Jürgen Plate

Mikroelektronik im Fernsehen

Teil 6

In der Mai-Folge wurde die recht umfangreiche 68008-Software kurz überflogen. Hier soll sie nun etwas umfassender besprochen werden. Zuerst sei aber auf ein kleines Hilfsmittel zum Einstellen der Kassettenrecorder-Platine hingewiesen.

Dabei handelt es sich um ein EPROM, das anstelle des letzten RAM-Bausteins in die Karte SBC-II gesteckt wird, das SCOP-PROM. Mit dem Programm in diesem EPROM wird der Computer zu einem digitalen Oszilloskop mit zwei Kanälen und einer Auflösung von etwa fünf Mikrosekunden. Die Eingänge des „SCOP" sind die Bits 0 und 1 von Port 0 der IOE-Karte, die auf Adresse 30H eingestellt sein muß. Die Messung an der CAS-Platine erfolgt in zwei Stufen: Zuerst wird der Pin 4 des Bausteins 6850 mit Bit 0 verbunden und der Menüpunkt 1 des Scop-Menüs gewählt (Start des Programms ab 8800H). Damit wird die Periodendauer gemessen. Nun drehen Sie am Trimmer 1, bis eine Periodendauer von 833 (± 5) µs angezeigt wird. Die zweite Messung erfolgt dann so, daß Bit 0 an Pin 3 des 6850 und Bit 1 an Pin 2 des 6850 angeschlossen werden. Nach der Wahl des Punktes 2 des SCOP-Menüs zeigen sich dann zwei Kurven übereinander – vorausgesetzt, Eingang des Kassetteninterfaces wurde, wie in *Bild 1* gezeigt, mit dem Ausgang verbunden. Nun muß nur noch die Dauer des 0-Signals mit Trimmer 2 auf drei Viertel der Periodendauer, also auf 625 µs eingestellt werden. Wenn Sie anschließend noch eine Probeaufnahme mit Ihrem Recorder machen, können Sie auch gleich sehen, ob die Polarität mit S1 umgeschaltet werden muß (*Bild 2*). Übrigens eignet sich nicht jeder Recorder für die Aufzeichnung; wenn Sie also erst einen kaufen, vergewissern Sie sich, daß Sie ein Umtausch- oder Rückgaberecht haben! Das SCOP-EPROM erhalten Sie bei den Bausatzlieferanten und beim Franzis-Software-Service.

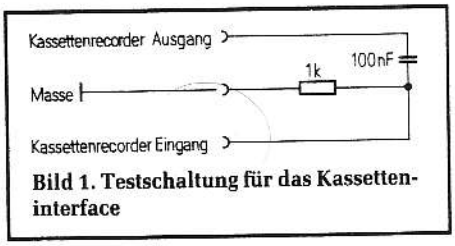


Bild 1. Testschaltung für das Kassetteninterface

Die 68008-Software

Das Menü des 68008-Grundprogramms ist wesentlich umfangreicher als das der Z80-Grundsoftware. An dieser Stelle werden nur die „Neuheiten" erläutert, also die Punkte, die es beim Z80 nicht gibt. Und das geht am besten mit Hilfe von *Bild 3*. (Eine genauere Beschreibung befindet sich im Begleitheft „Schaltpläne" vom Franzis-Software-Service, 8 DM, in dem sich auch Schaltungen und Datenblätter zum 68008 befinden.) Der erste Schirm gleicht dem des Z80; der einzige Unterschied besteht darin, daß bei der Adreßangabe beliebige Ausdrücke mit dezimalen, dualen und hexadezimalen Zahlen möglich sind. Daher müssen hexadezimale Adressen durch Dollar-Zeichen und Dualzahlen durch das Prozentzeichen eingeleitet werden. Der zweite Schirm enthält Editor, Assembler, Bibliothek und Optionen, auf die weiter unten eingegangen wird.

