Franz-Josef Katgely

mc-EPROM-Brenner für Einchip-Computer

Die mc-EMUFs sind so erfolgreich, weil man gängige Tischcomputer als Entwicklungssystem verwenden kann. Jetzt bekommen sie Konkurrenz aus dem eigenen Haus: Wir zeigen Ihnen nämlich, wie man mit dem mc-EPROM-Brenner (1) Einchip-Computer der MCS48-Familie programmiert.

Zwei typische Vertreter der MCS48-Familie, der 8748 und der 8749, die sich für die meisten Anwendungen eignen, haben 24 Ein-/Ausgabe-Leitungen, 1 bzw. 2 KByte Progammspeicher in Form eines EPROMS, 64 bzw. 128 Byte Datenspeicher (RAM), eine 8-Bit-CPU, einen Timer, einen Taktgenerator auf dem Chip und einen speziell auf Ein-/Ausgabe-Aufgaben abgestimmten Befehlssatz. Alle Assemblerbefehle werden in ein oder zwei Maschinenzyklen ausgeführt und garantieren dadurch eine schnelle Abarbeitung der einzelnen Befehle. Die Kosten für einen solchen Chip liegen für die wiederlöschbare Version mit Quarzfenster unter 30 DM und für die nur einmal programmierbare OTP-Version (One Time Programmable) im Plastikgehäuse ohne Quarzfenster unter 10 DM. Gerade der erstaunlich günstige Preis macht diese Chips für viele Anwender überaus interessant.

Arbeiten mit Einchip-Mikrocomputern

Cross-Assembler für Einchip-Mikrocomputer sind oft günstig verfügbar. Meist ist es auch möglich, einen bereits vorhandenen (Makro-)Assembler (z. B. MASM) so „umzustricken", daß er MCS48-Code erzeugt. Im Gegensatz zu Cross-Assemblern sind zur Programmierung von Einchip-Mikrocomputern geeignete Programmiergeräte leider

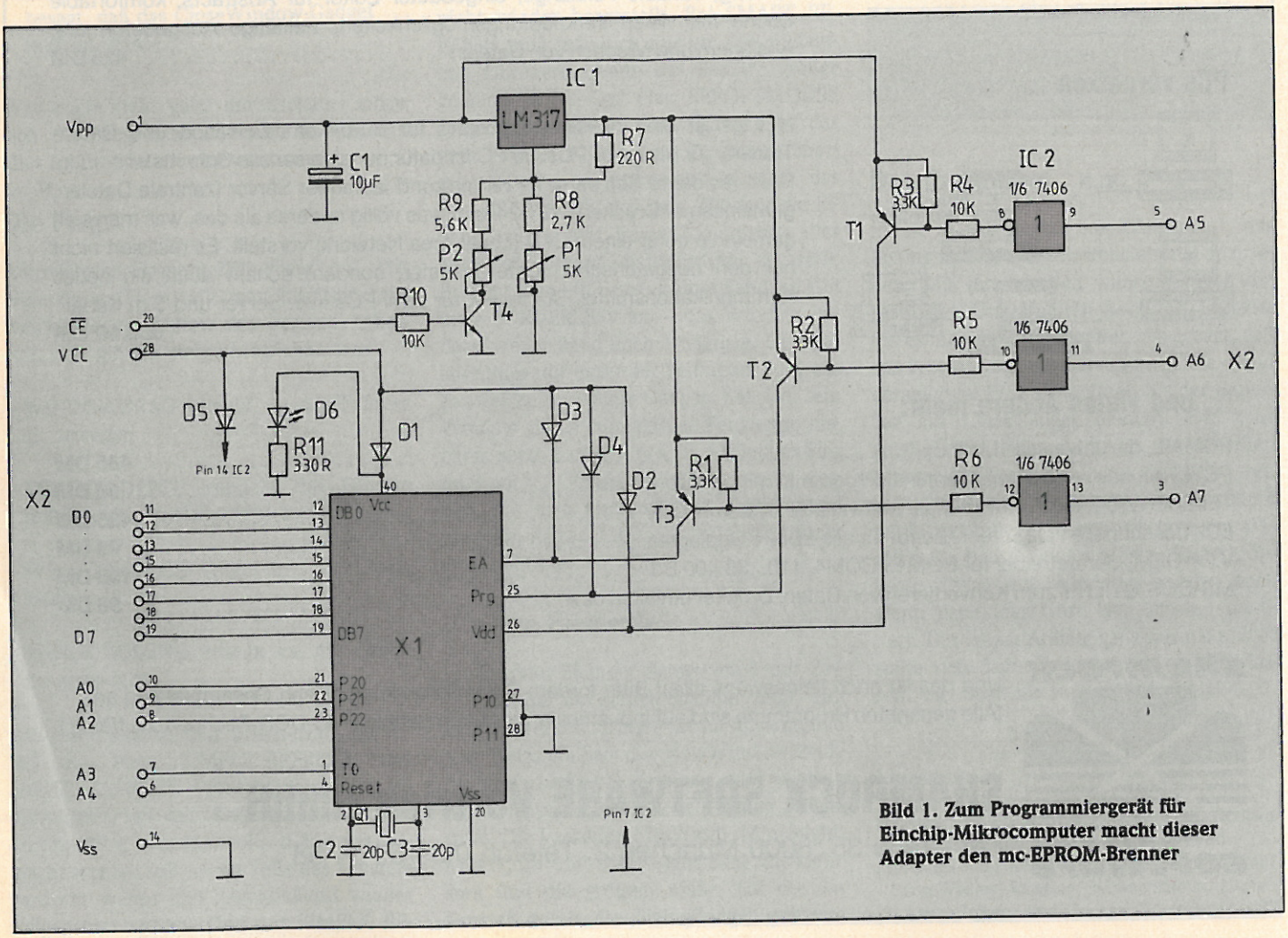

Bild 1. Zum Programmiergerät für Einchip-Mikrocomputer macht dieser Adapter den mc-EPROM-Brenner

