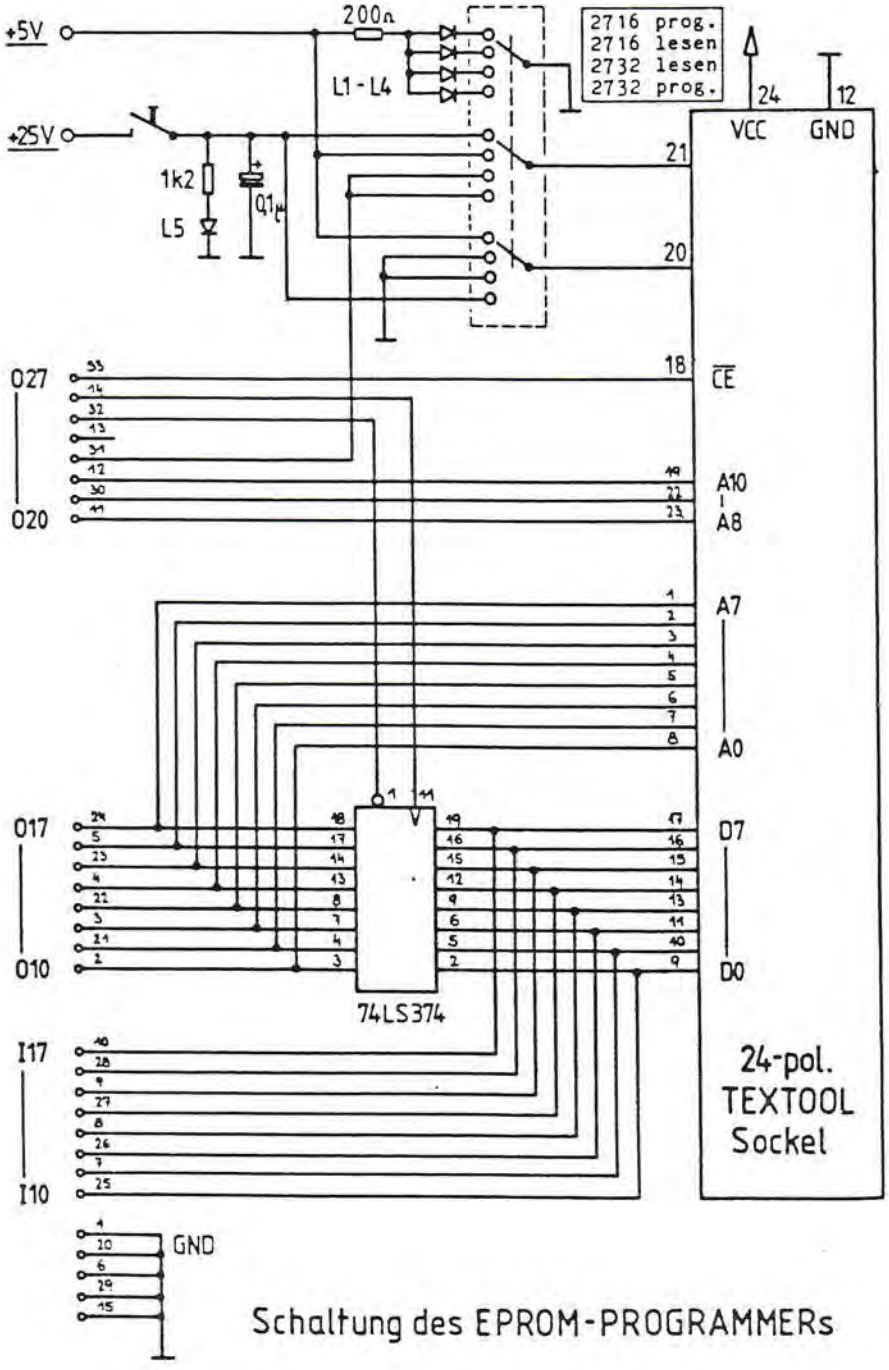
MZ-80B - intern Nr. 11

Bauanleitung für ein EPROM-Programmiergerät eischließlich Software

Programmiergeräte sind eine teure Angelegenheit: Meist besitzen sie eine eigene Mikroprozessorsteuerung und verfügen über einen grossen Speicherbereich. Entweder werden im "stand alone"-Betrieb Daten eingegeben und hierauf in ein EPROM eingebrannt, oder ein Computer steuert im "slave"-Betrieb das Programmiergerät über eine RS232-Schnittstelle. Wenige Anweisungen und die Daten werden übertragen, dann besorgt das Programmiergerät die korrekte Programmierung des EPROMs oder PROMs. Geräte der oberen Preisklasse verfügen über austauschbare Module, während jene der mittleren Preisklasse mit Wahlschaltern das Programmieren der gebräuchlichsten Speicherschaltkreise erlauben.

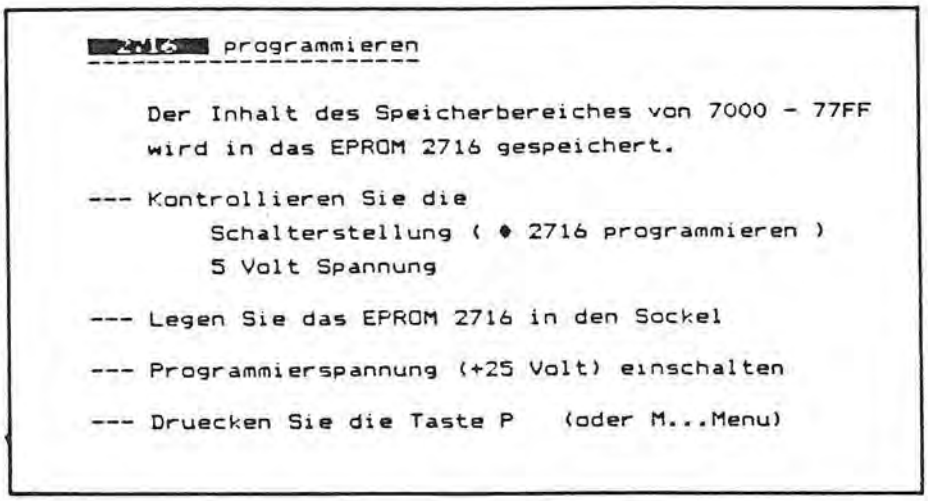
In manche Kleincomputer können preisgünstige Platinen eingesteckt oder extern angeschlossen werden, die die Programmierung von EPROMs ermöglichen. Meist erstreckt sich die Typenwahl aber nur auf EPROMs mit 2 KByte oder 4 KByte Speicherinhalt wie 2716, 2516, 2732, 2532 oder ähnliche. Da diese Festwertspeicher aber recht häufig zu finden sind, leisten diese kleinen Zusatzplatinen wertvolle Dienste.

Ein besonderer Anreiz liegt darin, den Hardwareaufwand auf ein Minimum zu beschränken und die zur Programmierung wesentlichen Grundaufgaben der Software zu übertragen. Dies ist ja eine Grundidee der gesamten Mikroprozessortechnologie,



Schaltung des EPROM-PROGRAMMERs





Bildschirmdialog am SHARP MZ80B

Bild 2: Bildschirmdialog

die wesentlich zur Flexibilitätssteigerung prozessorgesteuerter Geräte beiträgt.

DIE IDEE

Ist der Adressbereich eines Computers voll ausgeschöpft, etwa durch Vollbelegung des Speichers mit 64 KByte, oder durch Memorymapping nicht weiter verwertbar, so kann über Parallelports ein Adresszähler simuliert werden.

