P.R. Boldt

EPROM-Programmiergerät

Programme für SC/MP und Junior selbst "schießen"

Hier wird ein EPROMer, besser gesagt: ein EPROM-Programmiergerät, für die Elektor-Hauscomputer SC/MP und Junior vorgestellt. Die Abmessungen sind sehr "bescheiden": Die gesamte Elektronik hat auf einer Euro-Karte Platz. Trotzdem kann man mit diesem Gerät sowohl 2716- als auch 2732-EPROMs programmieren. Die EPROMs können nicht nur programmiert, sondern deren Inhalt kann gleichzeitig auch ausgelesen werden.

Vielleicht tun wir hier etwas zu viel des Guten. Nach der Veröffentlichung eines 2716-Programmiergeräts im Oktober-Heft 1981 schon wieder ein EPROMer? Nun — beide Schaltungen unterscheiden sich doch erheblich voneinander. Die erstgenannte Schaltung war insbesondere auch für "Elektor-fremde" Prozessoren geeignet, während der neue "Programmierer" speziell für die SC/MP- und 6502-Systeme entwickelt wurde. Mit diesem Gerät kann man sowohl 2716- als auch 2732-EPROMs programmieren und auslesen. Die Euro-Karte, auf der die Schaltung aufgebaut ist, enthält eine Messerleiste nach DIN, die genau auf den SC/MP-Bus und die Erweiterungs-Federleiste des Junior-Computers paßt. Es gibt also gute Gründe, allen Elektor-Computer-Besitzern die Schaltung so schnell wie möglich vorzustellen — denken wir.

Wie man solch ein 2716- (und auch 2732-)EPROM programmieren kann, wurde ausführlich im Oktober-Heft 1981 beschrieben. Dort wurde auch erwähnt, daß gerade der 2716-Typ sehr preiswert zu kaufen und darüber hinaus leicht zu programmieren ist. Wir möchten deshalb nur kurz wiederholen, wie die Programmierung funktioniert. An Pin 21 des 2716 (Pin 20 des 2732) legt man eine Programmierspannung von 25 V. Ein Programmier-Impuls von 50 ms Länge am $\overline{CE}$-Eingang reicht dann aus, um die an den Eingängen liegenden Daten in die entsprechende Adresse zu laden.

Steht bereits ein Computer zur Verfügung (Wofür sonst möchte man schließlich EPROMs programmieren?), dann liegt es natürlich nahe, den Computer auch zum Programmieren von EPROMs einzusetzen. Es gibt eigentlich nur einen Aspekt, der dagegen spricht: Den Junior-Computer kann man nicht in einem Schreibzyklus stoppen. Eine Adresse mit den darin enthaltenen Daten muß aber mindestens 50 ms an EPROM anliegen. Beim SC/MP ist das kein Problem! Für den Junior muß man deshalb einige Puffer vorsehen, in denen die Information zwischengespeichert werden kann. Die Platine enthält aus diesem Grunde die notwendigen Puffer. Bei Betrieb mit dem SC/MP-Prozessor kann man sie einfach mittels einiger Drahtstückchen überbrücken.

Die Schaltung

Bevor wir uns an die Programmierung begeben, ist es zunächst notwendig, die Funktionsweise des Gerätes zu beschreiben. Die Schaltung ist in Bild 1 dargestellt. Wie schon erwähnt, sind einige Puffer nötig, in denen die Daten und Adressen einige Zeit lang gespeichert werden können. Die Puffer sind in den IC1 . . . IC4 enthalten. Abhängig vom zu programmierenden EPROM-Typ müssen 11 (für 2716) und 12 (für 2732) Adreßbits in den Puffern zwischengespeichert werden. Deshalb wird die Leitung für das 12. Adreßbit über Schalter S1 wahlweise mit dem EPROM verbunden.

Die Daten-Leitungen DØ . . . D7 sind an zwei parallel geschaltete Zwischenspeicher ("Latch") angeschlossen (IC1 und IC2). Die Eingänge von IC1 sind mit den Ausgängen von IC2, und die Ausgänge von IC1 sind mit den Eingängen von IC2 verbunden. Auf diese Weise ist es möglich, die EPROM-Daten auch wieder über den Computer auszulesen. Außerdem sind noch einige Verbindungen zum Computer notwendig, die alle zum linken Rand von Bild 1 geführt sind.