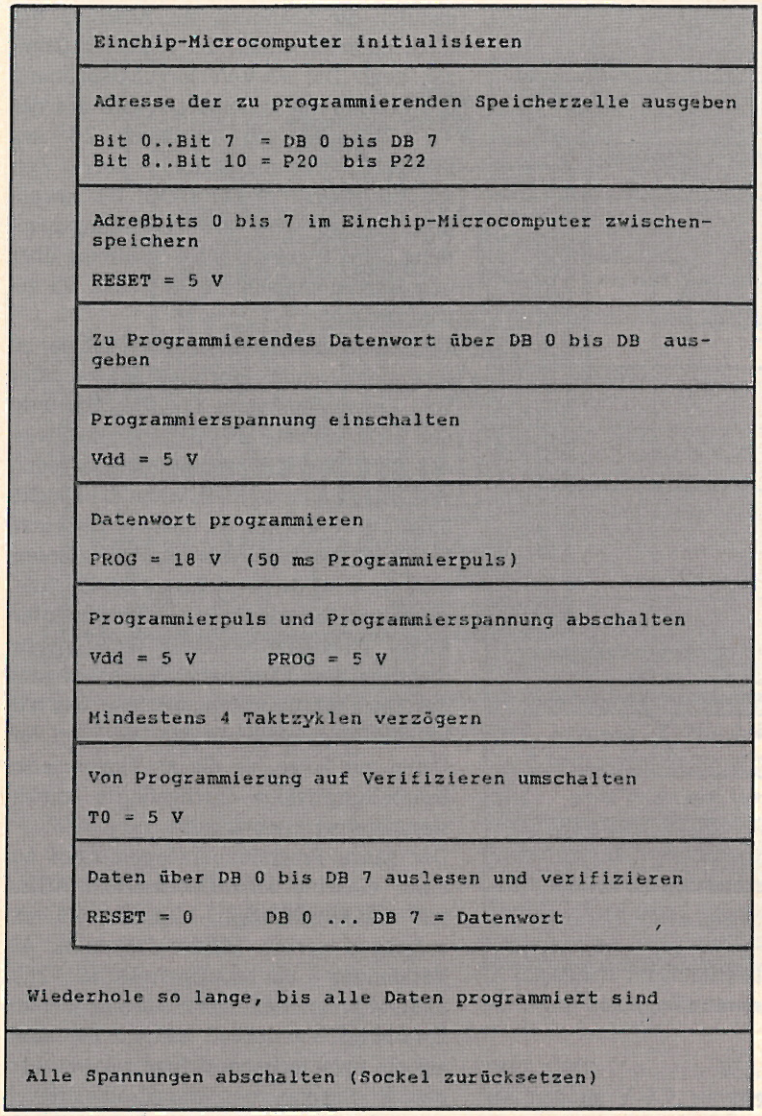

Bild 2. Programmieralgorithmus für die MCS48-Reihe

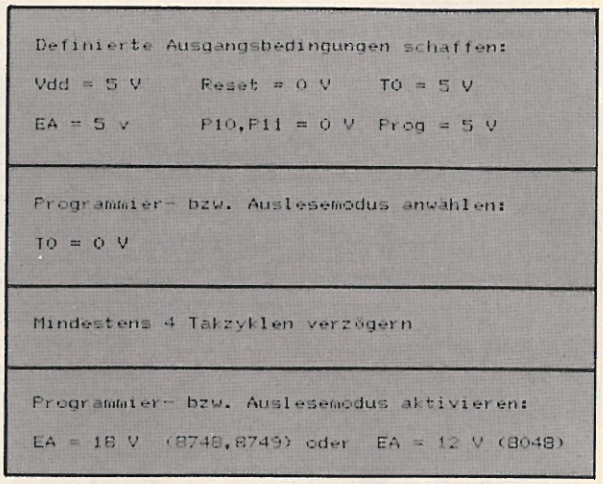

Bild 3. Struktogramm der MCS48-Initialisierung

nur zu haarsträubenden Preisen erhältlich, und oft sind diese Geräte dann nur ausschließlich zur Programmierung von Einchip-Mikrocomputern geeignet, d. h. es ist nicht möglich, gängige EPROMS damit zu programmieren. Die Preise für solche Geräte liegen im allgemeinen jenseits der 500-Mark-Schmerzschwelle. Billiger ist die Zusatzschaltung im *Bild 1.*

Ein Wort zu MCS48-Cross-Assembler

Im allgemeinen erzeugen diese Cross-Assembler als Ausgabedatei eine Datei im Intel-Hex-Format. Weil es sich hierbei um eine ASCII-Datei mit Adreßinformationen und Prüfsummen handelt, kann sie nicht direkt in den Programmspeicher der Einchip-Mikrocomputer programmiert werden. Diese Hex-Datei muß in eine Objekt-Datei umgewandelt werden. Ein für solche Umwandlungen gut geeignetes Turbo-Pascal-Programm wurde in [2] vorgestellt. Dieses Programm wandelt die Hex-Datei in eine direkt in den Einchip-Mikrocomputer programmierbare Objekt-Datei um.

Zur Programmierung von MCS48-Chips

Diese Einchip-Mikrocomputer müssen mit einem ganz speziellen Programmieralgorithmus programmiert werden und benötigen während der Programmierung zum Teil ungewöhnliche Spannungspegel. Der genaue Ablauf des Programmiervorgangs kann aus den in *Bild 2* und *Bild 3* abgedruckten Struktogrammen entnommen werden. Ferner wird auf das in *Bild 4* gezeigte Programmier-Timing-Diagramm hingewiesen.

