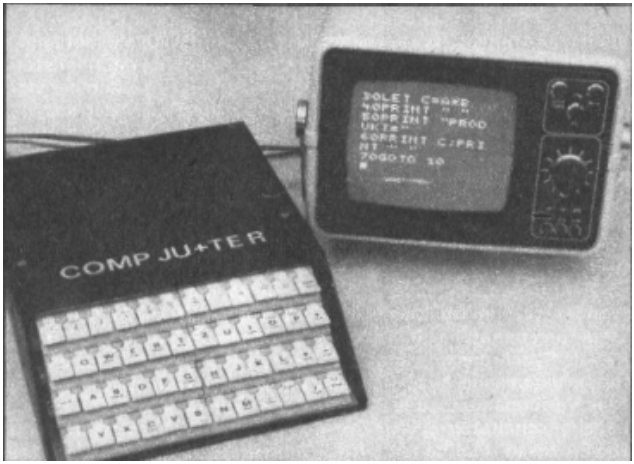
Dr. Helmut Hoyer

JU+TE Computer selbst gebaut
Teil II



Jugend+Technik 3/1988 - 7/1989

PDF-Version: Volker Pohlers, 2004

Inhalt

Graphik

Bereits im Heft 7/1987 wurde darauf hingewiesen, daß die weitestgehend programmtechnisch gestützte Bildschirmsteuerung des JU+TE-Computers ohne Hardware-Ergänzungen graphische Ausgaben gestattet. Der Bildwiederholspeicher (Adressen %FE00 bis %FFFF) enthält in jedem Bit die Helligkeitsinformation eines Bildpunktes. Man kann also mit den Prozeduren SETEB und SETEW direkt in BASIC-Anweisungen Bilder erzeugen. Abb. 1 gibt hierfür die Zuordnung von Speicheradressen zur Bildgeometrie. Dabei erscheint die Helligkeitsinformation des höchsten Bits jedes Bytes links, des niedrigsten rechts innerhalb des einer Speicherzelle zugehörigen Rechtecks. Acht Bytes enthalten die 64 Bildpunkte einer Zeile. 64 Zeilen gibt es insgesamt. Waagerechte Linien lassen sich mit einfachen Programmen erzeugen. Für das Variieren einzelner Bildpunkte gibt es aber nur recht umständliche BASIC-Anweisungen. Deswegen stellen wir hier das Einbeziehen von Maschinenprogrammen für Punktgraphik-Ausgaben vor.
Abb. 2 enthält sie in hexadezimaler Darstellung für den Speicherbereich von %FCA0 bis %FCFF. Im Betriebssystem des JU+TE-Computers gibt es kein Programm zur Eingabe solcher Informationen. Man kann sich aber leicht behelfen:
Zuerst wird das BASIC-Programm "HEX-EINGABE“ (Abb.3) eingetastet. Es gestattet das Eintragen jeweils zweier Bytes über INPUT-Anweisung in den RAM des Computers.

•

%FE00	%FE01	…	%FE07
%FE08	%FE09	…	%FE0F
%FE10			
…			
…			
…			
…			
%FFF8	%FFF9	…	%FFFF

f

```
10  PRINT “HEX-EINGABE“'
20  INPUT “AB ADR." A
30  PTH A,;INPUT ":" B
40  PROC SETEW[A,B]
50  LET A=A+2; GOTO 30
```

„

```
10  CALL%8DD
20  LET X=31,Y=31
30  LET A=GTC¤A3
40  PROC SETR[%5B,0]
50  IF A=0 THEN LET X=X+1
60  IF A=1 THEN LET X=X-1
70  IF A=2 THEN LET Y=Y+1
80  IF A=3 THEN LET Y=Y-1
90  CALL %FCBB
100 IF Z=0 THEN CALL %FCA0
110 ELSE ; CALL %FCB0
120 GOTO 30
```

,

```
FCA0: 70 FD 31 70 D6 FC CE 60 E2 52 32 92 30 50 FD AF
FCB0: 70 FD 31 70 D6 FC CE 42 32 8B F0 70 FD 31 70 D6
FCC0: FC CE B0 52 B0 53 72 23 EB E3 20 53 8B DF 0C FF
FCD0: 1C F8 B0 E2 38 51 56 E3 3F 90 E3 90 E3 10 E3 10
FCE0: E2 22 13 32 02 2C 01 38 4F 56 E3 3F 3E 00 E3 E0
FCF0: E2 76 E3 07 EB F7 90 E3 F0 E3 02 13 82 30 AF 00
```

Alle Angaben sollten hexadezimal gemacht werden, nach RUN also %FCA0 ENTER %70FD ENTER %3170 ENTER ... %AF00 ENTER. Zum Schluß wird mit RESET wieder der Grundzustand erreicht. Damit befinden sich die Maschinenprogramme zur Punktgraphik im Speicher. Die HEX-Eingabe hat ihren Zweck erfüllt und kann mit NEW wieder gelöscht werden. Damit stehen nun noch bei minimaler Speicherausrüstung 160 Bytes für ein BASIC-Programm mit Graphik-Ausgabe zur Verfügung.
Mit CALL-Anweisungen lassen sich hier über die Variablen X, Y (Koordinaten) und Z (Helligkeitswert) folgende Funktionen ausführen:
CALL%8DD : Bildschirm löschen
CALL %FCA0 : Bildpunkt (X, Y) setzen
CALL %FCB0 : Bildpunkt (X, Y) löschen
CALL %FCBB : Bildpunkt (X, Y) lesen
Beim Lesen eines Bildpunktes erhält Z den Wert 1, wenn gesetzt, und den Wert 0. wenn gelöscht. Ein Beispiel soll die Handhabung verdeutlichen (Abb. 4):
Die Anweisung 10 löscht den Bildschirm. die folgende setzt die Koordinaten X und Y auf Bildmitte. Es folgt das Holen eines ASCII von der Tastatur. Die AND-Verknüpfung mit 3 reduziert das Ergebnis auf den Bereich von 0 bis 3. Die Anweisung 40 löscht den Bildschirmpointer des Betriebssystems, damit alle Zeichendarstellungen in Folge der Tastenbetätigungen nur in der linken oberen Ecke erfolgen. Die nächsten vier Anweisungen variieren die Koordinaten

abhängig von der betätigten Taste. Danach wird mit CALL %FCBB der so adressierte Bildpunkt gelesen. Die Anweisung 100 bewirkt das Setzen. sofern er gelöscht war. Die nächste Anweisung behandelt den entgegengesetzten Fall, so daß insgesamt die Umkehrung des aktuellen Helligkeitswertes erfolgt. Das Programm gestattet somit das schrittweise Füllen des Bildschirms mit Polygonzügen.
Abschließend noch ein wichtiger Hinweis: mit SAVE können nur BASIC-Programme, nicht aber die in Abb. 2 dargestellten Maschinenprogramme auf Kassette gespeichert werden. Deren Hex-Eingabe ist damit nach jedem Einschalten des JU+TE-Computers neu erforderlich.

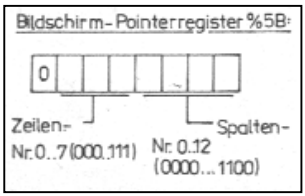
Reaktionsspiele

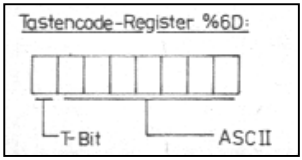
Obwohl die Programmiersprache BASIC die Rechenleistung nur sehr uneffektiv nutzt und der Einchip-Mikrorechner des JU+TE-Computers fast 90 Prozent der Zeit für die Bildschirm-Ausgabe aufwendet, können einfache Reaktionsspiele programmiert werden. Neben Fragen der Bildgestaltung spielen dabei die dynamische Tastenabfrage, das Erzeugen von Zufallszahlen und die Ausgabe akustischer Signale eine Rolle. Anhand zweier Beispiele werden geeignete Programmelemente vorgestellt.

Mondlandung

Die in unserem Beitrag zur Bildschirmgraphik dargestellten Möglichkeiten wollen wir zur Anzeige von Raumschiff und Mondoberfläche beim Programmbeispiel Mondlandung verwenden. Während links auf dem Bildschirm die Zahlenangaben erscheinen, soll die Mondlandung rechts als Graphik ablaufen. Das Byte rechts unten benutzt der Computer zur Anzeige des SHIFT-Zustandes, die zugeordnete RAM-Speicherzelle mit der Adresse %FFFF als Merkzelle. Um keinen Höhenschritt aufgeben zu müssen, weichen wir nach links aus. Im Bildwiederholspeicher werden helle Punkte mit 0 kodiert; durch
30 PROC SETEB[%FFFE,0]
entsteht daher ein weißer Strich als Symbol der Mondoberflache. Das Raumschiff soll als senkrechte Linie erscheinen. Dazu muß eine X-Koordinate gewählt werden, die bei der Zeichenausgabe nicht benutzt wird. Sonst würden sich Zahlenangaben und Graphik gegenseitig stören. Die Anweisung 20 bestimmt mit X=54 eine günstige Position. Mit
40 LET A=4,Y=H/10
werden die Anzahl der verwendeten Bildpunkte und die Bildschirmhöhe des Raumschiffes festgelegt. Die Anweisung 50 verhindert diese Darstellung, wenn das Raumschiff den Bildbereich verläßt.
60 CALL %FCA0;LET Y=Y+1,A=A-1;IF A>0 THEN GOTO 60
ist ein Anweisungszyklus, der mit dem Graphik-Unterprogramm ab Adresse %FCA0 vier Bildpunkte übereinander setzt. Das Löschen geschieht weiter unten mit der Anweisung 120 in ähnlicher Weise.
Wegen der graphischen Anzeige darf das Bild nicht rollen. Die Zahlenangaben müssen immer auf der gleichen Position erscheinen. Das TINY-MP-BASIC hat keine PRINT AT- und keine WINDOW-Anweisungen, doch läßt sich mit der Standard-Prozedur SETR der Bildschirmpointer setzen. Bei jeder Zeichenausgabe bestimmt der Inhalt des Registers %5B die Bildschirmposition. Nach der Ausgabe jedes Zeichens aktualisiert das Betriebssystem diesen Pointer. Die Anweisung 70 bewirkt daher, daß die folgenden PRINT-Anweisungen in der zweiten Zeile beginnen.

Das nächste Problem besteht in der dynamischen Tastenabfrage. Die Funktion GTC eignet sich nicht für Reaktionsspiele, da das Programm erst nach erfolgter Betätigung fortgesetzt würde. Hier fehlt die INKEY-Anweisung. Statt dessen kann man die Betriebssystem-Routine ab Adresse %C56 nutzen. die mit dem Tastencoderegister %6D arbeitet. Es speichert den Zeichencode (ASCII) der betätigten Taste und den Zustand der Tastatur (T-Bit=1 so lange Taste betätigt). Normalerweise verhindert das Betriebssystem hiermit mehrfache Auswertungen einer Tasteneingabe. Um eine dynamische Abfrage zu erreichen, muß vor dem Aufruf der genannten Betriebssystem-Routine das Tastencoderegister %6D gelöscht werden. Dies geschieht in der Zeile 90 mit der Prozedur SETR. Die AND-Verknüpfung mit 15 reduziert den Code auf die untersten vier Bit, so daß eine Zahl zwischen 0 und 15 entsteht. Die Multiplikation mit 3 erhöht diesen Wert auf eine für das Spiel günstige Größe. Wenn zum Zeitpunkt des Ausführens dieser Programmzeile keine Taste betätigt ist, erhält die Variable E den Wert 0. Im Spiel kann mit den Zifferntasten die Sinkgeschwindigkeit V dosiert verringert werden. Die übrigen Veränderungen gegenüber der Version aus Heft 12/1987 betreffen die Auswertung des Spiels.





RAM- Nutzung

%FE00...%FFFF: graphischer Bildwiederholspeicher
%FD80...%FDFF: Stapelspeicher (Stack)
%FD00...%FD7F: ASCII-Bildspeicher
%E000...%FCFF; BASIC-RAM (je nach Ausstattung)

Programmbeispiel Mondlandung

```
10 CALL %8DD;PRINT "MONDLANDUNG"
20 LET X=54,V=0,H=400,T=500
30 PROC SETEB[%FFFE,0]
40 LET A=4,Y=H/10
50 IF Y>60 THEN GOTO 70
60 CALL %FCA0;LET Y=Y+1,A=A-1;
   IF A>0 THEN GOTO 60
70 PROC SETR[%5B,%10]
80 PRINT "V="V;PRINT "H="H;
   PRINT "T="T
90 PROC SETR[%6D,0];CALL %C56;
   LET E=GETR[%6D]$A15*3
100 IF T<E THEN LET E=T
110 LET H=H-V-5+(E/2),V=V+10-E,
    T=T-E,A=4
120 LET Y=Y-1;CALL %FCB0;
    LET A=A-1;IF A>0 THEN GOTO 120
260 IF H>1000 THEN PRINT "RAUMSCHIFF
    VERSCHOLLEN !";GOTO 330
270 IF H>0 THEN GOTO 40
280 PRINT "KRATERTIEFE:"V," METER!"
290 IF V>50 THEN PRINT "WELTRAUM-
    ROWDY !";GOTO 330
300 IF V>30 THEN PRINT "SONNTAGS-
    FLIEGER !";GOTO 330
310 IF V>10 THEN PRINT "PROFI !"
320 ELSE; PRINT "KUENSTLER !"
330 PROC SETEB[%FFFE,V];WAIT 800;
    GOTO 10
```

Hase und Wolf

In TINY-MP-BASIC lassen sich Zeichenausgaben einfacher programmieren als Graphiken. Deshalb soll unser zweites Beispiel zeigen, wie man mit Symbolgraphik auch kompliziertere Spiele mit ausreichender Rechengeschwindigkeit programmieren kann.
Die Spielidee besteht darin, daß der vom Bediener gesteuerte Hase möglichst viele Doppelpunkte einsammelt, ohne sich vom computergesteuerten Wolf fangen zu lassen. Das Spielfeld besteht aus allen 8 x 13 Zeichenpositionen des Bildschirms, nur die rechte untere Ecke bleibt ungenutzt. Eine Ausgabe auf diese Position würde störendes Bildrollen verursachen. Zum Verständnis sind die Variablen wichtig: A: Bestwert, B: Spielstand, C: Zähler, D: Hase-Position, E: Wolf-Position, F: Anzahl dar restlichen Punkte, G: ASCII von der Tastatur, H: Bewegungsrichtung, J: Position für Unterprogramm, K: Zufallszahl, L: ASCII auf Bildposition J.

Programmbeispiel Hase und Wolf

```
10 LET A=0
20 LET B=0
30 PROC PTC[l2];PRINT "HASE UND WOLF"
40 LET C=103,F=101
50 PROC PTC[%3A];LET C=C-1;IF C>0 THEN GOTO 50
60 PROC SETR[%F3,255]
70 PROC SETR[%F2,3]
80 PROC SETR[%F1,15]
90 LET D=0,E=%7B
100 PROC SETR[%5B,0];PROC PTC[%48]
110 CALL %C56;LET G=GETR[%6D]$A%7F
120 LET H=0
260 IF G=%58 THEN LET H=1
270 IF G=%59 THEN LET H=-1
280 IF G=%0D THEN LET H=16
290 IF G=%2D THEN LET H=-16
300 LET J=D;GOSUB 820:LET D=J
310 PROC PTC[%48);PROC SETR[%6D,0]
320 IF L=%3A THEN LET B=B+1,F=F-1;PROC SETR[%F1,%8A]
330 IF D=E THEN GOTO 770
340 IF F=0 THEN GOTO 620
350 LET K=GETR[%F2]
360 IF K=1 THEN GOTO 560
370 IF K=2 THEN GOTO 540
380 LET H=16
520 IF D<E THEN LET H=-16
530 GOTO 560
540 LET H=1
550 IF D$A15<E$A15 THEN LET H=-1
580 LET J=E;GOSUB 820;LET E=J
570 PROC PTC[%57];PROC SETR[%F1,10]
580 IF L=%3A THEN LET F=F-1
590 IF D=E THEN GOTO 770
600 IF B<99 THEN WAIT 99-B
610 IF F>0 THEN GOTO 110
620 PROC PTC[12];PRINT "GRATULIERE !"B," PUNKTE"
630 WAIT 200;GOTO 30
770 PROC PTC[12];PRINT "PUNKTZAHL:   "B
780 IF A<B THEN LET A=B;PRINT "** REKORD! ** "
790 PRINT "BESTWERT:   "A
800 WAIT 400
810 PROC SETR[%6D,0];LET H=GTC;GOTO 30
820 PROC SETR[%5B,J]
830 PROC PTC[%20];LET L=%20
840 IF J+H$A15>12 THEN GOTO 890
850 IF J+H<0 THEN GOTO 890
880 IF J+H>%7B THEN GOTO 890
870 LET J=J+H
880 LET L=GETEB[%FD00+J]
890 PROC SETR[%5B,J];RETURN
```

Zunächst werden die Spielstand-Variablen gelöscht. Die Prozedur PTC[12] bewirkt wie CALL %8DD das Löschen von Bildschirm und Bildschirmpointer. Die Zeile 50 beschreibt das gesamte Spielfeld mit Doppelpunkten (ASCII: %3A).

Die Variable F wird auf einen um zwei geringeren Wert gesetzt, da die Startpositionen von Hase und Wolf abgerechnet werden müssen. Die folgenden drei Zeilen initialisieren den Zufallsgenerator, den der freie Zähler T1 realisiert.
60 PROC SETR [%F3,255] bewirkt ununterbrochenes Zählen mit einer Taktperiode von 63 µs.
70 PROC SETR[%F2,3] bestimmt den Zählumfang und damit die Anzahl der möglichen Zufallszahlen. Er kann zwischen 1 und 256 liegen. Für die Bewegung des Wolfes wird in der Zeile 350 eine von drei Taktiken ausgewählt. Das Lesen des Zählregisters %F2 hat den aktuellen Zählerstand als Ergebnis.
80 PROC SETR[%F1,15] gibt beide interne Zähler (T0 für Bilderzeugung und T1 als Zufallsgenerator) frei. In der nächsten Zeile erhalten Hase und Wolf ihre Startpositionen, Zeile 100 bewirkt die Anzeige eines H als Symbol für den Hasen in der linken oberen Ecke.
110 CALL %C56;LET G=GETR[%6D] ¤A%7F ermittelt in G den ASCII der gerade betätigten Taste. Die Zeilen 120 bis 290 leiten daraus die Richtung H des Hasens derart ab, daß X einen Schritt nach rechts, Y einen nach links, ENTER einen nach unten, – einen nach oben und alles andere keine Veränderung bewirken. Die Zeile 300 ruft das Unterprogramm zur Bestimmung der neuen Position des Hasens.
820 PROC SETR[%5B,J] setzt den Bildschirmpointer zunächst auf die alte Position. Hier bewirkt die nächste Anwei-

sung das Ausgeben eines Leerzeichens (%20). Die Zeile 840 testet die neue Position J+H auf Überschreitung des linken oder rechten Bildrandes, 850 auf die des oberen und 860 auf die der unteren Begrenzung. Nur bei Bewegungen innerhalb des Spielfeldes erhält J in der Zeile 870 den Wert der neuen Position.
880 LET L= GETEB[%FD00+J] liest aus dem ASCII Bildspeicher das dort befindliche Zeichen und speichert es in der Variablen L. Zeile 890 aktualisiert den Bildschirmpointer und bewirkt den Rücksprung ins Hauptprogramm. Die Zeile 310 schreibt das H (%48) auf die neue Hasenposition und löscht das Tastencoderegister. Hiermit werden die nächste dynamische Tastenabfrage vorbereitet und der Tasten-Piepton abgeschaltet. In der Zeile 320 erfolgt der Test. ob auf der neuen Hasen-Position ein Doppelpunkt (%3A) war. Gegebenenfalls werden der Spielstand erhöht, die Anzahl der verbleibenden Punkte verringert und mit PROC SETR[%F1,%8A] der Anschluß P36 als Ausgang des Zählers T1 vereinbart. Im folgenden werden bei Übereinstimmung von Hase- und Wolf-Position (D=E) das Spiel beendet und ausgewertet und bei vollständig abgesammelten Feld (F=0) mit einem neu gefüllten fortgesetzt. Sonst schließt sich die Bewegung des Wolfes abhängig von der ermittelten Zufallszahl K an. Bei K=1 geht der Wolf in die gleiche Richtung wie der Hase. Das bewirkt ein Weg-Abschneiden und gibt dem Wolf einen listigen Zug. Bei K=2 wird die horizontale, bei K=3 die vertikale Koordinate verbessert. Wenn die bereits optimal war, entsteht dabei eine ungünstige Bewegung. Das gibt dem Wolf einen verspielten Charakter und dem Hasen Tempovorteile. Die Zeile 560 aktualisiert entsprechend die Wolf-Position, die Zeile 570 kennzeichnet diese durch Anzeigen eines W (%57).
PROC SETR[%F1,10] schaltet P36 wieder als zählerunabhängigen Ausgang und gestattet beiden Zählern ansonsten ungestört fortgesetzte Funktion. Damit endet der bei erfolgreichem Sammeln des Hasen erzeugte Ton. Die Zeile 580 berücksichtigt das Absammeln durch den Wolf, wobei der Hase natürlich keine Punkte erhält. Die Zeile 600 gibt dem Spieler bei kleinem Spielstand zusätzliche Reaktionszeit. Solange sich noch Doppelpunkte auf dem Feld befinden (F>0), erfolgt danach der nächste Schritt des Hasen.
Die Zeilen 620 bis 810 dienen der Spielstandanzeige bei abgesammeltem Feld und der Beendigung des Spiels, wenn der Hase eingefangen ist. Die Zeile 810 bewirkt mit der Funktion GTC das Abwarten einer Tastenbetätigung, bevor ein neues Spiel gestartet wird.
In den abgebildeten Programmlisten erscheint anstelle des ¤ das übliche $. Beide Programme passen nicht in den RAM der Minimalkonfiguration, aber 1 KByte BASIC-RAM reicht bereits aus (kleinste Erweiterungsstufe). Die Mondlandung funktioniert nur mit den auf Seite 232 vorgestellten Graphik-Unterprogrammen.
Beim Programmieren eigener Spielideen ist zu beachten, daß der Zufallsgenerator nur dann pseudozufällige Zahlen zum Ergebnis hat, wenn der Programmablauf vor dem Lesen des Zählers von einer Bediener-Reaktion (INPUT oder GTC) verzögert wird. Der Zähler arbeitet determiniert, die Zufallsquelle ist die Reaktionszeit des Spielers! Man kann aber bei entsprechend eingeengtem Umfang gleich zwei oder mehrere Zahlen aus dem Zählerstand ableiten. Ist der Zählumfang mit PROC SETR[%F2,100] auf 100 festgelegt, erhält man mit
300 LET A=GETR[%F2]
310 LET B=A/10;IF B=0 THEN LET B=10
320 LET A=A¤M10+1
in A und B je eine Zahl zwischen 1 und 10. Die Ausgabe von Tönen setzt ebenfalls die Initialisierung des Zählers T1 voraus (vgl. Zeilen 60 und 70). Die Tonhöhe hängt vom Zählumfang ab. Der Ton beginnt mit dem Laden des Zähler-Steuerregisters %F1 mit der Zahl %8A und endet durch Laden von 10 (%0A). Dazwischen erfolgt die akustische Ausgabe unabhängig vom Programmablauf.

Dr. Helmut Hoyer

JU+TE-Computer-Tips

Nutzen der Ein-/Ausgabe-Signale

Auf der Prozessorplatine des JU+TE-Computers sind die 16 Anschlüsse der Ports 2 und 3 (P20...P27 und P30...P37) zugänglich. Sie können für die Ein- und Ausgabe von Signalen im TTL-Pegel benutzt werden. Das Betriebssystem belegt mit der Bildausgabe die Ausgänge P36 (akustische Ausgabe) und P37 (Synchronsignal und SAVE-Ausgang). Das Magnetbandinterface nutzt außerdem P30 für die Eingabe vom Magnetband. Die übrigen 13 Signale stehen bislang ungenutzt für den Anschluß weiterer Peripherie zur Verfügung.
Beim Port 3 liegen die Übertragungsrichtungen fest. P30 bis P33 sind Eingänge, P34 bis P37 Ausgänge. Sie können für spezielle Zwecke genutzt werden. P34 läßt sich z. B. als zusätzliches Adreßsignal (Bankumschaltung RAM/ROM), P30 bis P33 als Interrupteingang verwenden. Das wird mit dem Laden verschiedener Prozessor-interner Register vereinbart, was in der JU+TE später im Rahmen eines „ABC Einchip-Mikrorechner" erklärt wird.
Die Signale P20 bis P27 lassen sich in beiden Übertragungsrichtungen nutzen. Sie stehen beim JU+TE-Computer völlig uneingeschränkt zur Verfügung. Mit dem Laden des Registers Nr. %F6 wird festgelegt, welche Port 2-Signale Ausgänge sind. Eine 0 in diesem 8-Bit-Register bewirkt die Ausgaberichtung auf der entsprechenden Bitposition von Port 2. Mit SETR[%F6,7] werden z. B. P23 bis P27 Ausgänge, P20 bis P22 bleiben reine Eingänge. Sie sind über Registeradresse 2 zugänglich. Mit der Prozedur SETR[2, Ausdruck] gelangen die unteren acht Bit von Ausdruck in das Ausgaberegister von Port 2 und widerspiegeln sich an den in Ausgaberichtung vereinbarten Anschlüssen bis auf Widerruf mit den zugeordneten Spannungspegeln (0: 0...0,4V, 1: 2,4...5V). Die Eingabe gelingt mit der Funktion GETR[2]. Dabei erhält man auf den unteren acht Bit die aktuelle Belegung der Signale P20 bis P27. Das ist bei den Ausgabesignalen die zuletzt ausgegebene Information, bei den Eingabesignalen der von außen zugeführte Pegel. P30 bis P37 erreicht man in gleicher Weise über Registeradresse 3.
Bei der Beschaltung der Portanschlüsse ist zu beachten, daß nie Spannungen über 5P- oder unter 00-Potential auftreten. In Eingaberichtung betriebene Einchiprechner-Anschlüsse stellen eine rein kapazitive Last (ca. 5pF) dar, hier fließt praktisch kein Strom. Ausgänge können bei 0-Pegel mit 2 mA, bei 1-Pegel mit 0,25 mA belastet werden. Ein Lastwiderstand gegen 00 muß daher mindestens 7,5 kΩ, gegen 5P mindestens 2,4 kΩ betragen.
Das Schaltbild zeigt einen einfachen Verstärker, der den Anschluß von Leuchtdioden, kleinen Glühlampen oder Relais bis zur Leistungsgrenze der Stromversorgung gestattet. Beschaltet man damit die Port 2-Signale, läßt sich bereits ein einfaches, aber variantenreiches Lauflicht erzeugen.
Die programmtechnischen

Programmbeispiel Lauflicht

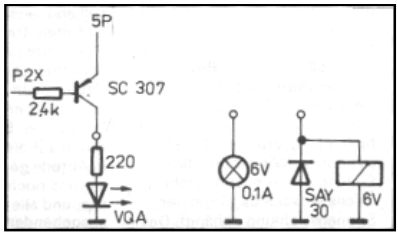
```
10  PRINT "LAUFLICHT"
20  PROC SETR[%F6,0]; LET A=1
30  INPUT "DAUER:"B
40  LET C=A¤A%110
50  IF C=%100 THEN LET
    A=A¤O%8000
80  IF C=%10 THEN LET
    A=A¤O%8000
70  LET A=RL[A¤A%8FFF]
80  PROC SETR[2,A]
90  WAIT B; PROC SETR[%6D,0]
100 CALL %C56
110 IF GETR[%6D]=0 THEN
    GOTO 40
120 GOTO 30
```

Port 2-Steuerregister %F6:

E/A	E/A	E/A	E/A	E/A	E/A	E/A	E/A
P27	P26	P25	P24	P23	P22	P21	P20

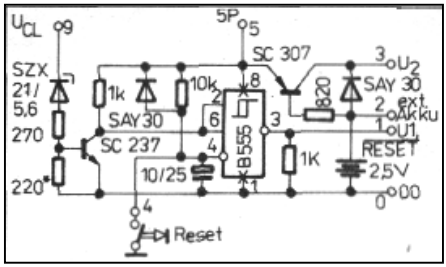
Ø: Ausgabe
1: Eingabe

Stromlaufplan des Ausgabeverstärkers



Zeichnungen: Hoyer, Schmidt

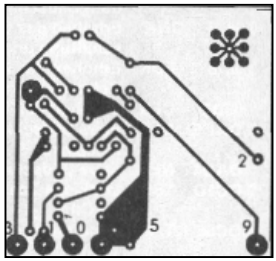
Möglichkeiten bieten diesbezüglichen Ideen breiten Realisierungsspielraum. Unser Beispiel erzeugt eine pseudostochastische Binärfolge (BPSS) mit wählbarer Taktfrequenz.
Die Anweisung 20 vereinbart alle Port 2-Anschlüsse als Ausgang und gibt unserem Schieberegister (Variable A) einen von Null verschiedenen Startwert. Es folgt die Eingabe der Taktperiode. Sinnvoll sind hier Zahlen zwischen 1 und 100. Ab der Anweisung 40 wird die Rückführung des Schieberegisters berechnet. Gut eignet sich hierzu die Antivalenz (XOR) der Bits 4 und 8. Die Variable C erhält diese beiden Stellen des Schieberegisters A, während alle anderen Positionen durch die AND-Verknüpfung mit 0 belegt werden. Von den vier möglichen Kombinationen der beiden interessierenden Bits erfüllen zwei die XOR-Verknüpfung. In diesen Fällen wird in der Anweisung 50 oder 60 die Position 15 (ganz links) unseres Schieberegisters gesetzt.
Die Anweisung 70 besorgt das Verschieben. Die AND-Verknüpfung löscht zunächst zur Vorbereitung des nächsten Zyklus die Position 14 (die Bits 9 bis 13 spielen keine Rolle). Die Prozedur RL bringt die Information von Stelle 14 auf Stelle 15 usw. von Stelle 1 auf Stelle 2, von 0 auf 1, und auf Position 0 erscheint der ursprüngliche Inhalt von Bit 15. Die Ausgabe erfolgt anschließend mit der Prozedur SETR[2,A].
Nach der Warteanweisung folgt die dynamische Tastenabfrage. Solange keine Taste betätigt wird, setzt das Programm ab Zeile 40 seine pseudostochastische Operation mit unveränderten Parametern fort. Sonst erwartet die Anweisung 30 eine neue Periodendauer-Eingabe.

