLS
 14 POKE 16435,255
 15 POKE 16437,255
 16 IF B<>0 THEN GOTO 18
 17 PRINT AT 20,4;"NACH LETZTEM
 BILD >BREAK<                DRU
ECKEN"
 18 FOR I=1 TO 36
 19 PRINT AT 5,1;"BILD-NUMMER "
;I;" WIRD PROJIZIERT"
 21 LET Z=INT ((65535-PEEK 1643
5-256*PEEK 16437)/10)
 22 IF B THEN IF Z<A(I) THEN GO
TO 21
 23 IF B=0 THEN IF INKEY$<>"" T
HEN GOTO 50
 24 IF B=0 THEN GOTO 21
 26 IF I=V THEN STOP
 28 POKE 23658,1
 30 PRINT
 32 POKE 23658,0
 40 NEXT I
 50 LET A(I)=Z
 55 GOTO 28
 60 LET I=0
 70 SAVE "DI■"
 72 IF I=0 THEN RUN
 74 LET B=1
 75 LET V=I
 80 GOTO 8
```

① **Steuerprogramm für Dia-Show:** Die Zeitpunkte für einen Diawechsel werden vom Inhalt der Systemvariablen FRAMES abgeleitet



② **Projektoranschluß:** Für jeden Diawechsel liefert der Computer einen kurzen Steuerimpuls

Nach dem Laden der gewünschten und bereits fertigen Dia-Show wartet der ZX 81 wieder auf die Startfreigabe. Sobald sie erteilt wurde, darf man sich genüßlich im Sessel zurücklehnen: Der Computer steuert den Bildwechsel jetzt selbsttätig genauso, wie Sie es ihm in der „Lernphase" beigebracht haben.

Klaus Stuchlich/-ll

ZX 81 à la carte

Brenner für 2716-EPROMs

Hier ist der eigentlich schon längst fällig gewesene EPROM-Brenner für den ZX 81. Er „schießt" maximal 2 KByte lange Maschinenprogramme in EPROMs vom Typ 2716, wobei am Bildschirm immer nur die gerade gültigen Bedienungshinweise eingeblendet werden.

Darüber, wie ein bereits geladenes 2716-EPROM an den ZX 81 angeschlossen wird, hat die FUNKSCHAU in Heft 10/1984 und in dem Sonderheft „ZX-81-Kochbuch" berichtet (Titel des Beitrags: Wiederbelebung). Jetzt geht es darum, Maschinenprogramme in EPROMs unterzubringen.

Sechs Bits bringen den Brenner in Fahrt

Die Wirkungsweise eines EPROM-Brenners ist schnell geklärt. Das Gerät muß dafür sorgen, daß die einzelnen 8-Bit-Speicherzellen in 2716-EPROMs richtig adressiert werden und zu jeder einzelnen Adresse auch das dafür vorgesehene Datenbyte geschickt wird. Liegt dann unter einer Adresse das richtige Datenbyte am EPROM an, dann muß mit einem „Programmierimpuls" dafür gesorgt werden, daß dieses Datenbyte in die adressierte Speicherzelle „gebrannt" wird. Danach kann die nächste Speicherzelle adressiert werden.

Diese Aufgabe übernimmt der EPROM-Brenner (*Bild 1*) gemeinsam mit dem ZX 81. Der Brenner ist sozusagen für die Hardware-Aufgaben zuständig, während der ZX 81 der Steuermann ist, der z. B. die richtigen Adreß- und Datenbytes für das EPROM parat hält.

Zum Betrieb des Brenners benötigt der ZX 81 ein 16-KByte-RAM. Nur dann ist ausreichend Platz für das Steuerprogramm und die zu programmierenden Bytes vorhanden. Da RAM-Erweiterungen die Busse des ZX 81 bereits stark belasten, und eine Überlastung der Busse zu schwer aufzuspürenden Störungen führen kann, sind sämtliche Signale, die der EPROM-Brenner an der ZX-81-Schnittstelle abgreift, über Treiberstufen gepuffert. Das erhöht zwar den Bauteileaufwand, kommt aber der Betriebssicherheit des Brenners zugute, besonders dann, wenn gleichzeitig noch weitere Peripheriegeräte (z. B. Drucker) an der ZX-81-Schnittstelle angeschlossen sind.