Der Programmieradapter

Der beschriebene Progammieradapter (Bild 1) stellt eine Erweiterung des in [1] veröffentlichten mc-EPROM-Brenners dar. Die Adapterplatine wird einfach in den 28poligen EPROM-Programmiersockel eingesteckt und mit der geänderten Software betrieben. Eingriffe am EPROM-Programmiergerät selbst sind, abgesehen von einer Änderung der Ansprechgeschwindigkeit der Überstromsicherung, nicht nötig. Zur Verwendung mit anderen EPROM-Programmiergeräten kann der Programmieradapter ohne Änderung übernommen werden. Lediglich die nötige Treibersoftware muß dann, entsprechend dem verwendeten Programmiergerät, geändert bzw. neu geschrieben werden. Die mc-EPROM-Brenner-Routinen können hierfür dann nur als Anhalt dienen.

Änderung an der mc-EPROM-Brenner-Hardware

Leider läßt es sich nicht umgehen, die Ansprechgeschwindigkeit der Überstromsicherung des mc-EPROM-Brenners herabzusetzen. Dies geschieht dadurch, daß C10 des mc-EPROM-Brenners durch einen Kondensator von 5 µF ersetzt wird. Hierdurch wird zwar die Empfindlichkeit der Überstromsicherung vermindert, jedoch ist immer noch ein in den meisten Fällen ausreichender Schutz des zu programmierenden Chips gewährleistet. Hier sollte der Anwender sich jedoch nicht scheuen, eigene Versuche anzustellen, um den für seine Anordnung optimalen Wert von C10 zu ermitteln. Der Wert von 5 µF stellt den in der Regel optimalen Kompromiß zwischen hoher Ansprechgeschwindigkeit (Empfindlichkeit der Überstromsicherung) und hoher Störsicherheit (Sicherheit gegen zu sensibles Verhalten der Überstromsicherung bei der Programmierung) dar.

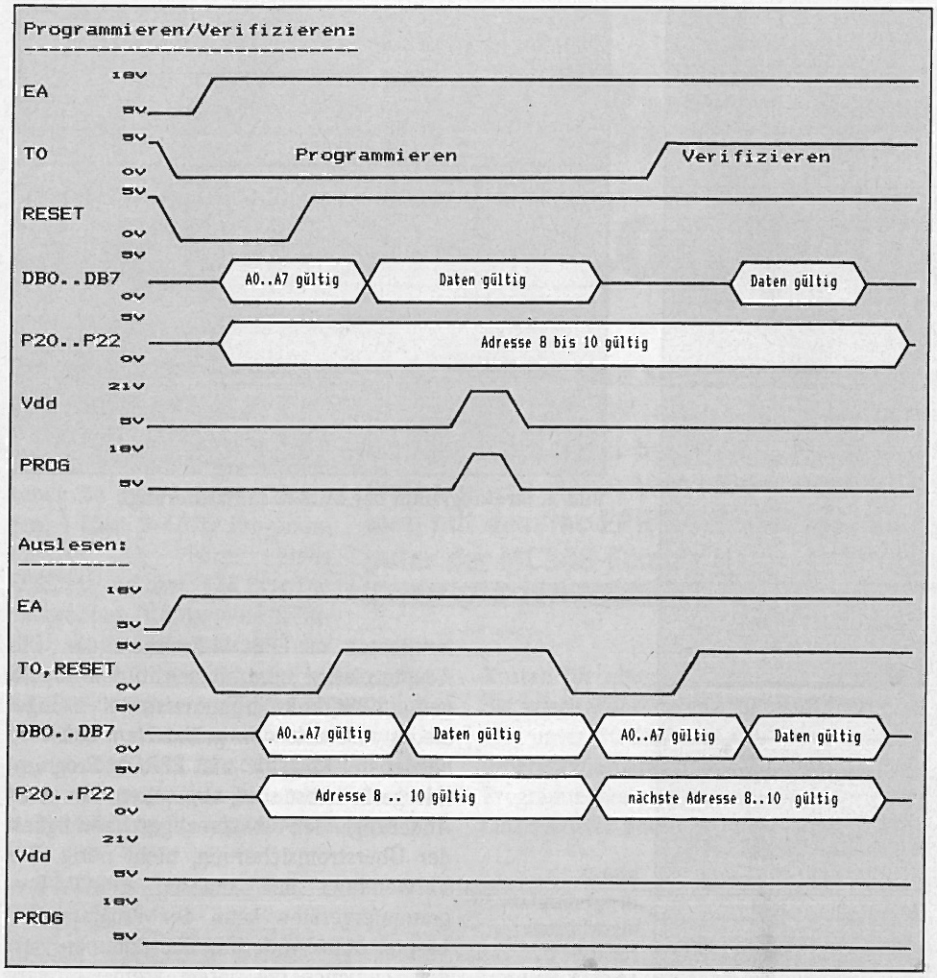

Bild 4. Zeitdiagramme für Programmieren/Verifizieren und Auslesen

Die Hardware im Detail

Alle Anschlußbezeichnungen des EPROM-Programmiergerätes beziehen sich auf die Anschlußbelegung des Typs 2764.

Die zur Bearbeitung des Typs MCS48 benötigten Spannungspegel werden durch IC1 (LM 317) erzeugt. Als Eingangsspannung für dieses IC dient die über Anschluß 1 des EPROM-Programmiersockels eingespeiste Programmspannung von 21 V. Mit Hilfe des Transistors T4 wird die Ausgangsspannung von IC1 zwischen 12 V (Auslesen/Verifizieren des 8048) und 18 V (Programmieren/Verifizieren/Auslesen des 8748 bzw. 8749) umgeschaltet. T4 erhält sein Eingangssignal über den Anschluß 20 des EPROM-Programmiergerätes (CE-Anschluß am EPROM-Programmiergerät).

Die Signale für die Anschlüsse V_{dd}, EA und Prog des Einchip-Mikrocomputers werden über die Transistoren T1...T3 an die entsprechenden Anschlüsse des Einchip-Programmiersockels X1 geschaltet. Die Transistoren erhalten ihr Eingangssignal aus OC-Invertern von IC2 (7406). Die Dioden D2...D4 verhindern, daß bei Signalspannungen von mehr als 5,5 V an den Anschlüssen V_{dd}, EA und Prog diese Signalspannungen über die +5-V-Versorgungsspannung kurzgeschlossen werden.