Der dritte Schirm ist wieder bekannt: Neu ist hier nur, daß zwischen Daten und Texten unterschieden wird. Daten werden wie gewohnt mit Anfangs- und Endadressen abgespeichert, bei Texten sucht sich das Programm Start- und Endadresse selbst. Beim Laden kann man für Texte eine neue Startadresse eingeben und so die Texte in andere Speicherbereiche bringen. Auch der nächste Schirm ist bekannt, neben den EPROM-Funktionen kam hier noch die Anzeige der RAM-Speicherbereiche hinzu. Auch den letzten Schirm des Hauptmenüs kennen Sie, wenn Sie die Karte SBC-II benutzen oder das Begleitbuch von R. D. Klein haben. Beim 68008 liegen alle I/O-Ports im Adreßbereich von $FFFFFF00 bis $FFFFFFFF (Sie haben richtig gezählt, es sind wirklich acht Stellen).

Mit der Einzelschrittfunktion können Sie das Programm Befehl für Befehl schrittweise ausführen. Wenn Sie vor dem Assemblieren im Optionen-Menü die Option „Debug an" gewählt hatten, erscheint neben dem Inhalt der Register auch die entsprechende *Quellzeile* des Programms auf dem Bildschirm.

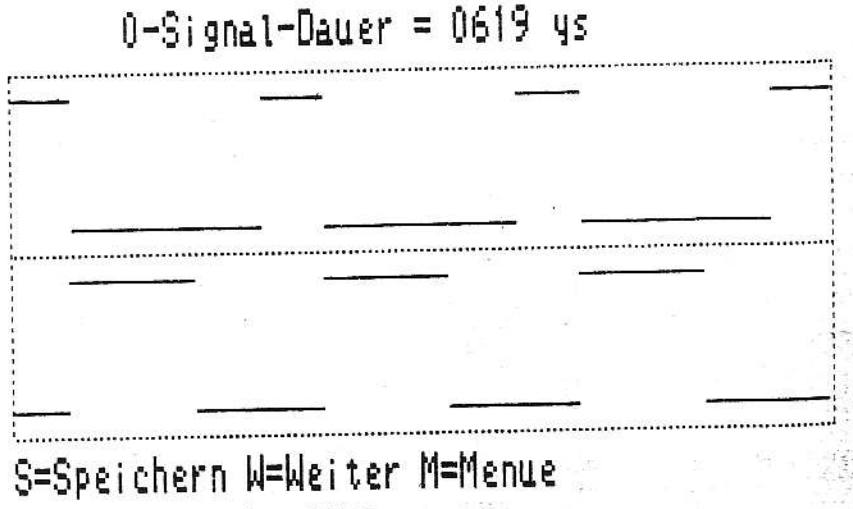


Bild 2. Der Bildschirm beim SCOP-Programm

Editor und Assembler

Mit dem Editor ist es möglich, Texte und Programme einzugeben. Das können Assembler- oder Pascal-Programme sein, aber auch beliebige Texte. Beim Start wird der Bildschirm gelöscht; am unteren Bildrand erscheint eine Statuszeile mit verschiedenen Informationen (Textstart und -ende, Zeichensatz usw.). Die Zeilen haben am linken Rand ein Delete-Zeichen, um zu zeigen, daß kein Text vorhanden ist. Der Editor ist bildschirmorientiert, was bedeutet, daß sie den Cursor frei auf dem ganzen Bildschirm bewegen können und alle Änderungen, die Sie auf dem Schirm machen, auch im Text erfolgen. Die meisten Steuerfunktionen werden durch Drücken der Control-Taste zusammen mit einer anderen Taste aufgerufen. Durch Drücken von Control und „J" erhalten Sie ein dreiteiliges Menü aller Steuerfunktionen, die das Einfügen und Löschen von einzelnen Zeichen, aber auch von ganzen Zeilen, das Suchen von Textstellen, das Ersetzen von Textstücken durch andere und vieles mehr enthalten. Eine umfassende Beschreibung des Editors würden den Rahmen dieses Artikels sprengen – außerdem ist die Computerei keine Sache, die Sie im „Trockenschwimmkurs" lernen können. Am besten, Sie probieren einfach einmal aus, was der Editor so alles kann.