Wie jedes Peripheriegerät, so muß auch der EPROM-Brenner vom ZX 81 adressiert werden. Er wird unter den Adressen C082h (Datenbyte) und C083h (Steuerwort) angesprochen. Der Brenner decodiert diese beiden Adressen mit den Gattern 4, 5 und 6. Diese Decodierung ist unvollständig: Da nur die Signale A0 und A15 ausgewertet werden, wird der Brenner nicht nur unter den genannten Adressen angesprochen, sondern auch unter 32 766 weiteren Adressen.

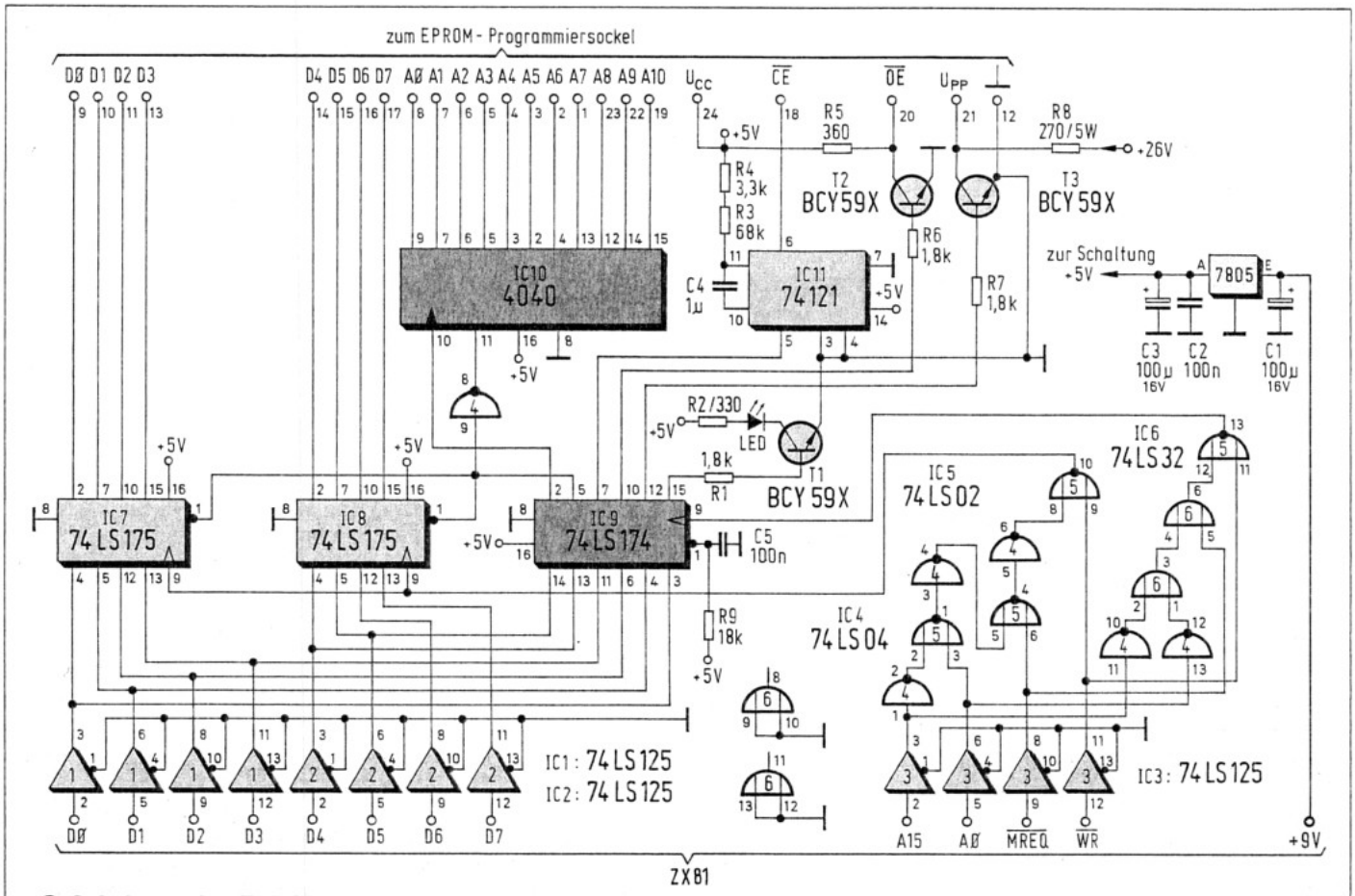
Da bei 16 KByte RAM oberhalb des 32-KByte-Adreßbereichs vom ZX 81 aber keine Speicherzellen angetroffen werden, kommt es nicht zu einer Doppeladressierung von RAM-Speicher und EPROM-Brenner. Wegen der unvollständigen Adreßdecodierung ist jedoch ein Betrieb des Brenners mit mehr als 16 KByte RAM unzulässig.

