Davor ist kein EPROM sicher

Programmiergerät für den Apple-II wird per Software eingestellt

Die Probleme, denen sich der Entwickler eines universellen EPROM-Programmiergerätes gegenübersieht, sind immer dieselben: uneinheitliche Pinbelegung der verschiedenen EPROM-Typen, unterschiedliche Programmierspannungen, zu wenig User-Port-Ausgänge und keine Programmierspannung am Computer. Für Apple-II-Besitzer ist damit jetzt Schluß.

Die verschiedenen Pinbelegungen werden in der Regel mit mehreren Umschaltern bewältigt; User-Port-Bits werden z. B. mit Adreßzählern eingespart; für die Programmierspannung gibt es oft nur einen Lötnagel – mit der kleingedruckten Aufforderung „+25 V".
Der Apple-II stellt an Ein-/Ausgabeleitungen zunächst nur den ‚Spiele-Port' mit vier TTL-Ausgängen, drei TTL-Eingängen und einer Spannung von +5 V zur Verfügung. Aber das reicht auch aus: Die Schnittstelle für die gesamte Versorgung des Programmiergerätes konnte auf sechs Anschlüsse reduziert werden: drei Portausgänge, ein Eingang, +5 V und Masse. Umschalter gibt es gar keinen, aber dennoch sind alle (üblichen) EPROM-Typen programmierbar (zumindest 2716, 2732, 2732A, 2764, 27128, 2758, 2508, 2516, 2532, 2564 sowie die entsprechenden C-Typen). Die für das Programmieren erforderlichen 25 V werden von der Schaltung selbst erzeugt. Als Spannungswandler und -regler dient ein NE555.

Timer NE555 als Spannungswandler

Die Idee, den Timer als Spannungswandler zu verwenden, entstand, als der in [1] benutzte TL497 nirgends erhältlich war. Die Schaltung ist so einfach und funktioniert so hervorragend, daß sie eigentlich längst als Standardapplikation bekannt sein müßte (*Bild 1*).
Der NE555 arbeitet als Multivibrator. Die beim Abschalten der Spule am Kollektor des BFX34 entstehenden Hochspannungsspitzen laden über D10 den Elko C2 auf. Übersteigt die Spannung das gewünschte Maß, wird der BC107 leitend. Er verringert dann über Pin 5 die Breite der Ausgangsimpulse und begrenzt damit den maximalen Spulenstrom sowie die Höhe der Induktionsspannungsspitzen so lange, bis sich ein Gleichgewicht einstellt. Die Ausgangsspannung ist an P1 einstellbar und kann bei der angegebenen Dimensionierung bis über 30 V (unbelastet) betragen. Höhere Ausgangsspannungen und -ströme können (falls für andere Anwendungen erwünscht) durch Vergrößern des 10-kΩ-Widerstandes R18 erreicht werden. Die Ausgangsimpulsbreite und der maximale Spulenstrom wachsen dann an, was eine entsprechend höhere Induktionsspannung ergibt. Der BFX34 ist dann gegebenenfalls zu kühlen.
Für das Programmiergerät wird eine Ausgangsspannung von +26 V (am Punkt ‚U') fest eingestellt, die bei Belastung mit 30 mA nicht wesentlich absinken darf.

Statt teurer Ports: ein Schieberegister

Den Kern des Programmiergerätes bildet ein 32 Bit langes Schieberegister mit parallelem Ausgang. Über einen seriellen Eingang werden Daten-, Adreß- und Steuerbits in geeigneter Reihenfolge (eben je nach Pinbelegung des EPROMs) eingeschoben und liegen nach einem Strobeimpuls an den 28 Pins der EPROM-Fassung an. Die Pinbelegung der verschiedenen EPROM-Typen wird also zur reinen Softwarefrage!

Zum Lesen überträgt ein weiteres 8-Bit-Schieberegister mit parallelem Eingangsregister (4021) die Daten bitweise zum seriellen Ausgang.
Die letzten sieben der 32 Bits steuern die Erzeugung und Verteilung der Programmierspannung, die Tristate-Ausgänge des Datenwortregisters IC 2 (müssen beim Auslesen des EPROMs hochohmig sein) und eine Kontroll-LED (D1).
Das Platinenlayout (*Bild 2*) erlaubt auch den getrennten Aufbau von Spannungswandler und Programmiergerät. Wer will, kann auf den Wandler ganz verzichten und die 26 V an der angegebenen Stelle (Lötpunkt U) einspeisen (*Bild 3*). Die rote LED (D2) signalisiert, daß die Programmierspannung anliegt und erlaubt gleichzeitig eine gewisse Spannungskontrolle (erlischt unterhalb 24 V).

Der Rest ist Software

Die Software besteht aus einem Basic-Rahmenprogramm (*Bild 4*) für den Benutzerdialog, das nach Angabe des EPROM-Typs jeweils ein typenspezifisches Maschinenprogramm zur Steuerung der Hardware zulädt.
Jedes Maschinenprogramm (*Bild 5*) umfaßt sechs Hauptroutinen (Initialisierung, Programmieren, Lesen, Vergleichen, Löschtest, Bildschirmausgabe) die von Basic aus aufgerufen werden. Der standardisierte Programm-Kopf enthält jeweils alle Einsprungadressen sowie die Spezifikationen des zugehörigen EPROM-Typs. Die Anpassung an verschiedene EPROM-Typen geschieht also im wesentlichen durch Änderungen der Parameter im Programmkopf.
Nach dem Laden des Maschinenprogramms wird von Basic aus zunächst die Initialisierungsroutine aufgerufen, die einen kurzen Funktionstest durchführt und das Programmiergerät in einen definierten Anfangszustand versetzt (Programmierspannung aus, alle Pins der EPROM-Fassung an Masse, grüne Kontroll-LED an). Diese Prozedur wird auch nach jeder Operation (Programmieren, Lesen etc.) wiederholt, so daß das EPROM stets gefahrlos gewechselt werden kann. Der Standardzustand wird jeweils durch die grüne LED signalisiert.
Das Unterprogramm zur Programmierung ist mit Abstand am längsten, es wurde aber viel Wert auf Sicherheit und Redundanz gelegt, um Bedienungs- und Programmierfehlern weitgehend vorzubeugen: Vor der Programmierung wird jedes Byte auf $FF getestet. Ist es nicht gelöscht, so erfolgt Rückfrage beim Benutzer, ob dennoch programmiert werden soll, zusammen mit der Angabe, ob

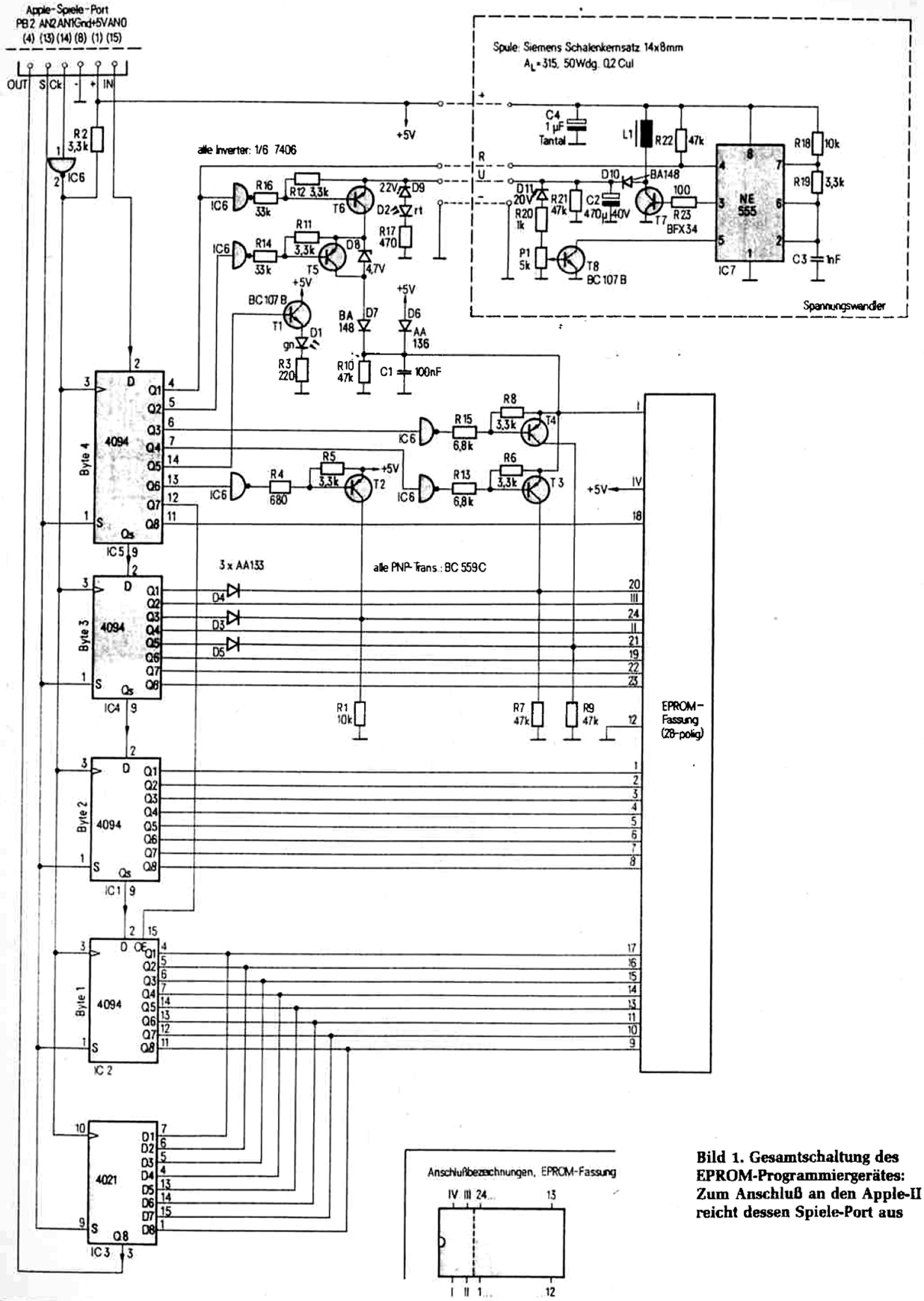

Bild 1. Gesamtschaltung des EPROM-Programmiergerätes: Zum Anschluß an den Apple-II reicht dessen Spiele-Port aus

