Rüdiger Bäcker

# EPROMs schnell programmiert

## Zeit sparen beim 68008-NDR-Computer

Für die Programmierung von EPROMs existieren unterschiedliche Methoden. Hier wird ein sehr schneller Algorithmus für den NDR-Klein-Computer vorgestellt. Mit diesem Algorithmus lassen sich alle Intel-kompatiblen Typen 2732 und 2764 mit großer Zuverlässigkeit programmieren. Die Programmierzeit beträgt für ein 2764 nur noch rund 30 Sekunden statt der bisher benötigten Zeit von etwa acht Minuten. Für den Betrieb mit der PROMmer-Baugruppe ist keinerlei Hardwareänderung erforderlich.

Es gibt mittlerweile mehrere Methoden, um ein EPROM mit den gewünschten Bytes zu „füttern". Wir wollen sie uns einmal anhand eines 2764 ansehen. Das 2764 ist ein EPROM mit einer Speicherkapazität von 8192 Bytes, also 8 KByte. Aus der Anschlußbelegung erkennen wir, daß 28 Pins nach außen geführt sind. Im einzelnen sind 8 Datenleitungen (D0–D7), 12 Adreßleitungen (A0–A12), eine Leitung für die 5-V-Versorgung, eine für die Masse und verschiedene Leitungen für die Steuerung von Programmierung und Leseoperationen vorhanden. Pin 26 ist nicht belegt, hier liegt bei einem EPROM mit höherer Speicherkapazität (27128) eine weitere Adreßleitung.

Die Steuerleitungen haben folgende Bedeutung:

- $\overline{CE}$: Pin 20 ist der Auswahleingang, ist er auf logisch 0, so werden die Ein-/Ausgangstreiber des EPROMs aktiviert.
- $\overline{OE}$: Pin 22 ist der Richtungseingang, er wird mit $\overline{CE}$ zusammen benötigt. Ist der Eingang auf 0, so sollen Daten gelesen werden.
- $\overline{PGM}$: Pin 27 ist der Eingang für den Programmierimpuls.
- Vpp: Pin 1 ist der Eingang für die Programmierspannung.

Daran, daß für die Programmierspannung ein eigener Eingang vorhanden ist, kann man erkennen, daß die Programmierspannung nicht identisch mit der Versorgungsspannung ist. Das hängt mit dem grundsätzlichen Aufbau eines EPROMs zusammen. Jede Speicherzelle besteht aus zwei NMOS-Transistoren; sie sind mit den Wortleitungen einerseits und mit den Bitleitungen andererseits verbunden. Wird nun eine im Verhältnis zur Versorgungsspannung wesentlich höhere Programmierspannung angelegt, so fließen durch den sogenannten Lawineneffekt Ladungsträger auf das Gate des Transistors. Da das Silizium sehr gut isoliert ist, fließen diese Ladungsträger nicht wieder ab, und der Zustand, der durch die Programmierung geschaffen wurde, bleibt für lange Zeit erhalten. Die einzige Möglichkeit, diesen Zustand zu ändern, besteht darin, die Speicherzelle mit UV-Licht zu bestrahlen. Da auch Sonnenlicht einen wesentlichen UV-Anteil besitzt, muß man das Fenster des EPROMs mit einem UV-dichten Etikett abdecken, um ein unbeabsichtigtes Löschen zu verhindern.

### Der Programmiervorgang

Ein Programmiervorgang muß demnach also grundsätzlich immer so aussehen, daß zunächst einmal die zu programmierende Speicherstelle adressiert wird. Dann muß noch der Auswahleingang ($\overline{CE}$) auf 0 gelegt werden. Ist so das richtige Byte ausgewählt, kann man die Programmierspannung anlegen. Wie lange die Programmierspannung an den Transistoren liegt, wird durch den Eingang $\overline{PGM}$ bestimmt: Solange der Eingang auf 0 liegt, wird sie an die Speicherstelle durchgeschaltet.

Beim NDR-Klein-Computer wird normalerweise der sogenannte Standard-Algorithmus angewendet, bei dem die Dauer des Programmierimpulses 50 ms beträgt. Dieser Algorithmus hat allerdings einige Nachteile. Zum einen dauert eine Programmierung hier pro Byte 60 ms, da noch 10 ms benötigt werden, um einen Kondensator des zeitbestimmenden Monoflops zu entladen. Zum anderen wird die Richtigkeit der programmierten Daten erst am Ende der Programmierung überprüft. Wird ein 2764 mit dieser Methode programmiert, so dauert die Programmierung über acht Minuten. Ist nun im ungünstigsten Fall gerade das erste Byte nicht richtig programmiert, so muß man acht Minuten warten, um festzustellen, daß der ganze Vorgang noch einmal wiederholt werden muß.