Die Adressen C082h und C083h wurden gewählt, weil diese Adressen bis auf den Wert von A15 mit den Adressen 4082h und 4083h übereinstimmen. Und da auch das 16-KByte-RAM-Modul unvollständig decodiert ist heißt dies im Klartext: Wenn der Brenner angesprochen wird, dann hat das auch Auswirkungen auf den Inhalt der Speicherzellen 4082h und 4083h. Das Steuerprogramm für den Brenner (Maschinenprogramm) darf deshalb nicht wie gewohnt mit Adresse 4082h beginnen, sondern mindestens zwei Adressen später. Tatsächlich beginnt es bei Adresse 4086h, da noch zwei Byte für eine Hilfsvariable vorgesehen wurden.

Ein unter Adresse C082h ausgegebenes Datenbyte wird vom Brenner in zwei 4fach Zwischenspeicher aufgenommen (IC 7 und IC 8), deren Ausgänge die Dateneingänge des EPROMs speisen. Und von dem unter Adresse C083h erteilten Steuerwort werden nur sechs Bit benötigt, die der Brenner in einem 6fach Zwischenspeicher aufnimmt (IC 9). Für den Brenner haben diese sechs Bit folgende Bedeutung:

- D0: Dieses Bit liefert das Taktsignal für den Zählerbaustein IC 10, der die Adressen an das EPROM anlegt.
- D1: Mit diesem Bit wird der Adreßzähler IC 10 zurückgesetzt, ebenso die Zwischenspeicher IC 7 und IC 8.
- D2: Mit diesem Bit wird ein Monoflop getriggert, das die 50 ms dauernden Programmierimpulse erzeugt. Auf eine Software-Lösung wurde hier verzichtet (50 ms durch Zeitschleife gewinnen), da der ZX 81 im SLOW-Modus die Zeitschleife durch den Bildaufbau unterbrechen würde und so die Dauer des Programmierimpulses unzulässig verlängert werden könnte (Zerstörung des EPROMs).
- D3: Dieses Bit legt gemäß der Programmiervorschrift für 2716-EPROMs den Anschluß $\overline{CE}$ am EPROM auf 5 V.
- D4: Mit diesem Bit wird die Programmierspannung (normalerweise 26 V) zum EPROM durchgeschaltet. Sie darf während der gesamten Dauer der Programmierung anstehen. Wenn der Transistor T3 zum Abschalten der Programmierspannung durchschaltet, muß der Widerstand R8 die Verlustleistung von rd. 2,6 W aufnehmen können. Eine Überlastung von T3 ist nicht zu befürchten, da der Transistor im Schaltbetrieb arbeitet.
- D5: Dieses Bit steuert eine Warn-LED. Die LED leuchtet immer dann, wenn man ein EPROM weder in den Programmiersockel stecken noch es aus diesem entfernen darf. Erlaubt ist dies nur bei dunkler LED.

Für die Programmierspannung ist leider ein externes Netzgerät erforderlich, dessen Ausgangsspannung am Brenner



① **Schaltung des EPROM-Brenners:** Der Betriebssicherheit zuliebe werden sämtliche Signale, die der ZX 81 bereitstellt, über Treiberstufen gepuffert

einzuspeisen ist. Der genaue Wert der Programmierspannung hängt vom EPROM-Fabrikat ab (üblich sind Werte zwischen 21 V und 26 V). Versuche, die Spannung mit einem Spannungswandler von der ZX-81-Versorgungsspannung abzuleiten, schlugen leider fehl. Wieder abhängig vom EPROM-Fabrikat kam es zu so starker Belastung der Versorgungsspannung, daß die Programmierspannung auf Werte knapp unter 20 V zusammenbrach. Im Interesse der Betriebssicherheit wurde deshalb die weniger elegante Lösung mit externem Netzgerät gewählt.