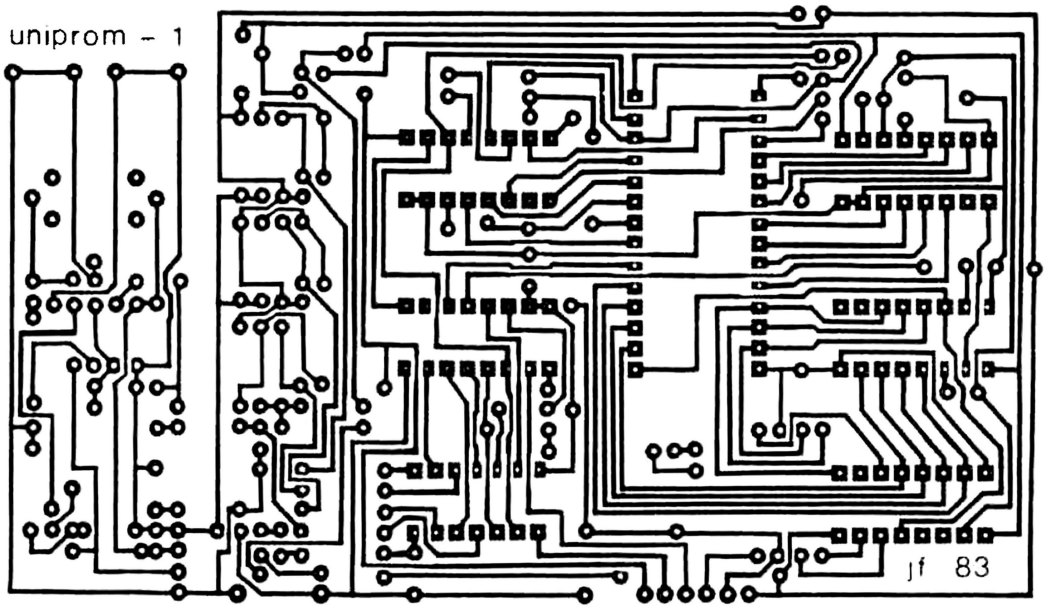

Bild 2. Layout der Platine;

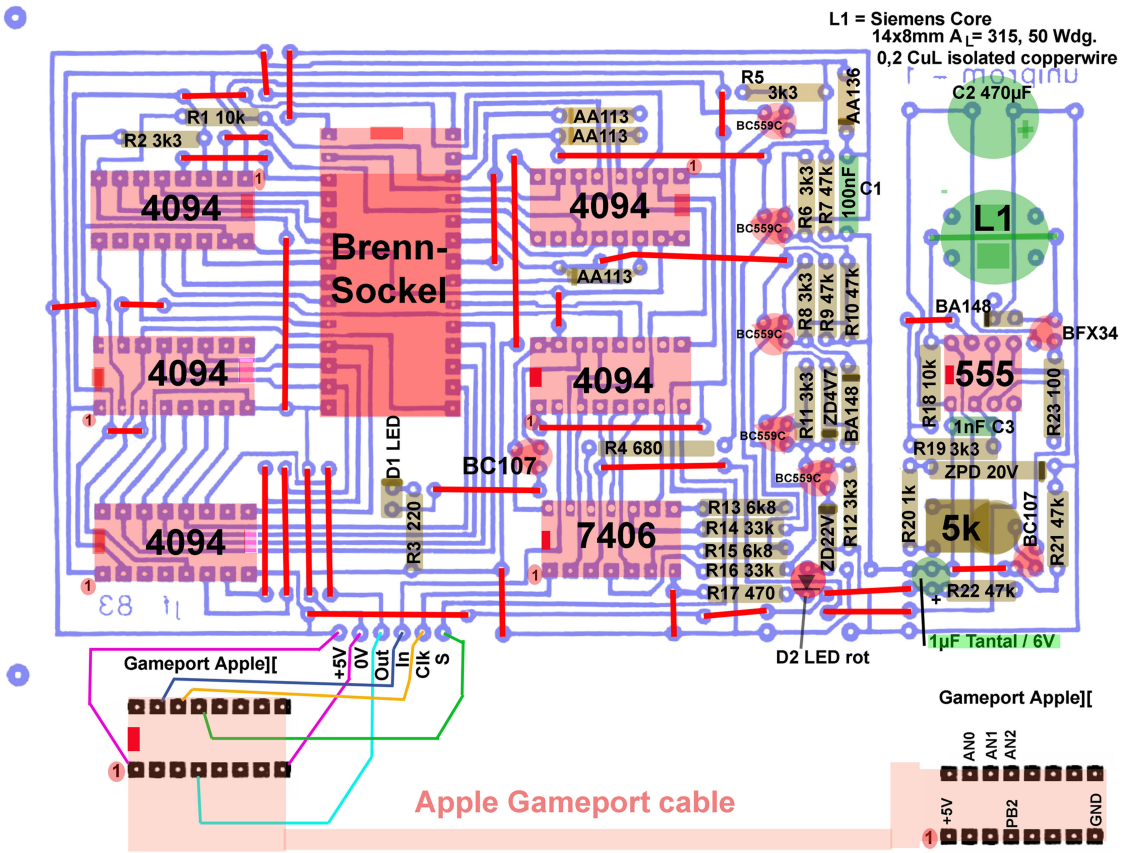

Bild 3. Bestückungsplan zu Bild 2

dies noch (fehlerfrei) möglich ist. Nach dem Programmieren wird jedes Byte sofort wieder gelesen und verglichen; beim Auftreten eines Fehlers erfolgen wiederum Rückmeldung und Anfrage, ob die Programmierung fortgesetzt werden soll. Das Unterprogramm zu Bildschirmausgabe ist vielleicht weniger üblich, aber praktisch: Es erlaubt ein Betrachten des EPROM-Inhaltes, ohne daß RAM-Speicherplatz zum Einlesen benötigt wird.

Für alle EPROM-Typen geeignet

Das Maschinenprogramm ist so aufgebaut, daß es sehr leicht an verschiedene EPROM-Typen angepaßt werden kann. Am Beispiel des 2716, dessen Steuerprogramm in Bild 5 abgedruckt ist, soll gezeigt werden, wie man dabei vorgeht: Zunächst stellt man mit Hilfe von Bild 1 und dem Datenblatt des EPROMs eine *Tabelle* auf. Aus ihr gehen alle Statusbits für die Betriebszustände ‚Read', ‚Verify', ‚Program' und ‚Program-Inhibit' hervor (mit ‚Program-Inhibit' ist hier der Zustand unmittelbar vor bzw. nach der aktiven Programmierphase gemeint). Man beachte, daß (etwa beim 2716) durch die Aktivierung von Bit 30 (V_{pp} an Pin 21) und Bit 27 (+5 V an Pin 24) Bit 20 und Bit 22 funktionslos werden (die Dioden D5 und D3 sperren) oder daß die Verbindung ‚V_{pp} an Pin 21' (Bit 30 = 1) auch im Read-Modus bestehen muß, damit Pin 21 an +5 V liegt. Man sollte also schon das Schaltbild zu Rate ziehen, auch weil so die genaue Funktion der Steuerbits (Bit 26...32) am leichtesten klar wird.

Die Schieberegister-Bytes 4 und 3 mit den Statusbits für ‚Read', ‚Verify', ‚Program' und ‚Program-Inhibit' werden dann in dieser Reihenfolge im Programmkopf eingetragen (ab Adresse $8018); Adreßbit-Stellen in Byte 3 bleiben dabei 0, die Adresse wird vom Programm per ODER-Verknüpfung eingesetzt. Das dafür zuständige Unterprogramm ‚SETADR' ist der zweite typenspezifische Teil der Software und muß

Tabelle: Statusbits für ‚Read', ‚Verify', ‚Program', ‚Program-Inhibit' bei verschiedenen EPROM-Typen