Ein Assembler übersetzt mnemonisch geschriebene Befehle (z. B. MOVE), einen Programmtext, die für den Menschen halbwegs verständlich sind, in die Maschinensprache. Der Assembler des 68008-Grundprogramms verarbeitet die Befehle, wie sie zum Beispiel im „16-Bit-Microprocessors User's Manual MC68000" der Firma Motorola beschrieben sind. Auch das Buch von Leventhal, „68000 Assembly Language Programming", McGraw-Hill, ist hier – obwohl in englisch – sehr zu empfehlen.

Einen Überblick des Befehlsvorrat vermittelt *Bild 4*. Lassen Sie sich nicht von der Vielzahl der Befehle ins Bockshorn jagen: Sie müssen nicht alle Befehle kennen, um ein 68008-Programm zu schreiben. Zudem können Sie auf über 60 Unterprogramme in der Grundsoftware zurückgreifen. Achten Sie bei der Programmerstellung darauf, daß der Speicherbereich von $8000 bis $8FFF für Systemvariable und die Symboltabelle reserviert ist.

Der Assembler kennt keine Makros. Bei Adressenangaben sind, wie schon bei der Adreßeingabe in anderen Menüpunkten, wieder beliebige Ausdrücke möglich, z. B. START + OFFSET*256.

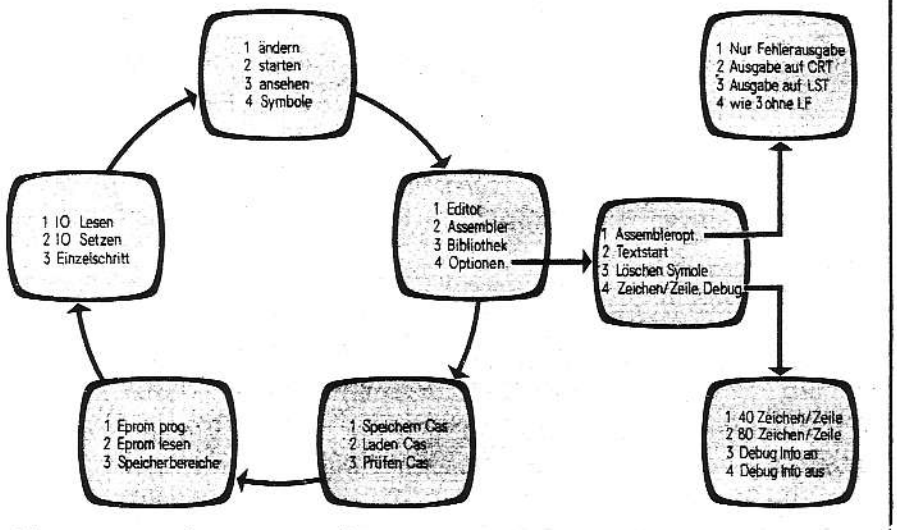


Bild 3. Die Menüs des 68008-Grundprogramms

Der Assembler kennt sechs Pseudooperationen:

ORG zur Festlegung der Startadresse, z. B. ORG $A000.

EQU zur Zuweisung eines Wertes an ein Symbol, z. B. IO EQU $FFFFFF30.

DC zur Definition von Daten und Texten. Dabei wird zwischen Byte (DC.B), Wort (DC.W) und Langwort (DC.L) unterschieden, z. B. DC.B „Guten Morgen"; DC.L 12345678.

DS zum Freihalten von Speicherbereichen. Auch hier gibt es DS.B, DS.W und DS.L, z. B. hält DS.B 35 genau 35 Bytes frei.

OFFSET Hier wird der Maschinencode für die Adresse 0 übersetzt, aber beginnend bei der angegebenen Adresse abgelegt (OFFSET $A000 legt den Code ab A000 ab).

END beendet das Assemblerprogramm.

Das Listing auf dem Bildschirm kann mit Control-S angehalten werden, mit Control-Q geht es dann weiter. Sie können die Assemblierung auch getrost mit der Reset-Taste abbrechen. *Bild 5* zeigt ein typisches Assemblerlisting.

Haben Sie das Programm dann endlich fertig, gibt es zwei Möglichkeiten, es für den späteren Gebrauch aufzubewahren. Entweder Sie speichern das Programm auf Kassette (oder später auf Diskette) oder Sie brennen ein EPROM. Damit sich die Programme im EPROM leicht aufrufen lassen, gibt es die Bibliotheksfunktion.