Stehen mindestens zwei 8-bit-Ausgabeports und ein 8-bit-Eingabeport zur Verfügung, so reduziert sich der Hardware-Aufwand für ein Programmiergerät auf einen IC 74LS374, der eine Doppelfunktion erfüllt: Für die Programmierung der EPROMs dient er als Zwischenspeicher für die Datenwerte, beim Lesen oder Verifizieren der EPROM-Inhalte sorgt er durch Tri-State-Ausgänge zur Trennung der Busrichtungen.

DIE AUSFUEHRUNG

Anschliessend wird eine konkrete Schaltungsvariante vorgestellt, die hard- und softwaremässig für den Personal Computer SHARP MZ80B mit Parallelportinterface MZ80I02 realisiert wurde, aber an jeden Kleincomputer mit dem Mikroprozessor Z80 übernommen werden kann. Es sind nur zwei Details zu beachten: Die Software wurde als Hybridprogramm abgefasst, das heisst der Dialogteil wurde in SHARP-BASIC geschrieben und kann mit geringfügigen Aenderungen in jede andere BASIC-Version übertragen werden.

Der Maschinenprogrammteil kann für Z80-Systeme ungeändert übernommen werden, es ist nur darauf zu achten, dass die Zeitschleife "T50MS", die die Länge des Programmierimpulses (50 Millisekunden) bestimmt, für einen Systemtakt von 4 MHz (Z80A) ausgelegt ist. Im Assemblerlisting sind die Befehlsausführungszeiten (in Mikrosekunden) angegeben. Bei anderem Systemtakt sind diese Werte proportional und der Schleifenzählerwert in BC ist dementsprechend zu ändern.

Zu beachten sind noch die Portadressen (Output $00, $01, Input $00). Das Maschinenprogramm benötigt weniger als 256 Bytes, zusätzlich werden 2 KByte bzw. 4 KByte als Speicher für die zu programmierenden Daten benötigt.

Das Programm wurde so erstellt, dass es auch auf Systemen mit nur 32 KByte läuft, ausser dem BASIC-Bereich wird nur der Adressbereich von 6F00-7FFF belegt.

DIE HARDWARE

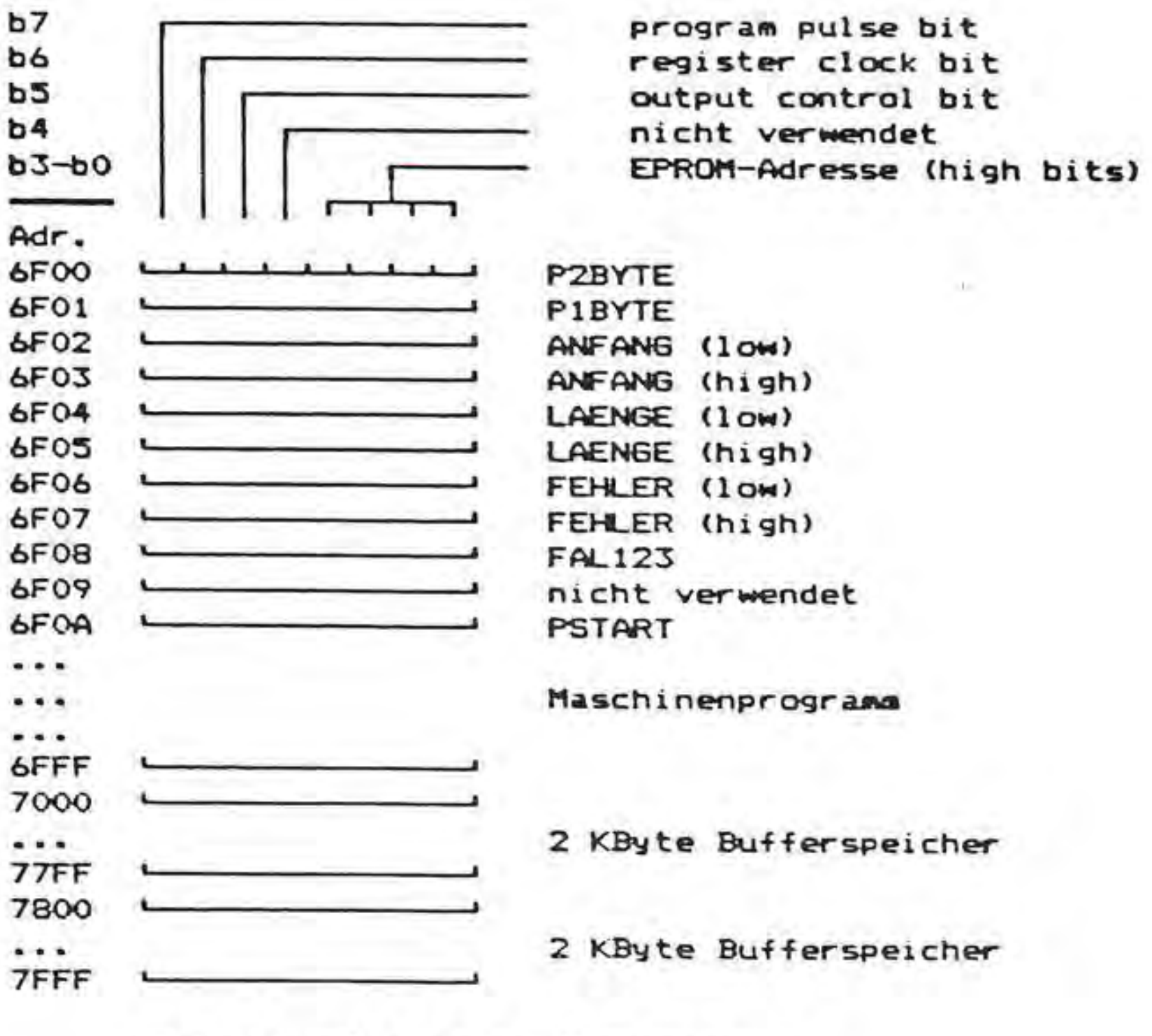
Im Prinzip reicht ein IC 74LS374 und ein 24-poliger Sockel für das Programmiergerät, sofern die drei notwendigen Ports zur Verfügung stehen. Extern muss noch für eine stabilisierte Spannung von +25 Volt gesorgt werden, die zur Programmierung notwendig ist. Hierzu reicht ein Netzgerät, bestehend aus einem 24-Volt-Trafo, Gleichrichter, Kondensator und einem einstellbaren Spannungsregler LM317. Bei seltenem Gebrauch lässt sich auch eine einfache stabilisierte 25-Volt-Spannung aus drei 9-Volt-Batterien erzielen.