Die Datenleitungen D0...D7 des EPROM-Programmiergerätes werden direkt mit den entsprechenden Anschlüssen des Einchip-Mikrocomputer-Programmiersockels verbunden. Ebenso werden die Adreßleitungen A0...A2 des EPROM-Programmiergerätes direkt an die Anschlüsse P20...P22 des Programmiersockels angeschlossen.

Die Adreßleitungen A3...A4 des Programmiergerätes werden an die Anschlüsse T0 und RESET des Einchip-Mikrocomputer-Programmiersockels geführt.

Die Anschlüsse P10 und P11 müssen laut Datenblatt der Firma Intel auf Massepegel liegen.

Der zu programmierende Einchip-Mikrocomputer wird über Anschluß 28 des EPROM-Programmiergerätes mit Betriebsspannung versorgt. Die Spannung am Anschluß 28 des EPROM-Programmiergerätes muß 6 V betragen, da die Rückstrom-Schutzdiode D2, je nach Stromfluß, einen erhöhten Spannungsabfall verursacht und dadurch die Spannung am Anschluß V_{dd} nicht mehr innerhalb der erlaubten Grenzen liegt. Die +5-V-Versorgungsspannung des Einchip-Mikrocomputers wird aus der +6-V-Spannung am Anschluß 28 des EPROM-Programmiergerätes über den Spannungsabfall an der Diode D1 gewonnen. Ebenso wird die Versorgungsspannung für IC2 aus der +6-V-Spannung über den Spannungsabfall an der Diode D5 gewonnen.

Die Leuchtdiode D6 signalisiert, daß der zu programmierende Einchip-Mikrocomputer vom Programmiergerät angesprochen wird und nicht aus dem Sockel entnommen werden darf. Der Mikrocomputer kann jederzeit entnommen werden, wenn die Leuchtdiode D6 verloschen ist. Sollte er bei leuchtender Diode doch versehentlich entnommen werden, darf er, solange D6 leuchtet, auf keinen Fall wieder in den Programmiersockel eingesetzt werden! Wird hierauf nicht geachtet, hätte das eine Beschädigung des Einchip-Mikrocomputers bzw. das fehlerhafte Programmieren oder Auslesen von Daten zur Folge, da die zu Beginn jedes Bearbeitungszyklus durchgeführte Initialisierung unwirksam würde.

Der ganze Programmieradapter kann auf einer einseitig kupferkaschierten Platine mit den Abmessungen 75 mm × 85 mm aufgebaut werden (*Bilder 5* und *6*). Als Verbindung der Adapterplatine zum EPROM-Programmiergerät-Sockel können 2,54-mm-Steckerleisten dienen, die dann auf die entsprechende Länge gekürzt werden. Hier sollten, um Kontaktschwierigkeiten zu vermeiden, nach Möglichkeit Steckleisten mit vergoldeten Kontakten Anwendung finden.

Als Programmiersockel (X1) sollte unbedingt ein Nullkraftsockel, am besten ein 40poliger Textool-Sockel, verwendet werden.

Abgleich des Programmieradapters

Setzen Sie während des Abgleiches keinen Chip in die Fassung des Programmieradapters! Eine Beschädigung wäre nicht auszuschließen! Zum Abgleich sind folgende Schritte nötig:

1. mc-EPROM-Brenner in Modus B schalten.
2. Adapterplatine in den EPROM-Sockel einsetzen.
3. Schließen Sie ein Voltmeter zwischen Anschluß 20 und Anschluß 7 des Einchip-Mikrocomputer-Programmiersokkels X1 an. Der Meßbereichsendwert des Meßgerätes sollte größer oder gleich 20 V sein.