Schieberег.-Byte	4								3							2		1
Schieberег.-Bit	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16...9	8...1
Steuerfunktion bzw. Sockelanschluß	V_{pp} an/ aus	V_{pp}: 25 V/ 21 V	V_{pp} ↓ Pin21	V_{pp} ↓ Pin20	gn. LED	+5 V ↓ Pin24	OE für Byte1	(18)	(20)	(III)	(24)	(II)	(21)	(19)	(22)	(23)	(1)..,(8)	(17)...(13), (11)...(9)
Pinfunktion 2716	–	–	–	–	–	–	–	$\overline{CE}$ (Pgm)	$\overline{OE}$	–	V_{cc}	–	V_{pp}	A10	A9	A8	A7...A0	D7...D0
Read	0	0	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Verify	1	1	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Program	1	1	1	0	0	1	1	1	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	1	0	0	1	1	0	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2732	–	–	–	–	–	–	–	$\overline{CE}$ ($\overline{Pgm}$)	$\overline{OE}$/ V_{pp}	–	V_{cc}	–	A11	A10	A9	A8	A7...A0	D7...D0
Verify	1	1	0	0	0	1	0	0	0	X	X	X	A11	A10	A9	A8	A7...A0	X...X
Program	1	1	0	1	0	1	1	0	X	X	X	X	A11	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	0	1	0	1	1	1	X	X	X	X	A11	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2732A	–	–	–	–	–	–	–	$\overline{CE}$ ($\overline{Pgm}$)	$\overline{OE}$/ V_{pp}	–	V_{cc}	–	A11	A10	A9	A8	A7...A0	D7...D0
Read	0	0	0	0	0	1	0	0	0	X	X	X	A11	A10	A9	A8	A7...A0	X...X
Verify	1	0	0	0	0	1	0	0	0	X	X	X	A11	A10	A9	A8	A7...A0	X...X
Program	1	0	0	1	0	1	1	0	X	X	X	X	A11	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	0	0	1	0	1	1	1	X	X	X	X	A11	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2764	–	–	–	–	–	–	–	$\overline{CE}$	$\overline{OE}$	$\overline{Pgm}$	–	A12	A11	A10	A9	A8	A7...A0	D7...D0
	0	0	0	0	0	0	0	0	0	1	X	A12	A11	A10	A9	A8	A7...A0	X...X
Verify	1	0	0	0	0	0	0	0	0	1	X	A12	A11	A10	A9	A8	A7...A0	X...X
Program	1	0	0	0	0	0	1	0	1	0	X	A12	A11	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	0	0	0	0	0	1	0	1	1	X	A12	A11	A10	A9	A8	A7...A0	D7...D0

Schieberegister-Byte	4								3							2		1
Schieberegister-Bit	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16...9	8...1
Steuerfunktion bzw. Sockelanschluß	V_{pp} an/ aus	V_{pp}: 25 V/ 21 V	V_{pp} ↓ Pin21	V_{pp} ↓ Pin20	gn. LED	+5 V ↓ Pin24	OE für Byte1	(18)	(20)	(III)	(24)	(II)	(21)	(19)	(22)	(23)	(1)...(8)	(17)...(13), (11)...(9)
Pinfunktion 2732	–	–	–	–	–	–	–	$\overline{CE}$	$\overline{OE}$	$\overline{Pgm}$	A13	A12	A11	A10	A9	A8	A7...A0	D7...D0
Read	0	0	0	0	0	0	0	0	0	1	A13	A12	A11	A10	A9	A8	A7...A0	X...X
Verify	1	0	0	0	0	0	0	0	0	1	A13	A12	A11	A10	A9	A8	A7...A0	X...X
Program	1	0	0	0	0	0	1	0	1	0	A13	A12	A11	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	0	0	0	0	0	1	0	1	1	A13	A12	A11	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2758	–	–	–	–	–	–	–	$\overline{CE}$ (Pgm)	$\overline{OE}$	–	V_{cc}	–	V_{pp}	A10 [1]	A9	A8	A7...A0	D7...D0
Read	0	0	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Verify	1	1	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Program	1	1	1	0	0	1	1	1	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	1	0	0	1	1	0	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2508	–	–	–	–	–	–	–	Pgm	$\overline{CS}$	–	V_{cc}	–	V_{pp}	–	A9	A8	A7...A0	D7...D0
Read	0	0	1	0	0	1	0	0	0	X	X	X	X	X	A9	A8	A7...A0	X...X
Verify	1	1	1	0	0	1	0	0	0	X	X	X	X	X	A9	A8	A7...A0	X...X
Program	1	1	1	0	0	1	1	1	1	X	X	X	X	X	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	1	0	0	1	1	0	1	X	X	X	X	X	A9	A8	A7...A0	D7...D0
Pinfunktion 2516	–	–	–	–	–	–	–	Pgm	$\overline{CS}$	–	V_{cc}	–	V_{pp}	A10	A9	A8	A7...A0	D7...D0
Read	0	0	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Verify	1	1	1	0	0	1	0	0	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Program	1	1	1	0	0	1	1	1	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	1	0	0	1	1	0	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2532	–	–	–	–	–	–	–	A11	($\overline{CS}$) $\overline{Pgm}$	–	V_{cc}	–	V_{pp}	A10	A9	A8	A7...A0	D7...D0
Read	0	0	1	0	0	1	0	A11	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Verify	0	0	1	0	0	1	0	A11	0	X	X	X	X	A10	A9	A8	A7...A0	X...X
Program	1	1	1	0	0	1	1	A11	0	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	1	0	0	1	1	A11	1	X	X	X	X	A10	A9	A8	A7...A0	D7...D0
Pinfunktion 2564	–	–	–	–	–	–	–	A11	$\overline{Pgm}$	$\overline{CS}$	V_{cc}	$\overline{CS}$	A12	A10	A9	A8	A7...A0	D7...D0
Read	0	0	0	0	0	1	0	A11	0	0	X	0	A12	A10	A9	A8	A7...A0	D7...D0
Verify	0	0	0	0	0	1	0	A11	0	0	X	0	A12	A10	A9	A8	A7...A0	D7...D0
Program	1	1	0	0	0	1	1	A11	0	0	X	0	A12	A10	A9	A8	A7...A0	D7...D0
Program-Inhibit	1	1	0	0	0	1	1	A11	1	0	X	0	A12	A10	A9	A8	A7...A0	D7...D0

[1]) A10 = 0 für 2758A, A10 = 1 für 2758B

der Pinbelegung des jeweiligen EPROMs angepaßt werden. Es befindet sich ganz am Ende des Maschinenprogramms, damit sich bei Änderungen keine Sprungadressen verschieben und Modifikationen auch ohne Assembler leicht durchführbar sind. Auf der Apple-Sammeldiskette 7 (AP007) unseres Software-Service sind Maschinenprogramme für alle EPROM-Typen.
Die letzten Bytes des Programmkopfes enthalten noch die Programmierzeit pro Byte in ms (in der Regel 50), die höchste Adresse des EPROMs ($07FF für 2716) und die Typenbezeichnung als ASCII-String mit einem Punkt am Ende. Nachdem auch diese Bytes (Adr. $8020-802F) dem Typ entsprechend eingestellt sind, ist die Anpassung der Software beendet. Das Maschinenprogramm wird unter dem Namen EPROM-XXXX (XXXX = Typenbezeichnung wie im Programmkopf, ohne Punkt) auf der Diskette abgelegt. Das Basic-Programm ruft es nach Angabe des gewünschten Typs unter diesem Namen auf, übernimmt die Startadressen der sechs Hauptroutinen, die Adressen für die Parameterübergabe, die höchste EPROM-Adresse und die Typenbezeichnung aus dem Programmkopf und verzweigt bei Anwahl einer Funktion in das entsprechende Unterprogramm.

Basic sorgt für Komfort

Das Basic-Programm ist zum großen Teil selbsterklärend, so daß auf eine ausführliche Beschreibung verzichtet werden kann. Die Auswahl der sieben Funktionen erfolgt menügesteuert, falsche oder sinnlose Eingaben werden nirgends akzeptiert, gegebenenfalls wird auf Fehler hingewiesen.
Der RAM-Bereich von $4000 bis $7FFF wird für die Aufnahme von EPROM-Daten freigehalten. Zur Eingabe von Daten vor der Programmierung bzw. zur Ausgabe oder Modifikation nach ‚Read' kann vom Programm aus vorübergehend in den Apple-II-Monitor verzweigt werden, wobei aber die nun ungeschützten Bereiche $0000...3FFF (Basic-Programm und -Variablen) und $8000...BFFF (Maschinenprogramm und DOS) tabu sind! Als Basisadresse für den Kopf des Maschinenprogramms wird $8000 erwartet; bei einer Änderung ist leldiglich die Variable ‚HRAM' am Anfang des Basic-Programms neu zu definieren.

Aufbau und Inbetriebnahme

Die Bestückung der Platine ist sicher problemlos; der Spannungswandler kann auch getrennt aufgebaut oder weggelassen werden.
Vor dem Anschluß an den steuernden Rechner sollte man sich gründlich vergewissern, daß die Programmierspannung nicht über Lötbrücken und andere verbotene Pfade zur Schnittstelle gelangen kann.
Der Abgleich der Programmierspannung auf 26 V am Punkt U (Lötnagel bei der roten LED) erfolgt am besten vor dem Einsetzen der Gatter- und CMOS-ICs (der Wandler ist dann automatisch aktiv); gleichzeitig kontrolliert man, daß an den Sockeln der Schieberegister nirgends Hochspannung anliegt und die Schaltfunktionen der 6 Transistoren.
Nach dem Start des Basic-Programms und der Angabe eines EPROM-Typs führt die Initialisierungsroutine des entsprechenden Maschinenprogramms automatisch einen kurzen Funktionstest durch. Danach sollte der Spannungswandler abgeschaltet sein, alle Pins der EPROM-Fassung (außer I und IV) an Masse liegen und die grüne LED leuchten. Ein EPROM kann nun eingesetzt werden.
Die Konzeption des Programmiergerätes ist so flexibel, daß ein Anschluß an andere Rechner keine Schwierigkeiten bereitet (die Anforderungen an die Schnittstelle sind wirklich minimal) und daß es mit geeigneter Steuersoftware auch die Bewältigung exotischer Pinbelegungen oder den Einsatz spezieller Programmiermethoden erlaubt.

Literatur

[1] *Gößler, Reinhard:* PROMmer und Drucker für Moppel. ELO 1983, Heft 1, Seite 22...28.
[2] *Konz, Michael:* CBM-EPROM-Programmer. mc 1982, Heft 3, Seite 72...76.
[3] *Wolf, Peter:* EPROM-Programmiergerät. mc 1983, Heft 7, Seite 78...82.
[4] *Minnich, Horst:* AIM programmiert EPROMs. mc 1982, Heft 8, Seite 70...72.