Die Bibliotheks-Funktion

Wenn Sie im EPROM ein Programm mit einem Bibliotheks-Vorspann versehen, kann dieses Programm nachher über das Bibliotheksmenü aufgerufen werden. Das Programm muß auf einer 2-KByte-Grenze beginnen. Der Kopf eines Bibliotheksprogramms hat folgendes Aussehen:

```
DC.B  $55,$AA,$01,$80
DC.B  „NAME___"          genau
                         8 Buchstaben
DC.L  Startadresse       (siehe unten)
DC.L  Programmlänge
DC.B  0 oder 1           Reloc
DC.B  0,0,0              (Reserve f.
                         Erweit.)
DC.L  0,0                hier weitere
                         Eintrittspunkte
                         oder 0,0
```

Die Startadresse hängt davon ab, ob das Programm verschieblich (relokativ) ist (Byte Reloc = 1), oder absolut (Byte Reloc = 0). Bei relokativen Programmen

Mnemonic	Operation	Assembler Syntax	Condition Codes X	N	Z	V	C
ABCD	Add Decimal with Extend	ABCD Dy,Dx ABCD -(Ay),-(Ax)	*	U	*	U	*
ADD	Add Binary	ADD <ea>,Dn ADD Dn,<ea>	*	*	*	*	*
ADDA	Add Address	ADDA <ea>,An	—	—	—	—	—
ADDI	Add Immediate	ADDI #<data>,<ea>	*	*	*	*	*
ADDQ	Add Quick	ADDQ #<data>,<ea>	*	*	*	*	*
ADDX	Add Extended	ADDX Dy,Dx ADDX -(Ay),-(Ax)	*	*	*	*	*
AND	AND Logical	AND <ea>,Dn AND Dn,<ea>	—	*	*	0	0
ANDI	AND Immediate	ANDI #<data>,<ea>	—	*	*	0	0
ASL, ASR	Arithmetic Shift	ASd Dx,Dy ASd #<data>,Dy ASd<ea>	*	*	*	*	*
Bcc	Branch Conditionally	Bcc <label>	—	—	—	—	—
BCHG	Test a Bit and Change	BCHG Dn,<ea> BCHG #<data>,<ea>	—	—	*	—	—
BCLR	Test a Bit and Clear	BCLR Dn,<ea> BCLR #<data>,<ea>	—	—	*	—	—
BRA	Branch Always	BRA <label>	—	—	—	—	—
BSET	Test a Bit and Set	BSET Dn,<ea> BSET #<data>,<ea>	—	—	*	—	—
BSR	Branch to Subroutine	BSR <label>	—	—	—	—	—
BTST	Test a Bit	BTST Dn,<ea> BTST #<data>,<ea>	—	—	*	—	—
CHK	Check Register Against Bounds	CHK <ea>,Dn	—	*	U	U	U
CLR	Clear an Operand	CLR <ea>	—	0	1	0	0
CMP	Arithmetic Compare	CMP <ea>,Dn	—	*	*	*	*
CMPA	Arithmetic Compare Address	CMPA <ea>,An	—	*	*	*	*
CMPI	Compare Immediate	CMPI #<data>,<ea>	—	*	*	*	*
CMPM	Compare Memory	CMPM (Ay)+,(Ax)+	—	*	*	*	*
DBcc*	Test Condition, Decrement and Branch	DBcc Dn, <label>	—	—	—	—	—
DIVS	Signed Divide	DIVS <ea>,Dn	—	*	*	*	0
DIVU	Unsigned Divide	DIVU <ea>,Dn	—	*	*	*	0
EOR	Exclusive OR Logical	EOR Dn,<ea>	—	*	*	0	0
EORI	Exclusive OR Immediate	EORI #<data>,<ea>	—	*	*	0	0
EXG	Exchange Registers	EXG Rx,Ry	—	—	—	—	—
EXT	Sign Extend	EXT Dn	—	*	*	0	0
JMP	Jump	JMP <ea>	—	—	—	—	—
JSR	Jump to Subroutine	JSR <ea>	—	—	—	—	—
LEA	Load Effective Address	LEA <ea>,An	—	—	—	—	—
LINK	Link and Allocate	LINK An,#<displacement>	—	—	—	—	—
LSL, LSR	Logical Shift	LSd Dx,Dy LSd #<data>,Dy LSd <ea>	*	*	*	0	*
MOVE	Move Data from Source to Destination	MOVE <ea>,<ea>	—	*	*	0	0
MOVE to CCR	Move to Condition Codes	MOVE <ea>,CCR	*	*	*	*	*
MOVE to SR	Move to the Status Register	MOVE <ea>,SR	*	*	*	*	*