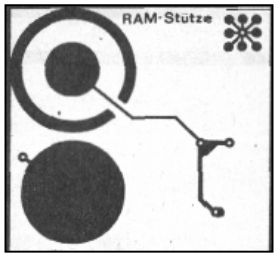


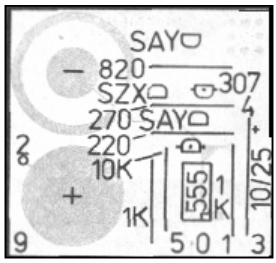
Stromlaufplan der RAM-Stütze

Batterie-Stützung für CMOS-RAMSchaltkreise

Ist der JU+TE-Computer mit CMOS-RAM-Schaltkreisen (U 224 D,U 6516 D oder HM 6264 LP)







Leiterplatte der RAM-Stütze (von oben nach unten) Lötseite, Bestükkungsseite, Bestückungsplan (SC 237 nicht gekennzeichnet)

ausgerüstet, kann der Speicherinhalt auch bei abgeschaltetem Gerät gesichert werden. Damit bleiben Programme beliebig lange erhalten, ohne daß sie jedes Mal von Kassette geladen werden müssen. Dazu erhalten die RAM-Schaltkreise ihre Betriebsspannung (U2) bei abgeschaltetem Computer aus Akkumulatoren. Außerdem muß unkontrolliertes Beschreiben der RAM-Zellen, was gewöhnlich beim Ein- und Ausschalten auftreten kann, vermieden werden. Unser Schaltungsvorschlag erzeugt dazu ein RESET-Signal für den Einchip-Mikrorechner. wenn die Spannung am Ladekondensator zu klein ist, die Betriebsspannung einen Kurzschluß hat oder die RESET-Taste betätigt ist. Dieses Signal wird beim Einschalten lange genug aktiv gehalten, daß ein regulärer Programmstart stattfindet. Das Rücksetzsignal erhalten auch die Speichermodule als U1 zum Passivieren der Schaltkreisfreigabe. Der HM 6264 LP

besitzt einen hierfür geeigneten 1-aktiven CS-Eingang. Bei den anderen Typen muß der Dekoder DS 8205D benutzt werden. Hier sind Pin 6 mit U1 statt 5P zu verbinden und alle benutzten Auswahlausgänge über Widerstände (ca. 10 kΩ) an U2 zu legen. In allen Fallen erfolgt die Spannungsversorgung der RAM-Schaltkreise mit U2. Die Leiterplatte wurde so gestaltet, daß sie direkt an die vier Versorgungsspannungs-Anschlüsse neben dem DS 8212 D auf der Prozessorplatine angeschlossen und dort senkrecht aufgesetzt werden kann. Der Elko am Pin 6 des Prozessors ist zu entfernen. Statt dessen erhält dieser Anschluß U1 als /RESET. Die Rücksetztaste wird jetzt am Anschluß 4 der Stützplatine kontaktiert. Ein weiterer Draht verbindet den Anschluß 9 mit dem Pluspol des Ladekondensators (2200/10) oder mit Pin 3 des Spannungsreglers B 3170 V. Unter Umständen muß der 220-Ohm Widerstand der Stützschaltung variiert werden, um das Erzeugen des Nullpegels an U1 bei einer Ladekondensatorspannung von 7,5 bis 8 V zu erreichen. Als Akkus können z. B. mit Hilfe aufgelöteter Metallstreifen und einer Bronzefeder zwei „Kosmos"-Zellen direkt auf der Platine befestigt werden. Sonst dienen zwei in Reihe geschaltete NC- oder Blei-Zellen als Stützbatterie am Anschluß 2. Der 820-Ohm Widerstand dient dem Nachladen.

Dr. Helmut Hoyer

Allgemein nutzbare Unterprogramme

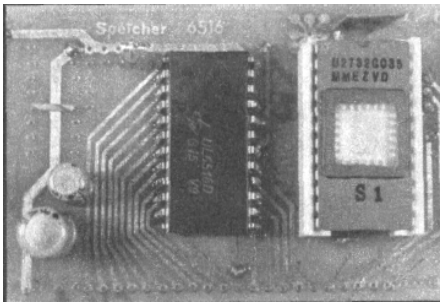
Einige Unterprogramme des Betriebssystems unseres Selbstbau-Computers können über CALL-Anweisungen effektiv in BASIC-Programmen verwendet werden. Sie benutzen die Register %10 bis %1F und %54 bis %6F ohne den zur Speicherung der Variablen reservierten Bereich (%20 bis %54) zu beeinflussen. Die für die Informationsübergabe genutzten Register sind in der folgenden Aufstellung angegeben.

Adresse	Name	Funktion
%0824	TBS	statische Tastenabfrage mit Ausführung auf dem Bildschirm. %5B: Bildschirmpointer, %5A: ASCII
%0827	ZBS	Ausgabe des in %5A übergebenen ASCII auf dem Bildschirm mit %5B als Pointer
%0875	INK	Erhöhen des Bildschirmpointers mit ggf. Zeilenschaltung und Bildrollen (Pointer: %5B)
%0878	EDK	Erhöhen des Bildschirmpointers um mehrere Schritte. Pointer: %5B, Schrittanzahl: %5C
%08DD	CLS	Löschen von Bildschirm und Pointer %5B
%0ACE	NLE	neue Bildschirmzeile (new line) ohne Freizeilensperre
%0AD4	NLN	neue Bildschirmzeile mit Freizeilensperre
%0C1D	TSK	statische Tastenabfrage. %6D: T-Bit, %5A: ASCII
%0C56	TAS	dynamische Tastenabfrage mit %6D: ASCII und T-Bit

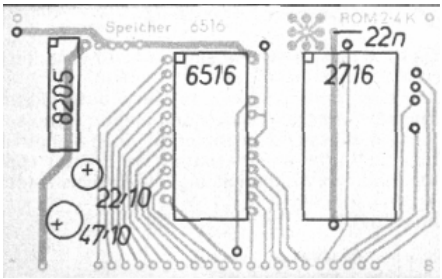
Hardware

Speichermodul mit U 6516 D

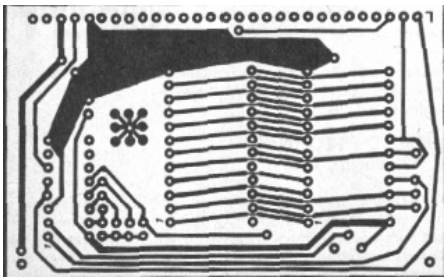
Das modulare Konzept des JU+TE-Computers gestattet des Ausrüsten mit den verschiedensten Speicherbauelementen. Nur der Einsatz dynamischer RAM ist nicht möglich, da das Auffrischen nicht gewährleistet wird. Während der Bilderzeugung variiert der Einchip-Mikrorechner zwar die unteren neun Adreßbits. Das läßt sich aber nicht für eine Refresh-Steuerung ausnutzen, da zwischen zwei Bildern etwa acht Millisekunden ohne dieses Variieren vergehen. Dynamische RAM gewährleisten gewöhnlich den Datenerhalt bis zu zwei Millisekunden Refresh-Periodendauer. Für den JU+TE-Computer müßte diese Zeit mehr als 10 ms betragen. In der Folge 7 der JU+TE-Computer-Selbstbauanleitung (Heft 1/88) wurden Hinweise für das Ausrüsten mit zusätzlicher Speicherkapazität veröffentlicht. Eine Leiterplatte für den 2K x 8 Bit-CMOS-RAM-Scheltkreis U 6516 D sind wir dabei schuldig geblieben. Da dieser Typ bereits seit geraumer Zeit produziert wird und daher auch etlichen JU+TE-Computer-Freunden zur Verfügung steht, holen wir das hiermit nach. Unser Schaltungsvorschlag gestattet das Ausnutzen aller Adreßbits zur Realisierung von 8 KByte RAM je Modul und die Batteriestützung (vgl. JU+TE 4/88).
Die minimale Bestückung erfordert einen U 6516 D. Der Dekoder DS 8205 D darf nicht



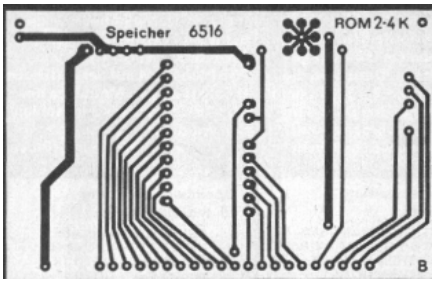
Speichermodul mit U 6516 D Foto: Domheim



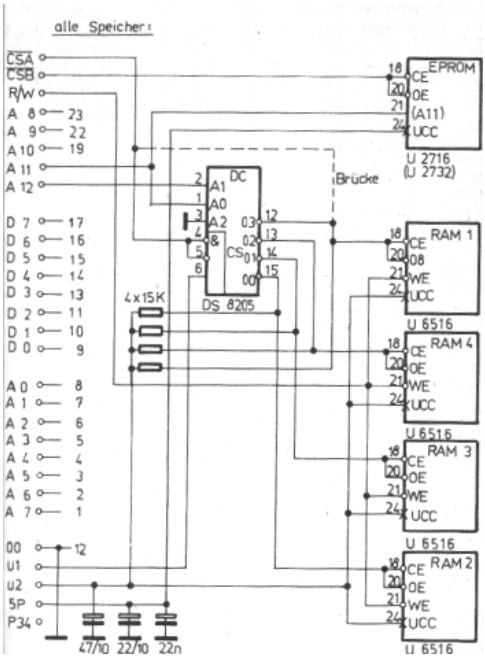
Bestückungsplan des Speichermoduls


Speichermodul Lötseite

eingesetzt werden. wenn der minimal bestückte Modul auf Steckplatz 1 zum Einsatz kommt. Hier muß der RAM-Schaltkreis nämlich gleichzeitig den BASIC-RAM (%E000 bis %E4FF) und den Operativspeicher des Betriebssysteme (%FD00 bis %FFFF) realisieren. Das gelingt durch Ignorieren der Adreßbits A11 und A12. Statt des Dekoders DS 8205 D sind ein Durchkontakt anstelle des Pins 16 und eine Drahtbrücke zwischen den Anschlußbohrungen für die Pins 5 und 12 einzusetzen. Unser Foto zeigt diese Bestükkungsvariante.



Speichermodul Bestückungsseite



Schaltbild des Speichermoduls mit U 6516 D

Zeichnungen: Hoyer, Grabowski

Das Aufstocken der RAM-Kapazität gelingt durch Aufsetzen weiterer U 6516 D. Dafür sind deren Pins 18 (chip enable /CE) und 20 (output enable /OE) vorsichtig abzuspreizen. Nachdem alle anderen Anschlüsse an die gleichen des bereits bestückten RAM angelötet sind, werden diese durch ein Stück Schaltdraht mit einem Ausgang des Dekoders (zuerst Pin 15, dann 14 und 13) verbunden. Pin 12 des Dekoders DS 8205 D, der anstelle der bei Minimalbestückung nötigen Drahtbrücke und des Durchkontakts einzusetzen ist, erzeugt das Auswahlsignal des untersten RAM-Schaltkreises.

Er belegt nun den Bereich von %E800 bis %FFFF, der den Operativspeicher des Betriebssystems einschließt. Die aufgesetzten maximal drei U 6516 D realisieren den BASIC-RAM (%E000 bis %E7FF, %E800 bis %EFFF und %F000 bis %F7FF). Für den EPROM (U 2716 C oder U 2732 C) wird eine 24polige Schaltkreisfassung bestückt. Eine Batteriestützung der RAM, mit der im Heft 4/88 veröffentlichten Ergänzung ist möglich, wenn die Auswahlsignale über Widerstände (10 bis 22 kΩ) mit der gestützten Versorgungsspannung U2 verbunden werden. Dafür vorgesehene Lötaugen befinden sich oben zwischen Dekoder und RAM-Schaltkreis. Solange keine Batteriestützung erfolgt, müssen die Sonderspannungen U1 und U2 des Modul-Bus mit 5P verbunden sein.

Auch wenn der Speichermodul auf Platz 1 mit vier U 6516 D und einem U 2716 C (U 2732 C) maxi mal bestückt ist, läßt die Belastbarkeit des Bussystems einen zweiten Modul (Platz 2) zu. Hier kann auch bei minimaler Bestückung bereits ein Dekoder DS 8205 D eingesetzt werden. Der RAM belegt den Bereich von %D800 bis %DFFF, der durch drei zusätzliche Schaltkreise auf %C000 bis %DFFF erweitert wird. Ein EPROM ist hier nicht erforderlich. Statt dessen kann ein U 6516 D gesteckt werden. Dazu muß die Verbindung von Pin 21 mit dem daneben liegenden Durchkontakt (A11) unterbrochen und statt dessen ein Schaltdraht zum Signal R/W (drittes Bussignal von links auf der Lötseite) eingesetzt werden. Dieser RAM belegt dann

die Adressen %2000 bis %27FF und erhält die nicht gestützte Betriebsspannung 5P. Damit können insgesamt ein EPROM und 18 KByte RAM oder zwei EPROM und 16 KByte RAM auf der Basis des U 6516 D an den Modulbus des JU+TE-Computers angeschlossen werden.

Dr. Helmut Hoyer

Software

Verwalten mehrerer BASIC-Programme

Der Editor des JU+TE-Computers setzt nur ein Programm im BASIC-RAM voraus. Er besitzt keine Komponenten, die das Unterscheiden mehrerer Programme ausführen. Bei entsprechend großer realisierter Speicherkapazität können sich aber auch mehrere Programme gleichzeitig im BASIC-RAM befinden. Insbesondere bei Batteriestützung der RAM-Schaltkreise bedeutet das, viel seltener den externen Massenspeicher (Magnetbandgerät) in Anspruch nehmen zu müssen. Wir stellen daher eine Möglichkeit vor, mit einem kleinen BASIC-Vorsatz mehrere Programme im RAM zu unterscheiden. Dabei wird ausgenutzt, daß der Editor die Anfangsadresse des BASIC-RAM im Doppelregister 6 (Register 6 und 7 des Einchip-Mikrorechners) ablegt. Nach jedem RESET steht dort die Adresse %E000. Hier beginnt unser Zusatzprogramm, das den Inhalt dieses Doppelregisters ggf. verändert. Im Interesse einer einfachen Lösung wird von zwei Voraussetzungen ausgegangen: Die BASIC-Programme seien nicht länger als 1 KByte (wie das Programmbeispiel Hase und Wolf in JU+TE 3/88) und beginnen mit der Zeilennummer 10.
Der verfügbare RAM wird in Scheiben zu je 1 KByte geteilt, in jeder kann sich ein Programm befinden. Die Scheiben werden von 1 beginnend durchnumeriert. Die Zuordnung zu Speicheradressen hängt von der realisierten RAM-Kapazität ab. Bei 8 KByte (erster Modul voll bestückt) beginnt der verfügbare Bereich bei %E000 und reicht bis %FCFF. In der ersten Scheibe befindet sich unser Verwaltungsprogramm (Manager):

Speicherverwaltung für 16 KByte:

```
1 PRINT "RAM-MANAGER"
2 INPUT "PROGRAMM-NR.:"A
3 IF A=9 THEN GOTO 10
4 IF A<1 THEN GOTO 1
5 IF A>16 THEN GOTO 1
6 PROC SETRR
  [6,A*%400+%BC00];END
```

Speicherverwaltung für 8 KByte:

```
1 PRINT "RAM-MANAGER"
2 INPUT "PROGRAMM-NR.:"A
3 IF A=1 THEN GOTO 10
4 IF A<1 THEN GOTO 1
5 IF A>8 THEN GOTO 1
6 PROC SETRR
  [6,A*%400+%DC00];END
```

1 PRINT "RAM-MANAGER"
2 INPUT "PROGRAMM-NR.:"A
Die Variable A erhält per Tasteneingabe die Nummer der ausgewählten RAM-Scheibe. Ist A gleich 1, bleibt die Anfangsadresse %E000. In diesem Fall wird mit der ersten Anweisung des eigentlichen Programms Nr. 1 fortgesetzt:
3 IFA=1 THEN GOTO 10
Andernfalls werden unzulässige Eingaben abgewiesen:
4 IF A < 1 THEN GOTO 1
5 IF A > 8 THEN GOTO 1
Einer gültigen Eingabe (beide Verzweigungsbedingungen nicht erfüllt) muß das Laden des Doppelregisters 6 mit der zugeordneten Adresse folgen. Für 2 ist das %E400, für 3 entsprechend %E800 usw.
6 PROC SETR [6,A*%400+%DC00]; END
Die END-Anweisung läßt nach der Ausschrift END 6 wieder den K-Kursor erscheinen. Alle Kommandos und Programm-Modifikationen beziehen sich nun auf die ausgewählte RAM-Scheibe. Nach Ausführung von SAVE und LOAD, nach einem nicht vereinbarten Kommando und dem Betätigen der RESET-Taste erfolgt jedoch ein Neuinitialisieren mit dem Laden des Doppelregisters 6 mit der Adresse %E000. Um wieder die gewünschte RAM-Scheibe zu erreichen, muß mit RUN erneut der RAM-Manager gestartet werden. Unsere Abbildung enthält das Speicherverwaltungsprogramm für 8 KByte und für 16 KByte RAM. Im zweiten Fall muß es dem Programm Nr.9 vorgesetzt werden. Die letzte Scheibe enthält jeweils nur 1/4 KByte, da der Rest vom Betriebssystem belegt ist. Übersteigt ein Programm die Länge von 1 KByte, kann die folgende RAM-Scheibe nicht genutzt werden.

Unterprogramm Quadratwurzel

Auf der Grundlage einer Einsendung von J. I. Kietzmann aus Neubrandenburg wollen wir

ein TINY-MP-BASIC-Programm zum Radizieren (Berechnen der Quadratwurzel) vorstellen. Es realisiert die im extended BASIC enthaltene Funktion SQR, genauer: die Anweisung LET Y= SQR(X). Der Aufruf aus dem Hauptprogramm, das der Variablen X zuvor das Argument zuweist, erfolgt mit GOSUB 1030.
Die Variable Y wird zunächst mit dem ersten Schätzwert

```
1030  LET Y=0
1040  LET Y=Y+1
1050  IF X/Y>Y THEN GOTO
        1040
1060  IF X/Y=Y THEN RETURN
1070  LET Y=Y-1;RETURN
```

geladen. Solange das Argument X größer als das Quadrat unseres Schätzwertes ist, wird dieser in der Zeile 1040 erhöht. Bei Gleichheit erfolgt der Rücksprung in der Zeile 1060 mit dem exakten Resultat. Ansonsten besteht das Ergebnis aus einer gebrochenen Zahl, deren ganzzahliger Anteil in der Zeile 1070 der Variablen Y zugewiesen wird. Der verbleibende Rest laßt sich mit dem Ausdruck X-(Y*Y) nachträglich berechnen. Bei negativen Argumenten ermittelt unser Programm das Ergebnis 0, erzeugt aber keine Fehlermeldung.

Zahlenratespiel

Einem Vorschlag von B. Piniek aus Berlin folgend stellen wir Euch ein Zahlenratespiel in TINY-MP-BASIC vor. Eine Besonderheit besteht hier im Erzeugen großer Zufallszahlen. Dazu wird ein Register benutzt, das das Betriebssystem zum Zählen der Bildschirmzeilen verwendet. Die Funktion GETR[%54] erzeugt eine Zahl zwischen 1 und 120 je nach Zeitpunkt im Verhältnis zur Bilderzeugung. Das läßt sich für eine 7-Bit-Zufallszahl nutzen. Gemeinsam mit den 8 Bit des freien Zählers T1 ergeben sich 15 Bit, die fast alle positiven Zahlen des im TINY MP BASIC verwendeten Formats darstellen können.
Das Betriebssystem des JU+TE-Computers verwaltet die Tastatur synchron zur Bilderzeugung. Daher erhält man direkt nach einer INPUT-Anweisung beim Lesen des Registers %54 keinen zufälligen Wert. In unserem Programmbeispiel wird deshalb in der Zeile 50 eingabeabhängig

```
10    CALL %8DD
20    PRINT "ZAHLENRATEN"
30    PROC SETR [%F3,255];
      PROC SETR [%F2,0];
      PROC SETR [%F1,10]
40    INPUT"↓GRENZE:"G
50    WAIT G¤M100+1; LET Z=1
60    LET A=GETR[%54]*256
      +GETR[%F2] ¤MG +1
70    PRINT "↓↓TIP:"Z
80    INPUT D
90    LET Z=Z+1
100   IF D>A THEN PRINT
      "ZU GROSS";GOTO 70
110   IF D<A THEN PRINT
      "ZU KLEIN";GOTO 70
120   PRINT"↓↓ * RICHTIG! *";
      LET A=GTC;GOTO 10
```

gewartet, wodurch ein quasi zufälliger Einfluß entsteht. Das Reduzieren mit ¤M100+1 begrenzt die Wartezeit. um das Spiel nicht allzusehr zu verzögern.
Die Zeile 60 stellt die zu ratende Zahl zusammen. Das Bilden des Divisionsrestes mit G begrenzt sie auf den Bereich bis zu dieser Grenze. Von Bedeutung ist, daß der Zähler T1 durch das Initialisieren mit SETR[%F2,0] über GETR[%F2] eine Zahl zwischen 0 und 255 erzeugt. Das dargestellte Programm paßt in den freien RAM der Minimalkonfiguration. Der Pfeil bezeichnet die Kursorsteuertaste, die in der TT-Tastatur links oben angeordnet ist. Bei Rechnern mit RAM-Erweiterung kann ab der Zeile 120 folgende Spielauswertung ergänzt werden:

```
120 PRINT "↓↓* RICHTIG! *"
130 LET B=0
140 LET B=B+1,G=G/2;
      IF G>0 THEN GOTO140
150 IF Z>B THEN PRINT
      "↓ ABER ES GEHT
      NOCH BESSER!"
160 ELSE ;PRINT "↓REIFE";
      PRINT "LEISTUNG!"
170 LET A=GTC;GOTO 10
```

Viel Spaß beim Ausprobieren!

Dr. Helmut Hoyer

Autocross

Anhand eines Reaktionsspiels sollen weitere Möglichkeiten effektvoller Ausgaben gezeigt werden. Bei der Bildschirmanzeige kommt das vom Betriebssystem ausgeführte automatische Bildrollen zur Anwendung, um die Bewegung einer Fahrbahn zu simulieren.