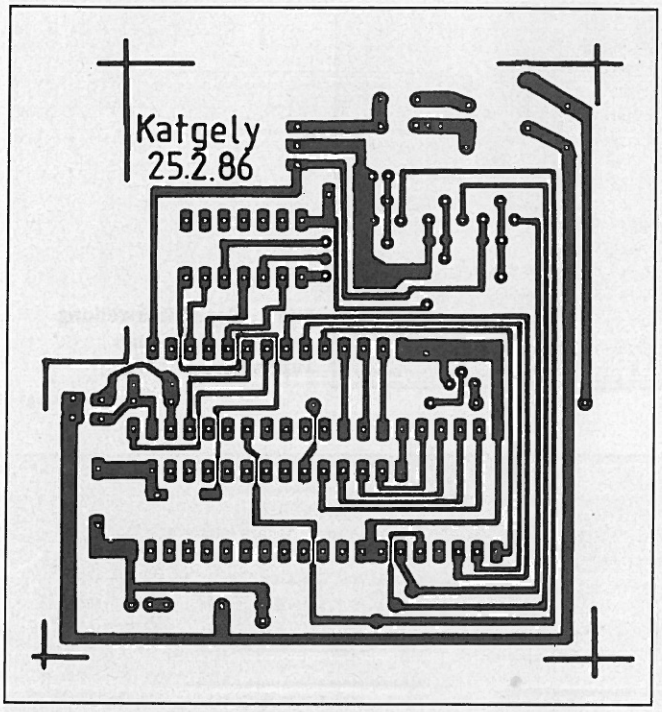

Bild 5. Platinen-Layout für den Programmieradapter

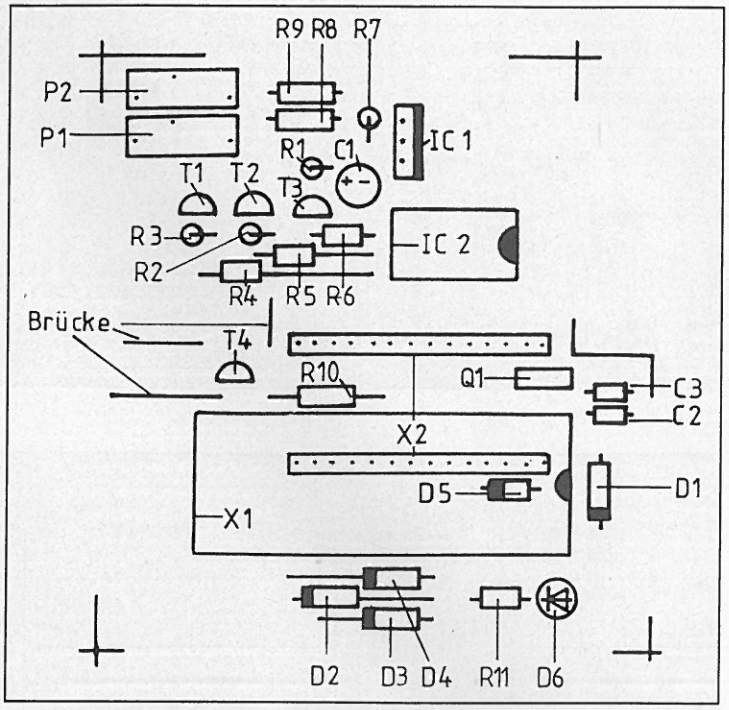

Bild 6. Bestückungsplan des Programmieradapters

4. Starten Sie das geänderte Turbo-Pascal-Programm.
5. Geben Sie ein: T 60
6. Geben Sie ein: F 0 B 2000 M FF
7. Geben Sie ein: P 0
8. Stellen Sie, während der Programmiervorgang läuft, mit P1 eine Spannung von 18 V ein. Sollte die Programmierung abgeschlossen sein, bevor Sie die korrekte Spannung eingestellt haben, wiederholen sie einfach die Schritte 7 und 8.
9. Nach abgeschlossener Programmierung und korrekter Einstellung geben Sie ein: T 62
10. Geben Sie ein: P 0
11. Stellen Sie mit P2 eine Spannung von 12 V ein. Sollte die Programmierung abgeschlossen sein, bevor Sie die korrekte Spannung eingestellt haben, wiederholen Sie einfach die Schritte 10 und 11.

Damit ist der Abgleich der Hardware abgeschlossen.

Die Treibersoftware

Zur Programmierung von MCS48-Einchip-Mikrocomputern muß sich der mc-EPROM-Brenner in jedem Fall in Modus B befinden. Dadurch beziehen sich alle hier beschriebenen Änderungen des Programmes auf die Original-Programmversion aus der mc 1/87 für Modus B. Die Besitzer des mc-EPROM-Brenners müssen lediglich die alten Programmteile gegen die geänderten Programmteile austauschen, die neu hinzugekommenen Funktionen und Prozeduren einfügen und die Pascal-Quelldatei neu compilieren.
In folgenden Programmteilen wurden Änderungen durchgeführt:

TYPE_CMD	READ_CMD
TEST_CMD	IDENTIFY_CMD
PROGRAM_CMD	VERIFY_CMD

Ferner wurden Änderungen in den zum Hauptprogramm MAIN gehörigen Deklarationen durchgeführt.
Für die Besitzer anderer EPROM-Programmiergeräte wird es schwieriger werden, die entsprechenden Änderungen durchzuführen. Sie sind gezwungen, die entsprechenden Routinen selbst zu schreiben.
Zur Anpassung der Software an andere EPROM-Programmiergeräte sollten die entsprechenden Datenblätter der Firma Intel zu Rate gezogen werden.

Die Software im Detail

Folgende Änderungen wurden in den Deklarationen des Hauptprogramms vorgenommen:

1. Die Felder RD, VER, PROG, PINH des Records EPROM_DATA wurden von array [3..4] of byte auf array [2...4] of byte geändert.
2. Der Typendeklarationsteil des Hauptprogramms wurde um die Deklaration des Typs MODI = (MCS48, EPROMS); erweitert.
3. Der Variablendeklarationsteil des Hauptprogramms wurde um die Deklaration der Variablen MODE : MODI;
   PRIMARAY : boolean; erweitert.
   Hierbei gibt die Variable MODE an, ob EPROMs oder Einchip-Mikrocomputer bearbeitet werden sollen. Die Variable PRIMARY gibt an, ob die für die MCS48-Einchip-Mikrocomputer erforderliche Initialisierung bereits durchgeführt wurde (PRIMARY = false) oder ob diese Initialisierung noch durchgeführt werden muß (PRIMARY = true).
4. Die Konstante EPROM_TYPES wurde von 18 auf 21 erhöht.
5. Das Array KNOWN_EPROMS wurde gemäß *Bild 7* geändert und erweitert. Dabei ist zu beachten, daß bei allen EPROM-Typen dieses Arrays die Felder RD, VER, PROG, PINH um $00 erweitert werden müssen. In Bild 7 ist diese Änderung am Beispiel des EPROMs 2764 gezeigt. Die dort vorgenommenen Änderungen müssen analog bei allen anderen EPROMs vorgenommen werden. Neu zu der Deklaration des Array KNOWN_EPROMs hinzugekommen sind die Elemente 8048, 8748 und 8749. Eine komplette Auflistung der Typenbeschreibungen aller bekannten EPROMs und Einchip-Mikrocomputer im neuen Format zeigt *Bild 8.*

```
(TYP  : '2764              '; LEN  : $2000       ;   TIME : 50          ;
 TNR  : '30'               ; VPP  : '21 V  '     ;   PGMD : 'N'         ;
                             RD   : ($C0,$10)    ;   VER  : ($40,$50)   ;
                             PROG : ($00,$56)    ;   PINH : ($40,$13) ),

Alte EPROM-Typenbeschreibung.

(TYP : '2764               '; LEN : $2000        ; TIME: 50             ;
 TNR : '30'                ; VPP : '21 V  '      ; PGMD: 'N'            ;
                             RD  : ($00,$C0,$10); VER : ($00,$40,$50)   ;
                             PROG: ($00,$00,$56); PINH: ($00,$40,$13)   ),

Neue EPROM-Typenbeschreibung.
```