Mnemonic	Operation	Assembler Syntax	Condition Codes X	N	Z	V	C
MOVE from SR	Move from the Status Register	MOVE SR,<ea>	—	—	—	—	—
MOVE USP	Move User Stack Pointer	MOVE USP,An MOVE An,USP	—	—	—	—	—
MOVEA	Move Address	MOVEA <ea>,An	—	—	—	—	—
MOVEM (see note)	Move Multiple Registers	MOVEM <register list>,<ea> MOVEM <ea>,<register list>	—	—	—	—	—
MOVEP	Move Peripheral Data	MOVEP Dx,d(Ay) MOVEP d(Ay),Dx	—	—	—	—	—
MOVEQ	Move Quick	MOVEQ #<data>,Dn	—	*	*	0	0
MULS	Signed Multiply	MULS <ea>,Dn	—	*	*	0	0
MULU	Unsigned Multiply	MULU <ea>,Dn	—	*	*	0	0
NBCD	Negate Decimal with Extend	NBCD <ea>	*	U	*	U	*
NEG	Two's Complement Negation	NEG <ea>	*	*	*	*	*
NEGX	Negate with Extend	NEGX <ea>	*	*	*	*	*
NOP	No Operation	NOP	—	—	—	—	—
NOT	Logical Complement	NOT <ea>	—	*	*	0	0
OR	Inclusive OR Logical	OR <ea>,Dn OR Dn,<ea>	—	*	*	0	0
ORI	Inclusive OR Immediate	ORI #<data>,<ea>	—	*	*	0	0
PEA	Push Effective Address	PEA <ea>	—	—	—	—	—
RESET	Reset External Devices	RESET	—	—	—	—	—
ROL, ROR	Rotate (without extend)	ROd Dx,Dy ROd #<data>,Dy ROd <ea>	—	*	*	0	*
ROXL, ROXR	Rotate with Extend	ROXd Dx,Dy ROXd #<data>,Dy ROXd <ea>	*	*	*	0	*
RTE	Return from Exception	RTE	*	*	*	*	*
RTR	Return and Restore Condition Codes	RTR	*	*	*	*	*
RTS	Return from Subroutine	RTS	—	—	—	—	—
SBCD	Subtract Decimal with Extend	SBCD Dy,Dx SBCD -(Ay),-(Ax)	*	U	*	U	*
Scc	Set according to Condition	Scc <ea>	—	—	—	—	—
STOP	Stop Program Execution	STOP #<data>	—	—	—	—	—
SUB	Subtract Binary	SUB <ea>,Dn SUB Dn,<ea>	*	*	*	*	*
SUBA	Subtract Address	SUBA <ea>,An	—	—	—	—	—
SUBI	Subtract Immediate	SUBI #<data>,<ea>	*	*	*	*	*
SUBQ	Subtract Quick	SUBQ #<data>,<ea>	*	*	*	*	*
SUBX	Subtract with Extend	SUBX Dy,Dx SUBX -(Ay),-(Ax)	*	*	*	*	*
SWAP	Swap Register Halves	SWAP Dn	—	*	*	0	0
TAS	Test and Set an Operand	TAS <ea>	—	*	*	0	0
TRAP	Trap	TRAP #<vector>	—	—	—	—	—
TRAPV	Trap on Overflow	TRAPV	—	—	—	—	—
TST	Test and Operand	TST <ea>	—	*	*	0	0
UNLK	Unlink	UNLK An	—	—	—	—	—