Das Basic-Programm prägt den Bildschirmdialog

Im Basic-Programm für den Brenner (*Bild 2*) reserviert, nach dem Start mit RUN, Zeile 10 erst einmal Platz für den Maschinencodeteil des Steuerprogramms. Diese Maschinencodes (Zeile 30) werden wie üblich mit einer Eingaberoutine (Zeile 40 bis 60) in die REM-Zeile gebracht. Der Aufruf der ersten Maschinenroutine in Zeile 110 sorgt dann dafür, daß am Programmiersockel mit Ausnahme der Versorgungsspannung (Pin 24) sämtliche Spannungen abgeschaltet werden.

Dann verzweigt das Programm zu einer Unterroutine (ab Zeile 9000), die für die Eingabe der zu programmierenden Bytes zuständig ist. Die Frage nach der Startadresse wird am besten mit 30000 beantwortet, da hier im RAM normalerweise freier Speicherplatz ist, ohne daß mit dem Wert von RAMTOP jongliert werden muß.

Nach Eingabe der Startadresse taucht am Bildschirm eine Zeile auf, die ab der Startadresse den Inhalt von zwölf Speicherzellen zeigt. Signalisiert ein Dutzend Nullen, daß der gewählte Speicherbereich tatsächlich frei ist, dann dürfen jetzt die zwölf ersten Bytes des Anwenderprogramms eingetippt werden (Hexcodes). Diese Werte werden nach NEWLINE in die angezeigte Zeile übernommen (und in den Speicher); anschließend sind die nächsten zwölf Byte an der Reihe. Mit der Eingabe von „S" wird diese Routine verlassen.

Jetzt hat man die Wahl, nur einen Teil des soeben eingegebenen Maschinenprogramms zur Programmierung ins EPROM freizugeben bzw. ein vielleicht bereits zuvor vorhandenes anschließendes Maschinenprogramm gleich mit zu schießen. Die Zahl der zu programmierenden Bytes wird mit der Unterroutine ab Zeile 9600 in eine für das Maschinencode-Steuerprogramm verwertbare 2-Byte-Zahl aufgespalten und in den Speicherzellen 40D3h/40D4h auf Abruf bereit gehalten. Die Addition von 1 ist erforderlich, da die Maschinenroutine zuerst dekrementiert und anschließend auf 0 abfragt.

Zeile 155 fordert nun auf, das EPROM in den Sockel zu stecken. Nach dem anschließenden NEWLINE wird als Anfangsadresse für das Anwenderprogramm im EPROM der Wert 0 vorgeschlagen. Ein davon abweichender Wert ist dann zweckmäßig, wenn man ein

```
  10 REM 00000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000000000000000000000000000