Um den Hardware-Aufwand gering zu halten, wurde auf eine Umschaltung auf die EPROM-Typen durch Software verzichtet und ein Vier-

LISTING 1

```
10 REM   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
15 REM   ♥                                      ♥
20 REM   ♥  E P R O M  -  P R O G R A M M E R   ♥
25 REM   ♥                                      ♥
30 REM   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
35 REM
40 REM
45 REM
50 REM   Dieses Programm erlaubt das Programmieren,
55 REM   Verifizieren und Lesen von EPROMs des Typs
60 REM            ____        ____
65 REM            2716   oder 2732
70 REM            ____        ____
75 REM   mit Hilfe der "EPROMMER" - Schaltung.
80 REM
85 REM   Hardware + SHARP MZ80B mit 32K oder 64K
90 REM              Parallelinterface MZ80I02
95 REM              oder gleichwertige Ausrüstung
100 CONSOLE C80
110 CURSOR 30,10: PRINT "Ein wenig Geduld!"
120 REM
130 REM --- Initialisierung
140 REM
150 REM  Portadressen sind       $00, $01, $00 (siehe Masch.prog.)
160 REM
170 SBLOCK=  7*4096            :REM  Erste Speicherstelle $7000
180 BUFFER=  6*4096+15*256     :REM  Adresse für Masch.prog. $6F00
190 P1BYTE=  BUFFER+1          :REM  Adresse für Byte für Outport 1
200 P2BYTE=  BUFFER            :REM  Adresse für Byte für Outport 2
210 ANFANG=  BUFFER+2          :REM  Adresse für Blockbeginn
220 LAENGE=  BUFFER+4          :REM  Adresse für Blocklänge
230 FEHLER=  BUFFER+6          :REM  Adresse für Fehlerzähler
240 FAL123=  BUFFER+8          :REM  Adresse für Falluntersch. (1,2,3)
250 PSTART=  BUFFER+10         :REM  Start des Maschinenprogramms
260 LIMIT    BUFFER            :REM  Trennung vom BASIC-Bereich
270 REM
280 REM --- Maschinenprogramm laden
290 REM
300 GOSUB 1450: MUSIC "+D1+E1+F1"
310 REM
320 REM --- Hauptmenu anzeigen
```

Fortsetzung des Programmausdruckes auf der nächsten Seite



Speicherbelegung

Bild 3: Speicherbelegung

stufenschalter mit zwei Schaltebenen vorgesehen, mit dessen Hilfe

2716 programmieren
2716 lesen/verifizieren
2732 lesen/verifizieren
2732 programmieren

gewählt werden kann.

Als zusätzlicher "Luxus" kann mit einer dritten Schalterebene die Schalterstellung über Leuchtdioden (L1-L4) signalisiert werden. Zudem ist es empfehlenswert, eine andersfarbige Leuchtdiode an die 25-Volt-Spannung anzuschliessen, um eine optische Rückmeldung zu erhalten, ob die Programmierspannung eingeschaltet ist oder nicht.

SCHALTUNGSAUFBAU

Aus Bild 1 ersehen Sie die Schaltung des Programmiergerätes. IC

```
330 REM
340 GOSUB 400: IF T=9 GOTO 1340
350 REM
360 ON T GOTO 610, 860, 1110, 610, 860, 1110, 340, 340
370 REM
380 REM ── Hauptmenu anzeigen
390 REM
                                    EPROM-Programmer
400 PRINT CHR$(6): C=20                  ↙
410 PRINT TAB(C); STRING$("█",37)
420 PRINT TAB(C); "[illegible]
430 PRINT TAB(C); STRING$("█",37): C=24
440 CURSOR C,6 : PRINT "1 -   2716  programmieren"
450 CURSOR C,8 : PRINT "2 -         verifizieren"
460 CURSOR C,10: PRINT "3 -         einlesen"
470 CURSOR C,13: PRINT "4 -   2732  programmieren"
480 CURSOR C,15: PRINT "5 -         verifizieren"
490 CURSOR C,17: PRINT "6 -         einlesen"
500 CURSOR C,20: PRINT "9 -   Programmende": C=20
510 CURSOR C,23: PRINT "Drücken Sie die gewünschte Trste !";
520 GET T$: IF (T$<"1")+(T$>"9") GOTO 520
530 T= ASC(T$)-ASC("0")
540 IF T<4 THEN EPROM$="[illegible]": E$="2716": A$="77FF": POKE LAENGE+1,$08
550 IF T>3 THEN EPROM$="[illegible]": E$="2732": A$="7FFF": POKE LAENGE+1,$10
560 PRINT CHR$(6)               ↑ 2716
570 RETURN                        2732
580 REM
590 REM ── Programm `programmieren`
600 REM
610 POKE FAL123,1
620 CURSOR C,1 : PRINT EPROM$; " programmieren"
630 CURSOR C,2 : PRINT "─────────────────"
640 CURSOR C,5 : PRINT "    Der Inhalt des Speicherbereiches von 7000 - "; A$
650 CURSOR C,7 : PRINT "    wird in das EPROM "; E$; " gespeichert."
660 CURSOR C,10: PRINT "─── Kontrollieren Sie die"
670 CURSOR C,12: PRINT "         Schalterstellung ( ◆ "; E$;" programmieren )"
680 CURSOR C,14: PRINT "         5 Volt Spannung"
690 CURSOR C,17: PRINT "─── Legen Sie das EPROM "; E$; " in den Sockel"
700 CURSOR C,20: PRINT "─── Programmierspannung (+25 Volt) einschalten"
710 CURSOR C,23: PRINT "─── Drücken Sie die Taste P   (oder M...Menu)"
720 GET T$: IF (T$<>"P")*(T$<>"M") GOTO 720
730 IF T$="M" GOTO 820   EPROM ↓          wird programmiert ↓
740 CURSOR C,23: PRINT "[illegible]";EPROM$;"[illegible]
750 IF T=4 THEN POKE $6F30,$A0: POKE $6FB1,$BF: POKE $6FB9,$FF
760 USR(PSTART)
770 IF T=4 THEN POKE $6F30,$20: POKE $6FB1,$FF: POKE $6FB9,$BF
780 MUSIC "+G1" : CURSOR C,17: PRINT STRING$("-",50)  ↙ Programmierung fertig!
790 CURSOR C,20: PRINT "[illegible]
800 CURSOR C,23: PRINT "[illegible]  Drücken Sie M (Menu)"
810 GET T$: IF T$<>"M" GOTO 810         ↖ +25Volt abschalten!
820 GOTO 340: REM Menu
830 REM
840 REM ── Programm `verifizieren`
850 REM
860 POKE FAL123,2
870 CURSOR C,1 : PRINT EPROM$; " verifizieren"
880 CURSOR C,2 : PRINT "─────────────────"
890 CURSOR C,6 : PRINT "    Der Inhalt des EPROMs ";E$;" wird mit dem Inhalt"
900 CURSOR C,8 : PRINT "    des Speicherbereiches 7000 - ";A$;" verglichen."
910 CURSOR C,12: PRINT "─── Kontrollieren Sie die"
920 CURSOR C,14: PRINT "         Schalterstellung ( ◆ "; E$; " lesen )"
930 CURSOR C,16: PRINT "         5 Volt Spannung"
940 CURSOR C,19: PRINT "─── Legen Sie das EPROM "; E$; " in den Sockel"
950 CURSOR C,23: PRINT "─── Drücken Sie die Taste V   (oder M...Menu)"
960 GET T$: IF (T$<>"V")*(T$<>"M") GOTO 960
970 IF T$="M" GOTO 1070
980 CURSOR C,23: PRINT"[illegible]";EPROM$;"
990 REM                 ↑ Verifizieren des EPROMs
1000 USR(PSTART)
1010 REM
1020 A=PEEK(FEHLER+1): B=PEEK(FEHLER): F=256*A+B: MUSIC "+G1"
1030 CURSOR C,19: PRINT STRING$("-",50)
1040 CURSOR C,23: PRINT"[illegible]    "; F; " Fehler   ";
1050 PRINT "───  Drücken Sie  M  (Menu)"
1060 GET T$: IF T$<>"M" GOTO 1060        ↖ Fertig!
1070 GOTO 340: REM Menu
1080 REM
1090 REM ── Programm `einlesen`
1100 REM
1110 POKE FAL123,3
1120 CURSOR C,1 : PRINT EPROM$; " einlesen"
1130 CURSOR C,2 : PRINT "─────────────────"
1140 CURSOR C,6 : PRINT "    Der Inhalt des EPROMs "; E$; " wird in den"
1150 CURSOR C,8 : PRINT "    Speicherbereich von 7000 - "; A$; " eingelesen."
1160 CURSOR C,12: PRINT "─── Kontrollieren Sie die"
1170 CURSOR C,14: PRINT "         Schalterstellung ( ◆ "; E$; " lesen )"
1180 CURSOR C,16: PRINT "         5 Volt Spannung
```

Fortsetzung des Programmausdruckes auf der nächsten Seite

74LS374 ist ein 8-bit-Register mit Tri-State-Ausgängen. Beim Lesen oder Verifizieren eines EPROMs werden die Ausgänge inaktiviert (High-Z), indem über Output-Port 2 Bit 5 auf logisch 1 gelegt wird.