Die erste Anweisung löscht die Variable F, die den jeweils aktuellen Rekord speichert. Beim Erzeugen der Überschrift und der Startlinie wird die Kursorsteuerfunktion ↓ verwendet, die sich gut zum Auflockern von Zeichenkettenausgaben eignet. Die Anweisung 30 initialisiert auch die Bahnposition A, die Autoposition C und den Kilometerzähler E. Mit der Anweisung 40 wird der Zähler T1 für einen Zählumfang von 3 eingestellt und gestartet. Er dient im Spiel als Zufallsgenerator für die Lage der Fahrbahn. Das Löschen des Tastencode-Registers %6D bereitet die spätere dynamische Abfrage vor.
Die Programmschleife, deren Ausführung einen Spielschritt bewirkt, beginnt mit der Anweisung 50. Nach Abschluß des zweiten Kilometers erzeugt sie eine lobende Anzeige, die Anweisung 60 dagegen eine schwindende Zeitbonifikation während des ersten Kilometers. In den Zeilen 70 und 80 wird abhängig von der in B gespeicherten Zufallszahl verzweigt. Die Anweisung 90 führt das Linksschwenken aus, wobei ein Verlassen des Darstellungsbereichs jedoch eine entgegengesetzte Bewegung auslöst. Entsprechendes realisiert die Zeile 110 für das Rechtsschwenken. In der Zeile 120 erhält das Bildschirm-Pointerregister %5B den Wert der Variablen A. Dieser liegt stets zwischen %70 und %77. Das entspricht einer Bildschirmposition in der untersten Zeile (vgl. JU+TE 3/88. S.234). Wird hierhin eine Ausgabe mit PRINT oder PTH ausgeführt, folgt automatisch als Zeilen-

```
10   LET F=0
20   CALL %8DD;
     PRINT "* AUTOCROSS *";
     PRINT "↓↓AN DEN START!";
     WAIT 150
30   PRINT "↓↓↓>>>>>>  <<<<<<";
     LET A=%74,C=%37,E=0
40   PROC SETR [%F3,255]
     PROC SETR [%F2,3]
     PROC SETR [%F1,10]
     PROC SETR [%6D,10]
50   IF E=2000 THEN PROC SETR[%5B,%20];
     PRINT "TOLLE FAHRT !";
     GOTO 120
60   IF E<990 THEN WAIT 990-E/10
70   LET B=GETR[%F2]
     IF B=1 THEN GOTO 120
80   IF B=2 THEN GOTO 110
90   LET A=A-1;
     IF A<%70 THEN LET A=%71
100  GOTO 120
110  LET A=A+1
     IF A>%77 THEN LET A=%76
120  PROC SETR[%5B,A];
     IF E<2000 THEN PRINT">>  <<"
130  ELSE ;
     PRINT">> <<"";
     IF E<3000 THEN WAIT 3000-E/10
140  CALL %C56;
     LET D=GETR[%6D] ¤A%7F
150  IF D=%59 THEN LET C=C-1
160  IF D=%58 THEN LET C=C+1
170  PROC SETR(%5B,C];
     PROC SETR[%6D,0];
     IP GETEB[%FD00+C]<>%20 THEN GOTO 190
l80  PROC PTC[%49];
     LET E=E+10;
     GOTO 50
190  PROC PTC[%58];
     PROC SETR[%5B,0]
200  PRINT " UNFALL !"
210  PRINT E," METER GEFAHREN !"
220  PROC SETR[%F1,%8A];
     LET A=0
230  WAIT 50;
     PROC SETR[%F2,6]
240  PROC PTC[8];
     PROC PTC[%20];
     PROC PTC[%40];
     WAIT 50
250  PROC SETR[%F2,12];
     LET A=A+1;
     IF A<13 THEN GOTO 230
260  IF E>F THEN LET F=E;
     PRINT "** REKORD **";
     PROC SETR[%F2,5];
     WAIT 300
270  PRINT "REKORDWEITE:"F," METER"
280  PROC SETR[%F1,10];
     WAIT 800;
     PROC SETR[%F1,%8A];GOTO 20
```

schaltung das Bildrollen. Wir nutzen dies zum Aufwärtsschieben aller vorheriger Ausgaben. Damit entsteht der Eindruck eines bewegten Bildes.
Während der ersten beiden Kilometer liegen zwei Leerzeichen zwischen den Begrenzungsmarken (Zeichenkette in Zeile 120). Danach folgt eine schmalere Bahn. Um die Eingewöhnung zu erleichtern, gibt die Anweisung 130 während des dritten Kilometers wieder eine immer kleiner werdende zusätzliche Reaktionszeit.
Wegen ELSE wird die Zeile 130 vor Erreichen des 3. Kilometers übergangen. Es folgt die dynamische Tasten-Abfrage, deren Ergebnis in D erscheint. Die Taste X (%58) bewirkt ein Fahren nach rechts und die Taste Y (%59) entgegengesetztes. Alle anderen Ergebnisse lassen unser Auto geradeaus fahren. In der Zeile 170 erhält der Bildschirm-Pointer die neue Position unseres Autos, die wegen der Initialisierung (Zeile 30) etwa in der Mitte des Bildes liegt. Es folgt das Ausschalten des Tasten-Beep mit dem Vorbereiten der nächsten dynamischen Abfrage. Für den Fall, daß die neue Autoposition nicht leer (%20) ist, wird mit der Zeile 190 fortgesetzt.
Sonst erzeugt die Anweisung 180 die Anzeige unseres Autos. Das verwendete I (%49) symbolisiert den typischen schlanken Rumpf mit abstehenden Rädern. Nach Erhöhen des Kilometerzählers E um zehn Meter erfolgt dann der Sprung zum nächsten Spielschritt.
Durch Verlassen der Fahrbahn erreicht der Interpreter die Zeile 190. Hier wird zunächst ein X als Symbol für das demolierte Auto angezeigt. Dann erhält der Bildschirm-Pointer den Wert 0, um die folgenden Ausgaben links oben beginnen zu lassen.
In der Zeile 220 werden der Zähler T1 als Quelle für die Akustische Ausgabe und die Variable A als Schleifenzähler initialisiert.
Ab der Zeile 230 tritt das Räumfahrzeug in Aktion. Zunächst wird einen Moment gewartet, dann der Piepton eine Oktave tiefer festgelegt. Die Anweisung 240 löscht das zuletzt ausgegebene Zeichen (8), erzeugt dort das Leerzeichen (%20) und gibt das ASCII %40 als Symbol des Räumfahrzeugs auf der nächsten Zeichenposition aus. Anschließend wird wieder einen Moment gewartet.
Die Zeile 250 verringert die Tonhöhe um eine weitere Oktave und organisiert den Durchlauf dieser Schleife 13mal. Ingesamt bewegt sich dabei das Räumfahrzeug über die Bildschirmzeile, in der das Auto die Fahrbahn verließ und nimmt alles dort dargestellte weg. Derweil simuliert der Zähler T1 ein Martinshorn. Die Tonhöhe wird durch den Zählumfang. die Dauer mit den WAIT-Anweisungen gesteuert.
Die Zeile 260 zeigt einen eventuellen neuen Rekord mit einer entsprechenden Ausschrift und einem anderen Ton an. In jedem Fall kommt durch die Zeile 270 der aktuelle Rekord zur Anzeige. In der Zeile 280 werden dann ein Ausschalten der akustischen Ausgabe für etwa sechs Sekunden und der Beginn eines neuen Spiels angewiesen. Das Programm benötigt fast 1 KByte BASIC-RAM. Viel Spaß bei der Rekordjagd!

Dr. Helmut Hoyer

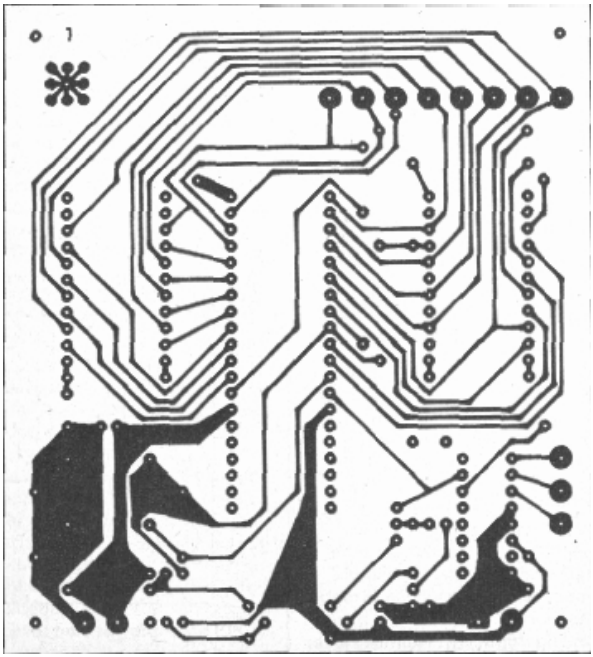
EPROM-Programmierzusatz

Wir beginnen heute eine wichtige Ergänzung für Mikrorechner vorzustellen, die sie als EPROM-Programmiergerät nutzbar macht. In diesem Heft beschreiben wir die Hardware, im Heft 11/1988 folgt dann ein Programm in TINY-MP-BASIC, das den Programmieralgorithmus deutlich macht. Wir hoffen, damit besonders den Freunden des JU+TE-Computers (Bauanleitung ab Heft 7/1987) eine interessante Selbstbauanleitung anzubieten, denn wer einmal einen JU+TEComputer arbeitsfähig aufgebaut hat und mit unserem Zusatz versieht, für den stellt das leidige Problem des EPROM-Programmierens dann keine Hürde mehr dar.

Elektrisch programmierbare und mit ultraviolettem Licht löschbare Festwertspeicher (EPROM) werden in jedem Mikrorechner gebraucht. Bei weitem nicht jeder Computer bietet aber die Möglichkeit, EPROM zu programmieren. Dazu reichen die für das Lesen verwendeten Signale und Betriebsspannungen nicht aus. EPROM speichern die Informationen in Form von kleinen Ladungen auf winzigen, völlig in Quarzglas (SiO_2) eingeschlossenen Siliziumstückchen. Die Ladungen können durch den normalen Betrieb im Rechner, bei dem nur Lese-Zugriffe möglich sind, nicht beeinflußt werden. Beim Löschen erhalten die Ladungsträger durch das UV-Licht genügend Energie, um das sonst ideal isolierende Quarzglas zu überwinden. Es lassen sich nur alle Speicherzellen gleichzeitig löschen, wobei alle Bits auf den Logikpegel 1 kommen.

Beim Programmieren werden gezielt mit Hilfe einer hohen Spannung (Vpp je nach Typ 12,5 V, 21 V oder 25 V) Ladungsträger auf diejenigen Siliziumstückchen „geschossen", die beim späteren Lesen eine 0 erzeugen sollen. Dazu muß die Programmierspannung etwa 50 ms lang auf die entsprechenden Speicherzellen einwirken, Da sich immer nur die acht Bit eines Bytes gleichzeitig behandeln lassen, benötigt diese Prozedur mindestens eine Minute je KByte.

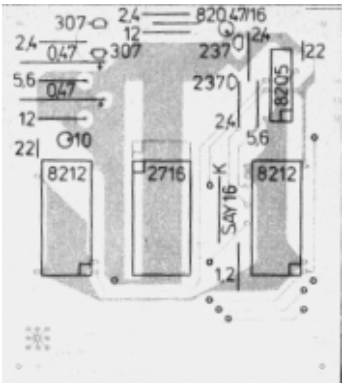
Leiterplatte des EPROM-Programmierzusatzes (Lötseite)



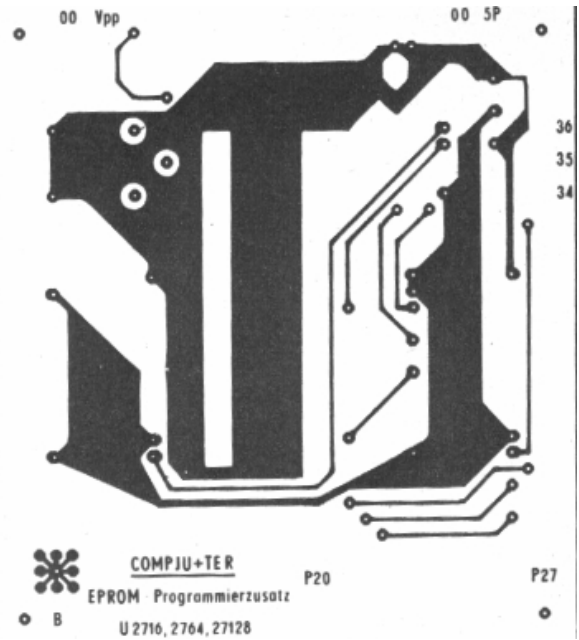
Schaltung

Beim Schaltungsentwurf wurde Wert gelegt auf ein Optimum zwischen Anwendungsbreite und gerätetechnischem Aufwand. Ziel war auch die Anpaßbarkeit an die verschiedensten Mikrorechner. Die Schaltung stellt die Erweiterung eines beliebigen Computers dar, der für den Anschluß nur ein bidirektionales 8-Bit Port und drei weitere TTL-Ausgänge besitzen muß. Es eignen sich daher neben dem JU+TE-Computer auch Kleincomputer zur Steuerung des Programmierzusatzes. Sogar die Lerncomputer POLY 880 und LC 80 besitzen geeignete Schnittstellen. Die 5 V-Betriebsspannung (5P), die der Programmierzusatz mit etwa 200 mA belastet, läßt sich in den meisten Fällen auch vom steuernden Rechner abgreifen. Die Programmierspannung Vpp muß dagegen mit einem zusätzlichen Stromversorgungsmodul bereitgestellt werden. Hier reichen bei den meisten EPROM-Typen 30 mA Belastbarkeit aus. Die Steuersignale P34, P35 und P36 bestimmen mittels des DS 8205 D den Zustand der Schaltung. Solange P36 1 Pegel besitzt, bleibt alles passiv. Sonst gibt es die folgende Zuordnung: Mit P34 = 1 und P35 = 1 werden die unteren acht Adreßbits vom bidirektionalen Port (Signale P20 bis P27) in den einen DS 8212 D geladen. Bei P34 = 1 und P35 = 0 speichert der andere DS 8212 D die höheren Adreßbits und zwei interne Steuersignale. Sie betreffen die Bildung der Schaltkreisauswahl /CE des EPROM und das Durchschalten der Programmierspannung auf Pin 23, was nur der Typ U 2716 erfordert. Mit P34 = 0 und P35 = 1 wird das Steuersignal /OE zum Lesen vom EPROM über das 8-Bit-Port aktiv. Die Kombination P34 = 0 und P35 = 0 ist dagegen während des Programmierimpulses erforderlich. Derweil stellt der steuernde Mikrorechner die einzuspeichernde Bitkombination über die Signale P20 und P27 bereit.

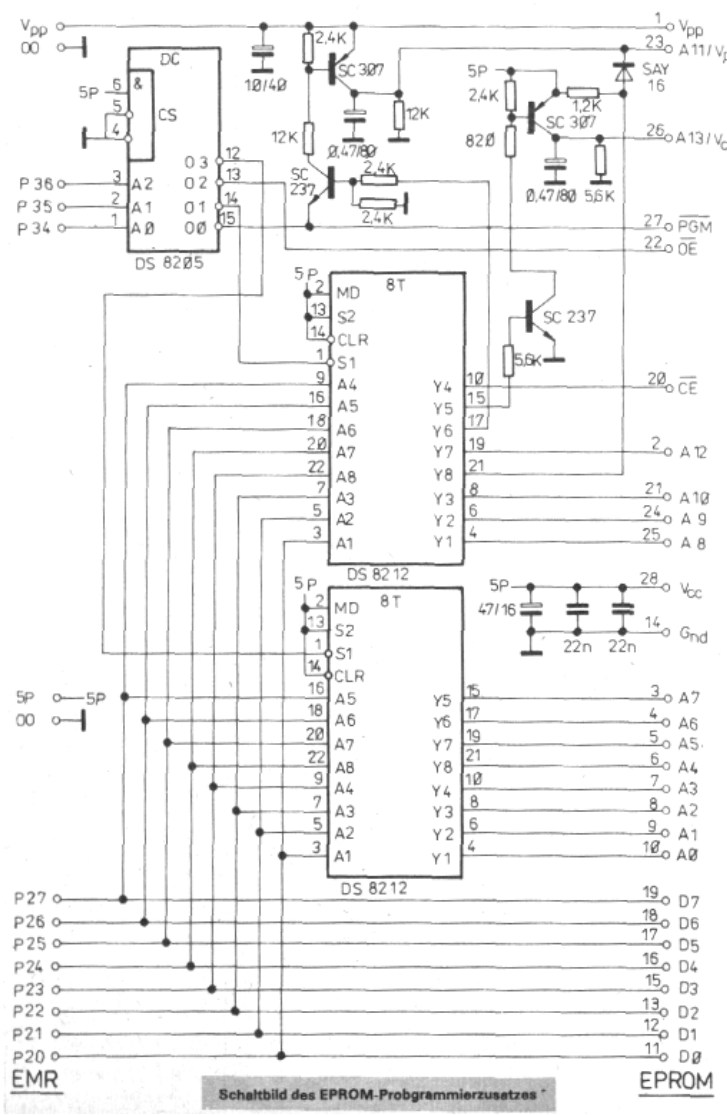
Bestückungsplan



Leiterplatte von der Bestückungsseite



Beim Übergang von einem zum anderen Zustand muß stets mit P36 = 1 alles passiv geschaltet werden, um das bidirektionale Port in die für die nächste Aktion erforderliche Übertragungsrichtung initialisieren und die ggf. erforderliche Bitkombination ausgeben zu können. Auf diese Weise lassen sich die EPROM-Typen U 2716, U 2764 und U 27128 handhaben. Die Anschlußbezeichnung im Schaltbild bezieht sich auf die beiden letztgenannten 28-poligen Schaltkreise.
Die Leiterplatte wurde so gestaltet, daß sie sich von Hand mit amateurmäßigen Mitteln fertigen läßt. Die zehn freien Durchkontakte sind im Bestückungsplan mit kleinen Kreisen markiert. Darüber hinaus dienen auch zahlreiche Bauelementeanschlüsse dem Verbinden von Löt- und Bestükkungsseite. Für den EPROM kann eine 28-oder 40-polige Schwenkhebelfassung verwendet werden. Ungenutzt bleiben stets nur über dem EPROM befindliche Anschlüsse.



Schaltbild des EPROM-Probgrammierzusatzes

(wird fortgesetzt in JU+TE 11/1988)

Zeichnungen Hoyer (3); Liebig

Dr. Helmut Hoyer

EPROM-Programmierzusatz

(Fortsetzung zu Teil 1/JU+TE 10/1988, S. 786 ff.)

Steuerung mit JU+TE-Computer

Am effektivsten ist das Steuern des Programmierzusatzes mit Maschinen bzw. Assembler-Programmen. Höhere Sprachen erfordern merklich mehr Rechenzeit. Als Beispiel eines geeigneten Steueralgorithmus stellen wir ein Programm in TINY-MP-BASIC vor, das die Handhabung des EPROM-Typs U 2716 C (K 573 РФ2) gestattet. Das Unterprogramm ab Zeile 500 lädt die EPROM-Adresse in die beiden Auffangregister (DS 8212 D) des Programmierzusatzes. Es setzt voraus, daß Port 2 (P20 bis P27) in Ausgaberichtung initialisiert ist und die Variable A die einzustellende Adresse enthält. Die Prozedur SETR beeinflußt einerseits die Steuersignale P34, P35 und P36, andererseits die Belegung von Port 2. Der Ausdruck A/256 erzeugt die höheren acht Bit der in A gespeicherten Adresse auf den bei der Ausgabe mit SETR benutzten unteren acht Bitpositionen. Die OR-Verknüpfung mit %48 bringt zusätzlich die Bits D6 und D3 auf 1. Dadurch werden die Spannungsversorgung dos EPROM (Vcc) und 1-Pegel am Programmierspannungsanschluß (A11/Vpp) gesichert. Die Schaltkreisauswahl /CE erhält den aktiven 0-Pegel, das Durchschalten der Programmierspannung zum EPROM wird gesperrt.
Die Anweisung 510 überträgt die so zusammengestellte Steuerinformation in den in der Schaltung oben dargestellten DS 8212 D. Die Zeile 520 lädt das andere Auffangregister mit den niederen acht Adreßbits, bevor die Anweisung 530 die Steuerung durch P36 = 1 passiviert und den Rücksprung (RETURN) in das Hauptprogramm auslöst.
Das Unterprogramm ab Zeile 550 dient auch dem Einstellen der Adresse aus der Variablen A, nur daß hier /CE den passiven 1-Pegel erhält und das Durchschalten der Programmierspannung ermöglicht wird.
Das dritte Unterprogramm ab Zeile 600 realisiert das Lesen der EPROM-Zelle, deren Adresse die Variable A bereitstellt, in die Variable B. Dazu wird nach dem Laden der Adresse Port 2 mit SETR[%F6,%FF] in Eingaberichtung geschaltet. Nach Aktivieren des Signals /OE erhält B über P20 bis P27 die Information vom EPROM. Es folgen das Passivieren von /OE, das Initialisieren von Port 2 auf Ausgabe und der Rücksprung.
Das Hauptprogramm blockiert während jedes Datenaustauschs mit dem EPROM-Programmierzusatz durch PROC SETR[%FB,%10] die Interruptannahme. Das ist nötig, damit die Bilderzeugung, die P36 beeinflußt, nicht die Kommunikation stört. Außerdem beschleunigt das die Arbeit des BASIC-Interpreters.
Die Zeile 10 initialisiert Port 2 und Port 3 und sorgt für ungefährliche Signale an der EPROM-Fassung. Nach Anzeige des Auswahl-Menüs und Freigabe der Bilderzeugung mit PROC SETR[%FB,%90] erfolgt das Eingeben von RAM-Anfangs- und -Endadresse in die Variablen C und D. Damit wird der Arbeitsspeicher im JU+TE-Computer festgelegt. Die Zeile 40 erwartet dann die Auswahl zwischen den drei angebotenen Diensten. Um z. B. einen Fehler bei der RAM-Adreßeingabe zu korrigieren, kann mit der Eingabe einer Zahl größer als drei neu begonnen werden.
Die Auswahl 1 führt zur RAM-Eingabe, die am besten hexadezimal erfolgen sollte. Sie behandelt immer gleich zwei Bytes auf einmal. Das erfordert eine geradzahlige RAM-Anfangsadresse. Nach Anzeige des alten RAM-Inhalts führt die Zeile 60 die Eingabe des neuen Inhalts aus. In Zeile 70 erfolgen das Weiterstellen der Adresse A und bei Erreichen des Endes des RAM-Bereichs das Anfordern einer neuen Auswahl.
Bei Auswahl 2 oder 3 wird durch die Anweisung 90 die EPROM-Anfangsadresse in die Variable F eingetragen. A erhält diese Zahl als Startwert, 1 in gleicher Weise die RAM-Anfangsadresse. In G wird die Grenze errechnet, bis zu der (ausschließlich) der EPROM zu behandeln ist. Übersteigt G die Zahl %800, reicht die Kapazität des U 2716 nicht aus.

Bei Auswahl 2 fordert unser Programm mit der Anweisung 110 das Überprüfen der Programmierspannung. Sie muß beim U 2716 zwischen 24 V und 25,5 V liegen. Die Bestätigung erfolgt mit dem Betätigen irgendeiner Taste. Nach Abschalten der Bilderzeugung beginnt dann das Programmieren. Den Zyklus eröffnen in der Zeile 120 das Einstellen der Adresse und die Ausgabe der RAM-Zelle (Adresse I) über Port 2. Die Zeile 130 realisiert den Programmierimpuls, während dessen die Programmierspannung Vpp zum Anschluß A11/Vpp durchgeschaltet wird. Wegen des abgeschalteten Bildinterrupts dauert ein WAIT-Durchlauf nur 1 ms. Nach Weiterstellen der Adressen A und I wird der Zyklus bis zum Erreichen des Grenzwertes G fortgesetzt. Vor Benutzung des Programms sollte die Zeile 130 genau überprüft werden, da eine Fehlfunktion an dieser Stelle zur Zerstörung des EPROM führen kann.
Nach Abschluß des Programmierens folgt ab Zeile 150 das Prüfen. G zählt die Fehler, H die von %FF verschiedenen EPROM-Inhalte. In J wird die zyklische redundante Kontrollsumme CRC gebildet. Es ist üblich, diese Zahl auf programmierten EPROM zu vermerken, um sich später durch wiederholte CRC-Berechnung vom fehlerfreien Speicherinhalt überzeugen zu können. Dieser Algorithmus läßt sich in sich in BASIC sehr schlecht realisieren, daher ruft die Anweisung 190 ein entsprechendes Maschinenprogramm.
Die Zeile 210 erzeugt eine ungefährliche Belegung von P34, P35 und P36 und ermittelt ggf. einen leeren EPROM dadurch, daß alle untersuchten Bytes im gelöschten Zustand vorgefunden wurden. Andernfalls bringt die Zeile die Anzahl der Fehler, die im Normalfall gleich null ist, zur Anzeige. Danach folgen noch die Ausgabe der Kontrollsumme, das Einschalten der Bilderzeugung und das Abwarten einer beliebigen Tastenbetätigung.

```
10   PROC SETR[%FB,%10];
     PROC SETR[%F7,1];
     PROC SETR[3,%70];
     PROC SETR[%F6,0];
     LET A=0;
     GOSUB 500
20   CALL %8DD;
     PRINT "U2716-HANDLER ";
     PRINT "1=HEX-EINGABE";
     PRINT "2=BRENNEN";
     PRINT "3=LESEN"
30   PROC SETR[%FB,%90];
     INPUT "RAM-ANF:"C;
     INPUT "RAM-END:"D;
     LET A=C
40   INPUT "AUSWAHL:"E;
     IF E>3
     THEN GOTO 10
50   IF E>1
     THEN GOTO 90
60   PTH A,":"GETEW[A];
     INPUT " NEU :"B;
     PROC SETEW[A,B]
70   LET A=A+2;
     IF A>D
     THEN GOTO 40
80   GOTO 60
90   INPUT "ROM-ANF:"F;
     LET A=F,G=D-C+F+1,I=C;
     IFG>%800
     THEN GOTO 90
100  IF E=3
     THEN GOTO 240
110  PRINT "VPP PRUEFEN !";
     LET B=GTC;
     PROC SETR[%FB,%10]
120  GOSUB 550;
     PROC SETR[3,%40];
     PROC SETR[2,GETEB[I]]
130  PROC SETR[3,0];
     WAIT 45;
     PROC SETR[3,%40]
140  LET A=A+1,I=I+1;
     IF A<G
     THEN GOTO 120
150  LET A=F,G=0,H=0,J=-1
160  GOSUB 600
170  IF B<>GETEB[C]
     THEN LET G=G+1
180  IF B<>%FF
     THEN LET H=H+1
190  CALL %E3C0
200  LET A=A+1,C=C+1;
     IF C<=D
     THEN GOTO 160
210  PROC SETR[3,%70];
     IF H=0
     THEN PRINT"ROM LEER"
220  ELSE ;
     PRINT G," FEHLER"
230  PTH "CRC:"J;
     PROC SETR[%FB,%90];
     LET B=GTC;
     GOTO 10
240  PROC SETR[%FB,%10]
250  GOSUB 600;
     PROC SETEB[I,B];
     LET A=A+1,I=I+1;
     IF A<G
     THEN GOTO 250
260  GOTO 150

500  PROC SETR[3,%50];
     PROC SETR[2,A/256¤O%48]
510  PROC SETR[3,%10];
     PROC SETR[3,%F0]
520  PROC SETR[2,A];
     PROC SETR[3,%30]
530  PROC SETR[3,%70];
     RETURN
550  PROC SETR[3,%50];
     PROC SETR[2,A/256¤O %E8];
     GOTO 510

600  GOSUB 500;
     PROC SETR[%F6,%FF];
     PROC SETR[3,%60];
     PROC SETR[3,%20]
610  LET B=GETR[2];
     PROC SETR[3,%60];
     PROC SETR[%F6,0];
     RETURN
```

Bei Auswahl 3 führt das Programm nach Ausschalten der Bilderzeugung ab Zeile 250 das Übertragen des EPROM-Inhalts in den RAM des JU+TE-Computers aus. Anschließend folgt auch hier der beschriebene Prüfalgorithmus.
Das BASIC-Programm belegt den Bereich von %E000 bis %E3BF. Die letzten beiden Bytes lauten %52 und %0D. Das Maschinenprogramm läßt sich mittels der Hex-Eingabe (Auswahl 1) dort anschließend eintragen. Dazu sollte der RAM-Anfang mit %E3BE gewählt werden, um sicher zu sein, daß nicht infolge eines Leerzeichens mehr oder weniger in irgend einer Zeichenkette ein abweichender Speicherbedarf des BASIC Programms vorliegt. Gegebenenfalls muß diese Adresse mit RESET und wiederholtem Aufruf des Hex-Eingabe Programms gesucht werden. Da 13 Byte Reserve bestehen, kann man sich hier den Gegebenheiten anpassen. Nach Anzeige von %520D muß auch %520D eingegeben und mit ENTER abgeschlossen wenden. Es folgt die Eingabe des Maschinenprogramms bis %0D und %00. Damit wird es für das Betriebssystem in das BASIC Programm integriert und bei SAVE und LOAD einbezogen. Nur LIST kommt mit dem Maschinencode nicht klar, was aber nicht schadet. Sofern das erste Byte (%31) nicht auf der Adresse %E3C0 steht, muß an schließend noch die Anweisung 190 korrigiert werden. Bei Verzicht auf die CRC-Berechnung kann man auch die Anweisung 190 und das Maschinenprogramm weglassen.