Mainprogramm in Applesoft Basic

```
1000  REM ************************************************************************
1010  REM *                                                                      *
1020  REM *     BASIC-RAHMENPROGRAMM FUER EPROM   PROGRAMMIERGERAET UNIPROM-1    *
1030  REM *                                                                      *
1040  REM *                                                                      *
1050  REM *                                                                      *
1060  REM ************************************************************************
1070  REM
1080  REM
1090  REM  LAEDT NACH ANGABE DES EPROM-TYPS 'XXXX' TYPENSPEZ. MASCHINENPROGRAMM
1100  REM  'EPROM-XXXX' ZUR ANSTEUERUNG DER HARDWARE ZU:
1110  REM
1120  REM  ERWARTETE STARTADRESSE:  HRAM+1 = 32768 - $8000
1130  REM
1140  REM  ALS INHALT DER ERSTEN BYTES ($80..) WIRD ERWARTET:
1150  REM
1160  REM  00,01  - STARTADRESSE INITIALISIERUNGSROUTINE
1170  REM  02,03  - STARTADRESSE PGM.-ROUTINE  (PROGRAMMIERUNG)
1180  REM  04,05  - STARTADRESSE READ-ROUTINE  (EPROM AUSLESEN)
1190  REM  06,07  - STARTADRESSE VERIFY-ROUTINE  (VGL. MIT APPLE-RAM)
1200  REM  08,09  - STARTADRESSE CLEAR?-ROUTINE  (LOESCHTEST)
1210  REM  0A,0B  - STARTADRESSE DISPL.-ROUTINE  (BILDSCHIRMAUSGABE)
1220  REM
1230  REM  0C,0D  - ZEIGER AUF ERROR BYTE
1240  REM  0E,0F  - ZEIGER AUF EPROM BYTE  (BEI FEHLERMELDUNG)
1250  REM  10,11  - ZEIGER AUF COMPUTER BYTE  (BEI FEHLERMELDUNG)
1260  REM  12,13  - ZEIGER AUF AKTUELLE ADRESSE, EPROM
1270  REM  14,15  - ZEIGER AUF ZIELADRESSE, EPROM  (LETZTES BYTE BEI PGM ETC.)
1280  REM  16,17  - ZEIGER AUF AKTUELLE ADRESSE, COMPUTER
1290  REM
1300  REM  21,22  - HOECHSTE EPROM ADRESSE
1310  REM  23-2F  - EPROM TYP  (ASCII-STRING MIT "." AM ENDE !)
1320  REM
1330  REM
1340  LET OLDHIMEM =  PEEK (115) + 256 *  PEEK (116): REM  ALTER HIMEM-WERT
1350  LET HRAM = 32767: REM  MASCHINENPGM. AB HARAM+1  (= $8000)
1360  LET LRAM = 16384: REM  BIS LRAM : BASIC-PROGRAMM UND -VARIABLEN
1370  REM LRAM...HRAM : FREI FUER EPROM-DATEN
1380  HIMEM: LRAM - 1
1390  REM
1400  REM  ASCII-STEUERZEICHEN
1410  REM  --------------------
1420  LET D$ =  CHR$ (4):BELL$ =  CHR$ (7):BS$ =  CHR$ (8):CR$ =  CHR$ (13)
1430  LET TOP% = 34: REM  TEXTFENSTER, OBERGRENZE
1440  LET TYP$ = "": REM  EPROM-TYP
1450  REM
1460  REM ************************************************************************
1470  REM  M E N U
1480  REM ************************************************************************
1490  REM
1500  POKE TOP%,0: HOME : POKE TOP%,6
1510  INVERSE : PRINT "STEUERPROGRAMM FUER EPROM-1": NORMAL
1520  IF TYP$ = "" THEN 1550
1530  VTAB 4: PRINT "EPROM-TYP : ";: INVERSE : PRINT TYP$: NORMAL
1540  REM  DRUCKT TYPENBEZEICHNUNG AUS MASCHINENPGM., BYTES $8024 ... :
1550  HOME : VTAB 6
1560  PRINT : PRINT "T - TYP ANGEBEN"
1570  PRINT : PRINT "P - PROGRAMMIEREN"
1580  PRINT : PRINT "R - EINLESEN"
1590  PRINT : PRINT "V - VERGLEICHEN"
1600  PRINT : PRINT "C - LOESCHTEST"
1610  PRINT : PRINT "D - BILDSCHIRMAUSGABE"
1620  PRINT : PRINT "M - MONITOR AUFRUF"
1630  PRINT : PRINT "E - PROGRAMMENDE"
1640  VTAB 21: HTAB 27: PRINT "?";: GET B$: PRINT B$;
1650  IF B$ = "T" THEN 1800
1660  IF B$ = "P" THEN 2390
1670  IF B$ = "R" THEN 2730
1680  IF B$ = "V" THEN 2840
1690  IF B$ = "C" THEN 3060
1700  IF B$ = "D" THEN 3280
1710  IF B$ = "M" THEN 3480
1720  IF B$ = "E" THEN 3610
1730  PRINT BELL$;
1740  GOTO 1640
1750  REM
1760  REM ************************************************************************
1770  REM  EPROM-TYP ANGEBEN, MASCHINENPROGRAMM LADEN, INITIALISIERUNG STARTEN
1780  REM ************************************************************************
1790  REM
1800  HOME
1810  INPUT "EPROM-TYP (Z.B. '2716') : ";T$
1820  ONERR  GOTO 2280: REM  FAENGT DOS-ERROR AB, FALLS GEW. MASCHINENPGM. FEHLT
1830  PRINT D$;"BLOAD EPROM-";T$
1840  POKE 216,0: REM  RESET ONERR-FLAG
1850  REM
1860  REM  STARTADRESSEN DER ASSEMBLER-ROUTINEN LADEN
1870  REM  ------------------------------------------
1880  REM  (ALLE DIREKTEN RAM-ADRESSEN SIND IN DIESEM PROGRAMM INTEGER-VARIABLEN)
1890  REM
1900  LET INIT% =  PEEK (HRAM + 1) + 256 *  PEEK (HRAM + 2)  65536
1910  LET PGM% =  PEEK (HRAM + 3) + 256 *  PEEK (HRAM + 4)  65536
1920  LET RD% =  PEEK (HRAM + 5) + 256 *  PEEK (HRAM + 6)  65536
1930  LET VFY% =  PEEK (HRAM + 7) + 256 *  PEEK (HRAM + 8)  65536
1940  LET CVFY% =  PEEK (HRAM + 9) + 256 *  PEEK (HRAM + 10)  65536
```

english menu:

"T - specify eprom typ => XXXX"
"P - program eprom"
"R - read eprom content"
"V - compare epromcontent <>RAM"
"C - Test eprom erased ?"
"D - Display eprom content"
"M - enter Apple II monitor"
"E - End of Program => Exit"