Die erforderliche Adressendekodierung übernimmt der 4-bit-Vergleicher IC5. Auf der einen Seite ist dieses IC mit den Adressenleitungen A12 . . . A15 verbunden. Auf der anderen Seite kann man den Adressenbereich mit Hilfe der Schalter S3 . . . S6 einstellen. Ein geschlossener Schalter bildet eine logische Null. Die Schalter sind den Adressenleitungen folgendermaßen zugeordnet: S3 → A12, S4 → A13, S5 → A14 und S6 → A15. Ein Adressenbereich umfaßt immer 4 K. Das scheint auf den ersten Blick sehr viel zu sein. Man muß aber bedenken, daß auch ein 4-K-EPROM (2732) programmiert werden kann. Bei der Programmierung eines 2716-EPROMs kann aus diesem Grunde sowohl der Bereich ab XØØØ als auch der ab X8ØØ adressiert werden.

IC6 ist ein astabiler Multivibrator mit Start/Stopp-Möglichkeit. Die Dimensionierung der frequenzbestimmenden Bauteile ist so ausgelegt, daß die Periodendauer 10 ms beträgt. Über einen Inverter ist der Ausgang des Multivibrators mit dem Schieberegister IC7 verbunden. Der Reset-Eingang von IC6 liegt am Q-Ausgang von FF2. Der Takt-Eingang dieses Flipflops, und auch der von FF1, erhält über N6 Impulse von Pin 36a der Messerleiste. Das bedeutet: Die Flipflops erhalten dann Taktimpulse, wenn der Prozessor einen Write-Befehl gibt.