Bedienungsanleitung

Maschinen-Unterprogramm:

```
E3C0:     31 30 48 23 B2 42 28 E4
E3C8:     F0 E4 56 E4 0F B2 42 28
E3D0:     E4 F0 E4 90 E4 58 E4 56
E3DB:     E4 1F B2 43 38 E4 48 E5
E3E0:     E0 E4 56 E4 F0 B2 43 56
E3E8:     E5 E0 B2 52 28 E4 38 E5
E3F0:     AF 0D 00 00 00 00 00 00
```

Der JU+TE-Computer benötigt insgesamt mindestens 2 KByte RAM, um mit dem beschriebenen Programm EPROM handhaben zu können. Das erste KByte (%E000 bis %E3FF) speichert dieses Programm, im zweiten KByte bleibt ein Viertel als Arbeitsbereich (%FC00 bis %FCFF) frei. Hiermit kann Record für Record in acht Schritten ein U 2716 mit beliebigem Inhalt programmiert werden. Bei größerer verfügbarer RAM-Kapazität lassen sich natürlich auch größere Blöcke bilden. Die Wahl von RAM-Anfang und -Ende erfolgen im Programm ohne Einschränkungen.
Die Programmierspannung Vpp darf ständig am Programmierzusatz angeschlossen sein. Ein EPROM kann aber grundsätzlich erst dann gesteckt werden, wenn das steuernde Programm läuft und irgendeine Eingabe erwartet. Eine falsche Polung birgt die Gefahr der Zerstörung des EPROM! Bevor mit RESET ein Abbruch des Programmablaufs erfolgt, ist der EPROM aus der Fassung zu nehmen. Vor dem Programmieren muß der EPROM leer sein. Zum Löschen eignen sich Quecksilber-Hochdrucklampen (HQL oder besser HQV), die z. B. in Bestrahlungsgeräten (Höhensonne) eingesetzt werden. Sonnenlicht besitzt nicht genug Energie zum sicheren Löschen. Kontrollieren läßt sich ein EPROM am besten mit Auswahl 3, wobei der RAM Bereich von 0 bis %7FF gewählt werden kann. Dort befindet sich kein RAM, so daß sich am Speicherinhalt im JU+TE-Computer nichts ändert. Aber der anschließende Test ermittelt, ob der EPROM leer ist. Das dauert insgesamt drei Minuten und 20 Sekunden.
Nach der Hex-Eingabe des gewünschten EPROM-Inhalts in den Pufferbereich %FC00 bis %FCFF kann gleich mit Auswahl 2 programmiert werden. Das dauert für ein Record (256 Bytes) knapp 40 Sekunden.
Der Test einschließlich der CRC-Berechnung bezieht sich auf den jeweils behandelten Bereich. Zum Ermitteln der Kontrollsumme des gesamten EPROM eignet sich das Lesen auf den Bereich von 0 bis %7FF (Auswahl 3).
Im JU+TE-Computer enthaltene Software kann ohne Hex-Eingabe in einen U 2716 übertragen werden. Zum Programmieren eines Betriebssystems-EPROM eignet sich die Auswahl des RAM Bereichs von %800 bis %FFF. Daß hier in Wirklichkeit gar kein RAM im JU+TE-Computer existiert, stört nicht. Das Programmieren des gesamten U 2716 dauert in summa gut fünf Minuten.
Die beschriebene Programmiereinrichtung wurde mit Schaltkreisen verschiedener Hersteller getestet und funktionierte dabei fehlerfrei. Für die

anderen Typen (U 2764 und U 27128) eignet sich die beschriebene Software nicht. Hier muß ein anderes Signalspiel realisiert werden. Das Lesen gelingt wie beim U 2716 mit /CE = 0 und /OE = 0. Zum Programmieren muß ebenfalls /CE = 0 (statt /CE = 1 beim U 2716) realisiert werden, wobei das Durchschalten von Vpp auf Pin 23 unterbleibt. Dazu sind die Bits D5 und D7 im oben dargestellten DS 8212 D, der die höheren Adreßbits speichert, mit 0 zu belegen. Das Bit D6 unterscheidet beim U 27128 als A13 die beiden 8-KByte-Bereiche. Beim U 2764 ist es gleichgültig. Bis auf diese Unterschiede beim Einstellen der höheren Adreßhälfte zum Programmieren kann der beschriebene Algorithmus ansonsten übernommen werden. Besser eignet sich aber ein „intelligenter" Algorithmus, der Programmierimpulse von 1 ms Dauer erzeugt, bis das betreffende Byte eingespeichert ist, und dann noch einen Impuls mit der doppelten Dauer zur Sicherheit anhängt. Dabei werden entschieden weniger als 50 ms je Byte benötigt, so daß sich höhere Lebensdauern und kürzere Programmierzeiten ergeben. Dieser Algorithmus läßt sich nicht in BASIC realisieren.
Zu beachten ist, daß diese höher integrierten EPROM einiger Hersteller (z. B Mitsubishi) eine Betriebsspannung von Ucc = 6 V benötigen und die Programmierspannung Vpp mit bis zu 100 mA belasten. Vpp beträgt bei den älteren Typen 21 V, bei neueren (Variante A) nur 12,5 V. Für den U 2732 eignet sich der EPROM-Programmierzusatz nicht, da hier A11 benötigt wird und der Programmierimpuls mit dem Signal /OE zu multiplexen ist.

Dr. Helmut Hoyer

Software

Wochentag

Unser Leser B. Wendt aus Kühlungsborn übermittelte das folgende Programm. Es gestattet auszurechnen, auf welchen Wochentag ein beliebiges Datum nach dem Gregorianischen Kalender fällt, Bei der Datumseingabe ist die Jahreszahl vollständig einzugeben, also zum Beispiel 1988. Nur 88 als Jahr eingegeben, führt zu einem falschen Wochentag.

```
10 PROC PTC[12]
20 PRINT"WOCHENTAGS-"
30 PRINT "BERECHNUNG"
40 INPUT “TAG:"T
50 INPUT "MONAT:"M
60 INPUT "JAHR:"J
70 LET H=J/100, V=J/400
80 LET Z=J+(J/4)-H+V+1$M7
90 LET I=J$M4
100 LET H=J$M100,V=J$M400
110 IF H=0
    THEN LET I=I+1
120 IF V=0
    THEN LET I=I-1
130 LET D=-30,B=1,A=0
135 IF I>0
    THEN LET A=1
140 IF M>2
    THEN LET D=D-1-A
158 IF M>8
    THEN LET B=2
160 LET D=D+30+(M+B$M2)
170 LET M=M-1
180 IF M>0
    THEN GOTO 160
190 LET C=D+T+Z+A+4$M7
200 GOTO C+210
210 PRINT "-SONNTAG-“;
    GOTO 230
211 PRINT "-MONTAG-";
    GOTO 230
212 PRINT "-DIENSTAG":
    GOTO 230
213 PRINT "-MITTWOCH-";
    GOTO 230
214 PRINT "-DONNERSTAG“;
    GOTO 230
215 PRINT “-FREITAG-";
    GOTO 230
216 PRINT"-SAMSTAG-"
230 WAIT 800; GOTO 10
```

Monophon

Mit Monophon wird ein Musikinstrument bezeichnet, mit dem man, wie bei einer Blockflöte, zu einer Zeit nur einen Ton erzeugen kann. Unser Programmbeispiel realisiert solch ein Instrument mit dem freien Timer T1 des JU+TE-Computers. Die Tastatur dient wie bei einer Orgel als Manual. Voraussetzung ist die Ausstattung des Computers mit einer RAM-Erweiterung (1 KByte-RAM reicht bereits aus) und einem akustischen Geber am Ausgang P36. Hier genügt schon das Magnetband-Interface, an dem ein Verstärker (z. B. auf Aufnahme geschalteter Kassettenrekorder) angeschlossen werden kann. Nach dem Löschen des Bildschirms und einer erklärenden Anzeige fordert unser Programm zu einer Eingabe auf. Sie entscheidet, ob beim Magnetband-Interface die Tonausgabe über die Diodenbuchse auf ein externes Gerät oder über den internen Lautsprecher erfolgt. Zunächst initialisiert die Anweisung 15 den Vorteiler von Timer T1 für eine Periodendauer von 8 µs. Das ist die Basis für die Tonerzeugung durch weitere Frequenzteilung mit dem Zähler T1. Danach wird die Bilderzeugung abgeschaltet, damit die Ausgabe von Synchronimpulsen über P37 nicht stört und die Tastenbehandlung möglichst umgehend die Zählersteuerung beeinflußt. Der Ausgang P37 erhält je nach Auswahl (INPUT-Anweisung) 0-Pegel (intern) oder 1-Pegel (extern). Die Anweisung 25 bestimmt P36 als Ausgang von T1 und sperrt den Timer. Damit führt sie zum Ausschalten der Tonerzeugung.
In der Zeile 30 erfolgt die dynamische Tastenabfrage, bei der in der Variablen A der ASCII der betätigten Taste entsteht. Ein Ergebnis unter 42 (z. B. keine Betätigung) führt über die Anweisung 35 zum Sperren des Timers und zur erneuten Abfrage. Ansonsten realisiert die Zeile 40 die Programmfortsetzung mit der Zeilennummer, die gleich dem ermittelten ASCII ist. Hier wird

```
10  CALL %8DD;
    PRINT" MONOPHON";
    PRINT "1= INTERN";
    INPUT "2=EXTERN"A
15  PROC SETR[%F3,%23];
    PROC SETR[%FB,%10];
    IF A=1 THEN
    PROC SETR[3,0]
20  ELSE; PROC SETR[3,%80]
25  PROC SETR[%F1,%80]
30  CALL %C56;
    LET A=GETR[%6D]$A%7F;
    PROC SETR [%6D,0]
35  IF A<42 THEN GOTO 25
40  GOTO A
42  LET B=101; GOTO 130
44  LET B=120; GOTO 130
46  LET B=107; GOTO 130
47  LET B=95: GOTO 130
66  LET B=160; GOTO 130
67  LET B=190; GOTO 130
68  LET B=202; GOTO 130
71  LET B=170; GOTO 130
72  LET B=151; GOTO 130
74  LET B=135; GOTO 130
76  LET V=114; GOTO 130
77  LET B=127; GOTO 130
78  LET B=143; GOTO 130
83  LET B=227; GOTO 130
86  LET B=180; GOTO 130
88  LET B=214; GOTO 130
89  LET B=240
130 PROC SETR [%F2,B];
    PROC SETR[%F1,%88];
    GOTO 30
```

ausgenutzt, daß bei GOTO ein Ausdruck, also auch eine Variable als Argument stehen darf. Das gestattet viel effektivere Verzweigungen als die IF-Anweisung. Als Ziel-Zeilen von GOTO A folgt praktisch eine Tabelle, die der Variablen B eine der betreffenden Taste zugeordnete Zahl zuweist und mit der Anweisung 130 fortsetzen läßt. Da die Zeilennummern dezimal angegeben werden müssen, der ASCII jedoch hexadezimal gebräuchlich ist, läßt sich die Zuordnung der einzelnen Tasten nicht auf den ersten Blick erkennen. Y wird z. B. mit %59 kodiert, dezimal heißt das 5+16+9=89. Diese Taste erzeugt den Grundton von 260,4 Hz (T=3840 µs). Die genaue Frequenz des Tones c^1 beträgt zwar 262 Hz, aber diese Abweichung fällt nicht ins Gewicht.

Viel bedeutender ist, daß sich auf der Basis der Zeitkonstanten 240, die das gemeinsame Vielfache einer großen Menge ganzer Zahlen darstellt, die Tonabstände mit Frequenzfehlern unterhalb von 0,5% realisieren lassen. Die reinen C-Dur-Intervalle erzeugt das Monophon sogar völlig exakt. Die jeweiligen Periodendauern sind über die untere Tastenreihe (Y bis /) den ganzen Tönen (o^1 bis e^2) zugeordnet. Die Halbtöne (cis, dis, fis, ...) realisieren die Tasten S, D, G, H, J, L und *. Beim Betätigen nicht vereinbarter Tasten findet der Interpreter keine Zeilennummer, die gleich A ist. In diesem Fall kommt die nächsthöhere Anweisung zur Ausführung. Damit sich zumindest diese stets auffinden läßt, hat die letzte Zeile die Nummer 130. Diese Zahl ist größer als der höchste vorkommende ASCII (OFF: %7F = 127). Hier erhält der Zähler T7 den Wert der Variablen B als Zeitkonstante und wird anschließend freigegeben. Das Programm setzt dann mit erneuter dynamischer Tastenabfrage fort.
Übrigens haben die ersten Zeilennummern einen Abstand von nur 5, damit der erste Teil des Programms vor der ersten Tastenkodezahl Platz findet. Mit GOTO A können allgemein alle ASCII ausgewertet werden. Nur die ENTER-Taste (%0D) bringt den Editor des JU+TE-Computers aus dem Rhythmus (Zeilenende-Byte), so daß die Zeilennummer 13 nicht verwendbar ist.
Viel Spaß beim Musizieren!

Dr. Helmut Hoyer

Malfix

Auf der Grundlage der Veröffentlichung in JU+TE 3/88 zum Thema „Graphik" entstand das folgende Malprogramm, mit dem man in einem Bildformat von 59*64 Bildpunkten Bilder erzeugen kann.
Durch Betätigen entsprechender Tasten kann ein Graphik-Kursor (blinkender Punkt) über den Bildschirm geführt werden. Entsprechende Kommandos legen fest, ob gemalt oder Gemaltes wieder gelöscht wird. Als Bewegungsrichtungen sind horizontale, vertikale und diagonale Richtung möglich.
Das erzeugte Bild kann auf Wunsch abgespeichert und ein abgespeichertes Bild wieder aufgerufen werden.
Das BASIC-Programm und der Speicherbereich zum Abspeichern eines Bildes passen in 1 KByte BASIC-RAM.
Soll ein gemaltes oder abgespeichertes Bild auch nach dem Ausschalten des Computers erhalten bleiben, so ist die bereits vorgestellte RAM-Stütze erforderlich.
Voraussetzung des Malprogramms ist das im Speicherbereich von %FCA0 bis %FCFF eingetragene Maschinenprogramm für Punktgraphik. Ist der Computer mit einer RAM-Stütze ausgerüstet, so braucht das Punktgraphikprogramm nur einmalig eingetragen zu werden. Ansonsten muß es nach jedem Einschalten vor dem eigentlichen Malprogramm eingetastet werden. Das geschieht mit dem BASIC-Hilfsprogramm "HEX-EINGABE" (Abb. 1). Nach dem Starten dieses Hilfsprogramms mit RUN erfolgt die Eingabe der in Abb. 2 dargestellten Angaben, jeweils 2 Bytes in hexadezimaler Darstellung, d. h. %FCA0 ENTER %70FD ENTER %3170 ENTER ... %AF00 ENTER. Mit RESET erreicht man wieder den Grundzustand und löscht mit NEW das Hilfsprogramm "HEX-EINGABE". Nun erfolgt die Eingabe des MALFIX-Programms.

Abb.1

```
10 PRINT "HEX-EINGABE"
20 INPUT "AB ADR."A
30 PTH A,; "INPUT":"B
40 PROC SETEW[A,B]
50 LET A=A+2: GOTO 30
```

Abb. 2

```
FCA0     70 FD 31 70 D6 FC CE 60 E2 52 32 92 30 50 FD AF
FCB0     70 FD 31 70 D6 FC CE 42 32 8B F0 70 FD 31 70 D6
FCC0     FC CE B0 52 B0 53 72 23 EB E3 20 53 8B DF 0C FF
FCD0     1C F8 B0 E2 38 51 56 E3 3F 90 E3 90 E3 10 E3 10
FCE0     E2 22 13 32 02 2C 01 38 4F 56 E3 3F 3E 00 E3 E0
FCF0     E2 76 E3 07 EB F7 90 E3 F0 E3 02 13 82 30 AF 00
```

In Anweisung 10 wird nach dem Schreiben der Überschrift mit der Wahl W ausgewählt, ob gemalt oder ein gespeichertes Bild auf den Bildschirm zurückgerufen werden soll. In der 1. Zeile von Anweisung 12 werden die Anfangsadressen vom Bildspeicher und vom Bildwiederholspeicher geladen.

Bild malen

```
10 CALL %8DD;
   PRINT "MAL-FIX";
   PRINT "1=MALEN";
   PRINT "2=HOLEN";
   INPUT "WAHL:"W;
   CALL %8DD
12 LET S=%E22B,T=%FE20;
   IF W=2
   THEN LET E=T,F=S;
   GOSUB 30
14 LET X=32,Y=30
16 CALL %C56;
   LET A=GETR[%6D]$A%7F;
   PROC SETR[%6D,0]
18 CALL %FCBB;
   IF Z=0
   THEN CALL %FCA0;
   WAIT 10; CALL %FCB0
20 ELSE ;
   CALL %FCB0;
   WAIT 10; CALL %FCA0
22 IF A<48
   THEN GOTO 16
24 IF C=1
   THEN CALL %FCA0
26 IF C=2
   THEN CALL %FCB0
28 GOTO A
30 LET G=236
32 PROC SETEW[E,
   GETEW[F]]; LET
   E=E+2,F=F+2,G=G-1;
   IF G>0
   THEN GOTO 32
34 RETURN
```

Für W=1 werden in Anweisung 14 die Anfangskoordinaten für den Graphik-Kursor auf Bildmitte festgelegt.
In Anweisung 16 erfolgt die dynamische Tastenabfrage, bei der in der Variablen A der ASCII der betätigten Taste entsteht. Ein Ergebnis unter 48 (z. B. keine Tastenbetätigung) führt über Anweisung 22 zur erneuten Tastenabfrage (GOTO 16). Vorher realisieren aber noch die Zeilen 18 und 20 das Blinken des Graphik-Kursors. Zunächst wird der Bildpunkt gelesen (CALL %FCBB). War er gelöscht, so wird er für eine bestimmte Zeit gesetzt (CALL %FCA0: WAIT 10) und anschließend wieder gelöscht (CALL %FCB0). Anderenfalls wird er für eine bestimmte Zeit gelöscht (Zeile 20) und dann wieder gesetzt.
Wurde aber eine Taste betätigt, so erfolgt die Abarbeitung der Anweisungen 24 und 26, die im Zusammenhang mit der Moduseinstellung beschrieben werden.
Nach dem Betätigen einer Taste steht also in der Variablen A (Zeile 16) der entsprechende ASCII-Code. Durch die Möglichkeit des Variableneinsatzes bei GOTO-Anweisungen (GOTO A) kann man direkt in die Zeilennummer springen, die gleich dem ermittelten ASCII ist. Da die Zeilennummern dezimal angegeben werden müssen, der ASCII aber hexadezimal gebräuchlich ist, empfiehlt sich das Anlegen einer Tabelle für die benötigten Tasten, z. B. Taste M (Malen): %4D, dezimal 4*16+13=77 Taste W (hoch): %57. dezimal 5*16+7=87.
In den entsprechenden Zeilen wird die erforderliche Richtungsänderung (Zeilen 65-69, 81-89) bzw. die Modusfestlegung (Zeilen 75-77) vorgenommen. Nach jeder Richtungsänderung des Graphik-Kursors wird die Einhaltung des Bildformates geprüft und ggf. korrigiert (Zeilen 36-44).

```
36 IF X>63
   THEN LET X=63
38 IF X<0
   THEN LET X=0
40 IF Y>59
   THEN LET Y=59
42 IF Y<1
   THEN LET Y=1
44 GOTO 16
48 LET E=S,F=T;
   GOSUB 30;
   CALL %8DD;
   END
65 LET X=X-1;
   GOTO 36
67 LET X=X+1,Y=Y-1;
   GOTO 36
68 LET X=X+1;
   GOTO 36
69 LET X=X+1,Y=Y+1;
   GOTO 36
75 LET C=0;
   GOTO 16
76 LET C=2;
   GOTO 16
77 LET C=1;
   GOTO 16
81 LET X=X-1,Y=Y+1;
   GOTO 36
87 LET Y=Y+1;
   GOTO 36
88 LET Y=Y-1;
   GOTO 36
89 LET X=X-1,Y=Y-1
127 GOTO 36
```

Bedienungsanleitung

1. Bild malen

Starten des Programms mit RUN, Eingebe W=1 und ENTER: Graphik-Kursor erscheint in der Bildmitte
Modusauswahl:
Taste M = Malen
Taste L = Löschen
Taste K = Kursorbewegen
Diese Festlegung gilt immer bis auf Widerruf.
Bewegungsrichtungen:

Der Dauerdruck einer Bewegungstaste erzeugt auch eine Dauerbewegung.

2. Bild abspeichern

Ein auf dem Bildschirm erzeugtes Bild kann jederzeit abgespeichert werden. Dazu wird die Taste 0 betätigt. Der blinkende Punkt „erstarrt" und das Bild wird ohne sichtbare Reaktion abgespeichert. Nach ca. 12 Sekunden wird als Quittung des Abspeicherns der Bildschirm gelöscht; es erscheint END 48. Danach kann das Programm erneut gestartet werden.

3. Aufrufen des abgespeicherten Bildes

Starten des Programms mit RUN, Eingabe W=2 und ENTER: zeilenweises Aufbauen des Bildes.
In der Bildmitte erscheint der Graphik-Kursor zum Zwecke des Weitermalens.
Nicht vergessen: Erst Modusauswahl, dann Kursor bewegen!

Die Programmschleife wird in jedem Fall durch den Sprung in Zeile 16 geschlossen.
Durch die Tastenbetätigung M, L oder K wird der Modus Malen, Löschen oder Kursorbewegen eingestellt, in Abhängigkeit davon C gesetzt und an entsprechender Stelle eine Reaktion ausgeführt. Ein Beispiel soll das verdeutlichen:
Es wird Taste M (Malen) betätigt. Somit wird beim weiteren Programmdurchlauf durch Zeile 28 (GOTO A) Zeile 77 angesprungen und C=1 festgelegt. Bei jedem folgenden Programmdurchlauf wird durch Zeile 24 ein Punkt gesetzt und bis auf Widerruf gemalt.
Beim Betätigen nichtvereinbarter Tasten findet der Interpreter keine Zeilennummer, die gleich A ist. In diesem Fall kommt die nachsthöhere Anweisung zur Ausführung. Damit auch in diesem Fall die Programmschleife geschlossen wird, erfolgt bei Zeile 127 ein entsprechender Sprung (127: höchster vorkommender ASCII).

Bild abspeichern

Wie in JU+TE 3/88 beschrieben, ist der Bildwiederholspeicher von %FE00 bis %FFFF untergebracht. Das entspricht einem Platzbedarf von %0200 Byte. Bedingt durch die Länge des BASIC-Programms (%E000 - %E227) stehen für die Bildspeicherung nur %01D8 Byte zur Verfügung, d. h. das Bildformat mußte um 5 Zeilen eingeengt werden. Eine Zeile wurde wegen des ohnehin störenden SHIFT-Striches vom unteren Bildrand, die vier weiteren Zeilen vom oberen genommen. So mit wird der Bildwiederholspeicher auf die Adressen %FE20 bis %FFF7 festgelegt.
Besteht der Wunsch zum Abspeichern eines Bildes, so wird die Taste 0 betätigt und über GOTO A die Zeilennummer 48 angesprungen. Dort werden die Startadressen des Bildwiederholspeichers der Variablen F (F=T=%FE20). die Startadresse des reservierten Bildspeicherbereiches der Variablen E (E =S=%E228) zugewiesen und das Unterprogramm 30 angesprungen.
32 PROC SETEW [E,GETEW[F]] liest das unter der in F stehenden Adresse des Bildwiederholspeichers befindliche Wort und speichert deren Inhalt an der in E stehenden Speicheradresse ab. Anschließend werden beide Adreßzahler um 2 erhöht, da wortweise umgeladen wird. Der Schrittzähler G verringert sich um 1. Zum Abspeichern des gewählten Bildformates von 59 Bildzeilen (1 Zeile entspricht 4 Worten) benötigt man G =59*4 =236 Schritte (30 LET G=236). Nach erfolgter Bildspeicherung und Rucksprung ins Hauptprogramm wird nach dem Löschen des Bildschirmes END 48 angezeigt.

Aufrufen des abgespeicherten Bildes

Für die Eingabe W=2 werden in Zeile 12 die Startadressen für Bildwiederholspeicher und Bildspeicher den Variablen E bzw. F (diesmal jedoch vertauscht) zugewiesen. Somit wird nach dem Sprung ins Unterprogramm 30 wortweise die Bildinformation aus dem Bildspeicherbereich in den

Bildwiederholspeicher umgeladen. Nach dem Rucksprung ins Hauptprogramm erfolgt ab Zeile 14 der schon beschriebene Programm durchlauf, d. h. es entsteht der blinkende Graphik-Kursor, der ein Weiterarbeiten an dem erzeugten Bild gestattet.

Bernhard Piniek

Vorankündigung

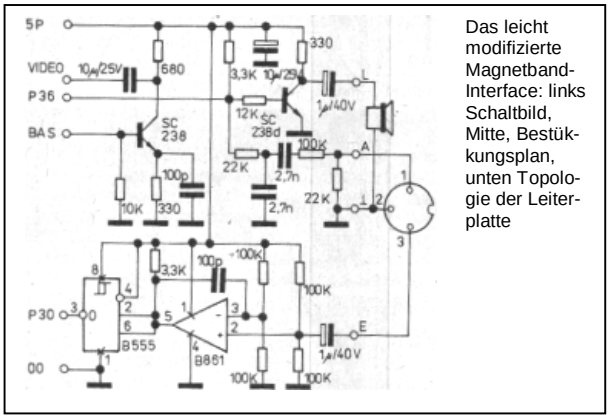
Seit eineinhalb Jahren veröffentlicht JU+TE Beitrage zum Selbstbau Computer. Er ist in TINY-MP-BASIC programmierbar und besitzt als Peripherie eine Schreibmaschinenähnliche Tastatur, einen auch graphisch nutzbaren Bildschirm und einen Anschluß für Magnetbandspeicher. Der RAM laßt sich Batterie-gestützt auslegen und kann mit Schaltkreisen verschiedener Integrationsgrade realisiert werden. Er erlaubt bei minimalem Hardware-Aufwand den Anschluß weiterer Geräte und eignet sich sogar zum Programmieren von EPROM. Die Möglichkeiten, besonders der Einsatz als Steuer- oder Prozeßrechner, sind damit nicht erschöpft. Deshalb veröffentlicht JU +TE demnächst ein erweitertes Betriebssystem (4 KByte).