Bild 4. Rahmenprogramm zur Steuerung des Benutzerdialogs:

```
1950  LET DSP% =  PEEK (HRAM + 11) + 256 *  PEEK (HRAM + 12) - 65536
1960  REM
1970  REM  RAM-ADRESSEN ZUR PARAMETERUEBERGABE ZUM/VOM MASCHINENPROGRAMM LADEN
1980  REM ------------------------------------------------------------
1990  REM
2000  LET ERR% =  PEEK (HRAM + 13) + 256 *  PEEK (HRAM + 14): REM  ERROR-BYTE
2010  LET EBYTE% =  PEEK (HRAM + 15) + 256 *  PEEK (HRAM + 16): REM  AKT. EPROM-BYTE
2020  LET CBYTE% =  PEEK (HRAM + 17) + 256 *  PEEK (HRAM + 18): REM  AKT. COMP.-BYTE
2030  LET A1% =  PEEK (HRAM + 19) + 256 *  PEEK (HRAM + 20): REM  AKT ADRESSE, EPROM
2040  LET A2% =  PEEK (HRAM + 21) + 256 *  PEEK (HRAM + 22): REM  ZIELADRESSE, EPROM
2050  LET A3% =  PEEK (HRAM + 23) + 256 *  PEEK (HRAM + 24): REM  AKT.ADR., COMPUTER
2060  REM
2070  REM  EPROM KENNDATEN LADEN
2080  REM ----------------------
2090  REM
2100  LET LIM =  PEEK (HRAM + 34) + 256 *  PEEK (HRAM + 35): REM  MAX. EPROM-ADR.
2110  LET TYP$ = "":I = 36
2120  LET C$ =  CHR$ ( PEEK (HRAM + I))
2130  IF C$ <  > "." THEN TYP$ = TYP$ + C$:I = I + 1: GOTO 2120
2140  REM
2150  REM  PROGRAMMIERGERAET INITIALISIEREN
2160  REM ----------------------------------
2170  CALL INIT%
2180  VTAB 10
2190  IF  PEEK (ERR%) = 5 THEN 2210
2200  PRINT "WENN GRUENE LED AN, EPROM EINSETZEN": GOTO 2230
2210  PRINT BELL$;"PROGRAMMIERGERAET ANSCHLIESSEN";CR$;CR$;"TESTSOCKEL LEER LASSEN!"
2220  LET TYP$ = ""
2230  GOSUB 4380: GOSUB 4380: REM  WARTEN
2240  GOTO 1500
2250  REM
2260  REM  FEHLERBEHANDLUNG (BEIM LADEN DES MASCHINENPROGRAMMS)
2270  REM -----------------------------------------------------
2280  POKE 216,0: REM  RESET ONERR-FLAG
2290  IF  PEEK (222) = 6 THEN 2320
2300  VTAB 10: PRINT BELL$;"DOS ERROR "; PEEK (222)
2310  GOSUB 4380: GOTO 1550
2320  VTAB 10: PRINT BELL$;"TYP NICHT BEKANNT"
2330  GOSUB 4380: GOTO 1550
2340  REM
2350  REM ************************************************************
2360  REM  PROGRAMMIEREN
2370  REM ************************************************************
2380  REM
2390  HOME
2400  LET TITEL$ = "PROGRAMMIEREN": GOSUB 3720
2410  VTAB 18: PRINT "START MIT <RET>    :"; SPC( 20): HTAB 22: GET A$
2420  IF A$ <  > CHR$ (13) THEN 1550
2430  VTAB 18: HTAB 1: FLASH : PRINT "PROGRAMMIERUNG LAEUFT";: NORMAL
2440  PRINT  SPC( 19)
2450  CALL PGM%
2460  IF  PEEK (ERR%) THEN 2520
2470  VTAB 18: PRINT BELL$;"PROGRAMMIERUNG BEENDET       <RET> ";: GET A$
2480  GOTO 1550
2490  REM
2500  REM  FEHLERBEHANDLUNG BEI RUECKSPRUNG NACH PGM.-ERROR
2510  REM -------------------------------------------------
2520  VTAB 18: PRINT BELL$;
2530  ON  PEEK (ERR%) GOTO 2540,2550,2560,2650
2540  PRINT "PROGRAMMIERUNG FEHLERHAFT :": GOTO 2570
2550  PRINT "EPROM NICHT GELOESCHT, PGM. UNMOEGLICH": GOTO 2570
2560  PRINT "EPROM NICHT GELOESCHT, ABER PGM. MOEGL."
2570  VTAB 20: GOSUB 4480: REM  FEHLERHAFTE BYTES ANZEIGEN
2580  VTAB 23: PRINT "PROGRAMMIERUNG FORTSETZEN  (J/N)  ?: ";B6$;: GET A$
2590  IF A$ = "J" THEN 2430
2600  GOTO 1550
2610  REM
2620  REM  FEHLER-ANZEIGE, NOCH NICHT INITIALISIERT
2630  REM ------------------------------------------
2640  PRINT CR$;BELL$
2650  PRINT "PROGRAMMIERGERAET NICHT INITIALISIERT"
2660  GOSUB 4380
2670  GOTO 1550
2680  REM
2690  REM ************************************************************
2700  REM  EINLESEN  EPROM -> APPLE
2710  REM ************************************************************
2720  REM
2730  HOME
2740  LET TITEL$ = "EINLESEN  EPROM -> APPLE": GOSUB 3720
2750  CALL RD%
2760  IF  PEEK (ERR%) = 4 THEN 2640
2770  VTAB 18: PRINT "DATEN EINGELESEN            <RET> ";: GET A$
2780  GOTO 1550
2790  REM
2800  REM ************************************************************
2810  REM  VERGLEICHEN  EPROM <-> APPLE
2820  REM ************************************************************
2830  REM
2840  HOME
2850  LET TITEL$ = "VERGLEICHEN  EPROM <-> APPLE": GOSUB 3720
2860  CALL VFY%
2870  IF  PEEK (ERR%) = 1 THEN 2940
2880  IF  PEEK (ERR%) = 4 THEN 2640
2890  VTAB 22: PRINT "VERGLEICH BEENDET           <RET> ";: GET A$
```

"If green LED on insert eprom"

"connect programming device"

"leave testsocket empty"

"typ not recognized"

"program eprom"

"start by pressing <Ret>"

"programming in progress"

"programming finished ------ press <Return>"

"programming error occured"
"eprom not erased program not possible"
"eprom not erased but program possible"

"shall i continue? J=yes N=no"

"programming device not initialized"

"read eprom to Apple"

"read data press <Return>"

"compare eprom <> Apple RAM"

"compare completed press <Return>"

```
2900  GOTO 1550
2910  REM
2920  REM  FEHLERBEHANDLUNG BEI VERIFY (BYTES ANZEIGEN)
2930  REM ------------------------------------------
2940  VTAB 18: PRINT BELL$;
2950  GOSUB 4480
2960  VTAB 22: PRINT "VERGLEICH FORTSETZEN  (J/N)  ?: ";BS$;: GET A$: PRINT A$
2970  IF A$ = "N" THEN 1550
2980  GOSUB 4730: REM  ADR. INCR.: VGL. BEIM  F O L G E N D E N  BYTE FORTSETZEN
2990  IF A1 < A2 THEN 2860
3000  GOTO 2890
3010  REM
3020  REM ************************************************************************
3030  REM  LOESCHTEST
3040  REM ************************************************************************
3050  REM
3060  HOME
3070  LET TITEL$ = "LOESCHTEST": GOSUB 3720
3080  CALL CVFY%
3090  IF  PEEK (ERR%) = 1 THEN 3160
3100  IF  PEEK (ERR%) = 4 THEN 2640
3110  VTAB 18: PRINT "LOESCHTEST BEENDET          <RET> ";: GET A$
3120  GOTO 1550
3130  REM
3140  REM  FEHLERBEHANDLUNG BEI CVFY  (BYTE ANZEIGEN)
3150  REM ------------------------------------------
3160  VTAB 15: PRINT BELL$;"EPROM :  ";: LET ADR% = A1%: GOSUB 4660
3170  PRINT " - ";: LET ADR% = EBYTE%: GOSUB 4600: PRINT
3180  VTAB 18: PRINT "LOESCHTEST FORTSETZEN  (J/N)  ?: ";BS$;: GET A$: PRINT A$
3190  IF A$ = "N" THEN 1550
3200  GOSUB 4730: REM  ADR. INCR.: TEST BEIM  F O L G E N D E N  BYTE FORTSETZEN
3210  IF A1 < A2 THEN 3080
3220  GOTO 3110
3230  REM
3240  REM ************************************************************************
3250  REM  BILDSCHIRMAUSGABE
3260  REM ************************************************************************
3270  REM
3280  HOME
3290  LET TITEL$ = "EPROM - BILDSCHIRMAUSGABE": GOSUB 3720
3300  LET AE = A2: REM  AE=EPROM-ENDADRESSE (LETZTES BYTE)
3310  HOME
3320  LET A2 = A1 + 127: REM  AUSGABE IN BLOCKS VON 128 BYTES, JEW. (A1...A2)
3330  IF A2 > AE THEN  LET A2 = AE
3340  LET A = A2:ADR% = A2%: GOSUB 4320: REM   A2 -> (A2%)
3350  PRINT
3360  CALL DSP%
3370  IF  PEEK (ERR%) = 4 THEN 2640
3380  PRINT CR$;CR$;"<RET> ";: GET A$
3390  IF A2 = AE OR A$ =  CHR$ (3) THEN 1550: REM  ABBRUCH MIT CTRL-C MOEGL.
3400  GOSUB 4730: REM  A1 INCR.: ANZEIGE BEIM  F O L G E N D E N  BYTE FORTSETZEN
3410  LET A2 = A2 + 128
3420  GOTO 3330
3430  REM
3440  REM ************************************************************************
3450  REM  MONITOR AUFRUF
3460  REM ************************************************************************
3470  REM
3480  HOME
3490  VTAB 4: PRINT "RUECKSPRUNG MIT  ";: FLASH : PRINT " 100G ";: NORMAL
3500  PRINT "    !": VTAB 6
3510  POKE 256,104: POKE 257,104: POKE 258,96
3520  REM  "PLA-PLA-RTS" FUER RUECKSPRUNG INS BASIC-PROGRAMM
3530  CALL  - 151
3540  REM  MONITOR-AUFRUF
3550  GOTO 1500
3560  REM
3570  REM ************************************************************************
3580  REM  PROGRAMMENDE
3590  REM ************************************************************************
3600  REM
3610  HIMEM: OLDHIMEM
3620  POKE TOP%,0
3630  HOME
3640  END
3650  REM
3660  REM ************************************************************************
3670  REM  UNTERPROGRAMM ZUR EINGABE DER ADRESSBEREICHE (HEXADEZIMAL)
3680  REM ************************************************************************
3690  REM
3700  REM  MONITOR ZERO-PAGE LOC.
3710  REM -------------------------
3720  LET PL% = 58:PH% = 59:ACC% = 69:XREG% = 70
3730  HOME
3740  IF TYP$ = "" THEN  PRINT BELL$;"TYP ANGEBEN !": GOSUB 4380: POP : GOTO 1550
3750  REM
3760  REM  EINGABE DER ADRESSBEREICHE (EPROM UND APPLE-RAM)
3770  REM ------------------------------------------
3780  VTAB 7: PRINT TITEL$
3790  FOR I = 1 TO  LEN (TITEL$): PRINT "-";: NEXT I: PRINT
3800  REM
3810  LET TXT$ = "EPROM, START-ADR.":LN = 10:ADR% = A1%: GOSUB 4150
3820  IF A > LIM THEN 3970
```

"continue compare test ? J=yes N=no "

"erase test"

"erase test successfull press <Return> to continue"

"continue erase test ? J=yes N=no"

"display eprom content"

"Jump back to BASIC:"

" 100G "

"specify eprom-type"