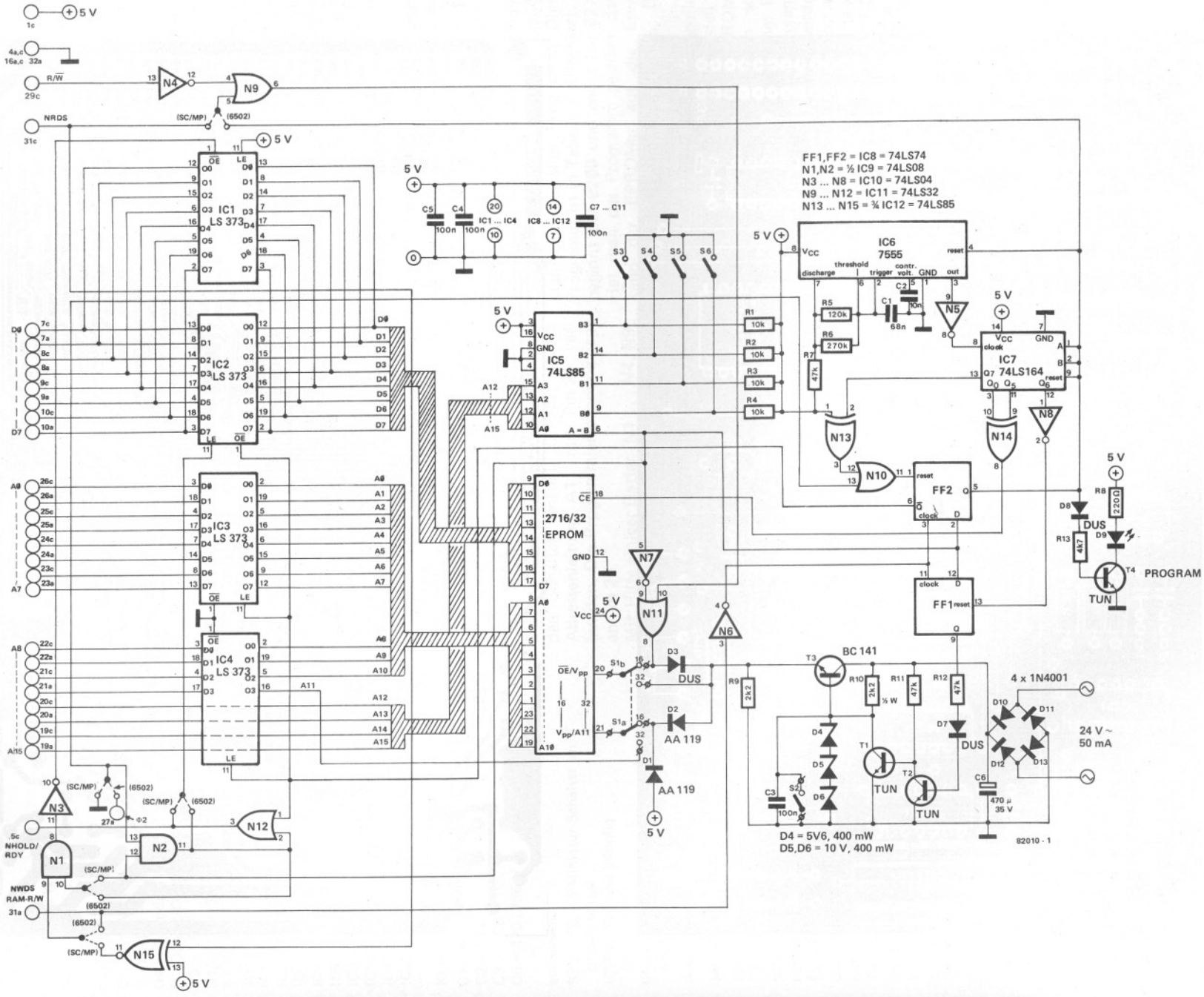
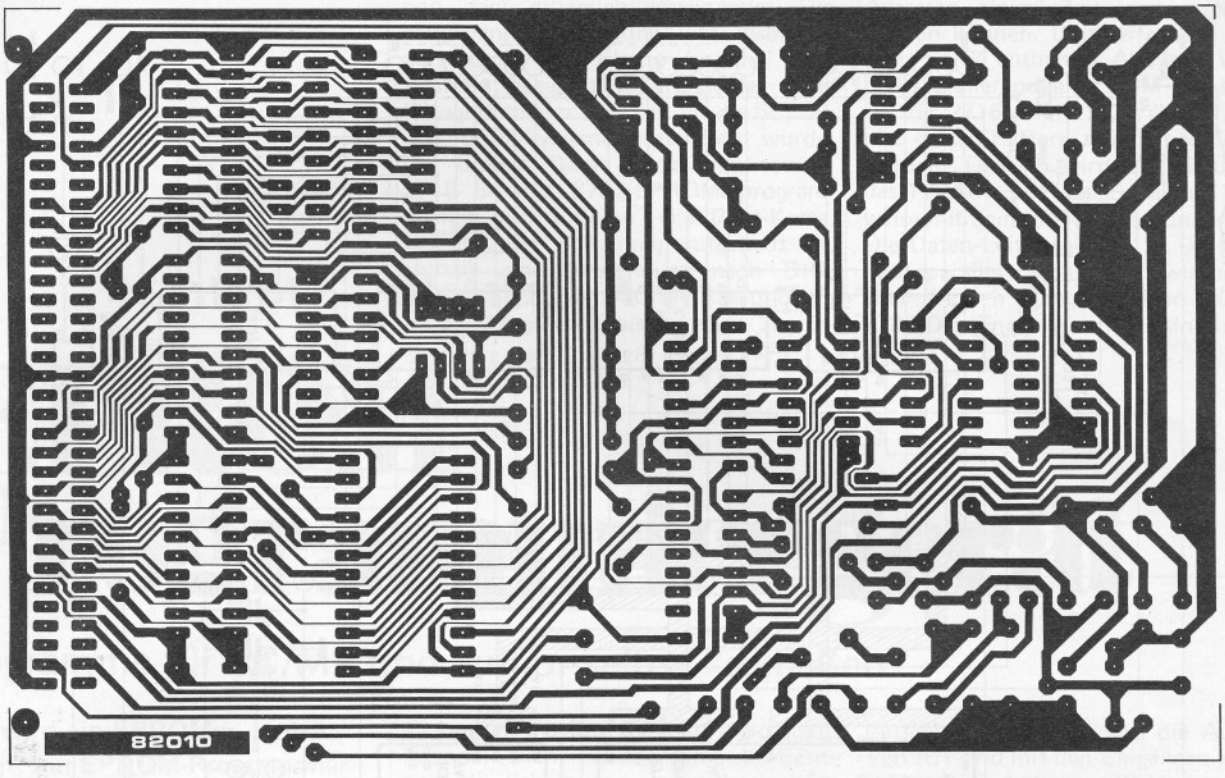


Bild 1. Die Schaltung des EPROMers. In der Mitte des Bildes erkennt man das EPROM, darüber den Adreßdekoder. Links davon sind die Puffer für Daten und Adressen dargestellt. Ganz rechts befindet sich die Logik zur Erzeugung der unterschiedlichen Steuerimpulse zur Programmierung. Rechts unten findet man das Netzteil zur Erzeugung der 25-V-Programmierspannung.