Beim Programmiervorgang dient dieses Register als Speicher für die zu programmierenden Datenbytes. Ein Low-high-low-Clock-Impuls über Bit 6 des Ports 2 bewirkt die Datenübernahme des Bytes, das über Port 1 ausgegeben wurde.

Bit 7 des Ports 2 ist für den Programmierimpuls verantwortlich. Die EPROM-Typen 2716 und 2732 sind (siehe Datenblatt) sehr einfach zu programmieren: wenn Adresse und Datenbyte anliegen und die Programmierspannung von 25 Volt am entsprechenden Pin anliegt, wird ein TTL-Impuls von 50 Millisekunden Länge an Pin 18 ausgegeben. Die einzelnen Bytes können in aufsteigender oder in beliebiger Reihenfolge programmiert werden.

Falls man das Programmiergerät an die Parallelportplatine des SHARP MZ808 anschliessen will, empfiehlt sich die Realisierung der Hardware in Form einer kleinen Platine, die über einen 37-poligen D-Stecker direkt an der Computerrückseite angesteckt wird. Führt man die Verdrahtung auf einer Lochrasterplatine in Wire-wrap-Technik aus, so ist der Aufwand recht gering, eine spätere Umverdrahtung für andere EPROM-Typen ist leicht ausführbar. Als EPROM-Sockel empfiehlt sich ein 24-poliger TEXTOOL-Sockel oder ein gleichwertiges Fabrikat mit Nullkraftkontakten.

DIE SOFTWARE

Das zur Programmierung gehörende Programm ist in zwei Abschnitte gegliedert: das Dialogprogramm, es ist in SHARP-BASIC abgefasst (Listing 1) und meldet sich mit einem Menu, das aus Bild 2 ersichtlich ist. Nach dem Drücken der entsprechenden Taste folgen die Anweisun-

gen zum Programmieren, Verifizieren (Vergleichen) oder Lesen eines EPROMs 2716 oder 2732.

Die Eingabe der Hexadezimalwerte, die in das EPROM geschrieben werden sollen, erfolgt mit dem Monitor in den Speicherbereich von 7000 bis 77FF (bzw. 7FFF). Natürlich besteht auch die Möglichkeit, ein kurzes BASIC-Unterprogramm für hexadezimale Dateneingabe anzufügen.

Der zweite Teil (im DATA-Bereich) enthält ein Z80-Maschinenprogramm (Listing 2), das vom BASIC-Programm geladen wird und durch USR(PSTART) von diesem aufgerufen wird.

Es führt in Sekundenbruchteilen das Einlesen oder Verifizieren eines EPROMs aus. Das Programmieren dauert rund zwei Minuten (EPROM 2716) oder vier Minuten (EPROM 2732), da pro Byte 50 Millisekunden Programmierdauer eingehalten werden müssen.

Dem Anwender steht es frei, das Programm zu ergänzen. Praktisch ist ein "empty check", das heisst, es wird überprüft, ob das EPROM vollständig gelöscht ist, ob alle Bytes auf hexadezimal FF gesetzt sind. Dies lässt sich auch erzielen, indem man alle Bytes des Bufferspeichers auf FF setzt und das EPROM verifiziert. Auch das Programmieren einzelner Bytes kann ergänzt werden, wird in den meisten Anwendungsfällen aber von untergeordneter Bedeutung sein.

Im vorliegenden Programm werden für die EPROMs 2716 oder 2732 folgende Befehle ausgeführt:

```
1190 CURSOR C,19: PRINT "—— Legen Sie das EPROM "; E$; " in den Sockel"
1200 CURSOR C,23: PRINT "—— Druecken Sie die Taste L    (oder M...Menu)"
1210 GET T$: IF (T$<>"L")&(T$<>"M") GOTO 1210
1220 IF T$="M" GOTO 1300
1230 REM
1240 USR(PSTART)
1250 REM
1260 MUSIC "+61": CURSOR C,19: PRINT STRING$("-",50)
1270 CURSOR C,23: PRINT "[redacted]";EPROM$;"[redacted] ";
1280 PRINT "Druecken Sie  M (Menu)"
1290 GET T$: IF T$<>"M" GOTO 1290        ↑ Fertig! EPROM   ↑ eingelesen
1300 GOTO 340: REM Menu
1310 REM
1320 REM —— Programmende
1330 REM
1340 CURSOR C,1 :PRINT "Programmende"
1350 CURSOR C,2 :PRINT "————————"
1360 CURSOR C,5 :PRINT "Programmierspannung (+25 Volt) abschalten!"
1370 CURSOR C,10:PRINT "Auf Wiedersehen!": PRINT: PRINT
1380 LIMIT MAX          : REM maximaler Speicherbereich
1390 END
1400 REM
1410 REM
1420 REM  ——  Maschinenprogramm laden (ab $6F00)
1430 REM      Programmstart bei PSTART (= $6F0A)
1440 REM
1450 RESTORE 1630
1460 N=0
1470 READ U$: IF U$="FINITO" GOTO 1520
1480 GOSUB 1560: REM Hex-Dez umwandeln
1490 POKE BUFFER+N,U
1500 N=N+1
1510 GOTO 1470
1520 RETURN
1530 REM
1540 REM  ——  HEX-DEZ
1550 REM
1560 U1= ASC(LEFT$(U$,1))-48:  IF U1>9 THEN U1=U1-7
1570 U2= ASC(RIGHT$(U$,1))-48: IF U2>9 THEN U2=U2-7
1580 U=  16*U1+U2
1590 RETURN
1600 REM
1610 REM  ——  Maschinenprogramm
1620 REM
1630 DATA 20,00,00,70,00,08,00,00,   00,00,E5,D5,C5,F5,CD,2F
1640 DATA 6F,2A,02,6F,ED,4B,04,6F,   3A,08,6F,FE,01,28,1F,FE
1650 DATA 02,28,3B,FE,03,28,56,CD,   2F,6F,F1,C1,D1,E1,C9,3E
1660 DATA 20,32,00,6F,D3,01,3E,00,   32,01,6F,D3,00,C9,3A,00
1670 DATA 6F,CB,AF,32,00,6F,D3,01,   CD,9A,6F,CD,CB,6F,78,B1
1680 DATA 20,F6,3A,00,6F,CB,EF,32,   00,6F,D3,01,18,C9,11,00
1690 DATA 00,ED,53,06,6F,CD,8A,6F,   BE,28,09,E5,2A,06,6F,23
1700 DATA 22,06,6F,E1,CD,CB,6F,78,   B1,20,EA,18,AA,CD,8A,6F
1710 DATA 77,CD,CB,6F,78,B1,20,F5,   18,9D,3A,01,6F,D3,00,3A
1720 DATA 00,6F,D3,01,00,00,00,DB,   00,C9,7E,D3,00,3A,00,6F
1730 DATA CB,F7,D3,01,CB,B7,D3,01,   3A,01,6F,D3,00,3A,00,6F
1740 DATA CB,FF,F3,D3,01,CD,BE,6F,   CB,BF,D3,01,FB,C9,F5,C5
1750 DATA 01,30,25,0B,78,B1,20,FB,   C1,F1,C9,E5,21,01,6F,34
1760 DATA AF,BE,20,04,21,00,6F,34,   E1,23,0B,C9,      FINITO
```

Listing 1 Programm für EPROM-PROGRAMMER

PROGRAMMIEREN

Der Speicherinhalt von 7000 bis 77FF (bzw. 7FFF) wird in das EPROM geschrieben.