"eprom startadress"

```
3830  LET A1 = A
3840  LET TXT$ = "EPROM,    END-ADR.":LN = 12:ADR% = A2%: GOSUB 4150
3850  IF A > LIM THEN 3970
3860  IF A < A1 THEN  PRINT BELL$: GOTO 3840
3870  LET A2 = A
3880  IF B$ = "C" OR B$ = "D" THEN  RETURN : REM  CVFY ODER DSP: KEINE RAM-ADR. NOETIG
3890  REM
3900  LET TXT$ = "APPLE, START-ADR.":LN = 15:ADR% = A3%: GOSUB 4150
3910  IF A < LRAM OR A + (A2 - A1) > HRAM THEN 4030
3920  LET A3 = A
3930  RETURN
3940  REM
3950  REM  FEHLERBEHANDLUNG: EPROM-ADRESSE ZU GROSS
3960  REM ------------------------------------
3970  PRINT CR$;CR$;CR$;BELL$;"AUSSERHALB EPROM-ADRESSBEREICH"
3980  GOSUB 4380
3990  GOTO 3720
4000  REM
4010  REM  FEHLERBEHANDLUNG: RAM-BEREICH DURCH PROGRAMM ODER VARIABLEN BELEGT
4020  REM ------------------------------------------------------------------
4030  VTAB 18: PRINT BELL$;"KEIN FREIER RAM-BEREICH"
4040  GOSUB 4380
4050  VTAB 18: PRINT  SPC( 23);CR$
4060  GOTO 3900
4070  REM
4080  REM
4090  REM ************************************************************************
4100  REM  UNTERPROGRAMM HEX-ADRESSE EINLESEN, UMWANDELN, ABSPEICHERN
4110  REM ************************************************************************
4120  REM
4130  REM  EINGABE IN ZEILE LN MIT TXT$ ALS PROMPT
4140  REM ----------------------------------------
4150  VTAB LN: PRINT TXT$;: INPUT " :  $":A$
4160  REM
4170  REM  UMWANDLUNG: HEXADEZIMALZAHL (A$) IN DEZIMALZAHL (A)
4180  REM ----------------------------------------------------
4190  LET A$ = "0000" + A$:A = 0:FAKT = 1:ERR = 0
4200  FOR I = 0 TO 3
4210  LET C$ =  MID$ (A$, LEN (A$) - I,1)
4220  IF C$ < "0" OR C$ > "F" THEN ERR = 1
4230  IF C$ <  = "9" THEN A = A + FAKT * ( ASC (C$) - 48): GOTO 4260
4240  IF C$ >  = "A" THEN A = A + FAKT * ( ASC (C$) - 55): GOTO 4260
4250  LET ERR = 1
4260  LET FAKT = 16 * FAKT
4270  NEXT I
4280  IF ERR THEN  VTAB LN: PRINT BELL$: SPC( 40): GOTO 4150
4290  REM
4300  REM  WERT VON A UNTER ADR%, ADR%+1 ABSPEICHERN: LOW-BYTE FIRST
4310  REM ------------------------------------------------------------
4320  LET HBYTE =  INT (A / 256):LBYTE = A - 256 * HBYTE
4330  POKE ADR%,LBYTE: POKE ADR% + 1,HBYTE
4340  RETURN
4350  REM
4360  REM  WARTESCHLEIFE
4370  REM ---------------
4380  FOR W = 1 TO 3000: NEXT W
4390  RETURN
4400  REM
4410  REM
4420  REM ************************************************************************
4430  REM  UNTERPROGRAMME ZUR HEXADEZIMALEN DATENAUSGABE
4440  REM ************************************************************************
4450  REM
4460  REM  APPLE- UND EPROM-BYTE MIT JEW. ADRESSEN ANZEIGEN
4470  REM ------------------------------------------------
4480  PRINT "APPLE :  ";
4490  LET ADR% = A3%: GOSUB 4660
4500  PRINT " - ";
4510  LET ADR% = CBYTE%: GOSUB 4600: PRINT
4520  PRINT "EPROM :  ";
4530  LET ADR% = A1%: GOSUB 4660
4540  PRINT " - ";
4550  LET ADR% = EBYTE%: GOSUB 4600: PRINT
4560  RETURN
4570  REM
4580  REM  (ADR%) HEXADEZIMAL DRUCKEN: BENUTZT MONITOR-ROUTINE 'PRBYTE' ($FDDA)
4590  REM ------------------------------------------------------------------
4600  POKE PL%,218: POKE PH%,253: POKE ACC%, PEEK (ADR%)
4610  CALL 65209
4620  RETURN
4630  REM
4640  REM  (ADR%+1,ADR%) HEX. DRUCKEN: BENUTZT MONITOR-ROUTINE 'PRNTAX' ($F941)
4650  REM ------------------------------------------------------------------
4660  POKE PL%,65: POKE PH%,249
4670  POKE ACC%, PEEK (ADR% + 1): POKE XREG%, PEEK (ADR%)
4680  CALL 65209
4690  RETURN
4700  REM
4710  REM  AKTUELLE ADRESSEN INKREMENTIEREN ((A1%) UND (A3%))
4720  REM ---------------------------------------------------
4730  LET A1 =  PEEK (A1%) * 256 +  PEEK (A1% + 1)
4740  LET A3 =  PEEK (A3%) * 256 +  PEEK (A3% + 1)
4750  LET A = A1 + 1:ADR% = A1%: GOSUB 4320
4760  LET A = A3 + 1:ADR% = A3%: GOSUB 4320
4770  RETURN
```

"eprom end adress"

"Apple start adress:"

" adress exceeds eprom adress area"

"RAM area not free / empty"

Assembler Code for Machinelanguageprogram

```
                        * APPLE II - MONITOR-UNTERPROGRAMME
                        *
                        PRYX2     EQU   $FD96      ; PRINT Y-REG, X-REG IN HEX
                        PRBL2     EQU   $F94A      ; PRINT BLANCS
                        PRBYTE    EQU   $FDDA      ; PRINT BYTE IN HEX
                        *
                        * APPLE II - I/O-ADRESSEN
                        *
                        OUT0      EQU   $C058      ; 0 -> AN0 (DATEN-AUSG.)
                        OUT1      EQU   $C059      ; 1 -> AN0
                        CLK0      EQU   $C05A      ; 0 -> AN1 (TAKT)
                        CLK1      EQU   $C05B      ; 1 -> AN1
                        STRB0     EQU   $C05C      ; 0 -> AN2 (STROBE)
                        STRB1     EQU   $C05D      ; 1 -> AN2
                        INP       EQU   $C063      ; TTL-EINGANG PB2 (DATEN-EING.)
                        *
                        * ZERO-PAGE-VARIABLEN
                        *
                        SRBYT     EQU   $0006      ; PROGRAMMIERGERAET-STATUS
                        SRBYT1    EQU   $0006
                        SRBYT2    EQU   $0007
                        SRBYT3    EQU   $0008
                        SRBYT4    EQU   $0009
                        *
                        INIFLG    EQU   $00EB      ; ENTHAELT #$96 NACH INITIALISIERUNG
                        COUNT     EQU   $00EC      ; ALLGEM. ZAEHLER
                        ERR       EQU   $00ED      ; ERROR-BYTE, BEDEUTUNG S.U.
                        EBYTE     EQU   $00EE      ; EPROM-BYTE BEI VFY ODER PGM
                        CBYTE     EQU   $00EF      ; COMPUTER-BYTE BEI VFY ODER PGM
                        *
                        EADR      EQU   $00FA      ; AKTUELLE ADRESSE, EPROM
                        EADRL     EQU   $00FA
                        EADRH     EQU   $00FB
                        EADR1     EQU   $00FC      ; ZIELADRESSE, EPROM
                        CADR      EQU   $00FE      ; AKTUELLE ADRESSE, COMPUTER
                        CADRL     EQU   $00FE
                        CADRH     EQU   $00FF
                        *
                        *-------------------------------------------------------
                        *
                        * BEDEUTUNGEN, ERROR-BYTE :
                        *
                        * ERR=0 - FUNKTION O.K.
                        * ERR=1 - PROGRAMMIERUNG AUSGEFUEHRT, FEHLERHAFT
                        * ERR=2 - EPROM-BYTE NICHT $FF, PROGR. UNMOEGLICH
                        * ERR=3 - EPROM-BYTE NICHT $FF, ABER PROGR. MOEGL.
                        * ERR=4 - PROGRAMMIERGERAET NICHT INITIALISIERT
                        * ERR=5 - PROGRAMMIERGERAET NICHT ANGESCHLOSSEN
                        *
                        *-------------------------------------------------------
                        *
                                  ORG   $8000
                        *
                        *-------------------------------------------------------
                        *
                        *=======================================================
                        * PROGRAMM - KOPF   (STANDARD-FORMAT!)
                        *=======================================================
                        *
                        * STARTADRESSEN DER HAUPTROUTINEN  (ZUM AUFRUF VON BASIC)
                        *
8000: 30 80             *         DA    INIT       ; INITIALISIERUNG
8002: B2 80             *         DA    PGM        ; PROGRAMMIERUNG
8004: EE 80             *         DA    READ       ; LESEN
8006: 01 81             *         DA    VFY        ; VERGLEICHEN
8008: 15 81             *         DA    CVFY       ; LOESCHTEST
800A: 27 81             *         DA    DSP        ; BILDSCHIRMAUSGABE
                        *
                        * SPEICHERPLAETZE ZUR PARAMETERUEBERGABE ZUM/VOM BASIC-PROGRAMM
                        *
800C: ED 00             *         DA    ERR        ; ERROR BYTE
800E: EE 00             *         DA    EBYTE      ; EPROM-BYTE
8010: EF 00             *         DA    CBYTE      ; COMPUTER-BYTE
8012: FA 00             *         DA    EADR       ; AKT. ADR., EPROM
8014: FC 00             *         DA    EADR1      ; ZIELADR. , EPROM
8016: FE 00             *         DA    CADR       ; AKT. ADR., COMPUTER
                        *
                        * STATUS-BITS FUER READ, VFY, PGM, PGM-INH.   *** 2716 ***
                        *
8018: 24 00             RDSTAT    DDB   %0010010000000000
801A: E4 00             VYSTAT    DDB   %1110010000000000
801C: E7 80             PGSTAT    DDB   %1110011110000000
801E: E6 80             IHSTAT    DDB   %1110011010000000
                        *
                        * WEITERE TYPENSPEZ. PARAMETER   *** 2716 ***
                        *
8020: 32                PGTIME    DFB   50          ; PROGRAMMIERZEIT IN MILLISEC.
8021: FF 07             *         DA    $07FF       ; HOECHSTE EPROM-ADRESSE
8023: 32 37 31
8026: 36 2E             *         ASC   '2716.'     ; EPROM-TYP ("." AM STRING-ENDE)
                        *         DS    $8030-*     ; DAMIT LAENGERE TYPENBEZ. OHNE
                        *                           ; CODEVERSCHIEBUNG MOEGL.
                        *
                        *=======================================================
                        * INITIALISIERUNG DES PROGRAMMIERGERAETES (SETZT GGF. ERR5)
                        *=======================================================
```