2



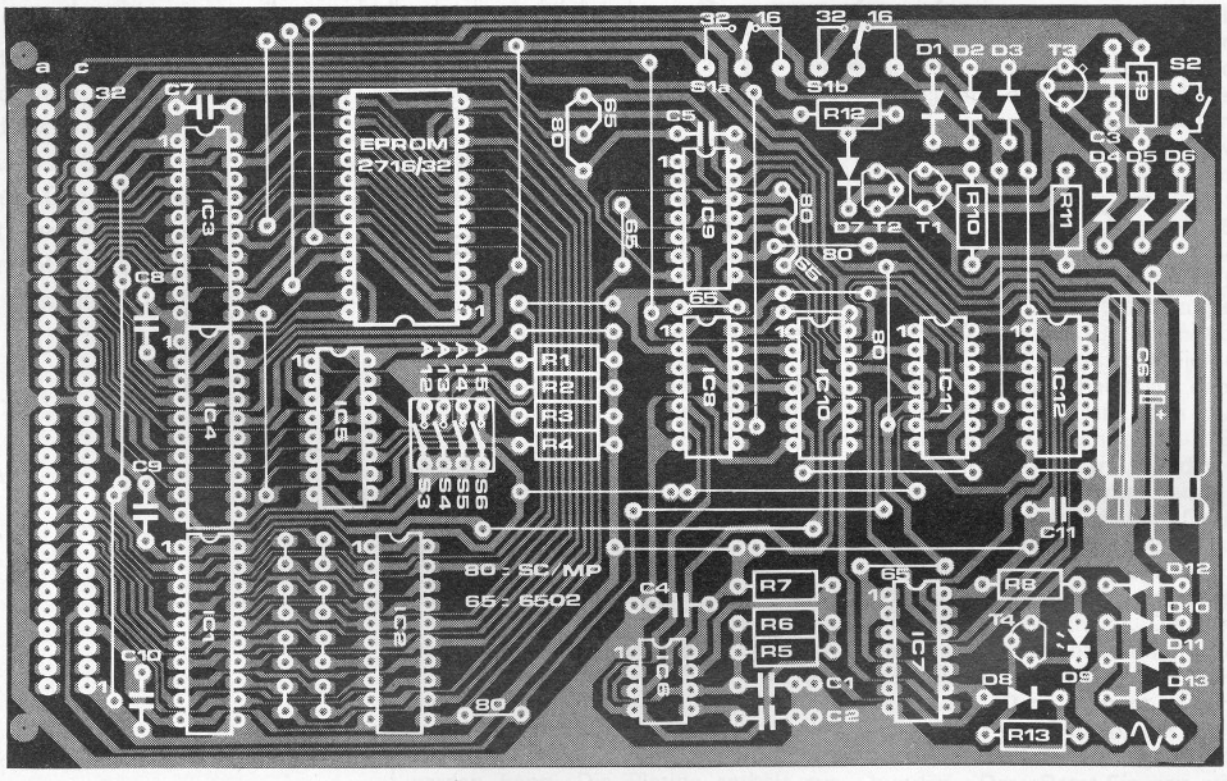


Bild 2. Layout und Bestückungsplan der Platine für den EPROMer. Die Fassung für das EPROM sollte vom Null-Kraft-Typ sein.

Stückliste

Widerstände:
R1 . . . R4 = 10 k
R5,R6 = 220 k
R7,R11,R12 = 47 k
R8 = 220 Ω
R9 = 2k2
R10 = 2k2/½ W
R13 = 4k7

Halbleiter:
D1,D2 = AA 119
D3,D7,D8 = DUS
D4 = Z-Diode 5V6/0,4 W
D5,D6 = Z-Diode 10 V/0,4 W
D9 = LED
D10 . . . D13 = 1N4001
T1,T2,T4 = TUN
T3 = BC 141
IC1 . . . IC4 = 74LS373
IC5 = 74LS85
IC6 = 7555
IC7 = 74LS164
IC8 = 74LS74
IC9 = 74LS08
IC10 = 74LS04
IC11 = 74LS32
IC12 = 74LS86

Kondensatoren:
C1 = 68 n
C2 = 10 n
C3 . . . C5 = 100 n
C6 = 470 µ/35 V

außerdem:
S3 . . . S6 = 4 Miniatur-Schalter im DIL-Gehäuse
IC-Test-Fassung 24polig ("Null Kraft")
64polige Messerleiste mit abgebogenen Anschlüssen nach DIN 41612