Attraktive Betriebssystem-Erweiterung

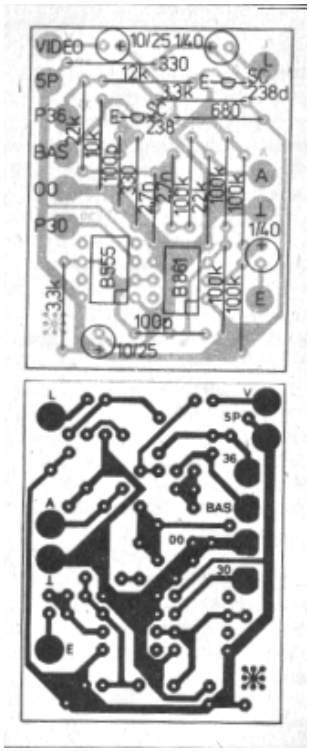
Der seit Juli 1987 von uns veröffentlichte Computer hat viel mehr Freunde gefunden. als zu erwarten war. Das Grundkonzept verwendet die Programmiersprache TINY-MP-BASIC, die sich gut für den Einstieg in die Mikrorechentechnik eignet. Sie läßt die Leistungsfähigkeit des Einchip-Mikrorechners UB 8830 D, der den Kern des JU+TE-Computers bildet, aber nur teilweise nutzen. Das erweiterte Betriebssystem erlaubt das Programmieren in Maschinensprache und damit wesentlich besseres Ausschöpfen der Rechenkapazität. Die dafür nötigen Grundlagen enthält unser „ABC Einchip-Mikrorechner" (vgl. JU+TE ab Heft 7/1988). Das neue Betriebssystem erlaubt außerdem, beliebige, sogenannte transidente Maschinenprogramme zu verwalten, so zum Beispiel den von uns bereits angekündigten FORTH-Compiler.
In Maschinensprache können weit aufwendigere Spiele als in BASIC realisiert werden. Auch der Einsatz als Prozeßrechner wird damit möglich. Die Bedingungen für den Anschluß peripherer Geräte verbessern sich. Demnächst möchten wir Euch z. B. mit Möglichkeiten zum Nutzen der Schreibmaschine „Erika 3004" als Drucker des JU+TE-Computers bekanntmachen. In Verbindung mit dem EPROM-Programmierzusatz (JU+TE 10/88 und 11/88) kann unser Selbstbau-Computer sogar zum Entwickeln von Programmen für andere Rechner, die einen Einchip-Mikrorechner-Schaltkreis enthalten, eingesetzt werden (EMR-ES; Einchip-Mikrorechner-Entwicklungssystem).
Gerätetechnisch erfordert das erweiterte Betriebssystem mindestens 2 KByte RAM und 4 KByte EPROM. Unsere Hex-Liste (vgl. S. 234f.) enthält dieses Programm für einen U 2716 C auf Modul 1 (statt des bisherigen) und einen zweiten auf Modul 2. Hierfür eignet sich die abgebildete einseitige Leiterplatte (S. 232), da Modul 2 keinen RAM enthalten muß.
Um auch Maschinenprogramme auf Kassette speichern zu können, mußte die Magnetbandsteuerung verändert werden. Das hat ein leicht modifiziertes Magnetband-Interface zur Folge. Die Abbildungen S.231 enthalten das Schaltbild, die Topologie und den Bestükkungsplan einer geeigneten Leiterplatte. Statt des überflüssig gewordenen DL 000 D wurde ein Transistor zum Erzeugen des Video-Signals darauf untergebracht. Dessen Ausgangselko sollte nur bei Bedarf bestückt, sonst durch eine Brücke ersetzt werden. Das in Heft 11/87 veröffentlichte Interface kann weiter genutzt werden, wenn der 22-nF-Kondensator entfernt, die Gegenkopplung (100 pF) zwischen den Pins 3 und 5 des Operationsverstärkers ergänzt und statt des 10-nF-Eingangskondensators ein Elko bestückt wird. Mit dem Einstellregler ist auf maximale Empfindlichkeit (Mitte) abzugleichen.
Bei vergrößerter RAM-Kapazität hätte auch eine feinere Bildauflösung realisiert werden können. Ein wirklicher Qualitätssprung erfordert aber unseres Erachtens einen integrierten Videocontroller (z. B. U 82720), den der Amateurhandel jedoch noch nicht bietet. Sein Erscheinen kann aber nicht ewig dauern. Bis zu seinem Einsatz im JU+TE-Computer wollten wir für einen recht kurzen Zeitraum keine Übergangsvariante einführen. So bleiben alle bislang veröffentlichten Programme auch unter dem erweiterten Betriebssystem unverändert lauffähig.

Installation

Man kann das erweiterte Betriebssystem mit dem JU+TE-Computer selbst installieren. Dazu bedarf es eines EPROM-Programmierzusatzes und zweier gelöschter U 2716 C. Den ersten EPROM kann man weitgehend aus dem alten Betriebssystem kopieren. Es reicht aus, die Bereiche von %0800 bis %0813 und von %0A38 bis %0A81 (EMR-Adressen) zu verändern. Das geschieht wie folgt:
1. Auswahl der RAM-Eingabe des BASIC-Programms „U2716-HANDLER". Von RAM-Adresse %FC00 bis %FC15 werden die ersten 22 Byte des neuen Systems (siehe Hex-Liste) geladen. Anschließend folgt das Programmieren (Brennen) ab ROM-Adresse 0 (Kontrollsumme CRC: %C408).
2. Kopieren des alten Systems ab RAM-Adresse %0816 bis %0A37 auf ROM-Anfang %0016 mit dem zweiten Programmiervorgang (CRC: %304A).



Das leicht modifizierte Magnetband-Interface: links Schaltbild, Mitte, Bestükkungsplan, unten Topologie der Leiterplatte



3. Eingabe des Programmteils von %0A38 bis %0A81 in den RAM-Bereich %FC00 bis %FC49 und Programmieren ab ROM-Adresse %0238 (CRC: %78CA).

4. Kopieren des alten Systems ab RAM-Adresse %0A82 bis %0FFF auf ROM-Anfang %0282 (CRC: %CE69).

Die Kontrollsummen (CRC) stimmen nur dann überein, wenn alles fehlerfrei geklappt hat. Das CRC des gesamten EPROM beträgt dann %B130, da an drei Stellen (Adressen %090C, %091A und %0F21) weitere Unterschiede zwischen neuem und altem System bestehen, die aber nur unbedeutende Schönheitskorrekturen betreffen.

Den Inhalt des zweiten EPROM muß man vollständig über die Tastatur eingeben. Das erfordert Ausdauer und Konzentration und sollte keinesfalls unter Zeitdruck begonnen werden. Man gibt immer einen Record (256 Bytes) in den RAM-Bereich %FC00 bis %FCFF ein und programmiert den EPROM in acht Stufen ab Adresse 0, %0100, %0200,..., %0700. Da das Programmierprogramm die jeweilige Kontrollsumme erst nach dem Brennen berechnet, darf bei der RAM-Eingabe kein Fehler unterlaufen. Das richtige CRC bestätigt die Fehlerfreiheit des jeweiligen Records.

Bedienungsanleitung

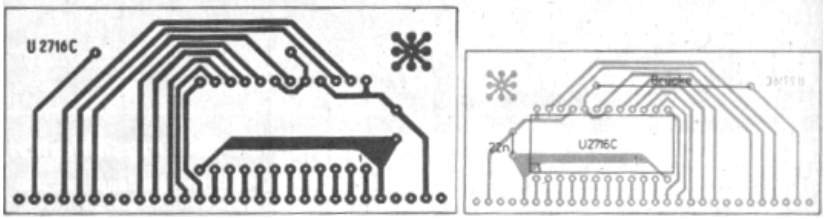
Wenn alles geklappt hat, und die neuen EPROM auf Modul 1 und 2 stecken, meldet sich das Betriebssystem mit einem Menü. Es bietet sechs Komponenten zur Auswahl:

• ***BASIC***

Nach Betätigen der Taste B erhält man das gewohnte Bild. Neu ist nur, daß nicht vereinbarte Kommandos ignoriert werden, statt, wie bisher eine Neuinitialisierung auszulösen. Der Rücksprung ins Menü gelingt nur mit der Reset-Taste. Beim Abspeichern des BASIC-Programms (nach SHIFT P) erwartet der Computer die Eingabe eines Namens. Er kann elf Zeichen enthalten, die weiteren werden weggelassen. Nach ENTER *beginnt bei abgeschaltetem Bildschirm (!)* das Ausgeben von Vorton, erstem Block mit Namen und Speicheradressen, die der Computer selbst berechnet hat, und weiteren Blöcken (je 128 Byte) mit dem eigentlichen Programm.

Beim Laden eines Programms (nach SHIFT 0) wird wie gewohnt bei Erkennen eines Fehlers abgebrochen. Mit Blocknummernfolge und Prüfsummen werden jetzt jedoch viel gründlichere Kontrollen ausgeführt. Wenn alles glatt geht, erscheinen „FF" (file found) und der Name des gelesenen Programms auf dem Bildschirm, sonst die Nummer des fehlerhaften Blocks.

• ***INIT***



Leiterplatte und Bestückungsplan für die EPROM-2-Platine auf Modul 2

Die Reset-Taste führt nur zum Neuinitialisieren des Prozessors und läßt den RAM-Inhalt unverändert. Die Eingabe eines Maschinenprogramms mit Marken setzt aber auch gelöschten Operativspeicher voraus. Nach Betätigen von /, SHIFT und 1 wird daher der gesamte RAM mit Nullbytes geladen. Um versehentliches Löschen zu erschweren, sind drei statt sonst einer Tastenbetätigung nötig.

• ***SAVE***

Die Taste *S* führt zum universellen KC-kompatiblen Kassetten Ausgabeprogramm. Der Einchip-Mikrorechner erwartet die Eingabe der Anfangs- und End-Adresse des auszulagernden Speicherbereichs vierstellig hexadezimal *ohne* vorangestelltes % und *ohne* ENTER. Als drittes muß der Name der Datei eingegeben und mit ENTER abgeschlossen werden. Der Einchip-Mikrorechner führt dann wie beschrieben die Ausgabe über das Magnetband-Interface aus. Es ist wichtig zu wissen, daß die neue Kassettensteuerung den Bereich von%FC00 bis %FC7F als Puffer benutzt. Im letzten KByte kann daher nur noch der RAM von %FC80 bis %FCFF z. B. für die Punktgraphik-Unterprogramme genutzt werden. Um sie auf Kassette zu speichern, ist FC80 FCFF PUNKTGRAFIK ENTER einzugeben. Der Systembereich ab %FD00 darf nicht mit ausgelagert werden.

• ***LOAD***

Vor dem Laden von Kassette (Taste L) bietet das erweiterte System die Möglichkeit, eine vierstellige Anfangsadresse einzugeben, ab der die gelesene Datei abgelegt wird. Mit ENTER verzichtet man auf dieses Angebot. Die Datei kommt dann auf den Bereich, von dem sie mit SAVE geholt wurde. Das Kassettengerät darf vor oder erst nach Aktivieren von LOAD gestartet werden. Das Speichern in den Systembereich ab %FD00 verändert den Stapelspeicher und ist daher verboten.

• ***DATA***

Mit *R* kann aus dem Menü oder dem Programm-Modus (PROG) die RAM- und Registeranzeige DATA gerufen werden. Sie zeigt ab der oben rechts stehenden Adresse 24 Bytes hexadezimal an. Adressen unter %0100 betreffen Register, ab %0100 Speicherzellen. In der untersten Zeile wird eine Eingabe auf die angezeigte Adresse angeboten, die mit den Tasten *0* bis *9* und *A* bis *F* hexadezimal erfolgen muß. Nach jeweils zwei Hexaziffern aktualisiert der Einchip-Mikrorechner die Anzeige und erhöht die Eingabeadresse. Sie kann zur Korrektur mit – zurückgestellt werden. + erhöht die Anzeigeadresse um 1, *ENTER* um 4. Nach *G* kann eine neue Anzeigeadresse vierstellig hexadezimal eingegeben werden. *OFF* (SHIFT ENTER) bewirkt die Rückkehr zum Menü oder Programm-Modus. Alle anderen Tasten aktualisieren die Anzeige und stellen die Eingabeadresse auf das erste angezeigte Byte.

• ***PROG***

Die Eingabe von Maschinenprogrammen erfolgt in einen gelöschten Speicherbereich. Besitzt der RAM keine Batteriestütze, muß deshalb nach dem Einschalten des JU+TE-Computers erst INIT ausgeführt werden. Da neben dem Maschinenprogramm automatisch eine Markentabelle (Verzeichnis der symbolischen Adressen) gespeichert wird, ist mindestens die doppelte RAM-Kapazität der reinen Programmlänge nötig. Die Markentabelle wächst in Richtung niedriger Adressen. Deshalb ist es sinnvoll, hiermit am Ende des zur Verfügung stehenden Bereichs zu beginnen. Nach Betätigen von *P* erwartet das Betriebssystem mit der Ausschrift *MTB* die hexadezimale Eingabe dieser Adresse. Soll der Bereich von %E000 bis %E3FF genutzt werden, erfolgt

entsprechend die Eingabe E3FF. Dadurch erscheint die Programmanzeige ab dieser Adresse. Mit G E000 muß sie auf den Anfang des Eingabebereichs eingestellt werden.
Neben PROG steht in der obersten Zeile der Inhalt des Flagregisters, der beim Programmtest eine große Rolle spielt. In den nächsten sechs Zeilen stehen die Adresse und der hexadezimale Code je eines Maschinenbefehls. In der untersten Zeile wird wieder die Eingabe angeboten. Sie erfolgt an die Stelle des zweiten Befehls. Das gestattet, sich stets an der vorherigen Eingabe, die darüber steht, zu orientieren. Zur Korrektur kann nach *Q* an die Stelle des ersten Befehls eingegeben werden.
Im Gegensatz zu DATA bewirkt *ENTER* hier die Anzeige ab nächstem Befehl. Mit einem Doppelpunkt (*:*) beginnt die Eingabe symbolischer Adressen. Eine Marke besteht aus drei beliebigen Zeichen, die dem Doppelpunkt folgen müssen. Vor Eingabe des Operationscodes werden links stehende Marken eingetastet. Sie erscheinen in der Befehlsanzeige anstelle der Adreßangabe. Nach dem Operationscode können rechts stehende Marken eingegeben werden. Das gelingt jedoch nur bei Sprungbefehlen. Sie erscheinen als Operand statt des zweiten bzw. zweiten und dritten Bytes in der Anzeige. Damit die Zuordnung der Marken zu den Befehlen nicht verlorengeht, dürfen Teile eines Programms nicht durch leeren Speicherbereich getrennt werden.
Zur Korrektur streicht *X* den aktuellen Befehl und läßt das folgende Programm aufrücken. Umgekehrt kann mit *I* ein Byte eingefügt werden (insert). Da bei ist zu beachten, daß das Einfügen eines Zweibytebefehls nach zweimal I, eines Dreibytebefehls nach dreimal I möglich wird.
Vor dem Testen muß der Einchip-Mikrorechner die Markenvereinbarungen in konkrete Operandenangaben umrechnen. Das macht er nach *T* ENTER. Bei fehlenden Zuordnungen (links stehende Marken) bricht er mit Angabe der gesuchten Bezeichnung ab, Zu große Distanzen bei Relativsprüngen (jr und djnz; vgl. „ABC Einchipmikrorechner" 10 oder 11) erkennt er jedoch nicht. Wenn das Adreßberechnen (Binden) vollständig klappt, entsteht wieder die normale Befehlsanzeige. Der Test kann beginnen.
Zum Testen gibt es vier Kommandos: *S* (Schrittbetrieb) bewirkt die Ausführung des als ersten angezeigten Befehls. Danach gelangt (auch nach Sprüngen) der als nächstes auszuführende Befehl an die erste Stelle. Die aktuelle Flagbelegung erscheint rechts oben. Registerinhalte können mit *R* überprüft werden. Nach Rückkehr von DATA zu PROG (OFF) ist die Weiterführung des Tests ohne Veränderung der Bedingungen möglich. *N* stoppt die Programmausführung erst bei Erreichen des als zweiten aufgelisteten Befehls. Damit lassen sich Unterprogrammsprünge und Schleifen beim Test übergehen. *H* vereinbart einen Haltepunkt, dessen Adresse vierstellig hexadezimal oder symbolisch (mit *:*) angefügt werden muß. Der nächste Stopp erfolgt dann bei Erreichen dieses Punktes im Programmablauf. *L* startet das Anwenderprogramm, ohne es wieder zu stoppen (Echtzeittest).
Das Betriebssystem benutzt die Register von %52 bis %7F. Sie dürfen vom Anwenderprogramm nicht genutzt werden. Variationen des Anzeigebereichs mit *G* behindern das Testen nicht. So können Programmpassagen auch leicht übersprungen werden. Nach Verändern von Befehlen ist jedoch erneutes Binden nötig. Aber auch das verändert weder die rechts oben angezeigten Flags noch die Register außerhalb des Systembereichs %52 bis %73.
Abschließend zu zwei Aspekten, die nach längerer Programmiererfahrung Bedeutung gewinnen können: Mit Maschinenprogrammen kann das Interruptsystem des Einchip-Mikrorechners genutzt werden. Die erweiterte Betriebssoftware

Kommandos bei DATA und PROG

0..9 und A..F	Hexadezimaleingabe
-	Adresse -1
+	Adresse + 1
Enter	Anzeige nächste Zeile
G adr.	Anzeige ab adr
OFF	Rücksprung

Weitere Kommandos bei PROG

: ASCII ASCII ASCII	Markeneingabe
Q	Eingabe in Zeile 1
X	Befehl streichen
I	Byte einfügen
T adr	Binden auf adr
S	Schritt-Test
N	Schleifen-Test
H adr	Haltepunkt auf adr
L	Echtzeit -Test
R	Aufruf DATA

adressiert die Interrupt-Serviceroutinen mit Register-indirekten Sprüngen:
IV0: %74 und %75
IV1: %76 und %77
IV2: %78 und %79
IV3: %7A und %7B
IV4: %7C und %7D
IV5: %7E und %7F
Das Betriebssystem benutzt den Timer T0 (IV4). Anwender-programme sollten ihn nicht in Anspruch nehmen, um schritt-weise Tests nicht zu behindern.
Auch Programme für andere Rechner mit Einchip-Mikrorechner-Schaltkreis kön-nen mit dem JU+TE-Computer entwickelt werden. Um sie auf den dort nötigen Adressen lauffähig zu machen, erfolgt das Binden mit *T* und der Adresse. auf der der oben angezeigte Befehl im anderen Rechner stehen soll. Anschlie-ßend kann z. B. per EPROM die entwickelte Software im Zielsystem installiert werden.

Dr. Helmut Hoyer

Die Freunde des JU+TE-Computers bitten wir, auch auf unseren Leserbriefseiten, Seiten 162 und 163, nachzu-schlagen.
Die Redaktion

Auf den Seiten 234 und 235 folgt der Abdruck des Listings. Das entfällt hier. CRC EPROM 1 (0800-0FFF): 8179, CRC EPROM 2 (2000-27FF): 6C9C. (vp)

Software

Reaktionstest

Heute stellen wir zwei Varianten dieses Spieles vor. Wie der Name schon sagt, kann mit diesem Spiel das Reaktionsvermögen einer Testperson überprüft werden. Ein Spiel besteht aus 10 Versuchen. Variante 1 ist kürzer als 256 Byte. Variante 2 bietet einen höheren Komfort und benötigt 1 KByte Speicherplatz.

Variante 1

```
 1 LET X=0,Z=0
 2 CALL %8DD;
   PRINT "REAKTIONSTEST";
   LET Y=0;
   PROC SETR[6D,0]
 3 PROC SETRR[%F2,%23];
   PROC SETR[F1,10]
 4 LET A=GETR[%F2];
   WAIT A+90;
   CALL %C56;
   IF GETR[%6D]<>0
   THEN PRINT SCHUMMEL";
   GOTO 8
 5 PRINT " LOS !"
 6 LET Y=Y+1;
   CALL %C56;
   IF GETR[%6D]=0
   THEN GOTO 6
 7 PRINT Y;
   PROC SETR[%6D,0];
   LET X=X+1,Z=Z+Y
 8 IF X<10
   THEN WAIT 200;
   GOTO 2
 9 PRINT " ";
   PRINT Z;
   LET A=GTC;
   GOTO 1
```

Es wird die Reaktionszeit gemessen vom Erscheinen eines Startkommandos auf dem Bildschirm bis zum Betätigen einer beliebigen Stoptaste, wobei ein Zufallsgenerator das Erscheinen des Startkommandos steuert. Die angezeigte Reaktionszeit ist nicht in Sekunden geeicht, um eine hohe Zeitauflösung zu erreichen. (1 Sekunde entspricht etwa dem Zahlenwert 00028). Aus Speicherplatzgründen mußten einstellige Zeilennummern gewählt werden (z. B. GOTO 6 = 2 Byte, GOTO 60 = 3 Byte). Vor jedem Spiel wird in Zeile 1 der Versuchszähler X und der Gesamtzähler Z auf Null gesetzt. Nachdem in Zeile 2 die Überschrift geschrieben, der Einzelzeitzähler gesetzt und mit PROC SETR [%6D,0] die dynamische Tastenabfrage vorbereitet wurde, erfolgt in Zeile 3 das Initialisieren des Zufallsgenerators. Der aus dem Zählregister %F2 gelesene Zufallswert wird der Variablen A zugeordnet und mit WAIT A+90 in eine akzeptable Zufallszeit umgewandelt. Somit erreicht man das Erscheinen des Startkommandos „LOS!" nach Ablauf einer zufälligen Zeit. Mit den restlichen Anweisungen in Zeile 4 wird auf vorzeitiges Betätigen einer Stoptaste reagiert und somit ein Schummeln verhindert. Nach Erscheinen des Startkommandos auf dem Bildschirm (Zeile 5) erhöht sich der Zeitzähler bis zum Betätigen einer beliebigen Stoptaste (Zeile 6). Diese Schleife wurde bewußt kurz gehalten (z. B. erfolgt kein Ausdruck der verstrichenen Zeit), um die größtmögliche Zeitauflösung zu erreichen. Nach erfolgter Stoptastenbetätigung wird der Zählvorgang beendet. Auf dem Bildschirm erscheint die benötigte Reaktionszeit (Zeile 7). Nach dem Löschen des Tastencoderegisters werden der Versuchszähler um 1 erhöht

```
10 CALL %8DD;
   PRINT "REAKTIONSTEST";
   PRINT "   DU HAST";
   PRINT " 10 VERSUCHE";
   PRINT " ";
20 PRINT "ES GEHT LOS !";
   WAIT 300;
   LET X=0,Z=0;
30 LET Y=0;
   PROC SETR[%6D,0]
40 CALL %8DD;
   PRINT "REAKTIONSTEST";
50 PROC SETRR[%F2,%23];
   PROC SETR[%F1,10];
60 LET A=GETR[%F2];
   WAIT A+90
70 LET A=A$M10+48
80 CALL %C56;
   IF GETR[%6D]<>0
   THEN PRINT "NICHT";
   PRINT "SCHUMMELN!";
   WAIT 200;
   GOTO 30
90 PROC SETR[%5B,%26]
100 PROC PTC[A];
   PRINT
110 LET Y=Y+1;
   CALL %C56;
   IF GETR[%6D]=0
   THEN GOTO 110
120 LET G=GETR[%6D]$A%7F;
   PROC SETR[%6D,0]
130 IF A<>G THEN
   PRINT "FALSCHE TASTE";
   PRINT "DAS SIND 50";
   PRINT "STRAFPUNKTE!"
135 IF A<>G
   THEN LET A=5,B=1,C=0
   LET D=0,Y=50;
   GOSUB 300;
   WAIT 300;
   GOTO 150
140 PRINT "ZEIT:"Y;
   WAIT 150
150 LET X=X+1,Z=Z+Y
160 IF X<10
   THEN GOTO 30
170 PROC SETR[%5B,%12];
   PRINT "BEENDET !";
   PRINT " "
180 LET Z=Z/10
190 IF Z>45 THEN
   PRINT "DISQUALIFI-";
   PRINT "ZIERT!";
   PRINT " "
195  IF Z>45
```

und die gerade benötigte Zeit zur Gesamtzeit addiert. Zeile 8 fragt ab, ob der Spieldurchgang (10 Versuche) beendet ist. Ist das nicht der Fall, so erfolgt selbständig ein weiterer Versuch. Bei Erreichen des Spielendes (Zeile 9) erfolgt der Ausdruck der Gesamtreaktionszeit. Eine beliebige Tastenbetätigung startet einen neuen Spieldurchgang.

```
   THEN LET A=10,B=2;
   LET C=0,D=10;
   GOSUB 300;
   GOTO 260
200 PRINT "DURCHSCHNITT";
   PROC SETR[%5B,%45]
205 PROC PTC[Z/10+48];
   PROC PTC[Z$M10+48];
   PRINT
210 IF Z<20
   THEN PRINT "SUPER !";
   GOTO 260
220 IF Z<25
   THEN PRINT "GUT !";
   GOTO 260
230 IF Z<30
   THEN PRINT "NA JA !";
   PRINT "GRAD' NOCH SO";
   GOTO 260
240 IF Z<35 THEN
   PRINT "LAHME ENTE !";
   GOT0 260
250 ELSE ;
   PRINT "BIST DU ETWA";
   PRINT "BETRUNKEN ?"
260 LET A=10,B=4,C=1,D=0;
   GOSUB 300
270 PROC SETR[%6D,0];
   LET A=GTC;
   GOTO 10
300 LET E=A
310 IF C=0
   THEN LET F=240,G=-20
320 ELSE ;
   LET F=10,G=20
330 PROC SETR[%F2,F];
   PROC SETR[%F1,%8A]
   LET E=E-1,F=F+G;
   WAIT D;
   IF E>0
   THEN GOTO 330
340 LET B=B-1;
   IF B>0
   THEN GOTO 300
350 PROC SETR[%F1,10];
   RETURN
```