  20 LET X=16518
  30 LET M$="3E183283C0C92A8440E
D5BD3403E223283C0CDAF401BCDAA401
DC29C4015C29C40C386403E233283C03
E223283C07E3283C023CDBD40C93E283
283C03E223283C00100330DC2CA4005C
2CA40C900000000ED4BD5403E183283C
03E183283C03E1A3283C00DC2E04005C
2E040C9"
  40 FOR I=1 TO LEN M$-1 STEP 2
  50 POKE X+INT ((I-1)/2),(CODE
M$(I)-28)*16+CODE M$(I+1)-28
  60 NEXT I
  70 PRINT AT 5,13;"ZX 81"
  80 PRINT AT 8,3;"EPROM - PROGR
AMMIERGERAET"
  90 PRINT AT 11,7;"FUER  2716 -
 EPROMS"
 100 PAUSE 250
 110 LET P=USR 16518
 120 PRINT "     EINGABE DES PROG
RAMMS"
 125 LET M=0
 130 PAUSE 250
 135 GOSUB 9000
 140 CLS
 141 PRINT "SOLLEN NUR DIE EINGE
GEBENEN ";M
 142 PRINT "BYTE PROGRAMMIERT WE
RDEN ? (J/N)"
 143 INPUT F$
 144 IF F$="J" OR F$="N" THEN GO
TO 146
 145 GOTO 143
 146 IF F$="J" THEN GOTO 149
 147 PRINT "WIEVIEL BYTE SOLLEN
PROGRAMMIERT WERDEN ? (2048 MAX)
"
 148 INPUT M
 149 GOSUB 9600
 150 CLS
 155 PRINT AT 13,4;"EPROM IN SOC
KEL STECKEN"
 156 IF INKEY$="" THEN GOTO 156
 157 CLS
 161 PRINT "SOLLEN DIE DATEN IM
EPROM AB"
 162 PRINT "ADRESSE 0 STEHEN ? (
J/N)"
 163 INPUT G$
 164 IF G$="J" OR G$="N" THEN GO
TO 166
 165 GOTO 163
 166 IF G$="J" THEN GOTO 170
 167 PRINT "AB WELCHER EPROM-ADR
ESSE SOLL   PROGRAMMIERT WERDEN
? (2047 MAX)"
 168 INPUT K
 169 GOSUB 9700
 170 PRINT AT 17,1;"ZUM STARTEN
 - S -  DRUECKEN"
 180 INPUT S$
 190 IF S$="S" THEN LET P=USR 16
524
 200 CLS
 210 IF S$="S" THEN PRINT "ENDE
DER PROGRAMMIERUNG"
 220 STOP
9000 CLS
9010 PRINT AT 21,0;"STARTADRESSE
 "
;
9020 INPUT A
9030 LET A=INT ABS A
9035 GOSUB 9500
9040 SCROLL
9050 PRINT A;TAB 8;
9060 FOR I=0 TO 11
9070 LET N=INT (PEEK (A+I)/16)
9080 PRINT CHR$ (N+28);CHR$ (PEE
K (A+I)-16*N+28);
9090 NEXT I
9100 INPUT D$
9110 IF D$="S" THEN RETURN
9120 IF D$="A" THEN GOTO 9010
9130 IF NOT LEN D$ THEN GOTO 924
0
9140 IF LEN D$<>2*INT (LEN D$/2)
 THEN GOTO 9100
9150 FOR I=1 TO LEN D$
9160 IF D$(I)<"0" OR D$(I)>"F" T
HEN GOTO 9100
9170 NEXT I
9180 PRINT AT 21,8;(D$+"
                ")( TO 24)
9190 FOR I=1 TO LEN D$ STEP 2
9200 POKE A,16*CODE D$(I)+CODE D
$(I+1)-476
9210 LET A=A+1
9215 LET M=M+1
9220 NEXT I
9230 GOTO 9040
9240 LET A=A+12
9245 LET M=M+12
9250 GOTO 9040
9500 LET H=INT (A/256)
9510 LET L=INT (A-(H*256))
9520 POKE 16517,H