In dem Moment, wo der Adressenbereich gefunden ist, wird der Ausgang "A = B" von IC5 logisch 1. Dieser Impuls wird an den Q-Ausgang von FF2 durchgeschaltet, und damit startet der Multivibrator IC6. Dies geschieht unter der Voraussetzung, daß die Flipflops einen Write-Befehl erhalten haben. Gleichzeitig mit dem Umschalten von FF2 kippt FF1, so daß eine Spannung von 25 V an den "Programmierpin" des EPROMs gelegt wird.
Zur Erzeugung der Programmierspannung von 25 V wurde ein kleines Netzteil entwickelt, das im wesentlichen aus den Transistoren T1 . . . T3 und den Dioden D4 . . . D6 besteht. Ein Brückengleichrichter mit Siebelko ist ebenfalls vorhanden, so daß man nur noch eine Wechselspannung von 24 V anschließen muß. Der Q-Ausgang von FF1 schaltet die 25-V-Spannung ein und aus. Mit Schalter S2 kann man zwischen "Programmieren/Lesen" und "Lesen" umschalten. Ist der Schalter geschlossen, dann wird auch die Programmierspannung zu Null (Betriebsart "Lesen"). In der anderen Schalterstellung ist die Betriebsart "Programmieren/Lesen" gewählt.
Doch nun wieder zurück zur Funktion des EPROMers: Die Programmierspannung ist eingeschaltet, und der Multivibrator IC6 "läuft". Im Rhythmus der Impulse von IC6 wird IC7 nun mit Einsen vollgeschrieben. 10 ms nachdem die 25-V-Spannung eingeschaltet wurde, wird Ausgang Q0 logisch 1. Dadurch erhält auch der Eingang $\overline{CE}$ des EPROMs eine "1". Durch die Verknüpfung der Ausgänge Q0 und Q5 mittels N14 liegt der "1"-Pegel 50 ms lang am $\overline{CE}$-Eingang an. 10 ms später wird Ausgang Q6 "1", und FF1 schaltet die Programmierspannung wieder ab. Während der gesamten Zeit sind die Daten und Adressen in den Latches gespeichert. Am Ausgang von FF2 ist über Transistor T4 eine LED angeschlossen, die während der Programmierung des EPROMs aufleuchtet.
Beim Lesen der Daten funktioniert die Hardware der Schaltung nicht so ohne weiteres, da der Prozessor keinen Write-Befehl gibt. In diesem Fall werden die Ausgänge von IC2 in den hochohmigen Zustand (einer von "Tri States") geschaltet, und IC1 wird freigegeben. Das läuft ungefähr darauf hinaus, daß die vom EPROM gelieferten Daten praktisch direkt mit den Datenleitungen des Prozessors verbunden sind. Schalter S1 dient zur Einstellung auf den EPROM-Typ. In der gezeichneten Stellung von S1 (2716) liegt die Programmierspannung an Pin 21, gleichzeitig ist der $\overline{OE}$-Eingang mit N11 verbunden. Wenn man den Schalter umstellt (2732), ist die Adressenleitung A11 mit Pin 21 verbunden. Die Programmierspannung liegt an Pin 20.
Mit Hilfe einiger Gatter (N1 . . . N4, N9, N11, N12 und N15) werden, abhängig vom gewählten Prozessor, aus den vorhandenen Steuersignalen weitere Steuersignale für die Schaltung abgeleitet.

Die Euro-Karte

In Bild 2 ist die Platine für den EPROMer dargestellt. Erfahrene Computer-Bauer werden sicherlich keine Schwierigkeiten bei der Bestückung haben. Die EPROM-Fassung sollte von sehr guter Qualität sein, da sie häufig benutzt wird. Ein paar Mark mehr als gewöhnlich sind in diesem Fall kein rausgeworfenes Geld! Am besten eignet sich eine sogenannte Null-Kraft-Fassung. Man setzt das IC in den Sockel und stellt den Kontakt mit einem Hebeldruck her.
Die Karte ist wie erwähnt für zwei unterschiedliche Systeme ausgelegt: SC/MP und Junior. Abhängig davon müssen die entsprechenden Drahtbrücken auf der Platine gelegt werden. Die Verbindungen zum Bus sind alle auf eine DIN-Leiste geführt. Man braucht nur noch eine Wechselspannung von 24 V anzuschließen.