Variante 2

Der wesentliche Unterschied zu Variante 1 besteht darin, daß nach Ablauf einer Zufallszeit eine Zahl zwischen 0 und 9 auf dem Bildschirm erscheint, die dann möglichst schnell gedrückt werden muß. Das Eingeben einer falschen Zahl wird mit 50 Zeitpunkten bestraft und zählt als ein Versuch. Des weiteren erfolgen ausführlichere Texthinweise während des Spiels, Ertönen kurzer Melodien zu bestimmten Anlässen und eine umfangreichere Auswertung nach Spielende. Nach dem Bildschirmlöschen und dem Schreiben der Überschrift werden in Zeile 20 der Versuchszähler X und der Gesamtzeitzähler Z auf Null gesetzt. Das Nullsetzen des Einzelzeitzählers geschieht nach jedem Versuch; deshalb bedarf es einer eigenen Zeilennummer (Zeile 30). PROC SETR [%6D,0] bereitet die dynamische Tastenabfrage vor, Zeile 40 bereinigt den Bildschirm; Zeile 50 initialisiert den Zufallsgenerator. In Zeile 60 wird der aus dem Zählregister %F2 gelesene Zufallswert der Variablen A zugeordnet und in zweifacher Weise verwendet. WAIT A+90 wandelt ihn in eine akzeptable Zufallszeit um und Zeile 70 korrigiert die ermittelte Zufallszahl auf ASCII-Code (z. B. dezimale 0 = %30, dezimale 5 = %35. Dazu muß jeweils %30 bzw. dezimal 48 addiert werden). Zeile 80 reagiert auf vorzeitiges Betätigen einer Stopptaste und verhindert somit ein Schummeln. In Zeile 90 erfolgt die Positionierung des Bildschirmpointers. Anschließend wird die Zutallszahl - ohne führende Nullen - zur Anzeige gebracht (Zeile 100). Von diesem Zeitpunkt an erhöht sich der Zeitzähler Y bis zum Betätigen einer Taste (Zeile 110). Im nächsten Schritt wird geprüft, ob die richtige Taste betätigt wurde, und das Tastencoderegister gelöscht (Zeile 120). Wurde eine falsche Taste betätigt (Zeile 130), so erfolgt nach dem entsprechenden Textausdruck die Initialisierung der Melodieparameter A-D und das Anrechnen der 50 Strafpunkte (Y=50). Danach erfolgt der Absprung ins Melodieprogramm (GOSUB 300). Das Anzeigen der Zeit wird verhindert (GOTO 150). Bei richtiger Tastenbetätigung erscheint auf dem Bildschirm die ermittelte Reaktionszeit (Zeile 140). In Zeile 150 werden der Versuchszähler um 1 erhöht und die gerade ermittelte Zeit zur Gesamtzeit addiert, Nach 10 Versuchen erfolgt die Auswertung (Zeile 160). Zunächst wird dem Spieler das Ende des Spiels mitgeteilt und danach der Durchschnitt der Reaktionszeit errechnet (Zeilen 170, 180). Bei Überschreitung eines

Einfache Melodieerzeugung

```
10  PROC SETR[%6D,0];
    PROC SETRR[%F2,%23];
    PROC SETR[%F1,10]
20  GOSUB 300
30  END
300 LET F=240;
    PROC SETR[%F1,%8A]
310 PROC SETR[%F2,F];
    LET F=F-20;
    IF F<10
    THEN GOTO 310
320 PROC SETR[%F1,10];
    RETURN
```

Maximalwertes wird der Spieler mit einer Melodie disqualifiziert (Zeile 190). Ansonsten erscheint der Durchschnittswert auf dem Bildschirm (Zeile 200). Da die Anzeige ohne die sonst üblichen führenden Nullen erfolgen soll, muß die Zehner- und Einerstelle extra ermittelt und angezeigt werden. Zunächst wird der Bildschirmpointer positioniert. PROC PTC [Z/10+48] errechnet die Zehnerstelle, korrigiert sie auf ASCII-Code und bringt sie zur Anzeige; PROC PTC [Z¤M10+48] verfährt ebenso mit der Einerstelle.
Es folgt die Auswertung der Reaktionszeit, die Ausgabe des entsprechenden Textes, sowie die Belohnung mit einer Melodie (Zeilen 210-260). Anschließend wird das Tastencoderegister gelöscht (Zeile 270). Nach einer Tastenbetätigung beginnt ein neues Spiel.
Nun noch die Beschreibung der Melodieerzeugung (Unterprogramm 300).
Die Erzeugung der Melodie könnte kurzer gehalten werden, würde aber nicht soviel Variationsmöglichkeiten der Parameter A-D bieten. Da der Speicherplatz vorhanden ist, wird eine ausführliche Variante vorgestellt.
Zur Erklärung der Parameter und deren sinnvoller Wahl:
A = Anzahl der Töne innerhalb einer Melodie (A = 2 ... 10)
B = Anzahl der Melodiedurchläufe (B = 1 ... 4)
C = steigende (C = 0) und fallende (C = 1) Melodie
D = Tonlänge (D = 0 ... 50)
E, F, G = Hilfsvariable
Die Parameter A-D werden bereits im Hauptprogramm festgelegt. Durch Änderung derselben können auf einfache Weise auch andere Melodien erzeugt werden.
In Zeile 300 wird zunächst die Information über die Anzahl der Töne gesichert, da dieser Parameter im weiteren Programmdurchlauf verändert wird. Für eine steigende Melodie (C = 0) wird der Anfangston auf F = 240 und die Schrittweite auf G = -20 und für eine fallende Melodie (C = 1) der Anfangston auf F = 10 und die Schrittweite auf G = 20 festgelegt (Zeilen 310 und 320). In Zeile 330 erhält der Zähler T1 den Wert der Variablen F als Zeitkonstante (PROC SETR [%F2,F]) und wird mit PROC SETR [%F1,%8A] für die Zeitdauer D freigegeben. Außerdem wird der Tonzähler E um 1 verringert und der neue Ton voreingestellt (LET F=F+G). Ist die Melodie zu Ende (E = 0), so wird geprüft, ob noch weitere Melodiedurchläufe folgen sollen. (Zeile 340). Wenn das nicht der Fall ist, so erfolgt das Ausschalten der Tonausgabe (Zeile 350) und der Rücksprung ins Hauptprogramm.
Die beschriebene Melodieerzeugung kann auch als Programmbaustein in anderen Programmen verwendet werden. Zu diesem Zweck müssen im Hauptprogramm an geeigneter Stelle der freie Zähler initialisiert, sowie die Parameter A-D sinnvoll festgelegt werden.
Eine sehr einfache Form einer kurzen Melodieerzeugung erhält man, wenn die Parameter A-D nicht frei wählbar sind. Das Programmbeispiel "Einfache Melodieerzeugung" zeigt eine solche Variante.

Bernhard Piniek

Kurzzeitwecker

```
10 CALL %8DD;
   PRINT "KURZZEIT-";
   PRINT "WECKER";
   INPUT "MINUTEN:"A
20 LET B=0;
   PROC SETR[%6D,0]
30 PROC SETR[%5B,%40];
   PRINT "SCHON"B;
   PRINT "NOCH "A-B
40 WAIT 9300;
   LET B=B+1;
   IF B<A
   THEN GOTO 30
50 PROC SETR[%5B,%40];
   PRINT "ZEIT";
   PRINT "ABGELAUFEN"
60 PROC SETR[%F3,%13];
   PROC SETR[%F1,%8A];
   LET A=0
70 LET A=A+1;
   PROC SETR[%F2,A];
   GOTO 70
```

Dieses Programm kann man zur Signalerzeugung nach einer einstellbaren Zeit verwenden, z. B. als Eieruhr oder als Wecker. Nach dem Löschen des Bildschirmes werden die Überschrift angezeigt und die Zeiteingabe in Minuten gefordert. Die Anweisung 20 löscht den Minutenzähler B und schaltet den Tastenton aus.
Mit der Anweisung 30 beginnt die Schleife. Zuerst wird der Bildschirm-Pointer (Register %5B) auf die Zeile Nr. 4 gestellt, wo die vergangene (B) und die noch ablaufende (A-B) Zeit zur Anzeige kommen. Die Zeile 40 wartet eine Minute und erhöht dann den Zähler B um 1. Wenn die Gesamtzeit A noch nicht erreicht ist, wird mit der Zeile 30 fortgesetzt.
Die Anweisung 50 stellt nach abgelaufener Zeit den Bildschirm-Pointer auf den Anfang der Zeile 4 und druckt auf den Bildschirm „Zeit abgelaufen”.

Die Zeile 60 schaltet den Timer T1 als Tongenerator auf P36. Hier muß ein Lautsprecher angeschlossen sein. Die Tonhöhe wird mit der Variablen A in der Anweisung 70 ständig verändert. Dadurch kann man den Weckton nicht überhören. Das Ausschalten erfolgt mit der RESET-Taste. Wenn der Kurzzeitwecker vor- oder nachgeht, muß die Zahl in der WAIT-Anweisung (Zeile 40) verändert werden.

Denis Hoyer

Computer-Tips

Die zahlreichen Leserbriefe lassen nicht nur die große Verbreitung des JU+TE-Selbstbau-Computers erkennen, sondern auch seine gute Nachbausicherheit. Trotzdem wollen wir einige Hinweise, die aus jahrelanger Erfahrung und einigen Zuschriften (für die wir uns herzlich bedanken) stammen, veröffentlichen. Damit soll eventuellen Schwierigkeiten bei der Inbetriebnahme neuer und der Erweiterung bereits funktionierender vorgebeugt werden. Wichtig ist es, Überlastungen an den Anschlüssen des Einchip-Mikrorechners sorgfältig zu vermeiden. Das kann sonst den teilweisen oder gar vollständigen Ausfall dieses Schaltkreises zur Folge haben. Zu den Vorsichtsmaßnahmen gehört auch, den Fernseher und Peripheriegeräte an den Ports nur im ausgeschalteten Zustand aller Komponenten anzuschließen oder zu trennen. Auch beim Löten am (natürlich ausgeschalteten) Computer sollte der Fernseher nicht angeschlossen sein, da er statische Aufladungen beachtlicher Stärke verursachen kann.

Gestörte Tonerzeugung

Das im Heft 12/87 (S. 931 f.) veröffentlichte Magnetbandinterface enthält einen Lautsprecher-Anschluß, auf den auch P37 wirkt. Das kann die akustische Ausgabe mit den Bildsynchronimpulsen überlagern. Daher empfiehlt es sich, die Verbindung zwischen den Pins 10 und 11 des DL 000 D auf dieser Leiterplatte zu trennen. Das im Heft 3/89 veröffentlichte Interface besitzt diese Schwäche nicht.

Langsame EPROM

Je mehr Speicherschaltkreise am Bus des JU+TE-Computers angeschlossen sind, desto kritischer werden durch die wachsende kapazitive Last die Zugriffszeiten. Im Extremfall können sie auf 250 ns zusammenschmelzen. Besonders unter den Basteltypen gibt es jedoch EPROM, die langsamer sind. RAM-Schaltkreise bereiten weniger Probleme (U 6516 D: typisch 150 ns).
Die kritische Verzögerungszeit besteht in der Reaktion auf das Aktivieren des /CE-Signals, das beim JU+TE-Computer gemeinsam mit aus dem Datastrobe-Signal des Einchip-Mikrorechners abgeleitet wird. Das hat den Vorteil, daß ein EPROM wenig Betriebsstrom verbraucht, solange er nicht angesprochen wird. Durch Verzicht auf diese Energie-Sparmaßnahme können langsame EPROM wie der U 2716 C 45 auch bei ausgebauter Speicherausstattung zum Einsatz kommen.
Dazu muß der Leiterzug zum /CE-Eingang (Pin 18) auf der Speichermodul-Platine so durchtrennt werden, daß das Auswahlsignal /CSB nur noch mit dem /OE-Eingang (Pin 20) verbunden bleibt. Der Schaltkreis reagiert auf diesen Eingang innerhalb von ca. 100 ns. /CE (Pin 18) erhält statt dessen aber einen kurzen Schaltdraht Massepotential. RAM-Schaltkreise erlauben eine solche Maßnahme nicht, da dies ungewolltes Verändern des Speicherinhalts gestatten und bei Typen mit Adreßpuffern das Übernehmen der Adreßinformation blockieren würde.

Schwaches Taktsignal

Das Taktsignal XTAL erzeugt im JU+TE-Computer eine Schwingschaltung aus zwei DL 000 D-Gattern. Hiermit wird durch Exemplarstreuungen nicht immer ein ausreichend hoher 1-Pegel erzielt. Erst ab 3,8 V ist die Funktionsfähigkeit des UB 8830 D unter allen Bedingungen sichergestellt. Hier hilft ein Pull-up Widerstand (ca. 750 Ω), der Pin 6 des DL 000 D mit der Versorgungsspannung 5P verbindet. Er kann auf der Lötseite ergänzt werden und gewährleistet einen sicheren Programmstart nach jedem Rücksetzen.

Instabiler Bildaufbau

Schwierigkeiten bei der Bilderzeugung können aus zu langsamen EPROM oder zu schwachem Taktsignal herrühren. Zuweilen spielen aber auch abweichende Signallaufzeiten pinkompatibler Schaltkreise, die z. B. statt des DL 074 D zum Einsatz kommen, eine Rolle.

Hier kann durch Variieren des 15-pF-Kondensators oder Ergänzen eines Kondensators (ca. 300 pF) zwischen Pin 8 des DL 000 D und Masse (00) Abhilfe geschaffen werden. Zu warnen ist vor sehr alten Basteltypen des D 195 D, bei denen zum Teil einige Flipflops gar nicht funktionieren. Das äußert sich dann in senkrechten Streifen auf dem Bildschirm. Waagerechte Streifen haben dagegen meist eine instabile Versorgungsspannung (zu schwacher Trafo) als Ursache.

Zeilennummern in TINY-MP-BASIC

Zum effektiven Strukturieren von BASIC-Programmen verwendet der Editor des JU+TE-Computers zwei spezielle Bytes: %0D (Zeilenende) und %00 (Programmende). Diese dürfen sonst nicht im BASIC-RAM erscheinen. Als niederer Teil der Zeilennummer wären sie aber denkbar. Der Editor ersetzt selbständig das Nullbyte mit %01. Daher erscheint bei LIST als Zeilennummer statt 0 die 1, statt 256 die 257, statt 512 die 513 usw. Nicht verwendet werden dürfen die Zeilennummern 13. 269, 525, 781 usw., da sie %0D als niederes Byte im dualen Zahlenformat besitzen Die höchstmögliche Zeilennummer ist 32767.

Magnetband-Interface

Obwohl der Operationsverstärker B 761 D des Magnetband Einganges nicht stark gegengekoppelt ist, kann eine Schwingneigung auftreten. Sie läßt sich mit einem Kondensator (ca. 22 pF) zwischen dem Ausgang (Pin 5) und dem Frequenzgang-Korrektureingang (Pin 6) bekämpfen. Bei Einsatz des Anfalltyps B 861 D als Operationsverstärker sollte dieser Kondensator vorsichtshalber grundsätzlich ergänzt werden. Mit dieser Maßnahme entsteht bereits am Ausgang des Operationsverstärkers ein sauberes Rechtecksignal. Das macht bei der Variante für das 4K-Betriebssystem den B 555 D entbehrlich. Er kann durch eine Brücke statt der Anschlüsse 2 und 3 ersetzt werden. Eine Kontroll-LED vermag der Operationsverstärker jedoch nicht zu steuern.

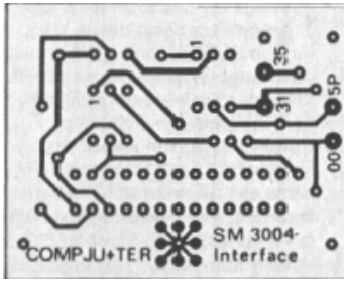
Dr. Helmut Hoyer

Schreibmaschine 3004 als Drucker

Das Büromaschinenwerk Sömmerda fertigt eine hochwertige Schreibmaschine mit der Bezeichnung „Erika 3004 electronic". Sie besteht aus einem Typenrad-Druckwerk, einer Elastomer-Tastatur und einer elektronischen Steuerung, die als Kern den Einchip Mikrorechner UB 8840 M enthält. Da sie einen Interface-Steckverbinder besitzt, eignet sie sich für die Kopplung mit Computern. Besonders nützlich ist ihr Einsatz als Drucker. Wir stellen die Kopplung des JU+TE-Computers mit der „S 3004" als Drucker vor.

Für den Datenaustausch verwendet die Schreibmaschine 3004 das Serieninterface (SIO) des Einchip-Mikrorechners. Daher müssen Zeichencodes und Druck-Kommandos im seriellen Format angeboten werden. Zur Geschwindigkeitsanpassung erzeugt die Schreibmaschine ein Rückmeldesignal, das mit 1-Pegel den Empfang und mit 0 Pegel die Ausführung eines Zeichens oder Kommandos quittiert. Für diese Signale werden die Pins P30 und P36 des in der Schreibmaschine enthaltenen Einchip-Mikrorechners verwendet. Sie liegen direkt am Interface-Steckverbinder an und sind daher sehr vorsichtig zu behandeln. Um allen denkbaren Schäden, für die der Hersteller natürlich nicht haften würde, vorzubeugen, empfiehlt sich dringend die Anwendung einer potentialtrennenden Interface-Schaltung. Unsere Abbildungen stellen eine einfache und erfolgreich erprobte Möglichkeit vor. Die einseitige Leiterplatte (Abb. unten) sitzt am 25poligen Flachsteckverbinder, der in die Interface-Buchse der Schreibmaschine paßt (vgl. Foto S. 377). Deren Konstruktion läßt wenig Platz, so daß die Schrauben zur Befestigung des Steckverbinders in der Leiterplatte versenkt werden müssen. Außerdem empfiehlt es sich, die 14 der Schreibmaschine nächsten Anschlußstifte zu kürzen und möglichst flach zu verlöten.

Für die Zeichendarstellung gilt nicht der Standardcode (ASCII). Statt dessen ist die Position des gewünschten Zeichens auf dem Typenrad der Schreibmaschine anzugeben. Mit Hilfe einer entsprechenden Tabelle für die im Computer verwendeten ASCII läßt sich dieser Code jedoch leicht erzeugen. Zusätzlich gibt es einige Druck-Kommandos, von denen das Lehrzeichen (%71) und die Zeilenschaltung (%77) die wichtigsten sind.



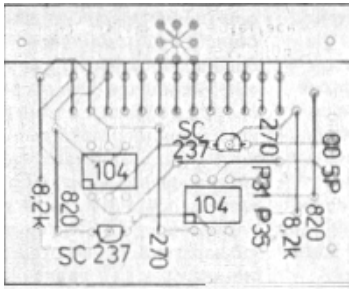
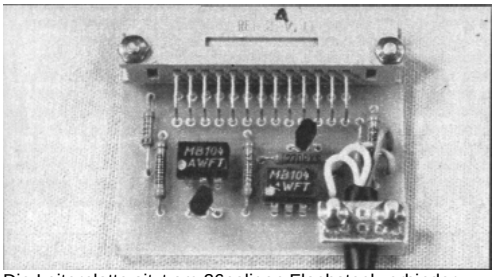


Abb. links außen: Layout der einseitigen Leiterplatte

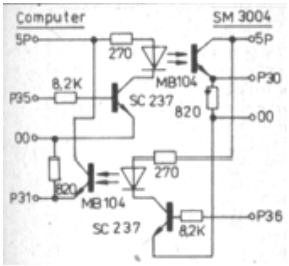
Abb. links: Bestükkungsplan



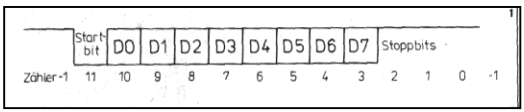
Die Leiterplatte sitzt am 26poligen Flachsteckverbinder.

Die Übertragung gelingt am einfachsten mit dem seriellen Interface. Es benutzt P37 als Ausgang. Da dieses Signal im JU+TE-Computer die Synchronimpulse zur Bilderzeugung ausgibt, würden sich Druckersteuerung und Bilderzeugung gegenseitig ausschließen. Die Übertragungsrate von nur 1200 Bit/s läßt aber genügend Zeit, das serielle Format (Abb. 1) programmtechnisch zu realisieren und P35 für die Ausgabe zu verwenden. Unser Beispiel basiert auf dieser Idee und läßt sich mit dem 4K-Betriebssystem (vgl. JU+TE 3/89) gut im RAM-Speicherbereich des JU+TE-Computers von %F900 bis %FA38 unterbringen. Dabei bleibt genug Platz für die Markentabelle, deren Beginn (höchste Adresse) mit %FBFF festgelegt wird. Die Eingabe der Codetabelle (Abb. 2) (%F900 bis %F93F) erfolgt mit der RAM- und Registeranzeige DATA, die der Programme (Abb. 3 bis 9) mit PROG, natürlich ohne die Kommentare. Voraussetzung ist selbstverständlich die Belegung dieses Bereichs im Computer mit einem RAM-Schaltkreis. Die Interruptserviceroutine ISR (Abb. 3) wird mit dem Timer T1 stets dann gerufen, wenn die Zeit für die Übertragung eines Bits (833 µs) abgelaufen und das nächste Bit auszugeben ist. Mit Hilfe eines Zählregisters (%3E) verfolgt sie den Übertragungszustand. Begonnen wird mit dem Zählerstand 11. Er entspricht dem Startbit. Über die Marke IS1 erfolgt die Ausgabe des zugeordneten 0-Pegels auf P35. Bei den Zählerständen 10 bis 3 werden nacheinander die Bits D0 bis D7 aus dem Schieberegister (%3F) über das C-Flag auf P35 übertragen. Danach folgen Stoppbits (vgl. Abb. 1). Um das nächste Zeichen zu übertragen, müssen das Schieberegister mit dem neuen Code und der Zähler mit der Zahl 11 (%0B) geladen werden.

Das erledigt das Unterprogramm DRU (Abb. 4). Es dient dem Drucken eines Zeichens und erwartet es als ASCII im Arbeitsregister r1. Die ersten beiden Befehle erzeugen in r0 und r1 die Tabellenadresse, die auf den dem ASCII zugeordneten Schreibmaschinencode weist. Dieser gelangt durch den dritten Befehl in r1. An der Marke DR1 wartet das Programm auf die Bereitschaft der Schreibmaschine. Danach wird das Aufrufen der Interruptserviceroutine aller 832 µs initialisiert. Das ist nicht ganz die exakte Periodendauer, liegt aber innerhalb des zulässigen Fehlerbereichs von ± 2%. Nach Laden des Schieberegisters mit dem Zeichencode (oder Druck-Kommando) erhält das Zählregister den Startwert 11. Es folgen die Freigabe der T1-Interruptannahme und an der Marke DR2 das Abwarten der Empfangsbestätigung der Schreibmaschine. Ist die Ausgabe beendet, erhält der Timer T0 wieder das Interruptrecht, während T1-Interrupts verboten bleiben. Damit wird wieder die Bilderzeugung freigegeben. die aus Zeitgründen während der Übertragung ausgesetzt war. Das macht sich durch ein Zucken auf dem Bild bei jedem Zeichen bemerkbar. Mit dem Unterprogramm DRU besitzt der JU+TE-Computer auf diese Weise eine Peripherie-



Schaltbild des potentialtrennenden Interfaces



Schnittstelle zum Drucken eines ASCII, das in r1 zu übergeben ist. Darauf können weitere Programme aufbauen. Effektiv ist das Abschreiben des Bildschirmes, da mit dieser Methode alle Bildausgaben wie Dezimal- und Hexadezimalanzeigen auch zum Drucken taugen. Das Programm HCB (Abb. 5) nutzt dazu den ASCII-Bildwiederholspeicher (%FD00 bis %FD7F) als Druckpuffer. Die erste Schleife ab Marke HC1 sucht die letzte genutzte Bildschirmposition. Die zweite Schleife ab Marke HC2 kopiert das Bild bis zu dieser Position auf den Drucker, wobei das noch ausstehende Unterprogramm NBA zum Berechnen der nächsten Bildadresse benutzt wird. Zum Drucken einer Zeile ist diese also zunächst auf den Bildschirm auszugeben und dann mit dem Unterprogramm HCB auf die Schreibmaschine zu übertragen.

Die Zeilenschaltung erfolgt mit dem Druck-Kommando %77. Zur Anpassung an das übliche A4-Format empfiehlt sich das Mitzählen der Zeilenschaltungen, um die Ausgabe zum Blattwechsel zu unterbrechen. Diesem Zweck dient das Unterprogramm ZLS (Abb. 6). Es benutzt r4 als Zeilenzähler und r3 als Speicher für die Zahl der Zeilen je Blatt. Vor Ausgabe des Druck-Kommandos wird der Zähler dekrementiert. Wenn dabei Null steht, erhält er erneut den Startwert. Außerdem erwartet das Programm dann eine Tastenbetätigung als Bestätigung des Papierwechsels, bevor mit dem Druck-Kommando %77 in r1 zum Ausgabeprogramm gesprungen wird. Die Marke DR1 dient als

2 Codetabelle

```
 F900: 71 42 43 41
 F904: 48 04 02 17
 F908: 1D 1F 1B 25
 F90C: 64 62 63 40
 F910: 0D 11 10 0F
 F914: 0E 0C 0B 0A
 F918: 09 08 13 3B
 F91C: 71 2E 71 35
 F920: 30 30 18 20
 F924: 14 34 3E 1C
 F928: 12 21 32 24
 F92C: 2C 16 2A 1E
 F930: 2F 14 36 33
 F934: 37 28 22 2D
 F938: 26 31 38 27
 F93C: 39 27 07 01
```

3 Interruptserviceroutine

```
*ISR: A6 3E 0B  Zähler = 11 ?
 F943 6B :IS1   dann Startbit
 F945 FB :IS2   wenn größer
 F947 A6 3E 03  Zähler kleiner 3 ?
 F94A 7B :IS3   dann Stoppbit
 F94C E0 3F     Schieben rechts,
 F94E 7B :IS3   wenn 1-Bit
*IS1: 00 3E     Zähler - 1
 F952 56 03 DF  P35 := 0
 F955 BF        Rücksprung
*IS3: 00 3E     Zähler - 1
*IS2: 46 03 20  P35 := 1
 F95B BF        Rücksprung
```

4 Druck eines Zeichens (UP)

```
*DRU: 0C F9      Tabellenadresse H
 F95E 26 E1 20   Adresse L aus ASCII
 F961 82 10      r1 := SM-Code
*DR1: 76 03 02   P31 = 1 ?
 F966 EB :DR1    dann warten
 F968 8F         Interruptsperre
 F969 E6 F3 37   PRE1 := 13 µs
 F96C E6 F2 40   T1 := 832 µs
 F96F E6 7E F9   ISR-Adresse H
 F972 E6 7F 40   ISR-Adresse L
 F975 46 F1 0E   T1-Start
 F978 19 3F      Schieberegister := Code
 F97A E6 3E 0B   Bitzähler := 11
 F97D E6 FB A0   Freigabe T1-Interrupts
*DR2: 76 03 02   P31 = 1 ?
 F983 6B :DR2    sonst warten
*DR3: 44 3E 3E   Ausgabe beendet ?
 F988 DB :DR3    sonst warten
 F98A 8F         Interruptsperre
 F98B E6 FB 90   Freigabe T0-Interrupts
 F98E AF         Rücksprung
```

5 Drucken des Bildinhalts

```
*HCB: 6C FD      BWS-Adresse H
 F991 7C 80      Endadresse L
*HC1: 00 E7      Adresse - 1
 F995 82 16      r1 := ASCII
 F997 A6 E1 20   Leerzeichen ?
 F99A 6B :HC1    dann weiter suchen
 F99C 58 E7      r5 := Bildendadresse
 F99E 7C 00      r7 := Anfangsadresse
*HC2: 82 16      r1 := ASCII
```