Bild 7. Gegenüberstellung der alten und neuen Typenbeschreibung

```
const   EPROM_TYPES = 21;  (* Anzahl der bekannten EPROMS u. SINGLECHIPS *)
        KNOWN_EPROMS : array [1..EPROM_TYPES] of EPROM_DATA =

((TYP  : '2716          ';   LEN  : $0800       ;   TIME : 00          ;
  TNR  : '10'           ;   VPP  : '      '     ;   PGMD : 'Z'         ;
                            RD   : ($00,$28,$00);   VER  : ($00,$00,$00) ;
                            PROG : ($00,$00,$00);   PINH : ($00,$00,$00) ),

 (TYP  : '2732          ';   LEN  : $1000       ;   TIME : 50          ;
  TNR  : '20'           ;   VPP  : '25 V  '     ;   PGMD : 'O'         ;
                            RD   : ($00,$20,$00);   VER  : ($00,$20,$18) ;
                            PROG : ($00,$20,$3C);   PINH : ($00,$20,$39) ),

 (TYP  : '2732A         ';   LEN  : $1000       ;   TIME : 50          ;
  TNR  : '21'           ;   VPP  : '21 V  '     ;   PGMD : 'O'         ;
                            RD   : ($00,$20,$00);   VER  : ($00,$20,$00) ;
                            PROG : ($00,$20,$24);   PINH : ($00,$20,$21) ),

 (TYP  : '2764          ';   LEN  : $2000       ;   TIME : 50          ;
  TNR  : '30'           ;   VPP  : '21 V  '     ;   PGMD : 'N'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$50) ;
                            PROG : ($00,$00,$56);   PINH : ($00,$40,$13) ),

 (TYP  : '2764          ';   LEN  : $2000       ;   TIME : 1           ;
  TNR  : '31'           ;   VPP  : '12,5 V'     ;   PGMD : 'A'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$C8) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$40,$8B) ),

 (TYP  : '2764          ';   LEN  : $2000       ;   TIME : 1           ;
  TNR  : '32'           ;   VPP  : '21 V  '     ;   PGMD : 'B'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$48) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$40,$0B) ),

 (TYP  : '2764          ';   LEN  : $2000       ;   TIME : 1           ;
  TNR  : '33'           ;   VPP  : '12,5 V'     ;   PGMD : 'C'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$C8) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$40,$8B) ),

 (TYP  : '2764          ';   LEN  : $2000       ;   TIME : 1           ;
  TNR  : '34'           ;   VPP  : '21 V  '     ;   PGMD : 'C'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$48) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$40,$0B) ),

 (TYP  : '27128         ';   LEN  : $4000       ;   TIME : 50          ;
  TNR  : '40'           ;   VPP  : '21 V  '     ;   PGMD : 'N'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$50) ;
                            PROG : ($00,$00,$56);   PINH : ($00,$40,$13) ),

 (TYP  : '27128         ';   LEN  : $4000       ;   TIME : 1           ;
  TNR  : '41'           ;   VPP  : '12,5 V'     ;   PGMD : 'A'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$C8) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$40,$8B) ),

 (TYP  : '27128         ';   LEN  : $4000       ;   TIME : 1           ;
  TNR  : '42'           ;   VPP  : '21 V  '     ;   PGMD : 'B'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$48) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$40,$0B) ),

 (TYP  : '27128         ';   LEN  : $4000       ;   TIME : 1           ;
  TNR  : '43'           ;   VPP  : '12,5 V'     ;   PGMD : 'C'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$C8) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$40,$8B) ),

 (TYP  : '27128         ';   LEN  : $4000       ;   TIME : 1           ;
  TNR  : '44'           ;   VPP  : '21 V  '     ;   PGMD : 'C'         ;
                            RD   : ($00,$C0,$10);   VER  : ($00,$40,$48) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$40,$0B) ),

 (TYP  : '27256         ';   LEN  : $8000       ;   TIME : 50          ;
  TNR  : '50'           ;   VPP  : '21 V  '     ;   PGMD : 'N'         ;
                            RD   : ($00,$80,$10);   VER  : ($00,$00,$51) ;
                            PROG : ($00,$00,$56);   PINH : ($00,$00,$17) ),

 (TYP  : '27256         ';   LEN  : $8000       ;   TIME : 1           ;
  TNR  : '51'           ;   VPP  : '12,5 V'     ;   PGMD : 'A'         ;
                            RD   : ($00,$80,$10);   VER  : ($00,$00,$C9) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$00,$8F) ),

 (TYP  : '27256         ';   LEN  : $8000       ;   TIME : 1           ;
  TNR  : '52'           ;   VPP  : '21 V  '     ;   PGMD : 'B'         ;
                            RD   : ($00,$80,$10);   VER  : ($00,$00,$49) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$00,$0F) ),

 (TYP  : '27256         ';   LEN  : $8000       ;   TIME : 1           ;
  TNR  : '53'           ;   VPP  : '12,5 V'     ;   PGMD : 'C'         ;
                            RD   : ($00,$80,$10);   VER  : ($00,$00,$C9) ;
                            PROG : ($00,$00,$CE);   PINH : ($00,$00,$8F) ),

 (TYP  : '27256         ';   LEN  : $8000       ;   TIME : 1           ;
  TNR  : '54'           ;   VPP  : '21 V  '     ;   PGMD : 'C'         ;
                            RD   : ($00,$80,$10);   VER  : ($00,$00,$49) ;
                            PROG : ($00,$00,$4E);   PINH : ($00,$00,$0F) ),

 (TYP  : '8048          ';   LEN  : $0400       ;   TIME : 00          ;
  TNR  : '60'           ;   VPP  : '      '     ;   PGMD : 'R'         ;
                            RD   : ($80,$81,$4D);   VER  : ($9B,$81,$4D) ;
                            PROG : ($F0,$81,$4D);   PINH : ($80,$81,$4D) ) ,

 (TYP  : '8748          ';   LEN  : $0400       ;   TIME : 50          ;
  TNR  : '61'           ;   VPP  : '21 V  '     ;   PGMD : 'N'         ;
                            RD   : ($80,$81,$4C);   VER  : ($9B,$81,$4C) ;
                            PROG : ($F0,$81,$4C);   PINH : ($80,$81,$4C) ) ,

 (TYP  : '8749          ';   LEN  : $0800       ;   TIME : 50          ;
  TNR  : '62'           ;   VPP  : '21 V  '     ;   PGMD : 'N'         ;
                            RD   : ($80,$81,$4C);   VER  : ($9B,$81,$4C) ;
                            PROG : ($F0,$81,$4C);   PINH : ($80,$81,$4C) ));
```