SC/MP + EPROMer

Besitzer des SC/MP-Systems können den EPROMer ganz einfach anwenden. Nach Einstellung der gewünschten Adresse auf der Karte wird zunächst S2 geschlossen. Dann setzt man das EPROM in die Fassung, steckt anschließend die Karte in die Bus-Leiste und beginnt mit der Programmierung. Voraussetzung dafür ist, daß die Programmierspannung anliegt und daß S2 wieder geöffnet wird. Nach der Programmierung sollte man S2 wieder schließen.
Eine spezielle Software ist für den EPROMer in diesem Fall nicht notwendig; ELBUG reicht hier völlig aus. Möchte man einen Speicherplatz programmieren, dann genügt es, MO . . . . YYYY einzugeben. YYYY ist die zu programmierende Adresse. Bei einem leeren EPROM erscheint "FF" als Antwort auf dem Display. Soll auf diesen Platz beispielsweise Ø8 gesetzt werden, dann muß man Ø8 eingeben. Und die Daten sind programmiert.
Sollen größere Daten-Blöcke einprogrammiert werden, dann empfiehlt sich eine Speicherung im RAM des Computers. Anschließend werden die Informationen mit einem Block-Transfer-Befehl in das EPROM geladen (BL . . . SSSS, EEEE, BBBB – S = Start-Adresse, E = End-Adresse, B = Adresse, in die der Block geladen wird).
Das Lesen des EPROM-Inhalts geschieht auf die gleiche Weise, als ob man eine normale Adresse lesen möchte.

Junior + EPROMer

Um den EPROMer mit dem Junior-Computer betreiben zu können, braucht man ein spezielles Programm. Dieses Programm ist in Tabelle 1 aufgelistet. Es beginnt bei Ø2ØØ und endet bei Ø277. Hat man das Programm geladen, dann kann der EPROMer auf die Erweiterungs-Leiste gesteckt werden. Der EPROMer ist jetzt betriebsbereit.
Man kann das Programm aus Tabelle 1 auch in ein EPROM laden, so daß es nicht jedes Mal, wenn der EPROMer in Betrieb genommen werden soll, eingegeben werden muß. Achtung: Die absoluten Sprungadressen im Programm müssen in diesem Fall angepaßt werden! Eine Möglichkeit besteht darin, das TM-EPROM auf der Interface-Karte mit dem Programm zu laden. Dieses EPROM hat nämlich noch einige freie Plätze. In den Adressen ØC8Ø . . . ØFFF steht überall FF. Im EPROM selbst sind das die Plätze 48Ø . . . 7FF. Läßt man das Programm statt bei Ø2ØØ, wie in Tabelle 1, bei ØC8Ø beginnen, kann das Programm in das TM-EPROM geladen werden. In diesem Fall muß man allerdings die absoluten Sprung-Adressen im Programm anpassen (alle 3-Byte-Befehle, die mit Ø2 enden). Im folgenden geben wir eine kurze Beschreibung der drei Programmteile PROGRAM, DUPLICATE und VERIFY.

Die Programmier-Routine

Auf die Speicherplätze MOVL, MOVH (ØØØ4, ØØØ5) werden das linke und das rechte Byte der Adresse, mit der man die Programmierung beginnen möchte, gesetzt. Danach wird die Programmier-Routine gestartet (in Tabelle 1 → Ø2ØØ). Auf dem Display erscheinen nun die