Bild 5. Maschinenprogramm für den EPROM-Typ 2716: Typenspezifisch sind nur die Teile in den Adreßbereichen $8018...$8027 und ab $8206

```
8030: 2C 5C C0  111  INIT    BIT  STRBO      ; 0 -> STROBE
8033: 2C 5A C0  112          BIT  CLKO       ; 0 -> CLOCK
                113
8036: A9 00     114          LDA  #$00       ; BIS LABEL CLRRTS: TEST OB
8038: 85 EB     115          STA  INIFLG     ; GERAET ANGESCHLOSSEN
803A: 85 07     116          STA  SRBYT2
803C: 85 08     117          STA  SRBYT3
803E: A9 02     118          LDA  #$02
8040: 85 09     119          STA  SRBYT4
8042: A9 69     120          LDA  #$69
8044: 85 06     121          STA  SRBYT1
8046: 20 D5 81  122          JSR  SFTIN1     ; #$69 EIN- UND AUSLESEN OHNE STRB.
8049: 20 F0 81  123          JSR  SHFTOT
804C: C9 69     124          CMP  #$69
804E: F0 2E     125          BEQ  ERR5       ; STRB. WAR 1 (VERBINDUNG UNTERBR.)
8050: 20 CB 81  126          JSR  SHFTIN     ; #$69 EIN- UND AUSLESEN
8053: 20 F0 81  127          JSR  SHFTOT
8056: C9 69     128          CMP  #$69
8058: D0 24     129          BNE  ERR5
805A: A9 96     130          LDA  #$96
805C: 85 06     131          STA  SRBYT1
805E: 20 CB 81  132          JSR  SHFTIN     ; #$96 EIN- UND AUSLESEN
8061: 20 F0 81  133          JSR  SHFTOT
8064: C9 96     134          CMP  #$96
8066: D0 16     135          BNE  ERR5
8068: 85 EB     136          STA  INIFLG     ; #$96 -> FLAG (INITIALIS. O.K.)
                137
806A: A9 00     138  CLRTS   LDA  #0
806C: 85 ED     139  ERRTS   STA  ERR
806E: A9 00     140          LDA  #$00       ; PROGRAMMIERGERAET INITIALISIEREN
8070: 85 06     141          STA  SRBYT1
8072: 85 07     142          STA  SRBYT2     ; ALLE PINS LOW, PGM-SPANNUNG
8074: 85 08     143          STA  SRBYT3     ; AUS, GRUENE LED AN:
8076: A9 08     144          LDA  #$08       ; SOCKEL FREI ZUM WECHSELN
8078: 85 09     145          STA  SRBYT4
807A: 20 CB 81  146          JSR  SHFTIN
807D: 60        147          RTS
                148
807E: A9 05     149  ERR5    LDA  #5
8080: D0 EA     150          BNE  ERRTS      ; (VERZWEIGT IMMER)
                151
                152
                153  ********************************************************
                154  * PROGRAMMIEREN  (SETZT GGF. ERR1 BIS ERR4)
                155  ********************************************************
                156
                157
8082: 20 4D 81  158  PGM     JSR  INICHK
8085: D0 62     159          BNE  ERR4       ; RUECKSPRUNG MIT ERR=4
                160
8087: 20 6C 81  161          JSR  PGMINH
808A: A9 14     162          LDA  #20
808C: A2 64     163  WAIT1   LDX  #100       ; 2 SEC. WARTEN, BIS
808E: 20 52 81  164          JSR  WAIT       ; PROGRAMMIERSPG. AUFGEBAUT
8091: 38        165          SEC
8092: E9 01     166          SBC  #1
8094: D0 F6     167          BNE  WAIT1
                168
8096: A5 ED     169          LDA  ERR        ; TESTE ALTES ERROR-BYTE (BEI
8098: C9 01     170          CMP  #1         ; WIEDEREINTRITT NACH ERROR):
809A: F0 41     171          BEQ  PGM1       ; - WEITER MIT NAECHSTEM BYTE
809C: B0 20     172          BCS  PGMGO      ; - PROGR., OBWOHL BYTE < $FF
                173
809E: A2 00     174  PGM0    LDX  #$00       ; TESTE NEUES BYTE VOR PGM.
80A0: A1 FE     175          LDA  (CADR,X)
80A2: 85 EF     176          STA  CBYTE      ; COMPUTER-BYTE -> CBYTE
80A4: 20 5B 81  177          JSR  PGMVFY
80A7: 20 A8 81  178          JSR  RDBYT1     ; (LESEN, PGM-SPG. BLEIBT EINGESCH.)
80AA: 85 EE     179          STA  EBYTE      ; EPROM-BYTE -> EBYTE
80AC: C9 FF     180          CMP  #$FF
80AE: F0 0E     181          BEQ  PGMGO      ; EPROM-BYTE = $FF, STARTE PGM.
80B0: 45 EF     182          EOR  CBYTE      ; TEST OB DENNOCH
80B2: 25 EF     183          AND  CBYTE      ; PROGRAMMIERUNG MOEGL.
80B4: D0 04     184          BNE  ERR2
80B6: A9 03     185  ERR3    LDA  #3
80B8: D0 B2     186          BNE  ERRTS      ; (VERZWEIGT IMMER)
80BA: A9 02     187  ERR2    LDA  #2
80BC: D0 AE     188          BNE  ERRTS      ; (VERZWEIGT IMMER)
                189
80BE: A5 EF     190  PGMGO   LDA  CBYTE      ; DATENBYTE -> SRBYT1
80C0: 85 06     191          STA  SRBYT1
80C2: 20 6C 81  192          JSR  PGMINH
80C5: 20 7D 81  193          JSR  PGPULS     ; PGM-IMPULS EIN!
80C8: AE 20 80  194          LDX  PGTIME
80CB: 20 52 81  195          JSR  WAIT       ; PROGRAMMIERUNG (50 MILLISEC.)
80CE: 20 6C 81  196          JSR  PGMINH
80D1: 20 5B 81  197          JSR  PGMVFY
80D4: 20 A8 81  198          JSR  RDBYT1     ; (LESEN, PGM-SPG. BLEIBT EINGESCH.)
80D7: 85 EE     199          STA  EBYTE
80D9: C5 EF     200          CMP  CBYTE      ; RUECKVERGLEICH
80DB: D0 08     201          BNE  ERR1       ; FEHLERHAFT: RET. MIT ERR=1
                202
80DD: 20 B2 81  203  PGM1    JSR  INCADR     ; NAECHSTES BYTE
80E0: D0 BC     204          BNE  PGM0
80E2: 4C 6A 80  205  CLRTS1  JMP  CLRTS      ; LETZTES BYTE: RET. MIT ERR=0
                206
80E5: A9 01     207  ERR1    LDA  #1
80E7: D0 83     208          BNE  ERRTS      ; (VERZWEIGT IMMER)
                209
80E9: A9 04     210  ERR4    LDA  #4
80EB: 4C 6C 80  211          JMP  ERRTS
                212
```

```
                        213
                        214 *-------------------------------------------------------
                        215 * EINLESEN EPROM -> APPLE  (SETZT GGF. ERR4)
                        216 *-------------------------------------------------------
                        217
                        218
80EE: 20 4D 81          219 READ    JSR  INICHK
80F1: D0 F6             220         BNE  ERR4        ; RUECKSPRUNG MIT ERR=4
80F3: 20 9E 81          221 READ1   JSR  RDBYTE
80F6: A2 00             222         LDX  #0
80F8: 81 FE             223         STA  (CADR,X)
80FA: 20 B2 81          224         JSR  INCADR
80FD: F0 E3             225         BEQ  CLRTS1      ; LETZTES BYTE; RET. MIT ERR=0
80FF: D0 F2             226         BNE  READ1
                        227
                        228
                        229 *-------------------------------------------------------
                        230 * VERGLEICHEN EPROM (-) APPLE  (SETZT GGF. ERR1, ERR4)
                        231 *-------------------------------------------------------
                        232
                        233
8101: 20 4D 81          234 VFY     JSR  INICHK
8104: D0 E3             235         BNE  ERR4        ; RUECKSPRUNG MIT ERR=4
8106: A2 00             236 VFY1    LDX  #0
8108: A1 FE             237         LDA  (CADR,X)
810A: 85 EF             238         STA  CBYTE
810C: 20 8E 81          239         JSR  CMPBYT      ; VGL. EPROM-BYTE MIT CBYTE
810F: B0 D4             240         BCS  ERR1        ; UNGLEICH, RET. MIT ERR=1
8111: F0 CF             241         BEQ  CLRTS1      ; LETZTES BYTE, RET. MIT ERR=0
8113: D0 F1             242         BNE  VFY1        ; NAECHSTES BYTE
                        243
                        244
                        245 *-------------------------------------------------------
                        246 * LOESCHTEST (ALLE BYTES $FF ?)   (SETZT GGF. ERR1, ERR4)
                        247 *-------------------------------------------------------
                        248
                        249
8115: 20 4D 81          250 CVFY    JSR  INICHK
8118: D0 CF             251         BNE  ERR4        ; RUECKSPRUNG MIT ERR=4
811A: A9 FF             252         LDA  #$FF
811C: 85 EF             253         STA  CBYTE       ; #$FF -> CBYTE
811E: 20 8E 81          254 CVFY1   JSR  CMPBYT      ; VGL. EBYTE MIT CBYTE
8121: B0 C2             255         BCS  ERR1        ; UNGLEICH, RET. MIT ERR=1
8123: F0 BD             256         BEQ  CLRTS1      ; LETZTES BYTE, RET. MIT ERR=0
8125: D0 F7             257         BNE  CVFY1       ; NAECHSTES BYTE
                        258
                        259
                        260 *-------------------------------------------------------
                        261 * EPROM BILDSCHIRMAUSGABE  (SETZT GGF. ERR4)
                        262 *-------------------------------------------------------