9530 POKE 16516,L
9540 RETURN
9600 LET D=INT (M/256)+1
9610 LET E=INT (M-((D-1)*256))
9620 POKE 16596,D
9630 POKE 16595,E
9640 RETURN
9700 LET B=INT (K/256)+1
9710 LET C=INT (K-((B-1)*256))
9720 POKE 16598,B
9730 POKE 16597,C
9740 LET P=USR 16599
9750 RETURN
```

② **Basic-Listing:** In Zeile 30 ist bereits der Maschinencode zum Steuern des EPROM-Brenners enthalten

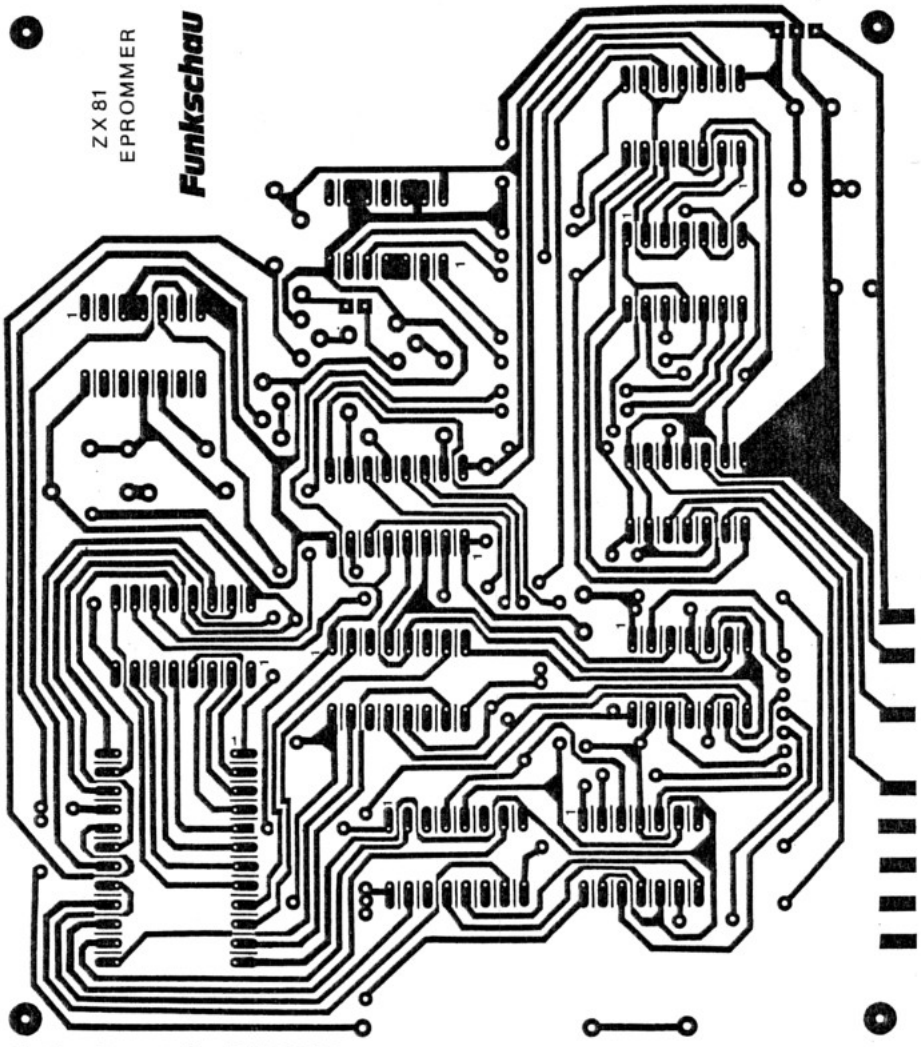
EPROM erst nach und nach mit Hilfsroutinen füllen möchte und nicht alle 2048 Speicherplätze auf einmal. Die Unterroutine ab Zeile 9700 übernimmt im Falle einer von 0 abweichenden Anfangsadresse die Einstellung des Adreßzählers auf den gewünschten Wert.

Mit dem Aufruf der eigentlichen Programmierroutine in Zeile 190 beginnt dann die Programmierung des EPROMs, die z. B. für 20 Byte etwa 5 s dauert. Zeile 210 bewirkt, daß man Gewißheit darüber erhält, daß die Programmierung des EPROMs abgeschlossen ist.

Das Basic-Programm hat lediglich die Aufgaben, das Maschinencode-Steuerprogramm im RAM zu verankern, den Bildschirmdialog zu ermöglichen und dem Maschinenprogramm Hilfsvariablen zur Verfügung zu stellen. Das eigentliche Kommando über den EPROM-Brenner führt das Maschinenprogramm, das aus sieben Routinen besteht (*Bild 3*).

Der Programmteil von Adresse 4086h bis 408Bh übernimmt, wie bereits angesprochen, das Abschalten der Spannungen am Programmiersockel. Das Datenregister wird gelöscht und der Adreßzähler im rückgesetzten Zustand gehalten: Der EPROM-Brenner kommt in einen definierten Grundzustand.

Der folgende Programmteil (408Ch bis 4095h) holt sich aus der Hilfsvariablen mit der Adresse 4084h/4085h den dort vom Basic-Programm in Zwei-Byte-Form untergebrachten Wert für die Startadresse. Eine weitere Hilfsvariable mit der Adresse 40D3h/40D4h hält die Zahl



Platinenlayout des EPROM-Brenners

der zu programmierenden Bytes bereit. Dazu ein Hinweis zu Bild 3: Die linke Spalte nennt die Adressen in Hex-Form, die mittlere Spalte zeigt die Mnemonics – aber mit Dezimalwerten – und die rechte Spalte gibt die Hex-Codes der Befehle an. Die eben genannten Adressen der Hilfsvariablen findet man also am schnellsten in der rechten Spalte, wie üblich freilich mit vertauschtem höher- und niederwertigerem Byte.