Ein weiteres Manko ist, daß auch Bytes mit dem Wert $FF programmiert werden: Neue bzw. gelöschte EPROMs haben in allen Speicherstellen ohnehin den Wert $FF stehen. Ist nur jedes fünfzigste Byte mit $FF zu programmieren, so würden schon drei Minuten gespart, wenn man diese Bytes überspringt.

### Es geht auch schneller

Es gibt jedoch auch noch einen anderen Algorithmus, der von Intel entwickelt worden ist und einige Vorteile vorzuweisen hat. Es wird dabei nicht mit einer festen Programmierimpulsdauer programmiert, sondern mit Programmierimpulsen von etwa 1 ms Dauer. Es wird nach jedem Impuls geprüft, ob die Speicherstelle schon den richtigen Wert hat. Ist dies der Fall, so wird noch einmal mit der doppelten bisher benötigten Zeit nachprogrammiert. Ist nach 15 Programmierimpulsen das Byte noch nicht richtig programmiert, so wird ebenfalls noch einmal mit der doppelten Zeit nachprogrammiert und dann das Byte nochmals geprüft. Falls dann der Inhalt noch immer nicht stimmt, wird dieses Byte als nicht programmierbar angesehen. Bytes, die mit $FF programmiert werden müßten, werden hier selbstverständlich übersprungen.
Der hier nun vorgestellte Algorithmus ist eng an den Intel-Algorithmus angelehnt, wurde jedoch an die Bedürfnisse des NDR-Klein-Computers angepaßt (*Bild*).
Beim NDR-Computer wird der Programmierimpuls durch ein Monoflop erzeugt.

```
020000                  ORG $20000
020000
020000                  * Q U I C K P R O M   6 8 - V 1.1
020000                  * FASTPROGRAMM - ROUTINE FUER DEN NDR-KLEIN-COMPUTER
020000                  * COPYRIGHT (C) 1985 BY :
020000                  * RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG 11
020000
020000                  KOPF:                                   * BIBLIOTHEKSEINTRAG
020000  55AA0180        DC.L $55AA0180
020004  51205072 6F6D20 DC.B 'Q Prom '
020008  20
020009  00000218        DC.L GETPAR-KOPF
020010  0000033C        DC.L ENDEA-KOPF
020014  01              DC.B 1
020015  00 00 00        DC.B 0,0,0
020018  00000000        DC.L 0,0
020020  00000000
020020  01 01           DC.B 1,1                                * VERSIONSNUMMER
020022  D0000001        DC.L $D0000001                          * SERIENNUMMER (D = DISK, E = EPROM)
020026                  DS 0                                    * WORTANGLEICH
020026
= 000000F4              BUFFER          EQU     $F4             * GRUNDPRG. EINBUF
= FFFFFF80              IPPORT          EQU     $FFFFFF80
= FFFFFF80              OPPORT          EQU     $FFFFFF80
= FFFFFF81              BUSYP           EQU     $FFFFFF91
= FFFFFF81              ADRP            EQU     $FFFFFFB1
= FFFFFF82              ZPORT           EQU     $FFFFFF82
020026
020026                  PRGLP:                          * PROGRAMMIERSCHLEIFE
020026                                                  * A0 = VON, D7 = BIS, D5 = NACH
020026  13D0 FFFFFF80   MOVE.B (A0),OPPORT              * BYTE AN PROMMER
02002C  0C10 00FF       CMP.B #$FF,(A0)                 * FF ?, BRAUCHT NICHT PROGRAMMIERT WERDEN
020030  6700 0060       BEQ NEXTBYTE                    * DANN NAECHSTES BYTE
020034  13C5 FFFFFF81   MOVE.B D5,ADRP                  * ADRESSE UNTERER TEIL
02003A  E05D            ROR #8,D5                       * UND ROTIEREN FUER MSB
02003C  1005            MOVE.B D5,D0
02003E  E15D            ROL #8,D5
020040  0200 001F       AND.B #%00011111,D0             * LED AN, ENABLE AUS, KEIN PRGM. IMPULS