Tabelle 1.

```
0010: 0200                    ORG   $0200
0020:
0030:                  DATE :  10-7-'81
0040:
0050:
0060:                  PAGE ZERO DATA BUFFERS :
0070:
0080: 0200             SAL     *     $0000  DATA BLOCK START ADDRESS
0090: 0200             SAH     *     $0001
0100: 0200             EAL     *     $0002  DATA BLOCK END ADDRESS + 1
0110: 0200             EAH     *     $0003
0120: 0200             MOVL    *     $0004  EPROM PROGRAM START ADDRESS
0130: 0200             MOVH    *     $0005
0140:
0150: 0200             INH     *     $00F9  DISPLAY BUFFER   ( DATA )
0160: 0200             POINTL  *     $00FA     "        "    ( ADDRESS L )
0170: 0200             POINTH  *     $00FB     "        "    ( ADDRESS H )
0180:
0190:                  EXTERNAL SUBROUTINES :
0200:
0210: 0200             GETBYT  *     $1D6F
0220: 0200             SCAND   *     $1D88
0230:
0240:                  JUNIOR MONITOR START :
0250:
0260: 0200             RESET   *     $1C1D
0270:
0280:
0290:                  PROGRAM START ADDRESS     : $0200  ( PROG )
0300:                  DUPLICATE START ADDRESS   : $0222  ( DUPL )
0310:                  VERIFY START ADDRESS      : $0233  ( VERIFY )
0320:
0330:
0340:                  ****************
0350:                  EPROM-PROGRAMMER
0360:                  ****************
0370:
0380: 0200 20 55 02  PROG    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0390: 0203 A0 00     PRGR    LDYIM $00     CLEAR Y-REGISTER
0400: 0205 B1 FA             LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H) AND
0410: 0207 85 F9             STAZ  INH     STORE THIS IN DISPLAY BUFFER INH
0420: 0209 20 6F 1D          JSR   GETBYT  READ TWO HEXKEYS AND STORE THEIR VALUE IN THE
0430:                                      ACCUMULATOR. RETURN WITH N=1 IF
0440:                                      ONLY HEXKEYS WHERE DEPRESSED.
0450:                                      IF A COMMAND KEY WAS DEPRESSED,
0460:                                      RETURN WITH N=0
0470: 020C 10 07             BPL   PR      COMMAND KEY DEPRESSED?
0480: 020E A0 00             LDYIM $00     CLEAR Y-REGISTER
0490: 0210 91 FA             STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0500:                                      THE EPROM MEMORY LOCATION
0510:                                      SPECIFIED BY POINTL(H)
0520: 0212 4C 03 02          JMP   PRGR
0530: 0215 C9 12     PR      CMPIM $12
0540: 0217 D0 35             BNE   PRGR    +KEY?
0550: 0219 E6 FA             INCZ  POINTL  INCREMENT ADDRESS BY ONE
0560: 021B D0 E6             BNE   PRGR
0570: 021D E6 FB             INCZ  POINTH
0580: 021F 4C 03 02          JMP   PRGR
0010: 0222 20 55 02  DUPL    JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0020: 0225 A0 00     DU      LDYIM $00
0030: 0227 B1 00             LDAIY SAL     GET DATA SPECIFIED BY SAL(H)
0040: 0229 91 FA             STAIY POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
0050:                                      THE EPROM MEMORY LOCATION
0060:                                      SPECIFIED BY POINTL(H)
0070: 022B 20 5E 02          JSR   INCMNT  INCREMENT SAL(H) AND POINTL(H) BY ONE
0080: 022E D0 F5             BNE   DU      NOT LAST ADDRESS
0090: 0230 4C 1D 1C          JMP   RESET   RETURN TO JUNIOR MONITOR
0100:
0110: 0233 20 55 02  VERIFY  JSR   TRF     TRANSFER MOVL(H) TO POINTL(H)
0120: 0236 A0 00     VER     LDYIM $00
0130: 0238 B1 FA             LDAIY POINTL  GET DATA SPECIFIED BY POINTL(H)
0140: 023A D1 00             CMPIY SAL     COMPARE THIS DATA WITH DATA SPECIFIED BY SAL(H)
0150: 023C F0 0F             BEQ   NEXT    DATA EQUAL?
0160: 023E 20 88 1D  ANYKEY  JSR   SCAND   DISPLAY EPROM ADDRESS AND DATA
```

```
0170: 0241 D0 FB          BNE   ANYKEY ANY KEY DEPRESSED?
0180: 0243 20 88 1D       JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
0190: 0246 D0 F6          BNE   ANYKEY ANY KEY DEPRESSED?
0200: 0248 20 88 1D NKEY  JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
0210: 024B F0 FB          BEQ   NKEY   NO KEY DEPRESSED?
0220: 024D 20 5E 02 NEXT  JSR   INCMNT INCREMENT SAL(H) AND POINTL(H) BY ONE
0230: 0250 D0 E4          BNE   VER    NOT LAST ADDRESS?
0240: 0252 4C 1D 1C       JMP   RESET  RETURN TO JUNIOR MONITOR
0250:
0260:                     ***********
0270:                     SUBROUTINES
0280:                     ***********
0290:
0300: 0255 A5 04    TRF   LDAZ  MOVL
0310: 0257 85 FA          STAZ  POINTL TRANSFER MOVL TO POINTL
0320: 0259 A5 05          LDAZ  MOVH
0330: 025B 85 FB          STAZ  POINTH TRANSFER MOVH TO POINTH
0340: 025D 60             RTS
0350:
0360: 025E 20 88 1D INCMNT JSR  SCAND  DISPLAY FOR ABOUT 5MS POINTH, POINTL
0370:                                  AND INH ( = EPROM ADDRESS AND DATA
0380:                                  ON THIS ADDRESS )
0390: 0261 E6 00          INCZ  SAL    INCREMENT SAL(H) BY ONE
0400: 0263 D0 02          BNE   INCDA
0410: 0265 E6 01          INCZ  SAH
0420: 0267 E6 FA    INCDA INCZ  POINTL INCREMENT POINTL(H) BY ONE
0430: 0269 D0 02          BNE   COMP
0440: 026B E6 FB          INCZ  POINTH
0450: 026D A5 01    COMP  LDAZ  SAH
0460: 026F C5 03          CMPZ  EAH    COMPARE EAH WITH SAH
0470: 0271 D0 04          BNE   RTRN   EAH NOT EQUAL SAH?
0480: 0273 A5 00          LDAZ  SAL
0490: 0275 C5 02          CMPZ  EAL    COMPARE EAL WITH SAL
0500: 0277 60       RTRN  RTS
```