Bild 4. Befehlssatz der Prozessorfamilie 68000

bezieht sich die Startadresse auf den Abstand zum ersten Byte des Kopfes ($55). Setzt man vor den Kopf eine Marke (z. B.: HEAD) und vor den ersten Befehl des Programms auch eine (z. B.: BEGIN), dann läßt sich die Startadresse definieren als:

```
DC.L   BEGIN-HEAD
```

Bei absoluten Programmen ist die Startadresse gleich der absoluten Speicheradresse des Programmanfangs. EPROMs mit relokativen Programmen lassen sich also in jede beliebige Fassung stecken. Hat man mehrere kleine Programme in einem EPROM (oder im RAM), so kann man die Köpfe der weiteren Eintrittspunkte (Entries) hintereinandersetzen. Die letzte Langwortdefinition ist dann nicht 0,0 sondern wieder $55,... – bis alle Entries definiert sind.

Optionen

Was noch bleibt, ist das Optionenmenü. Hier kann man die Ausgabe von Assembler und Pascal auf den Drucker umleiten, die Startadresse des Textpuffers einstellen, die Bildschirmausgabe von 40 Zeichen auf 80 Zeichen pro Zeile umschalten und für den Einzelschrittbetrieb die Debug-Info ein- und ausschalten. So ist es möglich, bei genügend großem Speicher mehrere Programmquellen unabhängig im Speicher zu halten und zu editieren. Der Assembler und auch das Pascal-System beziehen sich auf den eingestellten Textstart, der beim Einschalten auf $9000 gesetzt wird. Bei dem großen Adreßraum des 68008 spricht übrigens nichts dagegen, auch Texte oder Quellprogramme auf EPROMs abzulegen.

Neuigkeiten

Für die SBC-II-Karte sind gerade zwei neue Programme erschienen: Ein Basic-Interpreter, der anstelle des Grundprogramms eingesetzt wird, und GOSI, eine grafikorientierte Sprache, die der Sprache Logo recht ähnlich ist. Beide Pro-

```
Rolf-D.Klein  68000/68008  Assembler 3.1 (C) 1983
009C00
00A800                    ORG #A800
00A800
00A800
00A800                    ASSINIT:
00A800  4EB9 00001162      JSR @CLRSCREEN
00A806  4240               CLR D0
00A808  4241               CLR D1
00A80A  4EB9 000022DC      JSR @SETFLIP
00A810  4E75               RTS
00A812
00A812                    ASSPLATTE:     ; D0= HOEHE, D1=X , D2= Y
00A812  41F9 0000A828      LEA SCHEIBE,A0
00A818  4EB9 00003A0C      JSR @FIGUR
00A81E  4E75              RTS
00A820
00A820                    ASSFEST:
00A820  4EB9 00003A04      JSR @SETFIG
00A826  4E75               RTS
00A828
00A828
00A828                    SCHEIBE:
00A828  00 00 02 04 04     DC.B 0,0,2,4,4,4,4,6,0,0,10
00A82D  04 04 06 00 00
00A832  0A
0000    Fehler entdeckt
008A52  Ende-Symboltabelle
```

Bild 5. Listing des Assemblerteils für das Programm „Türme von Hanoi", dessen Pascal-Teil in der nächsten Folge besprochen wird

grammpakete werden auf zwei EPROMs 2732 zusammen mit einem Handbuch geliefert und sind bei den Bausatzherstellern sowie vom Franzis-Software-Service erhältlich (Basic und GOSI je 75 DM). Es sollte hier einmal ganz deutlich gesagt werden, daß Sie den NDR-Klein-Computer auch bauen können, wenn Sie die Fernsehserie noch nicht gesehen haben. Zum einen gibt das Begleitmaterial genügend Hilfestellung, und zum anderen können Sie sich mit Fragen an den Software-Service wenden.