020044  0000 0080       OR.B #%10000000,D0              * DANN ENABLE
020048  13C0 FFFFFFB2   MOVE.B D0,ADRP+1                * AN ADRESSPORT
02004E  0000 0020       OR.B #%00100000,D0              * UND PROGRAMMIERIMPULS
020052  13C0 FFFFFFB2   MOVE.B D0,ADRP+1                * AN ADRP
020058  0200 00DF       AND.B #%11011111,D0             * TRIGGER RUECKSETZEN
02005C  13C0 FFFFFFB2   MOVE.B D0,ADRP+1
020062                  TRREADY:                        * WARTET, BIS PRGM. PULS ZUENDE
020062  0839 0000       BTST #0,BUSYP                   * "1" = WARTEN
020066  FFFFFFB1
02006A  6600 FFF6       BNE TRREADY                     * BIS "0"
02006E  0200 001F       AND.B #%00011111,D0             * LESEN
020072  13C0 FFFFFFB2   MOVE.B D0,ADRP+1
020078  1039 FFFFFF80   MOVE.B IPPORT,D0                * PROGM. BYTE NACH D0
02007E  B010            CMP.B (A0),D0                   * SCHON OK ?
020080  6700 0010       BEQ NEXTBYTE                    * JA, DANN NAECHSTES BYTE
0200B4  5242            ADDQ #1,D2                      * ZAEHLER FUER PROGRAMMIERIMPULSE
0200B6  0C02 0003       CMP.B #3,D2
0200BA  6E00 0074       BGT ERROR                       * EPROM NICHT OK
0200BE  6000 FF96       BRA PRGLP                       * SONST WEITER PROGRAMMIEREN
020092
020092                  NEXTBYTE:
020092  5245            ADDQ #1,D5                      * ADRESSE 'nach' ERHOEHEN
020094  D1FC 00000001   ADDA.L #1,A0                    * ADRESSE SPEICHER ERHOEHEN
02009A  BEB8            CMP.L A0,D7                     * ALLES PROGRAMMIERT ?
02009C  6700 0008       BEQ ENDE                        * JA, DANN ENDE
0200A0  4282            CLR.L D2                        * NEIN, DANN IMPULSZAEHLER LOESCHEN
0200A2  6000 FFB2       BRA PRGLP                       * UND NAECHSTES BYTE PROGRAMMIEREN
0200A6
0200A6                  ENDE:
0200A6  41FA 0092       LEA OKTXT(PC),A0                * EPROM OK
0200AA  303C 0033       MOVE #$33,D0
0200AE  323C 000A       MOVE #10,D1
0200B2  343C 0041       MOVE #65,D2
0200B6  3E3C 000A       MOVE #!WRITE,D7
0200BA  4E41            TRAP #1
0200BC
0200BC                  ENDE1:
0200BC  13FC 0040       MOVE.B #%01000000,ADRP+1        * PROMMER NEUTRAL STELLEN
0200C0  FFFFFF82
0200C4  41FA 0115       LEA BLTXT(PC),A0
0200C8  343C 001E       MOVE #30,D2
0200CC  323C 000A       MOVE #10,D1
0200D0  303C 0021       MOVE #$21,D0
0200D4  3E3C 000A       MOVE #!WRITE,D7
0200D8  4E41            TRAP #1
0200DA  41FA 0126       LEA BLTXT1(PC),A0
0200DE  343C 0032       MOVE #50,D2
0200E2  323C 000A       MOVE #10,D1
0200E6  303C 0021       MOVE #$21,D0
0200EA  3E3C 000A       MOVE #!WRITE,D7
0200EE  4E41            TRAP #1
0200F0  3E3C 000C       MOVE #!CI,D7                    * TASTATUR ABFRAGEN
0200F4  4E41            TRAP #1
0200F6  0C00 0077       CMP.B #'w',D0                   * NOCH EIN EPROM PROGRAMMIEREN ?
0200FA  6700 011C       BEQ GETPAR                      * JA, DANN START
0200FE
0200FE                  ENDE2:                          * SONST ENDE
0200FE  4E75            RTS
020100
020100                  ERROR:
020100  41FA 001B       LEA PRGMERR(PC),A0              * EPROM NICHT ZU PROGRAMMIEREN
020104  303C 0033       MOVE #$33,D0
020108  323C 000A       MOVE #10,D1
02010C  343C 0041       MOVE #65,D2
020110  3E3C 000A       MOVE #!WRITE,D7                 * FEHLERMELDUNG AUSGEBEN
020114  4E41            TRAP #1
020116  6000 FF64       BRA ENDE1
02011A
```