Tabelle 1. Dieses Programm erlaubt den Betrieb des EPROMers zusammen mit dem Junior-Computer.

Adresse und die Daten, die in dieser Adresse stehen. FF erscheint, wenn das EPROM leer ist. Will man beispielsweise A9 programmieren, dann wird erst A gedrückt (die Anzeige im Display ändert sich nicht) und anschließend 9. Das Display erlischt für etwa 70 ms. Danach erscheinen Adresse und Daten A9 wieder auf dem Display. A9 ist in den entsprechenden Speicherplatz geladen.
Drückt man danach die Plus-Taste, dann erscheint die nächste Adresse auf dem Display. Auf die zuvor beschriebene Art können Daten geladen werden. Möchte man nur Daten lesen, dann braucht nur die Plus-Taste gedrückt werden, oder man kehrt durch Betätigung der Reset-Taste ins Monitor-Programm zurück.

Die Duplicate-Routine

Möchte man ein Programm, das irgendwo im Speicherbereich des Computers steht, in ein EPROM laden, dann wird die Duplicate-Routine verwendet.
Auf die Speicherplätze SAL, SAH (0000, 0001) wird die Start-Adresse des zu kopierenden Programms gesetzt. In EAL, EAH (0002, 0003) legt man die End-Adresse + 1. Und auf MOVL, MOVH (0004, 0005) wird die EPROM-Adresse gesetzt, in der das erste Byte des zu kopierenden Programm-Blocks stehen soll.
Danach kann man das Duplikat-Programm starten (Adresse 0222), und das Display erlischt. Nach einigen Programmierschritten "meldet" sich das Display aber mit der Anzeige des programmierten Speicherplatzes zurück. Auf diese Weise kann man den Programmiervorgang auf dem Display verfolgen. Allerdings ist die Anzeige immer nur sehr kurz und schwach!
Ist der Programmiervorgang beendet, dann meldet sich der Junior mit der Angabe der letzten Adresse + 1 zurück.

Die Verify-Routine

Mittels dieser Routine kann ein bestimmter Programm-Block mit dem EPROM-Inhalt verglichen werden. Auch in diesem Fall werden Start-Adresse, End-Adresse + 1 und Ziel-Adresse genau wie zuvor beschrieben eingegeben. Danach startet man das Programm auf 0233.
Wird nun ein Fehler entdeckt, stoppt das Programm, und auf dem Display wird die Adresse angezeigt, in der der Fehler steckt. Auch die Daten, die in diesem Moment an der Stelle stehen, werden angezeigt. Drückt man die RES- oder ST-Taste, dann läuft das Programm weiter.
Ist alles kontrolliert, meldet sich der Junior wieder mit der letzten Adresse + 1 zurück. Gleichzeitig ist also auch die Rückkehr ins Monitor-Programm erfolgt.

Schließlich bleibt nur noch zu sagen, daß mit Hilfe des EPROMers ein nützliches Gerät zur Verfügung steht, mit dem man EPROMs sehr leicht selbst programmieren kann. In der Praxis wird sich die Nützlichkeit sehr schnell erweisen.