Bild 8. Typenbeschreibung der bekannten Bausteintypen

```
Änderungen in TYPE_CMD:

     - Deklaration der Variablen TYPNUMBER im Typendeklarationsteil

          TYPNUMBER : integer;

     - Einfügen der Anweisungen

          val (KNOWN_EPROMS [W].TNR, TYPNUMBER, E1);
          case TYPNUMBER of
            60..69 : MODE := MCS48;
            else MODE := EPROMS;
          end; (* case *)

       vor die Anweisung

          if EPROM.LEN = $8000 ....

Damit sind die Änderungen an der Prozedur TYPE_CMD abgeschlossen.
```

```
Änderungen in TEST_CMD:

     - Deklaration der Variablen EMPTY im Variablendeklarationsteil.

          EMPTY : byte;

     - Einfügen der Anweisung

          if MODE = MCS48 then EMPTY := $00;
               else EMPTY := $FF;

       vor die DISP(15,11......) Anweisung.

     - Ersetzen der Anweisung

          DATA_EPROM := GET_EPROM (I);

       durch die Anweisung

          case MODE of
            MCS48 : DATA_EPROM := GET_MCS48 (I);
            EPROMS: DATA_EPROM := GET_EPROM (I);
          end; (* case *)

Weitere Änderungen müssen im TEST_CMD nicht durchgeführt werden.
```

```
Änderungen in VERIFY_CMD:

     - Ersetzen der Anweisung

          DATA_EPROM := GET_EPROM (I);

       durch die Anweisung

          case MODE of
            MCS48 : DATA_EPROM := GET_MCS48 (I);
            EPROMS: DATA_EPROM := GET_EPROM (I);
          end; (* case *)

  Damit sind alle nötigen Änderungen an VERIFY_CMD durchgeführt.
```

```
Änderungen in PROGRAM_CMD:

     - Ersetzen der Anweisung

          DATA_EPROM := PUT_EPROM (I, DATA_RAM);
```

Bild 9. Änderungen an der Original-Software

durch die Anweisung

```
case MODE of
  MCS48 : DATA_EPROM := PUT_MCS48 (I, DATA_RAM);
  EPROMS: DATA_EPROM := PUT_EPROM (I, DATA_RAM);
end; (* case *)
```

Diese Änderunge bezieht sich nur auf den 'N'-Fall (Normale Programmierung) der schon im PROGRAM_CMD vorhandenen case-Anweisung.

- Erweitern der übergeordneten case-Anweisung um den Fall 'R' (ROM - Version. Sie kann nur gelesen , aber nicht programmiert werden !).

```
'R': begin
       clrscr; write($07); gotoxy (18,10);
       writeln('*********************************');
       gotoxy (18,11);
       writeln('*       ROM-VERSION !!          *');
       gotoxy (18,12);
       writeln('*  Programmierung unmöglich ! *');
       gotoxy (18,13);
       writeln('*********************************');
       delay (1000);
       goto END1; (* Fehler ! Beende PROGRAM_CMD ! *)
```

- Einfügen der Anweisung

```
PRIMARY := true;
```

hinter das end; der übergeordneten case-Anweisung, unmittelbar vor der Anweisung gatoxy (1,19);.

- Ersetzen der Anweisung

```
DATA_EPROM := GET_EPROM (I);
```

im letzten Teil des PROGRAM_CMD's (12. Zeile von unten ab END1) durch die Anweisung

```
case MODE of
  MCS48 : DATA_EPROM := GET_MCS48 (I);
  EPROMS: DATA_EPROM := GET_EPROM (I);
end; (* case *)
```

Weitere Änderungen müssen im PROGRAM_CMD nicht vorgenommen werden.

Änderungen im IDENTIFY_CMD:

- Einfügen der Anweisung

```
case MODE of
  EPROMS: begin
```

als erste Anweisung in der Prozedur IDENTIFY_CMD.

- Einfügen der Zeilen

```
    end;
    MCS48 : begin
              clrscr; write($07); gotoxy (18,10);
              writeln ('*********************************');
              gotoxy (18,11);
              writeln ( *      MCS48-Einchip-         * );
              gotoxy (18,12);
              writeln ( *  Mikrocomputer können nicht  *');
              gotoxy (18,13);
              writeln ( *    identifiziert werden !    *');
              gotoxy (18,14);
              writeln ( *********************************');
              delay(1000);
            end;
  end; (* case *)
```