Fortsetzung folgt

400 Prozent Gewinn

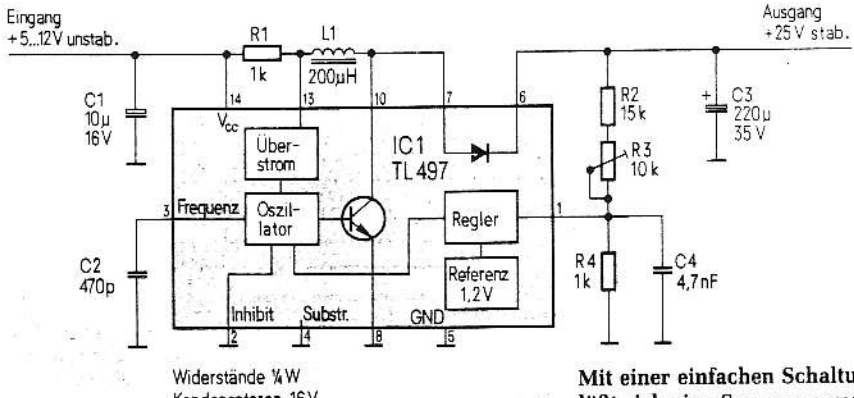
Beim Bau von EPROM-Programmiergeräten für den Anschluß an vorhandene Heimcomputer kommt sehr schnell die Frage auf, woher die 25-V-Spannung zur Programmierung zu beziehen ist. Ein separates Netzteil ist meist zu aufwendig und kommt daher genau so wenig in Betracht wie ein Anzapfen des vorhandenen Netzteils, da bei diesem die Spannung zu niedrig ist.

Die nachstehend beschriebene Schaltung (*Bild*) löst dieses Problem durch den Einsatz eines integrierten Schaltregler-ICs von Texas Instruments, das neben Referenzspannungserzeugung, Leistungstransistor, Diode und Regler alle erforderlichen aktiven Bauteile enthält. Der Eingang der Schaltung wird mit einer Spannung zwischen +5 V und +12 V verbunden, die nicht stabilisiert zu sein braucht. Je höher diese Spannung ist, umso größer ist auch die Belastbarkeit des Ausgangs und der Wirkungsgrad, wobei um die 80 % durchaus erreichbar sind.

Der Widerstand R1 verhindert zusammen mit der Überstromschutzschaltung eine Zerstörung des ICs durch Überstrom. Die Schwingfrequenz wird durch C2 bestimmt und beträgt ungefähr 20 kHz. Über R2...R4 wird die Ausgangsspannung der Schaltung an den Reglereingang geführt, wo sie mit der internen Referenzspannung von ca. 1,2 V verglichen wird. Da diese Spannung von IC zu IC unterschiedlich ist, erfolgt mit R3 ein Abgleich auf exakt 25 V Ausgangsspannung. Die beschriebene Schaltung wurde mit einem selbstgebauten Programmiergerät für 2716-EPROMs an einem Apple-II betrieben, wobei die Eingangsspannung der Floppy-Disk-Versorgung entnommen wurde.

Die Induktivität L1 besteht aus etwa 25 Windungen (nicht kritisch) Kupferlackdraht von 1 mm Ø auf einem Siferrit-Schalenkern von 14 × 8 mm und einem Al-Wert von 400 nH/W².

Harald Tillmann



Widerstände ¼ W
Kondensatoren 16 V
L1: ca. 25 Wdg. auf Siferrit-Schalenkern mit AL 400

Mit einer einfachen Schaltung läßt sich eine Spannung von 25 V erzeugen

Nur mit Codewort ladbar

Schreibt man beim VC-20 oder C-64 beispielsweise folgende Zeile mit 24 inversen T am Ende:

10 PRINT"LIST-TEST":REM"TTT...

dann wird die Zeile nach LIST nicht mehr auf dem Bildschirm erscheinen. Der Grund dafür ist, daß das inverse T denselben Code wie die Delete-Taste besitzt, so daß die entsprechende Zahl vorhergehender Zeichen gelöscht wird. Diesen Effekt kann man sich auch bei Disketten-Filenamen zunutze machen. Speichert man ein Programm mit

SAVE "ABCTTTTEST",8

auf Diskette ab, so erscheint bei der Directory-Anzeige nur TEST als Filename. LOAD"TEST",8 führt zu einem „File not found Error". Ein Laden ist nur mit dem Filenamen ABCTTTTEST möglich, worauf ein unbefugter kaum kommen wird.

Jürgen Bartz