Nach Übernahme der beiden Hilfsvariablen ins HL- bzw. DE-Registerpaar werden die Spannungen am Programmiersockel angelegt und die Rücksetzbedingung für das Datenregister sowie den Adreßzähler aufgehoben. Mit den beiden folgenden Programmzeilen (4098h und 409Bh) wird das erste zu programmierende Byte „gebrannt" und die Zahl der Bytes um 1 verringert.

Der Programmteil zwischen Adresse 409C und 40A7 programmiert nun alle übrigen Bytes (bis der Inhalt des DE-Registerpaares den Wert 0 hat). Anschließend wird der Brenner wieder in den definierten Grundzustand gebracht.

Jetzt folgen die vom Programm verwendeten Unterroutinen. Der Programmteil von Adresse 40AAh bis 40ACh setzt die Taktleitung des Adreßzählers auf H-Pegel, d. h. an Pin 10 von IC 10 tritt eine steigende Flanke auf, die jedoch noch nichts bewirkt. Das bleibt dem Programmteil von Adresse 40AFh bis 40BCh vorbehalten. Er sorgt zunächst für eine fallende Flanke auf der Taktleitung, worauf IC 10 den Zählerstand um 1 erhöht. Jetzt wird das Byte, das programmiert werden soll, aus dem RAM ins Datenregister gebracht und das HL-Registerpaar per INC-Befehl auf die Adresse des nächsten zu programmierenden Bytes eingestellt.

Dann wird eine Unterroutine aufgerufen, die zunächst von Adresse 40BDh bis 40C7h reicht. Sie erzeugt einen positiven Impuls am Takteingang des Monoflops (IC 11, Pin 5). Das Monoflop gibt daraufhin an Pin 6 den 50 ms dauernden Programmierimpuls aus.

```
4086 LD A,024          3E18
4088 LD (49283),A      3283C0
408B RET               C9
408C LD HL,(16516)     2A8440
408F LD DE,(16595)     ED5BD340
4093 LD A,034          3E22
4095 LD (49283),A      3283C0
4098 CALL 16559        CDAF40
4098 DEC DE            1B
409C CALL 16554        CDAA40
409F DEC E             1D
40A0 JP NZ,16540       C29C40
40A3 DEC D             15
40A4 JP NZ,16540       C29C40
40A7 JP 16518          C38640
40AA LD A,035          3E23
40AC LD (49283),A      3283C0
40AF LD A,034          3E22
40B1 LD (49283),A      3283C0
40B4 LD A,(HL)         7E
40B5 LD (49282),A      3282C0
40B8 INC HL            23
40B9 CALL 16573        CDBD40
40BC RET               C9
40BD LD A,038          3E26
40BF LD (49283),A      3283C0
40C2 LD A,054          3E22
40C4 LD (49283),A      3283C0
40C7 LD BC,13056       010033
40CA DEC C             0D
40CB JP NZ,16586       C2CA40
40CE DEC B             05