Schnittstelle zur Übergabe von Schreibmaschinencodes (statt ASCII bei DRU).
Das letzte Unterprogramm NBA (Abb. 7) ist das einfachste. Es erhöht den niederen Teil der Bildschirmadresse r7 und sichert dabei den lückenlosen Anschluß beim Übergang von einer Bildzeile zur nächsten.
Auf der Grundlage dieser Unterprogramme fällt es leicht, die Schreibmaschine zum Drucken von im JU+TE-Computer gespeicherten Programmen zu nutzen. Diesem Zweck dienen die Hauptprogramme BCY (Abb. 8) und MCY (Abb. 9), die über den Programm-Modus ab den Adressen %F9C9 und %FA03 gestartet werden können und effektive Erweiterungen des Betriebssystems darstellen.
Zum Drucken von BASIC-Programmen mit BCY wird das Doppelregister %10 benutzt, mit dem das Betriebssystem den BASIC-RAM adressiert. Durch Aufrufen der Systemkomponente BASIC und ggf. Einstellen des Manager-Programms muß das betreffende BASIC-Programm aktiviert werden. Nach ENTER erhalten %10 und %11 dessen Startadresse. Über RESET und PROG kann nun das Druckprogramm BCY genutzt werden.
Da BASIC-Zeilen etwa 100 Zeichen enthalten können, setzt es im Querformat eingespanntes Papier voraus. Ist einzeiliger Blatttransport eingestellt, passen so 40 Zeilen auf einen A4-Bogen. Nach Löschen des Bildschirmes werden zuerst die Zeilennummer, dann die BASIC-Zeile zur Anzeige gebracht und anschließend gedruckt. Der Abbruch erfolgt mit einem Sprung

```
 F9A2 D6 :DRU   Druck
 F9A5 D6 :NBA   nächste Adresse
 F9A8 A2 57     fertig ?
 F9AA FB :HC2   wenn nicht,
 F9AC AF        Rücksprung
6 Zeilenschaltung (UP)
*ZLS: 00 E4     Zeilenzähler - 1
 F9AF EB :ZL1   wenn größer 0
 F9B1 48 E3     Zähler := Zeilenzahl
 F9B3 B0 6D     Tastencoderegister := 0
*ZL2: D6 0C 56  dyn. Tastenabfrage
 F9B8 6B :ZL2   Warten auf Betätigung
*ZL1: 1C 77     Code Zeilenschaltung
 F9BC 8B :DR1   Ausführung
7 Berechnen der nächsten Bildadresse (UP)
*NBA: 7E        BWS-Adresse L + 1
 F9BF 66 E7 0D  Zeilenende ?
 F9C2 ED :NB1   wenn nicht
 F9C5 06 E7 03  Korrekturaddition
*NB1: AF        Rücksprung
8 Druck eines BASIC-Programms
*BCY: E6 1F 16  Pointer für LIST
 F9CC E6 6E 0C  UP-Adresse H für LIST
 F9CF E6 6F E6  UP-Adresse L für LIST
 F9D2 E6 43 28  Zeilenzähler := 40
 F9D5 E6 44 28  Zeilenzahl := 40
*BC1: D6 08 DD  Bild löschen
 F9DB 31 10     Registerpointer := %10
 F9DD 82 20     r2 := Zeilennummer H
 F9DF 42 22     Programmende ?
 F9E1 6D 08 12  dann Anfangsmenü
 F9E4 A0 E0     nächste Adresse
 F9E6 82 30     r3 := Zeilennummer L
 F9E8 56 E2 7F  Kennbit löschen
 F9EB D6 0A A3  Anzeige Zeilennummer
 F9EE E6 5B 04  Kursor := 4
 F9F1 D6 0D CC  Anzeige BASIC-Zeile
 F9F4 FD 08 12  wenn Syntaxfehler
 F9F7 A0 E0     nächste Adresse
 F9F9 31 40     Registerpointer := 140
 F9FD D6 :HCB   Druck des Bildinhalts
 F9FE D6 :ZLS   Zeilenschaltung
 FA01 8B :BC1   nächste BASIC-Zeile
9 Druck eines Maschinenprogramms
*MCY: 31 60     Registerpointer := %60
 FA05 D6 08 DD  Bild löschen
 FA08 D6 23 1F  Eingabe Anfangsadresse
 FA0B E9 5E     %5E := Adresse H
 FA0D F9 5F     %5F := Adresse L
 FA0F 20 5B     Kursor + 1
 FA11 D6 23 1F  Eingabe Endadresse
 FA14 A0 EE     Endadresse + 1
 FA16 31 40     Registerpointer := %40
 FA18 3C 38     Zeilenzahl := 56
 FA1A 4C 38     Zeilenzähler := 56
*MC1: D6 08 DD  Bild löschen
 FA1F D6 26 0E  Anzeige Programmzeile
 FA22 31 40     Registerpointer := %40
 FA24 D6 :ZLS   Zeilenschaltung
 FA27 D6 :HCB   Druck des Bildinhalts
 FA2A 08 5E     aktuelle Adresse H
 FA2C 18 5F     aktuelle Adresse L
 FA2E 24 6F E1  - Endadresse
 FA31 34 6E E0  - Endadresse H
 FA34 7B :MC1   wenn nicht fertig
 FA36 8D 08 12  Anfangsmenü
```

auf Adresse %0812 bei Programmende (0 als Zeilennummer H) oder Syntax-Fehler. Für das Drucken wird der Registerpointer auf %40 gestellt, um das LIST-Programm (%0DCC) nicht zu stören. Das Typenrad enthält übrigens weder < noch >. Diese Zeichen müssen daher von Hand nachgetragen werden.
Das Drucken von Maschinenprogrammen erfordert zuerst die Eingabe von Anfangs- und Endadresse. Das Doppelregister %5E verwaltet die aktuelle Adresse des zu druckenden Programms. Maschinenprogramme haben kurze Zeilen, so daß Hochformat mit 56 Zeilen eingestellt wird. Wie beim Druck von BASIC-Programmen erfolgt ein Halt erst nach Beschreiben eines Blattes. Kommentare können mit der Schreibmaschinen-Tastatur nachgetragen werden, bevor der Computer nach Blattwechsel eine Quittung bekommt.
Vor dem Start eines Druckprogramms ist nach dem Eingeben zu binden (PROG-Kommando T), damit die korrekten Adressen im Maschinenprogramm erscheinen. Zum Speichern der beschriebenen Programme auf Kassette muß bei SAVE der Adreßbereich von %F900 bis %BFFF vereinbart werden. Er schließt die Markentabelle ein.

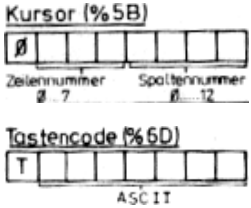
Dr. Helmut Hoyer

Foto: Behnisch, Zeichnungen: Hoyer; Liebig

Unterprogramme im 4K-Betriebssystem

Das erweiterte Betriebssystem des JU+TE-Computers enthält einige Unterprogramme, die sich beim Programmieren in Maschinensprache zuweilen gut nutzen lassen. Sie arbeiten mit den Registern %10 bis %1F und %50 bis %7F. Das Register %5A dient der Übergabe von Zeichencodes (ASCII). das Register %5B als Kursor. Es enthält stets die Zeichenposition, auf die die nächste Bildschirmausgabe erfolgt.
Bei der Hexadezimalanzeige werden Bytes in %5D, Doppelbytes dagegen in %5E und %5F übergeben. Die Eingabe von Bytes erfolgt in das Register %6C, von Doppelbytes in das Registerpaar %6E und %6F. Das T-Bit im Tastencoderegister %6D ist mit 1 belegt, wenn beim vorigen Aufruf einer Tastenabfrage eine Betätigung erkannt wurde. Einige Unterprogramme setzen spezielle Belegungen des Registerpointers RP und anderer in der Tabelle genannter Register voraus.
Der vom Betriebssystem benutzte Bereich kann durch die Ausführung der Unterprogramme verändert werden. Nutzerdaten sollten daher im Bereich von %20 bis %4F stehen. Werden die Programme des BASIC-Editors (%0AA3, %0DCC und %0E92) nicht benutzt, bleiben die Register %04 bis %4F frei.

Dr. Helmut Hoyer



Liste allgemein nutzbarer Unterprogramme des erweiterten Betriebssystems (4 KByte)

Adresse	Funktion	Bedingung
%0824	Tastenabfrage statisch mit Ausführung auf Bildschirm, Kursor (%5B) wird aktualisiert, %5A := ASCII	
%0827	Darstellen eines ASCII aus %5A auf dem Bildschirm, Kursor (%5B) wird aktualisiert	
%0872	Darstellen eines ASCII aus %5A auf dem Bildschirm, Kursor (%5B) wird aktualisiert. Steuerzeichen werden nicht erkannt	
%0875	Erhöhen des Kursors (%5B) um 1 mit ggf. Bildrollen	
%0878	Erhöhen des Kursors (%5B) um den Inhalt von %5C mit ggf. Bildrollen	
%08DD	Bild löschen, Kursor (%5B) := 0	
%0AA3	Dezimalausgabe aus %12 und %13 ohne führende Nullen	RP = #%10
%0ACE	Zeilenschaltung	
%0AD4	Zeilenschaltung mit Freizeilensperre	
%0B95	Darstellen eines ASCII aus %5A auf dem Bildschirm, Kursor bleibt unverändert, Steuerzeichen werden nicht erkannt	
%0C1D	Tastenabfrage statisch ohne Ausführung auf dem Bild-schirm %5A:= ASCII	
%0C56	Tastenabfrage dynamisch mittels Tastencoderegister (%6D): T-Bit und ASCII	
%0DCC	Anzeige einer BASIC-Zeile ohne Zeilennummer ab Adresse aus %10 und %11, die Adresse wird aktualisiert	RP = #%10 %6E = #%0C %6F = #%E6
%0E92	Dezimalanzeige aus %12 und %13 ohne führende Nullen mit Zeilenschaltung	RP = #%10
%20B6	Hexadezimalanzeige eines Bytes aus %5D, Kursor (%5B) wird aktualisiert	
%20CF	Hexadezimalanzeige der Speicherzelle mit der Adresse aus %5E und %5F, Adresse wird um 1 erhöht, der Kursor (%5B) wird aktualisiert	
%20DB	Hexadezimalanzeige eines Doppelbytes aus %5E und %5F, Kursor (%5B) wird aktualisiert	
%20E6	Berechnen der Byteanzahl in %6B eines Befehls mit dem Operationscode aus %5D	
%22E5	Hexadezimaleingabe eines Bytes in %6C mit Tastatur und Anzeige	RP = #%60
%231F	Hexadezimaleingabe eines Doppelbytes in %6E und %6F mit Tastatur und Anzeige	RP = #%60
%2352	RAM- und Registeranzeige	
%23E6	Programm-Modus	
%260E	Anzeige einer Maschinenprogrammzeile mit der Adresse aus %5E und %5F, Adresse und Kursor (%5B) werden aktualisiert, RP := #%60	
%268F	Magnetbandausgabe	RP = #%60
%27D1	Magnetbandeingabe	RP = #%60

Software

Pasch

Heute stellen wir Euch das bekannte Würfelspiel PASCH vor. Mit diesem Spiel stoßen wir bereits an Grenzen der Möglichkeiten mit TINY BASIC. Alle möglichen Variablen A bis Z werden benutzt, einige sogar mehrfach. Der Speicherplatzbedarf für dieses Programm beträgt %0CBC, also gut 3 KByte. Das heißt, der Rechner muß eine der folgenden RAM-Konfigurationen haben:

- 10x U 224 oder
- 3x U 6516 oder
- 1 x U 6264.

Auch die Laufzeiten einiger Programmschleifen wirken sich schon spürbar aus. In einigen Fällen konnten bestimmte Routinen nur durch erhöhten Softwareaufwand auf vertretbare Laufzeiten gebracht werden, so das „Würfeln" und der Bildaufbau der fünf Würfelwerte. Um bei der Zeilennumerierung unter 1000 zu bleiben, reichte die Zehner-Schrittweite nicht mehr aus. Die Programmlänge läßt eine detaillierte Programmbeschreibung nicht zu. Die Kenntnis einzelner Algorithmen aus vorrangegangenen Veröffentlichungen setzen wir voraus.

Spielbeschreibung

Das JU+TE-PASCH-Spiel ist mit den bekannten Spielregeln identisch. Es wird mit fünf Würfeln gespielt. Jede Partie besteht aus 15 Runden. Ziel des Spieles ist es, möglichst viele der in der Tabelle aufgeführten Würfelkombinationen zu erreichen, um somit eine möglichst hohe Gesamtpunktzahl zu erzielen. Jede Wertungsrunde besteht aus drei Würfen, die aber nicht alle zur Anwendung kommen müssen. Nach dem ersten und zweiten Wurf könnt ihr selbst entscheiden, welche Würfel ihr nochmals würfeln und welche ihr bereits für die Wertung behalten möchtet. Spätestens nach dem dritten Wurf erfolgt die Wertung. Dann müßt ihr Euch entscheiden, in welche Rubrik der Tabelle ihr Euren Wurf eintragt. Allerdings kann in jede Rubrik nur einmal ein Wert eingetragen werden. Da die Rundenzahl begrenzt ist, muß zum Spielende in jeder Rubrik ein Wert (auch 0 ist ein Wert) stehen.

Rubriken

1: bis 6: Hier werden Einsen bis Sechsen gesammelt. Wenn die Summe in diesen sechs Rubriken zum Spielende 63 oder mehr beträgt, erhaltet ihr zusätzlich einen Bonus von 50 Punkten.
E: (Ein Paar) Eingetragen wird ein Wurf mit mindestens zwei gleichen Würfelwerten (z. B. 4-2-3-1-2, Gutschrift: 4 Punkte). Sind zwei Paare vorhanden, so wird das höhere Paar gewertet.
Z: (Zwei Paare) Die Paare müssen voneinander unterschiedlich sein.
D: (Drei Gleiche) Eingetragen wird ein Wurf mit mindestens drei gleichen Würfelwerten.
V: (Vier Gleiche) In diese Rubrik kann auch ein Pasch eingetragen werden.
<: (Kleine Straße) Hier kann nur der Wurf 1, 2, 3, 4, 5 eingetragen werden.
>: (Große Straße) Sie besteht aus dem Wurf 2, 3, 4, 5, 6.
K: (Kuchen) Ein Kuchen besteht aus zwei und drei Gleichen. Ein Pasch wird nicht als Kuchen gewertet.
C:(Chance) Es wird die Summe aller fünf Würfel gebildet und eingetragen, unabhängig von der Zusammensetzung des Wurfes.
P: (Pasch) Hier kann nur ein Wurf mit fünf gleichen Würfelwerten eingetragen werden. Die Gutschrift beträgt immer 50 Punkte.

Ablauf des Spieles

Nach dem Start mit RUN erscheint nach einer bestimmten Würfelzeit der 1. Wurf mit den fünf Würfelwerten und dem Angebot zum Löschen einiger oder aller Würfel.
Löschen von Würfeln: Soll zum Beispiel der 2. und 5. Würfel gelöscht werden, so betätigt man die Zifferntasten 2 und 5; und es erscheint auf diesen Positionen eine 0. Mit ENTER folgt der 2. Wurf. Auf den gelöschten Positionen erscheinen neue Würfelwerte. Es besteht wiederum die Möglichkeit zum Löschen beliebiger Würfel in der beschriebenen Weise. Mit ENTER wird schließlich der 3. Wurf ausgeführt. Ein Löschen nach dem 3. Wurf ist nicht mehr möglich. Soll nach dem 1. oder 2. Wurf nicht gelöscht werden, so kann man mit SHIFT-ENTER sofort zur Wertung gelangen.
Wertung: Nach der Aufforderung zur Wertung, die akustisch begleitet wird, muß sich der Spieler entscheiden, in welche Rubrik der Tabelle er sein Würfelergebnis eintragen möchte. Durch Druck auf die entsprechende Taste erfolgt der Wertungseintrag. Mögliche

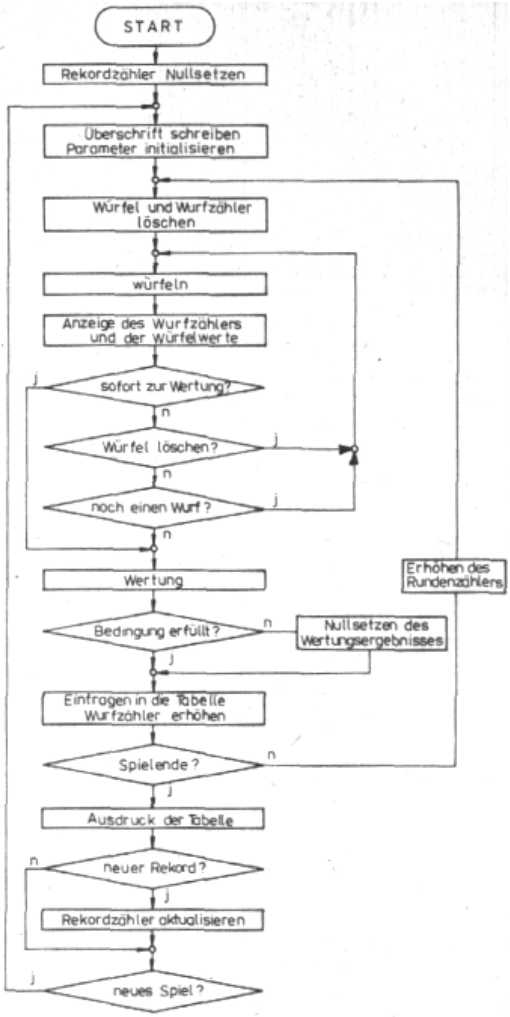
Wertungen sind: 1, 2, 3, 4, 5, 6, E, Z, D, V. <, >. K, C. P (vgl.: „Rubriken"). Um Fehleingaben zu vermeiden, muß das Eintragen einer Wertung über SHIFT erfolgen (z. B. SHIFT-P). Die für einen Wertungseintrag zu betätigenden Tasten sind identisch mit den Tabellenbezeichnungen. Der Wertungseintrag wird nur ausgeführt, wenn eine vereinbarte Taste betätigt wurde und der entsprechende Tabellenplatz frei ist. Nach dem Wertungseintrag folgt die nächste Runde.

Anzeige: Während des Spiels werden auf dem Bildschirm oben rechts der aktuelle Bonuszähler und in der dritten bzw. vierten Bildschirmzeile die noch freien Rubriken der Tabelle angezeigt.

Tabellenaufruf: Die Tabelle erscheint selbständig nur nach Spielende. Sie kann aber jederzeit bei "Programmstillstand" über LIST aufgerufen werden, so vor oder nach dem Löschen und vor dem Eintragen einer Wertung. Durch beliebige Tastenbetätigung gelangt man anschließend wieder an die alte Stelle des Spieldurchlaufs.

Bonuszähler: Wenn die Summe der Rubriken 1 bis 6 nach Spielende 63 oder mehr beträgt. wird ein Bonus von 50 Punkten gewährt. Um die Übersicht zu erleichtern. zählt der Bonuszähler rückwärts und zeigt an, wieviel Punkte noch zum Erreichen des Bonus benötigt werden.

Weiterer Spielverlauf: Im weiteren Verlauf des Spieles müssen nach und nach alle Tabellenplätze belegt werden. Erhält man einen unpassenden Wurf. so muß man sich trotzdem für einen Tabellenplatz entschei-



den. Dort wird dann allerdings selbständig eine 0 eingetragen.

Spielende: Nach 15 Runden sind alle Tabellenplätze belegt und die Tabelle gelangt zur Anzeige. Durch Betätigung der ENTER-Taste folgt die Ergebnisanzeige.

Neues Spiel: Ein neues Spiel beginnt durch beliebige Tastenbetätigung.

Berhard Piniek

```
5     LET V=0
10    CALL %8DD;
      PRINT "PASCH    B:"
15    PROC SETRR[%F2,%613];
      PROC SETR[%F1,10]
20    LET A=-1,B=A,C=A,D=A;
      LET E=A,F=A,G=A,H=A
25    LET I=A,J=A,K=A,L=A;
      LET M=A,N=A,O=A,U=0
30    GOSUB 700;
      GOSUB 800
35    LET P=0,Q=0,R=0,S=0;
      LET T=0,Z=0
40    LET Z=Z+1;
      IF P=0
      THEN GOSUB 160;
      LET P=W
45    IF Q=0
      THEN GOSUB 160;
      LET Q=W
50    IF R=0
      THEN GOSUB 160;
      LET R=W
55    IF S=0
      THEN GOSUB 160;
      LET S=W
60    IF T=0
      THEN GOSUB 160;
      LET T=W
65    PROC SETR[%5B,%40];
      PRINT "WURF:";
      LET X=Z,Y=%45;
      GOSUB 170;
      LET X=P,Y=%50;
      GOSUB 170
70    LET X=Q,Y=%52;
      GOSUB 170;
      LET X=R,Y=%54;
      GOSUB 170
75    LET X=S,Y=%56;
      GOSUB 170;
      LET X=T,Y=%5B;
      GOSUB 170;
      PRINT
80    IF P=Q
      THEN IF P=R
      THEN IF P=S
      THEN IF P=T
      THEN LET X=4;
      GOSUB 470
85    IF Z>2
      THEN PROC SETR[%6D,0];
      GOTO 180
90    PROC SETR[%5B,%60];
      PRINT "LOESCHEN:"
95    LET X=GTC$A%7F;
      PROC PTC[8]
100   IF X=%2D
      THEN GOSUB 600;
      GOTO 65
105   IF X=%D
      THEN GOTO 40
110   IF X=%7F
      THEN LET Z=3;
      GOTO 180
115   IF X=%31
      THEN LET P=0,X=0,Y=%50;
      GOSUB 170;
      GOTO 90
120   IF X=%32
      THEN LET Q=0,X=0,Y=%52;
      GOSUB 170;
      GOTO 90
125   IF X=%33
      THEN LET R=0,X=0,Y=%54;
      GOSUB 170;
      GOTO 90
130   IF X=%34
      THEN LET S=0,X=0,Y=%56;
      GOSUB 170;
      GOTO 90
135   IF X=%35
      THEN LET T=0,X=0,Y=%5B;
      GOSUB 170;
      GOTO 90
140   IF X<%31
      THEN GOTO 95
145   IF X>%35
      THEN GOTO 95
150   GOTO 180
160   WAIT GETR[%54]$M7+1;
      LET W=GETR[%F2];
      RETURN
170   PROC SETR[%5B,Y];
      PROC PTC[X+48];
      RETURN
180   PROC SETR[%5B,%60];
      PRINT "WERTUNG:";
      LET X=25;
      GOSUB 470
185   LET X=GTC$A%7F;
      PROC PTC[8]
190   IF X=%2D
      THEN GOSUB 600;
      GOTO 65
195   IF X=%21
      THEN IF A<0
      THEN GOTO 275
200   IF X=%22
      THEN IF B<0
      THEN GOTO 280
205   IF X=%23
      THEN IF C<0
      THEN GOTO 285
210   IF X=%24
      THEN IF D<0
      THEN GOTO 290
215   IF X=%25
      THEN IF E<0
      THEN GOTO 295
220   IF X=%26
      THEN IF F<0
      THEN GOTO 300
225   IF X=%55
      THEN IF G<0
      THEN GOTO 305
230   IF X=%4A
      THEN IF H<0
      THEN GOTO 325
235   IF X=%54
      THEN IF I<0
      THEN GOTO 340
240   IF X=%46
      THEN IF J<0
      THEN GOTO 350
245   IF x=%3C
      THEN IF K<0
      THEN GOTO 360
250   IF X=%3E
      THEN IF L<0
      THEN GOTO 370
255   IF X=%5B
      THEN IF M<0
      THEN GOTO 380
260   IF X=%53
      THEN IF N<0
      THEN GOTO 400
265   IF X=%40
      THEN IF O<0
      THEN GOTO 405
270   GOTO 180
275   LET Z=0;
      GOSUB 500;
      LET A=X*Y;
      GOTO 415
280   LET X=2,Z=0;
      GOSUB 510;
      LET B=X*Y;
      GOTO 415
285   LET X=3,Z=0;
      GOSUB 510;
      LET C=X*Y;
      GOTO 415
290   LET X=4,Z=0;
      GOSUB 510;
      LET D=X*Y;
      GOTO 415
295   LET X=5,Z=0;
      GOSUB 510;
      LET E=X*Y;
      GOTO 415
300   LET X=6,Z=0;
      GOSUB 510;
      LET F=X*Y;
      GOTO 415
305   LET Z=2;
      GOSUB 500;
      IF X>6
      THEN LET G=0;
      GOTO 415
310   LET G=X,Z=2,X=X+1;
      GOSUB 510;
      IF X>6
      THEN GOTO 320
315   LET G=X
320   LET G=2*G;
      GOTO 415
325   LET Z=2;
      GOSUB 500;
      IF X>6
      THEN LET H=0;
      GOTO 415
330   LET H=X,Z=2,X=X+1;
      GOSUB 510;
      IF X>6
      THEN LET H=0;
      GOTO 415
335   LET H=2*(H+X);
      GOTO 415
340   LET Z=3;
      GOSUB 500;
      IF X>6
      THEN LET I=0;
      GOTO 415
345   LET I=3*X;
      GOTO 415
350   LET Z=4;
      GOSUB 500;
      IF X>6
      THEN LET J=0;
```

```
     GOTO 415
355  LET J=4*X;
     GOTO 415
360  LET Z=2;
     GOSUB 500;
     LET K=P+Q+R+S+T;
     IF X=7
     THEN IF K=15
     THEN GOTO 415
365  LET K=0;
     GOTO 415
370  LET Z=2;
     GOSUB 500;
     LET L=P+Q+R+S+T;
     IF X=7
     THEN IF L=20
     THEN GOTO 415
375  LET L=0;
     GOTO 415
380  LET Z=3;
     GOSUB 500;
     IF X>6
     THEN LET M=0;
     GOTO 415
385  LET M=X,Z=2;
     GOSUB 500;
     IF M>X
     THEN GOTO 395
390  LET Z=2,X=X+1;
     GOSUB 510;
     IF X>6
     THEN LET M=0;
     GOTO 415
395  LET M=(3*M)+(2*X);
     GOTO 415
400  LET N=P+Q+R+S+T;
     GOTO 415
405  IF P=Q
     THEN IF P=R
     THEN IF P=S
     THEN IF P=T
     THEN LET O=50
410  ELSE ;
     LET O=0
415  LET U=U+1;
     IF U<15
     THEN CALL %8DD;
     PRINT "PASCH     B:";
     GOSUB 700;
     GOSUB 800;
     GOTO 35
420  GOSUB 600;
     CALL %8DD;
     PRINT " ERGEBNISSE   ";
     LET X=1;
     GOSU8 470;
     PROC SETR[%6D,0]
425  LET W=A+B+C+D+E+F;
     IF W>62
     THEN LET W=W+50;
     PRINT "BONUS :     50"
430  ELSE ;
     PRINT "KEINEN BONUS!"
435  LET W=W+G+H+I+J+K+L;
     LET W=W+M+N+O
440  PRINT "SUMME :"W;
     IF W>V
     THEN LET V=W;
     PRINT "NEUER"
445  PRINT "REKORD:"V;
     LET J=GTC;
     GOTO 10
470  LET W=200;
     PROC SETR [%F1,%8A]
480  PROC SETR [%F2,W];
     LET W=W-X;
     IF W>100
     THEN GOTO 480
490  PROC SETR[%F1,10];
     PROC SETR[%F2,6];
     RETURN
500  LET X=1
510  LET Y=0
520  IF X=P
     THEN LET Y=1
530  IF X=Q
     THEN LET Y=Y+1
540  IF X=R
     THEN LET Y=Y+1
550  IF X=S
     THEN LET Y=Y+1
560  IF X=T
     THEN LET Y=Y+1
570  IF Y>=Z
     THEN RETURN
580  LET X=X+1;
     IF X>6
     THEN RETURN
590  GOTO 510
600  CALL %8DD;
     PROC SETR[%6D,0];
     PRINT "TABELLE  B:"
610  PRINT "1:     2:";
     PRINT "3:     4:";
     PRINT "5:     6:"
620  PRINT "E:   Z:   D:";
     PRINT "V:   <:   >:";
     PRIHT "K:   C:   P:";
     GOSUB 700
630  LET X=A,Y=%12;
     GOSUB 960;
     LET X=B,Y=%19;
     GOSUB 960;
     LEI X=C,Y=%22;
     GOSUB 960
640  LET X=D,Y=%29;
     GOSUB 960;
     LET X=E,Y=%32;
     GOSUB 960;
     LET X=F,Y=%39;
     GOSUB 960
650  LET x=G,Y=%42;
     GOSUB 960;
     LET X=H,Y=%47;
     GOSUB 960;
     LET X=I,Y=%4B;
     GOSUB 960
660  LET X=J,Y=%52;
     GOSUB 960;
     LET X=K,Y=%57;
     GOSUB 960;
     LET X=L,Y=%5B;
     GOSUB 960
670  LET X=M,Y=%62;
     GOSUB 960;
     LET X=N,Y=%67;
     GOSUB 960;
     LET X=O,Y=%6B;
     GOSUB 960;
     PRINT
680  LET W=GTC;
     IF U<15
     THEN CALL %8DD;
     PROC SETR[%6D,0];
     PRINT "PASCH   B:";
     GOSUB 700;
     GOSUB 800
690  RETURN
700  LET W=0;
     IF A>0
     THEN LET W=A
710  IF B>0
     THEN LET W=W+B
720  IF C>0
     THEN LET W=W+C
730  IF D>0
     THEN LET W=W+D
740  IF E>0
     THEN LET W=W+E
750  IF F>0
     THEN LET W=W+F
760  LET W=63-W;
     IF W<0
     THEN LET W=0
770  LET X=W,Y=%B;
     GOSUB 960;
     RETURN
800  PRINT " ";
     IF A<0
     THEN PROC PTC[%31]
810  IF B<0
     THEN PROC PTC[%32]
820  IF C<0
     THEN PROC PTC[%33]
830  IF D<0
     THEN PROC PTC[%34]
840  IF E<0
     THEN PROC PTC[%35]
850  IF F<0
     THEN PROC PTC[%36]
860  IF G<0
     THEN PROC PTC[%45]
870  IF H<0
     THEN PROC PTC[%5A]
880  IF I<0
     THEN PROC PTC[%44]
890  IF J<0
     THEN PROC PTC[%56]
900  IF K<0
     THEN PROC PTC[%3C]
910  IF L<0
     THEN PROC PTC[%3E]
920  IF M<0
     THEN PROC PTC[%4B]
930  IF N<0
     IHEN PROC PTC[%43]
940  IF O<0
     THEN PROC PTC[%50]
950  RETURN
960  PROC SETR[%5B,Y];
     IF X/10=0
     THEN PROC PTC[%1B];
     GOTO 980
970  PROC PTC[X/10+48]
980  IF X<0
     THEN PROC PTC[%2D];
     RETURN
990  PROC PTC[X$M10+48];
     RETURN
```