VERIFIZIEREN

Der Inhalt des EPROMs wird mit dem Speicherinhalt von 7000 bis 77FF (bzw. 7FFF) verglichen. Sollten Abweichungen auftreten, so werden die fehlerhaften Bytes gezählt.

LESEN

Der Inhalt des EPROMs wird in den Speicherbereich von 7000 bis 77FF (bzw. 7FFF) eingelesen.

DER PROGRAMMABLAUF

Im Speicherbereich werden folgende Werte abgelegt (Bild 3). P2BYTE und P1BYTE enthalten die EPROM-Adressen, die über Port 2 und Port 1 ausgegeben werden. In ANFANG ist die Anfangsadresse 7000 des Speicherbereichs abgelegt. In LAENGE steht die Blocklänge, entweder 0800 (=2048 dezimal) für EPROM 2716

oder 1000 (=4096 dezimal) für EPROM 2732.

Die Speicherstelle FAL123 wird vom BASIC-Programm mit 1, 2 oder 3 belegt, je nachdem, ob das EPROM programmiert, verglichen oder eingelesen wird.

Bei der Programmierung wird ein Datenbyte in das 8-bit-Register geschrieben, die Adresse an Port 2 und Port 1 ausgegeben und schliesslich der Programmierimpuls von 50 Millisekunden Dauer ausgegeben. Um die Zeitschleife nicht zu stören, wird für diese 50 ms Dauer die Interruptmöglichkeit abgeschaltet und anschliessend wieder eingeschaltet.

Nach jeder Byteausgabe werden die Adresszeiger für den Speicher und das EPROM erhöht, bis alle Bytes programmiert sind.

Nach der Fertigmeldung am Bildschirm empfiehlt sich das Verifizieren des EPROMs, wobei der Schalter auf "LESEN" umzustellen ist. Beim Verifizieren werden die programmierten Bytes mit dem Speicherinhalt verglichen. Sollten Unstimmigkeiten auftreten, so wird die Anzahl der fehlerhaften Bytes angezeigt.

WICHTIGE TIPS

Kontrollieren Sie die Verdrahtung Ihrer Platine, lassen Sie es nicht auf einen "Rauchtest" ankommen. Die Numerierung der Portlinien am linken Rand des Schaltplanes bezieht sich auf den 37-poligen Stecker der SHARP MZ80I02-Platine.

Achten Sie auf die richtige Eingabe der Hexwerte in den DATA-Zeilen des BASIC-Programms. Ein einziger falscher Wert kann zur Selbstzerstörung des gesamten Programms führen (vor dem Testen speichern!).

Achten Sie beim Programmieren auf die richtige Schalterstellung, damit die 25-Volt-Spannung nicht am falschen Pin anliegt!

LISTING 2

```
*****************************************************
*                                                   *
*     Maschinenspracheprogramm zu  E P R O M M E R  *
*                                                   *
*****************************************************

Das folgende Programm im Z80-Code ist im BASIC-Programm
enthalten und wird von diesem geladen und ausgeführt.
Die Zeitschleife für den 50-Millisekunden-Programmier-
impuls ist für 4-MHz-Takt (Z80A) ausgelegt.

                        ORG     6F00H

0000                    PORT1   EQU     00H
0001                    PORT2   EQU     01H
0000                    IPORT   EQU     00H

6F00  20                P2BYTE: DEFB    20H
6F01  00                P1BYTE: DEFB    00H
6F02  7000'             ANFANG: DEFW    7000H
6F04  0800'             LAENGE: DEFW    0800H
6F06  0000'             FEHLER: DEFW    0000H
6F08  0000'             FAL123: DEFW    0000H

6F0A  E5                PSTART: PUSH    HL              ;Register im Stack speichern
6F0B  D5                        PUSH    DE
6F0C  C5                        PUSH    BC
6F0D  F5                        PUSH    AF
6F0E  CD 6F2F'                  CALL    INIT            ;Initialisierung der Output-Ports
6F11  2A 6F02'                  LD      HL,(ANFANG)     ;Blockbeginn
6F14  ED 4B 6F04'               LD      BC,(LAENGE)     ;Blocklänge
6F18  3A 6F08'                  LD      A, (FAL123)     ;Fallunterscheidung 1/2/3:
6F1B  FE 01                     CP      01H
6F1D  28 1F                     JR      Z, PROGR        ;Programmieren
6F1F  FE 02                     CP      02H
6F21  28 3B                     JR      Z, VERIF        ;Verifizieren
6F23  FE 03                     CP      03H
6F25  28 56                     JR      Z, LESEN        ;Lesen
6F27  CD 6F2F'          RUECK:  CALL    INIT
6F2A  F1                        POP     AF              ;Register rückspeichern
6F2B  C1                        POP     BC
6F2C  D1                        POP     DE
6F2D  E1                        POP     HL
6F2E  C9                        RET

6F2F  3E 20             INIT:   LD      A,20H           ;Initialisierung der Output-Ports
6F31  32 6F00'                  LD      (P2BYTE),A
6F34  D3 01                     OUT     (PORT2),A       ;Byte 0010 0000
6F36  3E 00                     LD      A,00H
6F38  32 6F01'                  LD      (P1BYTE),A
6F3B  D3 00                     OUT     (PORT1),A       ;Byte 0000 0000
6F3D  C9                        RET
6F3E  3A 6F00'          PROGR:  LD      A,(P2BYTE)      ;Programmieren von 2K/4K-Byte
6F41  CB AF                     RES     5,A             ;Bit 5 = 0
6F43  32 6F00'                  LD      (P2BYTE),A
6F46  D3 01                     OUT     (PORT2),A       ;Register-Output aktivieren
6F48  CD 6F9A'          PROG1:  CALL    WRITBY          ;Programmieren eines Bytes
6F4B  CD 6FCB'                  CALL    NEXTAD          ;Adressen erhöhen, Zähler dekrem.
6F4E  78                        LD      A,B
6F4F  B1                        OR      C
6F50  20 F6                     JR      NZ,PROG1        ;fertig? - nein, nochmals
6F52  3A 6F00'                  LD      A,(P2BYTE)      ;ja
6F55  CB EF                     SET     5,A             ;Bit 5 = 1
6F57  32 6F00'                  LD      (P2BYTE),A
6F5A  D3 01                     OUT     (PORT2),A       ;Register-Output inaktivieren
6F5C  18 C9                     JR      RUECK           ;Rücksprung

6F5E  11 0000'          VERIF:  LD      DE,0000H        ;Verifizieren
6F61  ED 53 6F06'               LD      (FEHLER),DE     ;Fehlerzähler = 0000
6F65  CD 6F8A'          VER1:   CALL    READBY          ;ein Byte von EPROM lesen
6F68  BE                        CP      (HL)            ;und vergleichen mit Speicher
6F69  28 09                     JR      Z,VER2          ;identisch? - ja
6F6B  E5                        PUSH    HL              ;nein - Fehlerzähler erhöhen
6F6C  2A 6F06'                  LD      HL,(FEHLER)
6F6F  23                        INC     HL
6F70  22 6F06'                  LD      (FEHLER),HL
6F73  E1                        POP     HL
6F74  CD 6FCB'          VER2:   CALL    NEXTAD          ;Adresszeiger erhöhen
6F77  78                        LD      A,B
6F78  B1                        OR      C
6F79  20 EA                     JR      NZ,VER1         ;fertig? - nein, nochmals
6F7B  18 AA                     JR      RUECK           ;ja - Rücksprung
```

Fortsetzung des Programmausdruckes auf der nächsten Seite

oder 1000 (=4096 dezimal) für EPROM 2732.