40CF JP NZ,16586       C2CA40
40D2 RET               C9
40D3 NOP               00
40D4 NOP               00
40D5 NOP               00
40D6 NOP               00
40D7 LD BC,(16597)     ED4BD540
40DB LD A,024          3E18
40DD LD (49283),A      3283C0
40E0 LD A,027          3E1B
40E2 LD (49283),A      3283C0
40E5 LD A,026          3E1A
40E7 LD (49283),A      3283C0
40EA DEC C             0D
40EB JP NZ,16608       C2E040
40EE DEC B             05
40EF JP NZ,16608       C2E040
40F2 RET               C9
```

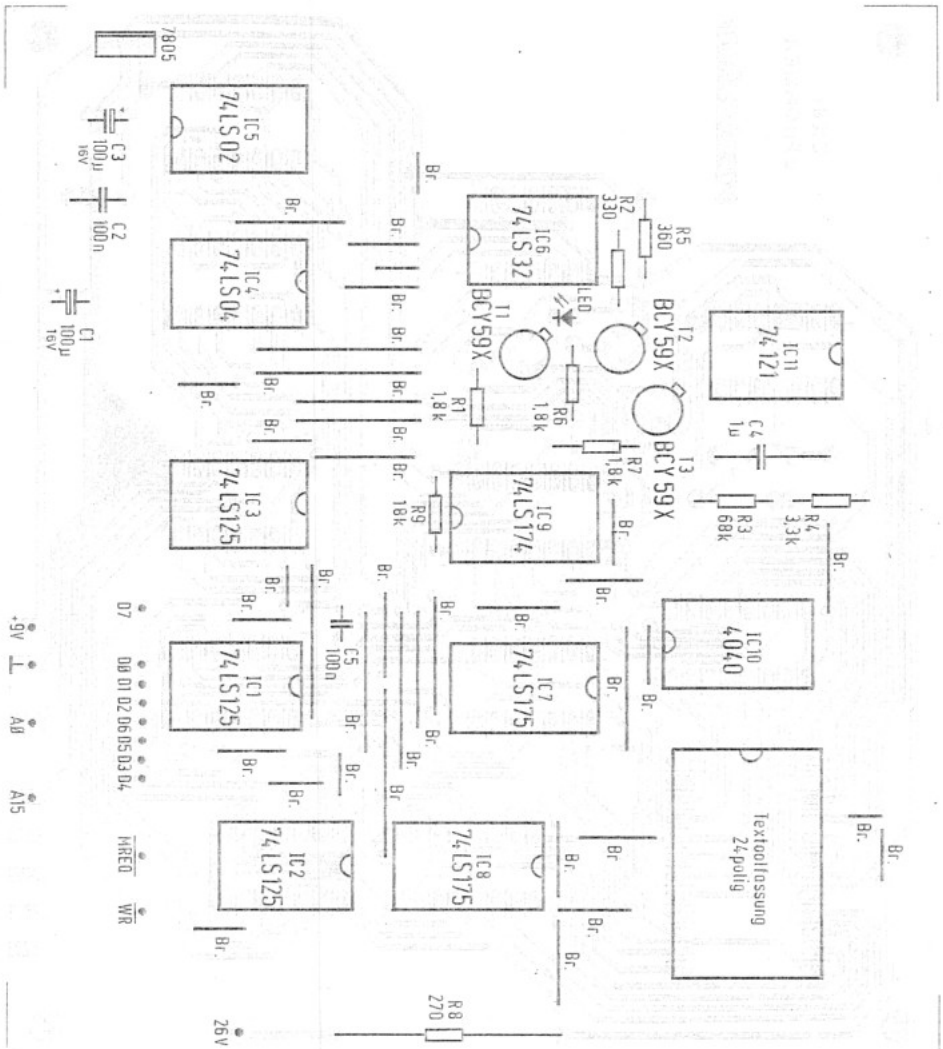
③ **Assembler-Listing:** Das Programm darf nicht wie gewohnt mit Adresse 4082h beginnen

Zwischen Adresse 40CAh und 40D2h durchläuft das Programm eine Warteschleife, um das Ende des Programmierimpulses abzuwarten. Damit Bauteiletoleranzen aufgefangen werden, dauert die Warteschleife sicherheitshalber etwa 54 ms.

Der vorletzte Programmteil zwischen Adresse 40D7h und 40DDh holt sich aus einer Hilfsvariablen bei Adresse 40D5h/40D6h die Information, wie viele Takte auf den Adreßzähler gegeben werden müssen, wenn diser voreingestellt werden soll, also praktisch die Adresse, ab der im EPROM die Bytes untergebracht werden sollen. Damit die Zählung auch bei 0 beginnt, wird der Adreßzähler zurückgesetzt.

Der letzte Programmteil ab Adresse 40E0h erzeugt den Takt für Pin 10 des Adreßzählers (IC 10), indem zuerst eine steigende und dann eine fallende Flanke „programmiert" wird. Danach wird die Information, wieviele Takte noch nötig sind, um 1 verringert. Hat diese Information den Wert 0 erreicht, dann ist der Adreßzähler so voreingestellt, wie es verlangt wurde.

Michael Knispel



Bestückungsplan für den EPROM-Brenner

Für engagierte Anwender:

Mit mc lernen Sie Computer von Grund auf verstehen. Ausführliche Funktionsbeschreibungen von Rechner-Hardware und gut kommentierte Programm-Listings bieten Ihnen den richtigen Einstieg ins ernsthafte Computern.

Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.

Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.

In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.

Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.
Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.
mc können Sie aber auch auf andere Art kennenlernen.
Kostenlos und unverbindlich.
Die Abrufkarte dafür finden Sie an der Umschlagklappe.

mc
Die Mikrocomputer-Zeitschrift