Software

Einmaleins

Dieses Programm nutzt drei Unterprogramme. Ab Anweisung 500 wird der Inhalt der Variablen H dezimal ohne führende Nullen (Bereich 0 bis 199) auf dem Bildschirm angezeigt. Die Zeile 550 ergänzt diese Ausgabe von H mit einem Gleichheitszeichen und fordert eine Zahleneingabe in A. Das dritte Unterprogramm ab Anweisung 600 dient der akustischen Ausgabe. Es nutzt die Variablen I (Anzahl der Wiederholungen) und J (Anfangstonhöhe) zum Festlegen verschiedener Tonfolgen durch das Hauptprogramm. K bestimmt den jeweiligen Zählumfang des Timers T1 und damit die Tonhöhe.
Nach Eingabe des Grenzwertes (Schwierigkeitsgrad, sinnvoll von 10 bis 199) erhalten A und B Zufallswerte bis zu dieser Grenze sowie G bis 4. Danach werden die Operanden sortiert, so daß B die größere und C die kleine Zahl erhält. Mit G verzweigt das Programm zum Addieren (G=1), Subtrahieren (G=2), Multiplizieren (G=3) oder Dividieren (G=4). Dem Stellen der Aufgaben in B und C folgen das Einschätzen jeder Antwort und das Benoten von jeweils zehn Ergebnissen. Das Programm paßt in 1 KByte RAM.

```
10    CALL %8DD;
      PRINT "EIN MAL EINS";
      INPUT "BIS ZUR "D;
      LET E=0,F=0
15    IF D>199
      THEN PRINT "VIEL ZU
      HOCH";
      WAIT 300;
      GOTO 10
20    PROC SETRR[%F2,255];
      PROC SETR[%F1,10]
30    WAIT A$M17+1
40    LET A=GETR[%54]$MD+1;
      LET B=GETR[%F2]
50    LET G=B$M4+1,B=B$MD+1
60    IF A<B
      THEN LET C=A
70    ELSE ;
      LET C=B,B=A
80    PROC SETR[%5B,%30];
      PRINT "WIEVIEL IST"
90    GOTO G*100
100   LET B=B-C,H=B;
      GOSUB 500;
      PROC PTC[%2B];
      LET H=C;
      GOSUB 550
110   IF A=B+C
      THEN GOTO 150
120   PRINT "# FALSCH ! #"
      LET J=3,I=10;
      GOSUB 600;
      GOTO 160
150   PRINT "* RICHTIG ! *";
      LET E=E+1,J=30,I=1;
      GOSUB 600
160   LET F=F+1;
      IF F<10
      THEN PROC SETR[%5B,%40];
      PRINT " ";
      PRINT " ";
      GOTO 30
170   PRINT "   NOTE:";
      PROC SETR[%5B,%69];
      LET H=6-(E/2);
      GOSUB 500
180   LET J=10,I=E+1;
      G0SUB 600;
      GOTO 10
200   LET H=B;
      GOSUB 500;
      PROC PTC[%2D];
      LET H=C;
      G0SUB 550
210   IF A=B-C
      THEN GOTO 150
220   GOTO 120
300   LET C=C$M10$O4;
      LET B=B/C,H=B;
      GOSUB 500;
      PROC PTC[%2A];
      LET H=C;
      GOSUB 550
310   IF A=B*C
      THEN GOTO 150
320   GOTO 120
400   LET C=C$M10$O4;
      LET B=B/C*C,H=B;
      GOSUB 500;
      PROC PTC[%2F];
      LET H=C;
      GOSUB 550
410   IF B/C=A
      THEN GOTO 150
420   GOTO 120
500   IF H>99
      THEN PROC PTC[%31];
      LET H=H-100;
      GOTO 520
510   IF H/10=0
      THEN GOTO 530
520   PROC PTC[H/10+48]
530   PROC PTC[H$M10+48];
      RETURN
550   GOSUB 500;
      PROC PTC[%3D];
      INPUT A;
      RETURN
600   PROC SETR[%6D,0];
      LET K=J
610   PROC SETR[%F2,K];
      PROC SETR [%F1,%8A];
      LET K=K-1;
      IF K>=0
      THEN GOTO 610
620   LET I=I-1;
      IF I>0
      THEN GOTO 600
630   PROC SETR[%F1,10];
      WAIT GETR[%54];
      RETURN
```

Kleines Einmaleins

Das Programm von R. Schittko aus Merseburg stellt Multiplikationsaufgaben. Die obere Grenze der möglichen Faktoren bestimmt die erste Eingabe (Zeile 40). Nach Anzeige einer Aufgabe mit zufälligen Operanden wird das Ergebnis erwartet und bewertet. Jeweils zehn Antworten wertet das Programm aus. Ein neues Spiel wird mit RUN gestartet. Das kleine Einmaleins erfordert einen RAM-Bereich von mehr als 1/4 KByte. Viel Spaß beim Multiplizieren!

```
10    CALL %8DD
20    PRINT "KLEINES 1*1"
30    PROC SETRR[%F2,255];
      PROC SETR[%F1,10]
40    PRINT " ";
      INPUT "GRENZE (1-99)"G;
      LET Z=0,F=0
50    GOSUB 500;
      LET B=A
60    GOSUB 500
70    PRINT A,"*"B,"=?";
      PROC SETR[%5B,
      GETR[%5B]-15]
80    INPUT E;
      IF A*B=E
      THEN PRINT "* RICHTIG *"
90    ELSE ;
      LET F=F+1;
      IF F<4
      THEN PRINT "ETWAS MEHR";
      PRINT "KONZENTRATION"
100   ELSE ;
```

```
     PRINT "NUN NIMM DICH";
     PRINT "ZUSAMMEN !"
110  LET Z=Z+1;
     IF Z<10
     THEN GOTO 50
120  WAIT 200;
     PRINT "VON 10";
     PRINT "AUFGABEN";
     PRINT "WAREN"10-F;
     PRINT "RICHTIG."
130  IF F<2
     THEN PRINT "* PRIMA *"
140  ELSE
     PRINT "MEHR UEBEN !"
150  PRINT " ";
     END
500  WAIT G$M100+1;
     LET A=GETR[%F2]$MG+1;
     RETURN
```

Master Mind

Unser Logikspiel basiert auf einer Programmidee von H. Dönelt aus Kamenz. Es ermittelt mit dem Timer T1 als Zufallszahlengenerator eine vierstellige Zahl mit den Ziffern 1 bis 6. Dazu wird der Zufallswert dreimal durch 6 geteilt, wobei der Rest jeweils eine zu suchende Ziffer bestimmt. Die Anweisung 70 erzeugt die vierte Stelle aus einem wiederholten Zugriff auf den aktuellen Zählerstand des Timers T1. Dieses Verfahren läßt die gleiche Ziffer auch auf mehreren Stellen zu. Die Zeile 100 fordert die Eingabe eines Tips in die Variablen E bis H. Die Zähler für das Bewerten der Richtigkeit (I und S) werden hier gelöscht. Bei der Suche nach Übereinstimmung nutzt das Programm die Möglichkeit, die Variablen als Register anzusprechen, wobei wegen des begrenzten Bereichs nur die niederen Bytes eine Rolle spielen (A: Register %21, B: %23,..., H: %2F). Die Variablen J und K dienen der Adressierung dieser Register. Bei Feststellen einer Übereinstimmung werden der betreffende Zähler erhöht sowie im Unterprogramm 400 ein kurzer Piepton erzeugt und beide beteiligten Variablen verändert, damit sie nicht wiederholt in Rechnung gehen. Die Zeile 190 stellt anschließend den ursprünglichen Inhalt der Variablen A bis D wieder her.
Die Anweisung 210 quittiert jede auf der richtigen Position geratene Ziffer mit einem Sternchen (%2A), die 230 jeden weiteren Treffer auf falscher Position mit einem O (%4F). Bei vier Volltreffern verzweigt die Zeile 240 zur lobenden Schlußanzeige. Nach acht Fehlversuchen erscheinen statt dessen die zu ratende Zahl und eine tadelnde Bemerkung. Ein neues Spiel beginnt mit einer beliebigen Tastenbetätigung. Das Programm benötigt 1 KByte.

```
10   PROC SETRR[%F2,7];
     PROC SETR[%F1,10]
20   CALL %8DD;
     PRINT " MASTER MIND";
     WAIT 100
30   LET I=GETR[%F2],T=1
40   LET A=I$M6+1,I=I/6
50   LET B=I$M6+1,I=I/6
60   LET C=I$M6+1
70   LET D=GETR[%F2]$M6+1
80   PRINT "4 MAL 1 AUS 6"
90   PROC PTC[%32];
     PROC PTC[T+%30];
     PROC PTC[%3A]
100  LET E=GTC$A7,
     F=GTC$A7,I=0;
     LET G=GTC$A7,
     H=GTC$A7,S=0;
     PROC PTC[32]
110  LET J=%21,K=%29
120  IF GETR[J]=GETR[K]
     THEN LET I=I+1;
     GOSUB 400
130  LET J=J+2,K=K+2;
     IF J<%28
     THEN GOTO 120
140  LET J=%21
150  LET K=%29
160  IF GETR[J]=GETR[K]
     THEN LET S=S+1;
     GOSUB 400
170  LET K=K+2;
     IF K<%30
     THEN GOTO 160
180  LET J=J+2;
     IF J<%28
     THEN GOTO 150
190  LET J=I,A=A$A7,B=B$A7;
     LET C=C$A7,D=D$A7
200  IF I=0
     THEN GOTO 220
210  PROC PTC[%2A];
     LET I=I-1;
     GOTO 200
220  IF S=0
     THEN GOTO 240
230  PROC PTC[%4F];
     LET S=S-1;
     GOTO 220
240  PRINT ;
     LET T=T+1;
     IF J=4
     THEN GOTO 300
250  IF T<9
     THEN GOTO 90
260  LET K=%21
270  PROC PTC[GETR[K]+%30];
     LET K=K+2;
     IF K<%28
     THEN GOTO 270
280  PRINT " DU NASE!";
     PROC SETR[%F3,31]
290  GOTO 310
300  PRINT "# RICHTIG ! #";
     PROC SETR[%F3,15]
310  PROC SETR[[%F1,%8A];
     LET E=64
320  LET E=E+4;
     PROC SETR[%F2,E];
     IF E<256
     THEN GOTO 320
330  PROC SETR[%F1,10];
     LET E=GTC;
     GOTO 10
400  PROC SETR[%F1,%8A];
     PROC SETR[J,GETR[J]+16];
     PROC SETR[K,8];
     PROC SETR[%F1,10];
     RETURN
```

Römische Zahlen

Dieses Programm bietet das Umrechnen von natürlichen in römische Zahlen und umgekehrt an. Nach Wahl 1 testet es die möglichen Werte aus und bringt die zugehörigen Zeichen ggf. mit der Prozedur PTC auf den Bildschirm. Umgekehrt werden die eingegebenen römischen Ziffern ab Zeile 200 in die entsprechenden Dezimalwerte gewandelt (Variable C) und in A aufaddiert. Die Anweisung 330 berücksichtigt das Subtrahieren vorangestellter niederwertiger Ziffern mit Hilfe der Variablen D, die den

jeweils letzten Eingabewert speichert. Ab Zeile 400 erzeugt das Programm eine Schlußmelodie, bevor es seine Dienste erneut anbietet.

```
10    CALL %8DD;
      PRINT "ROEMISCHE";
      PRINT "ZAHLEN";
      PRINT "1:N->R";
      PRINT "2:R->N";
      INPUT "WAHL:"W
20    IF W=2
      THEN GOTO 200
30    IF W<>l
      THEN PRINT "?";
      WAIT 100;
      GOTO 10
40    PRINT "NATUERLICHE";
      INPUT "ZAHL:"A;
      PRINT "ROEMISCH:"
50    IF A>999
      THEN PROC PTC[%4D];
      LET A=A-1000;
      GOTO 50
60    IF A>899
      THEN PROC PTC[%43];
      PROC PTC[%4D];
      LET A=A-900
70    IF A>499
      THEN PROC PTC[%44];
      LET A=A-500
80    IF A>399
      THEN PROC PTC[%43];
      PROC PTC[%44];
      LET A=A-400
90    IF A>99
      THEN PROC PTC[%43];
      LET A=A-100;
      GOTO 90
100   IF A>89
      THEN PROC PTC[%58];
      PROC PTC[%43];
      LET A=A-90
110   IF A>49
      THEN PROC PTC[%4C];
      LET A=A-50
120   IF A>39
      THEN PROC PTC[%58];
      PROC PTC[%4C];
      LET A=A-40
130   IF A>9
      THEN PROC PTC[%58];
      LET A=A-10;
      GOTO 130
140   IF A>8
      THEN PROC PTC[%49]
      PROC PTC[%58];
      LET A=A-9
150 IF A>4
      THEN PROC PTC[%56];
      LET A=A-5
160 IF A>3
      THEN PROC PTC[%49];
      PROC PTC[%56];
      LET A=A-4
170   IF A>0
      THEN PROC PTC[%49];
      LET A=A-1;
      GOTO 170
180   GOTO 400
200   LET A=0,D=0,C=0
210   PRINT "ROEMISCHE";
      PRINT "ZAHL:"
220   LET B=GTC$A%7F;
      LET C=0
230   IF B=%4D
      THEN LET C=1000
240   IF B=%44
      THEN LET C=500
250   IF B=%43
      THEN LET C=100
260   IFB8=%4C
      THEN LET C=50
270   IF B=%58
      THEN LET C=10
280   IF B=%56
      THEN LET C=5
290   IF B=%49
      THEN LET C=1
300   IF B=%0D
      THEN GOTO 350
310   IF C=0
      THEN PRINT ;
      PRINT "FALSCHE EIN-";
      PRINT "GABE,";
      PRINT "NOCH EINMAL!";
      GOTO 200
320   LET A=A+C
330   IF C>D
      THEN LET A=A-D-D
340   LET D=C;
      GOTO 220
350   PRINT "NATUERLICHE";
      PRINT "ZAHL:"A
400   PROC SETR[%6D,0];
      PROC SETRR[%F2,3];
      PROC SETR[%F1,%8A];
      LET A=256
410   PROC SETR[%F2,A];
      LET A=A-1;
      IF A>0
      THEN GOTO 410
420   PROC SETR[%F1,10];
      GOTO 10
```

Dr. Helmut Hoyer

Speichern von Maschinenprogrammen

Werden im JU+TE-Computer Maschinen(unter-)programme benötigt und ist keine RAM-Stütze vorhanden, so ist es recht mühsam und zeitaufwendig, nach jedem Ausschalten des Computers das Programm "HEX-EINGABE" einzuladen und den Maschinencode einzutippen. Die Prozedur „PROC" erlaubt in Verbindung mit „SETEW" das Beschreiben von Doppelspeicherzellen, was sich für diese Zwecke recht gut nutzen läßt. Anhand der Grafik-Routine aus JU+TE 3/88, S. 232 soll das Prinzip erläutert werden:

```
200   CALL %8DD
210   PRINT "GRAFIK";
      PRINT "JU+TE 3/88";
      PRINT "S.232"
220   LET A=%FCA0
300   LET B=%70FD; GOSUB 800
310   LET B=%3170; GOSUB 800
...
770   LET B=%AF00; GOSUB 800
780   END
800   PROC SETEW[A,B]
810   LET A=A+2
820   RETURN
```

Nach "RUN" wird der Programmname angezeigt. In Zeile 220 weist man der Variablen A die Anfangsadresse des Maschinenprogramms zu. Von Zeile 300 bis Zeile 770 werden die Variablen B jeweils Doppelbytes zugeordnet und mit Hilfe des Unterprogramms ab Zeile 800 in die Speicherzellen eingeschrieben. Während des Programmlaufs ist ein Ton im Lautsprecher hörbar, danach meldet sich der Computer mit "END 780". Jetzt kann das BASIC Programm mit "NEW" wieder gelöscht werden.
Die Länge des Programms hängt vor allem von der Anzahl der Bytes des Maschinenprogramms ab, ist aber in jedem Fall wesentlich umfangreicher als dieses. Bei JU+TE-Computern mit ausreichend großer Speicherkapazität ist es auch möglich, das Programm gemeinsam mit dem entsprechen den Hauptprogramm (z. B. "MALFIX") abzuspeichern. Dabei wird vor dem Hauptprogramm eine GOTO-Anweisung zur Anfangszeile des Hilfsprogramms eingefügt (GOTO 200)

Die Zeile 780 ersetzen wir durch eine GOTO Anweisung zum Start des Hauptprogramms (meist GOTO 10). Nach dem ersten Durchlauf können wir dann die erste Zeile löschen, damit bei einem eventuellen Neustart nicht noch einmal das ohnehin schon im Speicher stehende Maschinenprogramm eingeschrieben wird.

Ingolf Haß

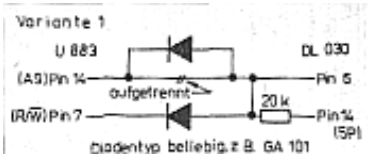
Pins 1 bis 5 und 8 bis 13 des DL 008 biegen wir ab. Pin 5 (DL 008) wird mit Pin 14 (U 883) und Pin 4 (DL 008) mit Pin 7 (U 883) verbunden.
Jetzt können auf dem Bild nur noch Streifen auftreten, wenn ein Programm aus dem Grafikspeicher liest. Das Betriebssystem macht dies nur beim Bildschirmrollen.

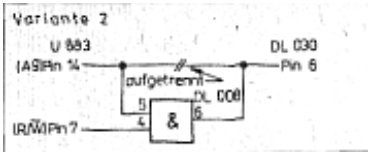
Harun Scheutzow

Hardware

Störende Streifen

Diese kleine Schaltung soll die weißen Streifen auf dem Bildschirm unter der letzten Zeile des Bildes, also nur einen Schönheitsfehler, beseitigen und ist nicht notwendig für die Funktion des COMP JU+TE R. Nur eine kurze Erklärung der Funktionsweise: Die Streifen entstehen durch ständige Schreibzugriffe auf den Grafikspeicher zur Darstellung des flackernden Kursors. Die Bildausgabe erfolgt aber nur mit Lesezugriffen. Die Schaltung sorgt dafür, daß wirklich nur bei Lesezugriffen etwas auf den Bildschirm kommt.
Es werden zwei Varianten der Schaltung dargestellt, die erste mit zwei Dioden und einem Widerstand, die zweite mit einem Schaltkreis DL 008 oder PL 008. Auf jeden Fall ist der Leiterzug (auf der Platinenoberseite) von Pin 14 des U 883 zu Pin 6 des DL 030 aufzutrennen und dann die Schaltung 1 oder 2 einzubauen, siehe Schaltplan. Bei Schaltung 2 wird der DL 008 direkt auf den DL 030 aufgelötet, aber nur mit den Pins 6, 7 und 14. Die





Leserbriefseiten

(Ausschnitte der Leserbriefseiten, den JU+TE-Computer betreffend.)

JU+TE 3/89, S. 163

Achtung! Leiterplatten für den JU+TE-Computer

Für die vielen Freunde des JU+TE-Selbstbau-Computers, vor altem aber auch für diejenigen, die es erst noch werden möchten, heute eine erfreuliche Mitteilung:

Ab sofort können begrenzte Stückzahlen amateurgerechter, ungebohrter Leiterplatten zum JU+TE-Computer bestellt werden bei:

Gerlich
Markscheiderweg 08/417
Neubrandenburg, 2000.

Die Leiterplatten werden per Nachnahme zugeschickt. Bei der anhaltend großen Nachfrage bitten wir Euch aber, etwas Wartezeit einzuplanen. Bestellungen, die bisher bei JU+TE erfolgten, haben wir ebenfalls schon weitergereicht. Bitte also, keine Bestellungen mehr an JU+TE senden!
Wie wir vorn Hersteller erfuhren, können folgende Leiterplatten bestellt werden:

- Rechnerplatine (Abb. 6/7 in JU+ TE 8/87 S. 635 und 9/87 S. 696)
- Speicherplatine U 2716 (Abb. 10/11 in 9/87 S. 699)
- Magnetbandinterface 2K (Abb. 19 in 12/87 S. 932)
- EPROM-Programmierzusatz (in 10/88 S. 786/787)
- Magnetbandinterface 4K (in 3/89 S. 231)
- vereinfachte Speicherplatine U 2716 für 4K (in 3/89 S. 232)

Weitere Anfragen betreffs Leiter platten bitte an obige Adresse richten.

JU+TE-Computer-Sprechstunde

Am 20. April 1989 bieten wir Euch die Möglichkeit, in der Zeit von 15 Uhr bis 17.30 Uhr in unserer Redaktion unter Berlin 2233 432 oder 433 anzurufen. Eure Fragen zum JU+TE-Computer wird unser Autor Dr. Helmut Hoyer direkt am Telefon beantworten. Darüber hinaus wären wir Euch natürlich für Hinweise zum JU+TE-Computerklub dankbar. Die nächste JU+TE-Computersprechstunde mit EPROM-Programmiermöglichkeiten bereiten wir für das Pfingsttreffen der FDJ im Mai vor. Näheres im April-Heft.

JU+TE 4/89

JU+TE-Computer zum Pfingsttreffen der FDJ

Während des Pfingsttreffens der FDJ in Berlin bieten wir Euch an drei Tagen im Palast der Republik im 1. Geschoß (Platzseite) neben dem Mitmachen beim JU+TE/nl-Verkehrspreisausschreiben einen kleinen Service zum JU+TE-Selbstbeucomputer:

- am 12. Mai 1989 von 9 bis 18 Uhr,
- am 13. Mai 1989 von 10 bis 18 Uhr und
- am 14. Mai 1989 von 9.30 bis 18 Uhr.

Wir haben folgende Angebote:

- Brennen Eurer mitgebrachten EPROM U2716 C mit dem 4K-Betriebssystem;
- Überspielen von Software auf Eure Kassetten;
- der Autor des Computers, Dr. Hoyer, stellt sich Euren Fragen;
- Spiele;
- Weiterleiten von Leiterplattenwünschen;
- Vermitteln von Adressen für die Bauelementebeschaffung;
- Verkauf von JU+TE-Heften, darunter Heft 3/89, das die Beschreibung des erweiterten Betriebssystems enthält.

Natürlich erwarten wir auch Eure Hinweise und Ratschläge zum weiteren Ausbau des JU+TE-Computers. Also wir erwarten Euch zu Pfingsten!

JU+TE 6/89

+++ JU+TE-Computer-Information ++++++++++++

Heißgelaufen!

Für den 20. April 1989 hatten wir zu einer JU+TE-Computer-Sprechstunde per Telefon eingeladen. Es war ein Versuch. Doch nun sind wir in die Pflicht genommen: Fast 100 Computerfreunde aus allen Himmelsrichtungen den Republik stellten in den zwei Stunden ihre Fragen, und jeder hatte ein Paket davon auf Lager. Da liefen die Telefonleitungen wirklich heiß. Aber es hat uns Spaß gemacht.

JU+TE Computerclub

Neue JU+TE-Computer-Sprechstunde

Für alle Interessenten, vor allem auch diejenigen, die am 20. April telefonisch kein Glück hatten, führen wir die Sprechstunde mit dem Autoren des JU+TE-Computers, Dr. Helmut Hoyer, am 6. Juli 1989 in der Zeit von 15 Uhr bis 17.30 Uhr durch. Ihr erreicht uns unter Berlin 2233432 oder 22334 33.

Problem Bauanleitung

Ein großes Problem für viele unserer Leser ist offensichtlich das Beschaffen der Bauanleitung, die wir ab JU+TE-Heft 7/1987 veröffentlichten. Wir haben deshalb eine kleine Broschüre vorbereitet, die voraussichtlich Ende des Jahres im Buchhandel erhältlich sein wird. Sie beinhaltet die komplette Bauanleitung vom Aufbau des Grundgerätes bis zum Entwicklungssystem und natürlich auch Softwarebeispiele. Wenn der Auslieferungstermin festliegt, werden wir das rechtzeitig in JU+TE ankündigen.

Leiterplatten-Service

Nach wie vor können Leiterplatten für den JU+TE-Computer nur bei folgender Adresse bestellt werden (vgl. JU+TE 3/1989 Seite 163): Gerlich, Markscheiderweg 08/417, Neubrandenburg, 2000. Die Zusendung erfolgt per Nachnahme. Wir bitten Euch, keine Bestellungen mehr an JU+TE zu senden. Aber Achtung: Nicht doppelt bestellen! Bereits bei uns eingegangene Bestellungen haben wir weitergeleitet.