Die Speicherstelle FAL123 wird vom BASIC-Programm mit 1, 2 oder 3 belegt, je nachdem, ob das EPROM programmiert, verglichen oder eingelesen wird.

Bei der Programmierung wird ein Datenbyte in das 8-bit-Register geschrieben, die Adresse an Port 2 und Port 1 ausgegeben und schliesslich der Programmierimpuls von 50 Millisekunden Dauer ausgegeben. Um die Zeitschleife nicht zu stören, wird für diese 50 ms Dauer die Interruptmöglichkeit abgeschaltet und anschliessend wieder eingeschaltet.

Nach jeder Byteausgabe werden die Adresszeiger für den Speicher und das EPROM erhöht, bis alle Bytes programmiert sind.

Nach der Fertigmeldung am Bildschirm empfiehlt sich das Verifizieren des EPROMs, wobei der Schalter auf "LESEN" umzustellen ist. Beim Verifizieren werden die programmierten Bytes mit dem Speicherinhalt verglichen. Sollten Unstimmigkeiten auftreten, so wird die Anzahl der fehlerhaften Bytes angezeigt.

WICHTIGE TIPS

Kontrollieren Sie die Verdrahtung Ihrer Platine, lassen Sie es nicht auf einen "Rauchtest" ankommen. Die Numerierung der Portlinien am linken Rand des Schaltplanes bezieht sich auf den 37-poligen Stecker der SHARP MZ80I02-Platine.

Achten Sie auf die richtige Eingabe der Hexwerte in den DATA-Zeilen des BASIC-Programms. Ein einziger falscher Wert kann zur Selbstzerstörung des gesamten Programms führen (vor dem Testen speichern!).

Achten Sie beim Programmieren auf die richtige Schalterstellung, damit die 25-Volt-Spannung nicht am falschen Pin anliegt!

LISTING 2

```
*****************************************************
*                                                   *
*     Maschinenspracheprogramm zu  E P R O M M E R  *
*                                                   *
*****************************************************

Das folgende Programm im Z80-Code ist im BASIC-Programm
enthalten und wird von diesem geladen und ausgeführt.
Die Zeitschleife für den 50-Millisekunden-Programmier-
impuls ist für 4-MHz-Takt (Z80A) ausgelegt.

                        ORG     6F00H

0000                    PORT1   EQU     00H
0001                    PORT2   EQU     01H
0000                    IPORT   EQU     00H

6F00  20                P2BYTE: DEFB    20H
6F01  00                P1BYTE: DEFB    00H
6F02  7000'             ANFANG: DEFW    7000H
6F04  0800'             LAENGE: DEFW    0800H
6F06  0000'             FEHLER: DEFW    0000H
6F08  0000'             FAL123: DEFW    0000H

6F0A  E5                PSTART: PUSH    HL              ;Register im Stack speichern
6F0B  D5                        PUSH    DE
6F0C  C5                        PUSH    BC
6F0D  F5                        PUSH    AF
6F0E  CD 6F2F'                  CALL    INIT            ;Initialisierung der Output-Ports
6F11  2A 6F02'                  LD      HL,(ANFANG)     ;Blockbeginn
6F14  ED 4B 6F04'               LD      BC,(LAENGE)     ;Blocklänge
6F18  3A 6F08'                  LD      A, (FAL123)     ;Fallunterscheidung 1/2/3:
6F1B  FE 01                     CP      01H
6F1D  28 1F                     JR      Z, PROGR        ;Programmieren
6F1F  FE 02                     CP      02H
6F21  28 3B                     JR      Z, VERIF        ;Verifizieren
6F23  FE 03                     CP      03H
6F25  28 56                     JR      Z, LESEN        ;Lesen
6F27  CD 6F2F'          RUECK:  CALL    INIT
6F2A  F1                        POP     AF              ;Register rückspeichern
6F2B  C1                        POP     BC
6F2C  D1                        POP     DE
6F2D  E1                        POP     HL
6F2E  C9                        RET

6F2F  3E 20             INIT:   LD      A,20H           ;Initialisierung der Output-Ports
6F31  32 6F00'                  LD      (P2BYTE),A
6F34  D3 01                     OUT     (PORT2),A       ;Byte 0010 0000
6F36  3E 00                     LD      A,00H
6F38  32 6F01'                  LD      (P1BYTE),A
6F3B  D3 00                     OUT     (PORT1),A       ;Byte 0000 0000
6F3D  C9                        RET
6F3E  3A 6F00'          PROGR:  LD      A,(P2BYTE)      ;Programmieren von 2K/4K-Byte
6F41  CB AF                     RES     5,A             ;Bit 5 = 0
6F43  32 6F00'                  LD      (P2BYTE),A
6F46  D3 01                     OUT     (PORT2),A       ;Register-Output aktivieren
6F48  CD 6F9A'          PROG1:  CALL    WRITBY          ;Programmieren eines Bytes
6F4B  CD 6FCB'                  CALL    NEXTAD          ;Adressen erhöhen, Zähler dekrem.
6F4E  78                        LD      A,B
6F4F  B1                        OR      C
6F50  20 F6                     JR      NZ,PROG1        ;fertig? - nein, nochmals
6F52  3A 6F00'                  LD      A,(P2BYTE)      ;ja
6F55  CB EF                     SET     5,A             ;Bit 5 = 1
6F57  32 6F00'                  LD      (P2BYTE),A
6F5A  D3 01                     OUT     (PORT2),A       ;Register-Output inaktivieren
6F5C  18 C9                     JR      RUECK           ;Rücksprung

6F5E  11 0000'          VERIF:  LD      DE,0000H        ;Verifizieren
6F61  ED 53 6F06'               LD      (FEHLER),DE     ;Fehlerzähler = 0000
6F65  CD 6F8A'          VER1:   CALL    READBY          ;ein Byte von EPROM lesen
6F68  BE                        CP      (HL)            ;und vergleichen mit Speicher
6F69  28 09                     JR      Z,VER2          ;identisch? - ja
6F6B  E5                        PUSH    HL              ;nein - Fehlerzähler erhöhen
6F6C  2A 6F06'                  LD      HL,(FEHLER)
6F6F  23                        INC     HL
6F70  22 6F06'                  LD      (FEHLER),HL
6F73  E1                        POP     HL
6F74  CD 6FCB'          VER2:   CALL    NEXTAD          ;Adresszeiger erhöhen
6F77  78                        LD      A,B
6F78  B1                        OR      C
6F79  20 EA                     JR      NZ,VER1         ;fertig? - nein, nochmals
6F7B  18 AA                     JR      RUECK           ;ja - Rücksprung
```