                        263
                        264
8127: 20 4D 81          265 DSP     JSR  INICHK
812A: D0 BD             266         BNE  ERR4        ; RUECKSPRUNG MIT ERR=4
812C: A9 08             267 DSP1    LDA  #8          ; NACH JE 8 BYTES: NEUE ZEILE
812E: 85 EC             268         STA  COUNT
8130: A6 FA             269         LDX  EADRL
8132: A4 FB             270         LDY  EADRH
8134: 20 96 FD          271         JSR  PRYX2       ; 'CR' UND EPROM ADRESSE DRUCKEN
8137: A2 01             272 DSP2    LDX  #1
8139: 20 4A F9          273         JSR  PRBL2       ; 1 BLANC DRUCKEN
813C: 20 9E 81          274         JSR  RDBYTE      ; BYTE EINLESEN
813F: 20 DA FD          275         JSR  PRBYTE      ; BYTE IN HEX DRUCKEN
8142: 20 B2 81          276         JSR  INCADR      ; ADRESSE INKREMENTIEREN
8145: F0 9B             277         BEQ  CLRTS1      ; LETZTES BYTE, RET. MIT ERR=0
8147: C6 EC             278         DEC  COUNT
8149: F0 E1             279         BEQ  DSP1        ; NEUE ZEILE
814B: D0 EA             280         BNE  DSP2        ; NAECHSTES BYTE
                        281
                        282
                        283 *-------------------------------------------------------
                        284 * ALLGEMEINE HILFSROUTINEN
                        285 *-------------------------------------------------------
                        286
                        287
814D: A5 ER             288 INICHK  LDA  INIFLG      ; CHECK, OB INITIALISIERT
814F: C9 96             289         CMP  #$96
8151: 60                290         RTS              ; MIT Z-FLAG=1, FALLS O.K.
                        291
                        292 *-------------------------------------------------------
                        293
8152: A0 C8             294 WAIT    LDY  #200        ; WARTEN: (X-REG) MILLISEC.
8154: 88                295 LOOP1   DEY              ; LOOP1  =  1 MILLISEC.
8155: D0 FD             296         BNE  LOOP1
8157: CA                297         DEX
8158: D0 F8             298         BNE  WAIT
815A: 60                299         RTS
                        300
                        301 *-------------------------------------------------------
                        302
815B: AD 1A 80          303 PGMVFY  LDA  VYSTAT      ; VFY-STATUS AUSGEBEN
815E: 85 09             304         STA  SRBYT4
8160: AD 1B 80          305         LDA  VYSTAT+1
8163: 85 08             306         STA  SRBYT3
8165: 20 06 82          307         JSR  SETADR
8168: 20 C8 81          308         JSR  SHFTIN
816B: 60                309         RTS
                        310
                        311 *-------------------------------------------------------
                        312
816C: AD 1E 80          313 PGMINH  LDA  IHSTAT      ; PGM-INHIBIT AUSGEBEN
816F: 85 09             314         STA  SRBYT4
8171: AD 1F 80          315         LDA  IHSTAT+1
8174: 85 08             316         STA  SRBYT3
```

```
8176: 20 06 82 317             JSR  SETADR
8179: 20 CB 81 318             JSR  SHFTIN      : (DATEN + ADRESSBITS LIEGEN AN)
817C: 60       319             RTS
               320
               321 *-------------------------------------------------------
               322
817D: AD 1C 80 323 PGPULS      LDA  PGSTAT      : PROGRAMMIERIMPULS STARTEN
8180: 85 09    324             STA  SRBYT4
8182: AD 1D 80 325             LDA  PGSTAT+1
8185: 85 08    326             STA  SRBYT3
8187: 20 06 82 327             JSR  SETADR
818A: 20 CB 81 328             JSR  SHFTIN
818D: 60       329             RTS
               330
               331 *-------------------------------------------------------
               332
818E: 20 9E 81 333 CMPBYT      JSR  RDBYTE      ; LIES EPROM-BYTE, VGL. MIT CBYTE
8191: 85 EE    334             STA  EBYTE       ; FALLS GLEICH, INCR. ADRESSEN
8193: C5 EF    335             CMP  CBYTE
8195: D0 05    336             BNE  NOTEQU
8197: 20 B2 81 337             JSR  INCADR
819A: 18       338             CLC
819B: 60       339             RTS              ; CARRY=0: BYTES GLEICH
819C: 38       340 NOTEQU     SEC
819D: 60       341             RTS              ; CARRY=1: BYTES VERSCHIEDEN
               342
               343 *-------------------------------------------------------
               344
819E: AD 18 80 345 RDBYTE      LDA  RDSTAT      : EPROM-BYTE -> ACCU
81A1: 85 09    346             STA  SRBYT4
81A3: AD 19 80 347             LDA  RDSTAT+1
81A6: 85 08    348             STA  SRBYT3
81A8: 20 06 82 349 RDBYT1      JSR  SETADR
81AB: 20 CB 81 350             JSR  SHFTIN
81AE: 20 F0 81 351             JSR  SHFTOT
81B1: 60       352             RTS
               353
               354 *-------------------------------------------------------
               355
81B2: A2 02    356 INCADR     LDX  #2          ; CADR UND EADR INCR.
81B4: B5 F9    357 CMPLOP     LDA  EADR-1,X    ; CHECK OB LETZTES BYTE
81B6: D5 FB    358             CMP  EADR1-1,X
81B8: D0 04    359             BNE  NOTYET
81BA: CA       360             DEX
81BB: D0 F7    361             BNE  CMPLOP
81BD: 60       362             RTS              ; Z-FLAG=1 : ZIEL ERREICHT
81BE: E6 FE    363 NOTYET     INC  CADRL
81C0: D0 02    364             BNE  *+4
81C2: E6 FF    365             INC  CADRH
81C4: E6 FA    366             INC  EADRL
81C6: D0 02    367             BNE  *+4
81C8: E6 FB    368             INC  EADRH
81CA: 60       369             RTS              ; Z-FLAG=0 : NOCH NICHT AM ZIEL
               370
               371 *-------------------------------------------------------
               372
81CB: 20 D5 81 373 SHFTIN      JSR  SFTIN1      : STATUS-BYTES -> S-R. UND STRB.
81CE: 2C 5D C0 374             BIT  STRB1
81D1: 2C 5C C0 375             BIT  STRB0
81D4: 60       376             RTS
               377
               378 *-------------------------------------------------------
               379
81D5: A2 00    380 SFTIN1     LDX  #0          ; STATUS-BYTES -> SHFT-R. SCHIEBEN
81D7: A0 08    381 LOOP2      LDY  #8
81D9: B5 06    382             LDA  SRBYT,X     ; NAECHSTES STATUS-BYTE LADEN
81DB: 2C 58 C0 383 LOOP3       BIT  OUT0        ; DATENBIT EINSTELLEN
81DE: 4A       384             LSR
81DF: 90 03    385             BCC  *+5
81E1: 2C 59 C0 386             BIT  OUT1
81E4: 20 FF 81 387             JSR  CLOCK       ; SHIFT-REG. TAKT
81E7: 88       388             DEY              ; BIT-ZAEHLER: Y-REG
81E8: D0 F1    389             BNE  LOOP3
81EA: E8       390             INX              ; BYTE-ZAEHLER: X-REG
81EB: E0 04    391             CPX  #4
81ED: D0 E8    392             BNE  LOOP2
81EF: 60       393             RTS
               394
               395 *-------------------------------------------------------
               396
81F0: A2 80    397 SHFTOT     LDX  #$80        ; EPROM DATENBYTE -> X-REG SCHIEBEN
81F2: AD 63 C0 398 LOOP4       LDA  INP         ; (DATENBIT = MSB)
81F5: 2A       399             ROL              ; -> CARRY
81F6: 8A       400             TXA
81F7: 6A       401             ROR              ; ANHAENGEN
81F8: AA       402             TAX
81F9: 20 FF 81 403             JSR  CLOCK       ; SHIFT-REG. TAKT
81FC: 90 F4    404             BCC  LOOP4       ; CARRY (VON ROR) SIGNALIS. 8. BIT
81FE: 60       405             RTS              ; DATENBYTE JETZT IN X-REG UND ACCU
               406
               407 *-------------------------------------------------------
               408
81FF: 2C 5B C0 409 CLOCK       BIT  CLK1        ; SHIFT-REG. TAKT
8202: 2C 5A C0 410             BIT  CLK0
8205: 60       411             RTS
               412
               413 *-------------------------------------------------------
               414
8206: A5 FA    415 SETADR     LDA  EADRL       ; EPROM-ADRESSE IN STATUS-BYTES
8208: 85 07    416             STA  SRBYT2      ; EINFUEGEN (GGF. EINBELGG. BEACHTEN)
820A: A5 FB    417             LDA  EADRH
820C: 05 08    418             ORA  SRBYT3
820E: 85 08    419             STA  SRBYT3
8210: 60       420             RTS
```