vor das die Prozedur IDENTIFY_CMD abschließende end; .

```
procedure INIT_MCS48;
begin
  OUT_BYTE [1] := $00;
  OUT_BYTE [2] := ((EPROM.RD [2] or $08) and $7f);
  OUT_BYTE [3] := EPROM.RD [3];
  OUT_BYTE [4] := EPROM.RD [4];
  PUT_DATA;
  OUT_BYTE [2] := OUT_BYTE [2] or $80;  (* EA auf 18 V *)
  PUT_DATA;
  PRIMARY := false;
end;

function GET_MCS48 (ADR1 : real) : byte;
const   LATCH_READ = $18;
var     ADR : integer;
begin
  ADR := trunc (ADR1);
  if PRIMARY then INIT_MCS48;   (* Wurde schon initialisiert ? *)
  OUT_BYTE [1] := lo (ADR);
  OUT_BYTE [2] := hi (ADR) or EPROM.RD [2];
  OUT_BYTE [3] := EPROM.RD [3];
  OUT_BYTE [4] := EPROM.RD [4];
  PUT_DATA;
  OUT_BYTE [2] := OUT_BYTE [2] or LATCH_READ;
  PUT_DATA;
  OUT_BYTE [4] := OUT_BYTE [4] and $FB;
  PUT_DATA;
  GET_MCS48 := GET_DATA;
end;

function PUT_MCS48 (ADR1 : real; Data : byte): byte;
const    LATCH_WRITE = $10;
var      ADR : integer;
begin
  ADR := trunc (ADR1);
  if PRIMARY then INIT_MCS48; (* Wurde schon initialisiert ? *)
  OUT_BYTE [1] := lo (ADR);
  OUT_BYTE [2] := hi (ADR) or EPROM.PINH [2];
  OUT_BYTE [3] := EPROM.PINH [3];
  OUT_BYTE [4] := EPROM.PINH [4];
  PUT_DATA;
  OUT_BYTE [2] := OUT_BYTE [2] or LATCH_WRITE;
  PUT_DATA; (* A0 bis A7 im Singelchip zwischenspeichern *)
  OUT_BYTE [1] := DATA;
  OUT_BYTE [2] := hi (ADR) or EPROM.PROG [2];
  OUT_BYTE [3] := EPROM.PROG [3];
  OUT_BYTE [4] := EPROM.PROG [4];
  PUT_DATA; (* Datenbyte in Singlechip brennen *)
  delay (EPROM.TIME);
  OUT_BYTE [2] := EPROM.PINH [2] or LATCH_WRITE;(* Prog.puls aus *)
  PUT_DATA;
  OUT_BYTE [2] := OUT_BYTE [2] or $08; (* TO auf H *)
  OUT_BYTE [4] := OUT_BYTE [4] and $FB;
  PUT_DATA;
  PUT_MCS48 := VERIFY_MCS48 (ADR);
end;
```

Bild 10. Zusätzliche Prozeduren und Funktionen

```
IC 1            LM 317
IC 2            7406

T1..T3          BC 327
T4              BC 546

D1, D2          1 N 4002
D3..D5          1 N 4148
D6              LED 5mm rot

C1              TANTALELKO 10 uF, 35 V
C2,C3           20 pF, keramisch

R1..R3          3,3 K, 1/4 Watt
R4..R6          10  K, 1/4 Watt
R7              220 R, 1/4 Watt
R8              2,7 K, 1/4 Watt
R9              5,6 K, 1/4 Watt
R10             10  K, 1/4 Watt
R11             330 R, 1/4 Watt
P1              5 K, Spindeltrimmer
P2              5 K, Spindeltrimmer

Q1              Quarz 3 MHz

X1              Textoolsockel 40 polig
X2              Pfostensteckerleisten 2,54 mm
                einfach, 2x14 Kontakte vergoldet

1 x 14poliger DIL-Sockel
1 x Platine gemäß Layout Bild 8
```

Bild 11. Mit diesen Bauteilen wird der Programmieradapter aufgebaut

6. Die Deklaration der Konstanten EMPTY entfällt. An ihre Stelle tritt eine in der Prozedur TEST_CMD deklarierte Variable EMPTY. Diese Änderung wurde dadurch nötig, daß der Programmspeicher der MCS48-Einchip-Mikrocomputer im unprogrammierten Zustand $00 enthält, wogegen unprogrammierte EPROMs $FF enthalten. Um eine einwandfreie Erkennung gelöschter Bausteine in jeder Betriebsart zu gewährleisten, muß je nach Betriebsmode (MCS48, EPROMs) ein anderes EMPTY-Muster verwendet werden.

Bild 9 zeigt die restlichen Änderungen an der Software. Die Prozedur- und Funktionsdeklarationen wurden um die Prozedur INIT_MCS48 und die Funktionen GET_MCS48, PUT_MCS48 und VERIFY_MCS48 erweitert. Das Listing der neuhinzugefügten Funktionen kann *Bild 10* entnommen werden. Beim Einbau der neuen Prozeduren bzw. Funktionen sind die Prozedur INIT_MCS48 und die Funktion GET_MCS48 vor die Funktion GET_EPROM und die Funktionen VERIFY_MCS48 und PUT_MCS48 vor die Funktion PUT_EPROM in den Quellcode einzubinden.

Inbetriebnahme

Sind alle Änderungen im Programm vorgenommen und ist das geänderte Programm compiliert, die Hardware aufgebaut und abgeglichen, dann steht einer Inbetriebnahme nichts mehr im Wege. Um unnötigen Ärger zu vermeiden, empfiehlt es sich jedoch, nicht sofort einen Einchip-Mikrocomputer in den Programmieradapter einzusetzen und „loszuprogrammieren". Vielmehr ist es sinnvoll, den EPROM-Brenner erst einmal ohne eingesetzten Baustein zu betreiben und verschiedene Spannungspegel zu überprüfen. So sollten Sie überprüfen, ob die an Anschluß 7 anliegende Spannung beim Betrieb nicht über 12 V (für 8048/8049) bzw. 18 V (für 8748/8749) liegt. Ebenso empfiehlt sich eine Kontrolle der Betriebsspannung am Anschluß 40. Diese Spannung sollte nicht über 5,5 V liegen.

Literatur

[1] *Seng, Peter:* mc-EPROM-Brenner, mc 1987, Heft 1, Seite 46...58.

[2] *Plate, Jürgen:* Sichere Übertragung. mc 1987, Heft 3, Seite 104..108.

[3] *Pelka, Horst:* Der Ein-Chip-Mikrocomputer, Franzis-Verlag.

[4] Microcontroller Handbook, Fa. Intel.