Fortsetzung des Programmausdruckes auf der nächsten Seite

```
6F7D  CD 6F8A'    LESEN:  CALL    READBY          ;Lesen - ein Byte einlesen
6F80  77                  LD      (HL),A          ;und speichern
6F81  CD 6FCB'            CALL    NEXTAD          ;Adresszeiger erhöhen
6F84  78                  LD      A,B
6F85  B1                  OR      C
6F86  20 F5               JR      NZ,LESEN        ;fertig? - nein, nochmals
6F88  18 9D               JR      RUECK           ;ja - Rücksprung

                          ; READBY liest ein Byte des EPROMs (Adresse in
                          ;       P2BYTE/P1BYTE) in den Accu ein

6F8A  3A 6F01'    READBY: LD      A,(P1BYTE)      ;Adresse ausgeben
6F8D  D3 00               OUT     (PORT1),A
6F8F  3A 6F00'            LD      A,(P2BYTE)
6F92  D3 01               OUT     (PORT2),A
6F94  00                  NOP                     ;Verzögerung
6F95  00                  NOP
6F96  00                  NOP
6F97  DB 00               IN      A,(IPORT)       ;Byte einlesen
6F99  C9                  RET                     ;Rücksprung
                          ; WRITBY schreibt das Byte in (HL) in das EPROM
                          ;       (Adresse in P2BYTE/P1BYTE)

6F9A  7E          WRITBY: LD      A,(HL)          ;Byte nach Accu
6F9B  D3 00               OUT     (PORT1),A       ;Ausgabe an Port1
6F9D  3A 6F00'            LD      A,(P2BYTE)
6FA0  CB F7               SET     6,A             ;Bit 6 = 1
6FA2  D3 01               OUT     (PORT2),A       ;Register-Clock LOW-HIGH
6FA4  CB B7               RES     6,A             ;Bit 6 = 0
6FA6  D3 01               OUT     (PORT2),A       ;Register-Clock HIGH-LOW
6FA8  3A 6F01'            LD      A,(P1BYTE)      ;EPROM-Adresse LOW-Byte
6FAB  D3 00               OUT     (PORT1),A       ;an Port1 ausgeben
6FAD  3A 6F00'            LD      A,(P2BYTE)      ;EPROM-Adresse HIGH-Bits
6FB0  CB FF               SET     7,A             ;Bit 7 = 1, Progr.impuls
6FB2  F3                  DI                      ;Interrupt abschalten
6FB3  D3 01               OUT     (PORT2),A       ;Progr.impuls einschalten
6FB5  CD 6FBE'            CALL    T50MS           ;Verzögerung 50 Millisekunden
6FB8  CB BF               RES     7,A             ;Bit 7 = 0
6FBA  D3 01               OUT     (PORT2),A       ;Progr.impuls abschalten
6FBC  FB                  EI                      ;Interrupt ermöglichen
6FBD  C9                  RET                     ;Rücksprung

                          ; T50MS ist eine Verzögerungsschleife
                          ;       für 50 Millisekunden (4 MHZ-Takt!)

6FBE  F5          T50MS:  PUSH    AF              ;2.75 Mikrosekunden
6FBF  C5                  PUSH    BC              ;2.75
6FC0  01 2530'            LD      BC,2530H        ;2.5
6FC3  0B          LOOP:   DEC     BC              ;1.5  / 5.25 us * 9520
6FC4  78                  LD      A,B             ;1    /
6FC5  B1                  OR      C               ;1    /
6FC6  20 FB               JR      NZ,LOOP         ;1.75 /
6FC8  C1                  POP     BC              ;2.5
6FC9  F1                  POP     AF              ;2.5
6FCA  C9                  RET                     ;2.5

                          ; NEXTAD erhöht den Inhalt der EPROM-Adress-
                          ;       speicher P2BYTE/P1BYTE um eins
                          ;       erhöht HL und dekrementiert BC

6FCB  E5          NEXTAD: PUSH    HL              ;EPROM-Adresse erhöhen
6FCC  21 6F01'            LD      HL,P1BYTE       ;Adresse LOW-Byte erhöhen
6FCF  34                  INC     (HL)
6FD0  AF                  XOR     A
6FD1  BE                  CP      (HL)            ;wenn 00
6FD2  20 04               JR      NZ,NEXT1
6FD4  21 6F00'            LD      HL,P2BYTE       ;dann Adresse HIGH-Bits erhöhen
6FD7  34                  INC     (HL)
6FD8  E1          NEXT1:  POP     HL
6FD9  23                  INC     HL              ;Speicherzeiger erhöhen
6FDA  0B                  DEC     BC              ;Bytezähler dekrementieren
6FDB  C9                  RET                     ;Rücksprung
                  END
Symbols:
ANFANG  6F02    FAL123  6F08    FEHLER  6F06    INIT    6F2F
IPORT   0000    LAENGE  6F04    LESEN   6F7D    LOOP    6FC3
NEXT1   6FD8    NEXTAD  6FCB    P1BYTE  6F01    P2BYTE  6F00
PORT1   0000    PORT2   0001    PROG1   6F48    PROGR   6F3E
PSTART  6F0A    READBY  6F8A    RUECK   6F27    T50MS   6FBE
VER1    6F65    VER2    6F74    VERIF   6F5E    WRITBY  6F9A

No error(s)
```

Listing 2 Assemblerprogramm

* * *

```
6F7D  CD 6F8A'    LESEN:  CALL    READBY          ;Lesen - ein Byte einlesen
6F80  77                  LD      (HL),A          ;und speichern
6F81  CD 6FCB'            CALL    NEXTAD          ;Adresszeiger erhöhen
6F84  78                  LD      A,B
6F85  B1                  OR      C
6F86  20 F5               JR      NZ,LESEN        ;fertig? - nein, nochmals
6F88  18 9D               JR      RUECK           ;ja - Rücksprung

                          ; READBY liest ein Byte des EPROMs (Adresse in
                          ;       P2BYTE/P1BYTE) in den Accu ein

6F8A  3A 6F01'    READBY: LD      A,(P1BYTE)      ;Adresse ausgeben
6F8D  D3 00               OUT     (PORT1),A
6F8F  3A 6F00'            LD      A,(P2BYTE)
6F92  D3 01               OUT     (PORT2),A
6F94  00                  NOP                     ;Verzögerung
6F95  00                  NOP
6F96  00                  NOP
6F97  DB 00               IN      A,(IPORT)       ;Byte einlesen
6F99  C9                  RET                     ;Rücksprung
                          ; WRITBY schreibt das Byte in (HL) in das EPROM
                          ;       (Adresse in P2BYTE/P1BYTE)

6F9A  7E          WRITBY: LD      A,(HL)          ;Byte nach Accu
6F9B  D3 00               OUT     (PORT1),A       ;Ausgabe an Port1
6F9D  3A 6F00'            LD      A,(P2BYTE)
6FA0  CB F7               SET     6,A             ;Bit 6 = 1
6FA2  D3 01               OUT     (PORT2),A       ;Register-Clock LOW-HIGH
6FA4  CB B7               RES     6,A             ;Bit 6 = 0
6FA6  D3 01               OUT     (PORT2),A       ;Register-Clock HIGH-LOW
6FA8  3A 6F01'            LD      A,(P1BYTE)      ;EPROM-Adresse LOW-Byte
6FAB  D3 00               OUT     (PORT1),A       ;an Port1 ausgeben
6FAD  3A 6F00'            LD      A,(P2BYTE)      ;EPROM-Adresse HIGH-Bits
6FB0  CB FF               SET     7,A             ;Bit 7 = 1, Progr.impuls
6FB2  F3                  DI                      ;Interrupt abschalten
6FB3  D3 01               OUT     (PORT2),A       ;Progr.impuls einschalten
6FB5  CD 6FBE'            CALL    T50MS           ;Verzögerung 50 Millisekunden
6FB8  CB BF               RES     7,A             ;Bit 7 = 0
6FBA  D3 01               OUT     (PORT2),A       ;Progr.impuls abschalten
6FBC  FB                  EI                      ;Interrupt ermöglichen
6FBD  C9                  RET                     ;Rücksprung

                          ; T50MS ist eine Verzögerungsschleife
                          ;       für 50 Millisekunden (4 MHZ-Takt!)

6FBE  F5          T50MS:  PUSH    AF              ;2.75 Mikrosekunden
6FBF  C5                  PUSH    BC              ;2.75
6FC0  01 2530'            LD      BC,2530H        ;2.5
6FC3  0B          LOOP:   DEC     BC              ;1.5  / 5.25 us * 9520
6FC4  78                  LD      A,B             ;1    /
6FC5  B1                  OR      C               ;1    /
6FC6  20 FB               JR      NZ,LOOP         ;1.75 /
6FC8  C1                  POP     BC              ;2.5
6FC9  F1                  POP     AF              ;2.5
6FCA  C9                  RET                     ;2.5

                          ; NEXTAD erhöht den Inhalt der EPROM-Adress-
                          ;       speicher P2BYTE/P1BYTE um eins
                          ;       erhöht HL und dekrementiert BC

6FCB  E5          NEXTAD: PUSH    HL              ;EPROM-Adresse erhöhen
6FCC  21 6F01'            LD      HL,P1BYTE       ;Adresse LOW-Byte erhöhen
6FCF  34                  INC     (HL)
6FD0  AF                  XOR     A
6FD1  BE                  CP      (HL)            ;wenn 00
6FD2  20 04               JR      NZ,NEXT1
6FD4  21 6F00'            LD      HL,P2BYTE       ;dann Adresse HIGH-Bits erhöhen
6FD7  34                  INC     (HL)
6FD8  E1          NEXT1:  POP     HL
6FD9  23                  INC     HL              ;Speicherzeiger erhöhen
6FDA  0B                  DEC     BC              ;Bytezähler dekrementieren
6FDB  C9                  RET                     ;Rücksprung
                  END
Symbols:
ANFANG  6F02    FAL123  6F08    FEHLER  6F06    INIT    6F2F
IPORT   0000    LAENGE  6F04    LESEN   6F7D    LOOP    6FC3
NEXT1   6FD8    NEXTAD  6FCB    P1BYTE  6F01    P2BYTE  6F00
PORT1   0000    PORT2   0001    PROG1   6F48    PROGR   6F3E
PSTART  6F0A    READBY  6F8A    RUECK   6F27    T50MS   6FBE
VER1    6F65    VER2    6F74    VERIF   6F5E    WRITBY  6F9A

No error(s)
```

Listing 2 Assemblerprogramm

* * *

```
6F7D  CD 6F8A'    LESEN:  CALL    READBY          ;Lesen - ein Byte einlesen
6F80  77                  LD      (HL),A          ;und speichern
6F81  CD 6FCB'            CALL    NEXTAD          ;Adresszeiger erhöhen
6F84  78                  LD      A,B
6F85  B1                  OR      C
6F86  20 F5               JR      NZ,LESEN        ;fertig? - nein, nochmals
6F88  18 9D               JR      RUECK           ;ja - Rücksprung

                          ; READBY liest ein Byte des EPROMs (Adresse in
                          ;       P2BYTE/P1BYTE) in den Accu ein

6F8A  3A 6F01'    READBY: LD      A,(P1BYTE)      ;Adresse ausgeben
6F8D  D3 00               OUT     (PORT1),A
6F8F  3A 6F00'            LD      A,(P2BYTE)
6F92  D3 01               OUT     (PORT2),A
6F94  00                  NOP                     ;Verzögerung
6F95  00                  NOP
6F96  00                  NOP
6F97  DB 00               IN      A,(IPORT)       ;Byte einlesen
6F99  C9                  RET                     ;Rücksprung
                          ; WRITBY schreibt das Byte in (HL) in das EPROM
                          ;       (Adresse in P2BYTE/P1BYTE)

6F9A  7E          WRITBY: LD      A,(HL)          ;Byte nach Accu
6F9B  D3 00               OUT     (PORT1),A       ;Ausgabe an Port1
6F9D  3A 6F00'            LD      A,(P2BYTE)
6FA0  CB F7               SET     6,A             ;Bit 6 = 1
6FA2  D3 01               OUT     (PORT2),A       ;Register-Clock LOW-HIGH
6FA4  CB B7               RES     6,A             ;Bit 6 = 0
6FA6  D3 01               OUT     (PORT2),A       ;Register-Clock HIGH-LOW
6FA8  3A 6F01'            LD      A,(P1BYTE)      ;EPROM-Adresse LOW-Byte
6FAB  D3 00               OUT     (PORT1),A       ;an Port1 ausgeben
6FAD  3A 6F00'            LD      A,(P2BYTE)      ;EPROM-Adresse HIGH-Bits
6FB0  CB FF               SET     7,A             ;Bit 7 = 1, Progr.impuls
6FB2  F3                  DI                      ;Interrupt abschalten
6FB3  D3 01               OUT     (PORT2),A       ;Progr.impuls einschalten
6FB5  CD 6FBE'            CALL    T50MS           ;Verzögerung 50 Millisekunden
6FB8  CB BF               RES     7,A             ;Bit 7 = 0
6FBA  D3 01               OUT     (PORT2),A       ;Progr.impuls abschalten
6FBC  FB                  EI                      ;Interrupt ermöglichen
6FBD  C9                  RET                     ;Rücksprung

                          ; T50MS ist eine Verzögerungsschleife
                          ;       für 50 Millisekunden (4 MHZ-Takt!)

6FBE  F5          T50MS:  PUSH    AF              ;2.75 Mikrosekunden
6FBF  C5                  PUSH    BC              ;2.75
6FC0  01 2530'            LD      BC,2530H        ;2.5
6FC3  0B          LOOP:   DEC     BC              ;1.5  / 5.25 us * 9520
6FC4  78                  LD      A,B             ;1    /
6FC5  B1                  OR      C               ;1    /
6FC6  20 FB               JR      NZ,LOOP         ;1.75 /
6FC8  C1                  POP     BC              ;2.5
6FC9  F1                  POP     AF              ;2.5
6FCA  C9                  RET                     ;2.5

                          ; NEXTAD erhöht den Inhalt der EPROM-Adress-
                          ;       speicher P2BYTE/P1BYTE um eins
                          ;       erhöht HL und dekrementiert BC

6FCB  E5          NEXTAD: PUSH    HL              ;EPROM-Adresse erhöhen
6FCC  21 6F01'            LD      HL,P1BYTE       ;Adresse LOW-Byte erhöhen
6FCF  34                  INC     (HL)
6FD0  AF                  XOR     A
6FD1  BE                  CP      (HL)            ;wenn 00
6FD2  20 04               JR      NZ,NEXT1
6FD4  21 6F00'            LD      HL,P2BYTE       ;dann Adresse HIGH-Bits erhöhen
6FD7  34                  INC     (HL)
6FD8  E1          NEXT1:  POP     HL
6FD9  23                  INC     HL              ;Speicherzeiger erhöhen
6FDA  0B                  DEC     BC              ;Bytezähler dekrementieren
6FDB  C9                  RET                     ;Rücksprung
                  END
Symbols:
ANFANG  6F02    FAL123  6F08    FEHLER  6F06    INIT    6F2F
IPORT   0000    LAENGE  6F04    LESEN   6F7D    LOOP    6FC3
NEXT1   6FD8    NEXTAD  6FCB    P1BYTE  6F01    P2BYTE  6F00
PORT1   0000    PORT2   0001    PROG1   6F48    PROGR   6F3E
PSTART  6F0A    READBY  6F8A    RUECK   6F27    T50MS   6FBE
VER1    6F65    VER2    6F74    VERIF   6F5E    WRITBY  6F9A

No error(s)
```

Listing 2 Assemblerprogramm

* * *