**68008-Assemblerlisting zum schnellen EPROM-Programmieren mit dem NDR-Klein-Computer**

```
02011A 4570726F6D206E PRGMERR:    DC.B 'Eprom nicht zu programmieren !',0
020121 69636874207A75
020128 2070726F677261
02012F 6D6D696572656E
020136 202100
020139 00             DS 0
02013A
02013A 4570726F6D206F OKTXT:      DC.B 'Eprom ok',0
020141 6B00
020143 00             DS 0
020144
020144 51756963686B7072 PTXT1:    DC.B 'Quickprom 68',0
02014B 6F6D206368B00
020151 284329203139938 PTXT2:     DC.B '(C) 1985 - Ruediger Baecker',0
020158 352020527565
02015F 64696765722042
020166 6165636B657200
02016D 766F6E2020203D PTXT3:      DC.B 'von   ==>',0
020174 3D3E00
020177 626973202020 PTXT4:        DC.B 'bis   ==>',0
02017E 3D3E00
020181 6E616368202020 PTXT5:      DC.B 'nach  ==>',0
020188 3D3E00
02018B 733A2053746172 PTXT6:      DC.B 's: Start  f: Fehler  w: weiter  m: Menue',0
020192 742020663A2046
020199 65686C65722020
0201A0 773A2077656974
0201A7 657220206D3A20
0201AE 4D656E756500
0201B4 4570726F6D2077 PTXT7:      DC.B 'Eprom wird programmiert - bitte warten',0
0201BB 6972642070726F
0201C2 6772616D6D6965
0201C9 727420202206269
0201D0 74746520776172
0201D7 74656E00
0201DB 20202020202020 BLTXT:      DC.B '                                        ',0
0201E2 20202020202020
0201E9 20202020202020
0201F0 20202020202020
0201F7 20202020202020
0201FE 20202000
020202 20202020202020 BLTXT1:     DC.B '                    ',0
020209 20202020202020
020210 20202020202000
020217 00             DS 0
020218
020218
020218
020218                GETPAR:                 * HIER PROGRAMMSTART
020218 3E3C 0011      MOVE #!CLPG,D7          * BILDSCHIRM LOESCHEN
02021C 4E41           TRAP #1
02021E 41FA FF24      LEA PTXT1(PC),A0        * TEXTE AUSGEBEN
020222 303C 0043      MOVE #$43,D0            * SCHRIFTGROESSE
020226 343C 00C8      MOVE #200,D2            * POSITION ETC.
02022A 323C 000A      MOVE #10,D1
02022E 3E3C 000A      MOVE #!WRITE,D7
020232 4E41           TRAP #1
020234 41FA FF1B      LEA PTXT2(PC),A0
020238 303C 0032      MOVE #$32,D0
02023C 0442 001E      SUB #30,D2
020240 3E3C 000A      MOVE #!WRITE,D7
020244 4E41           TRAP #1
020246 41FA FF25      LEA PTXT3(PC),A0
02024A 303C 0021      MOVE #$21,D0
02024E 0442 001E      SUB #30,D2
020252 3E3C 000A      MOVE #!WRITE,D7
020256 4E41           TRAP #1
02025B 41FA FF1D      LEA PTXT4(PC),A0
02025C 0442 0014      SUB #20,D2
020260 3E3C 000A      MOVE #!WRITE,D7
020264 4E41           TRAP #1
020266 41FA FF19      LEA PTXT5(PC),A0
02026A 0442 0014      SUB #20,D2
02026E 3E3C 000A      MOVE #!WRITE,D7
020272 4E41           TRAP #1
020274 41ED 00F4      LEA BUFFER(A5),A0
020278 343C 008C      MOVE #140,D2
02027C 323C 00B4      MOVE #180,D1
020280 303C 0021      MOVE #$21,D0
020284 363C 0008      MOVE #8,D3              * MAX 8 ZEICHEN
020288 3E3C 000B      MOVE #!READ,D7          * ADRESSE 'von' IN BUFFER EINLESEN
02028C 4E41           TRAP #1
02028E 41ED 00F4      LEA BUFFER(A5),A0
020292 4280           CLR.L D0
020294 3E3C 001D      MOVE #!WERT,D7          * WERT ERMITTELN
020298 4E41           TRAP #1
02029A 2240           MOVEA.L D0,A1           * UND 'von' RETTEN
02029C 41ED 00F4      LEA BUFFER(A5),A0       * DANN WEITER MIT 'bis' UND 'nach'
0202A0 343C 007B      MOVE #120,D2
0202A4 323C 00B4      MOVE #180,D1
0202A8 303C 0021      MOVE #$21,D0
0202AC 363C 0008      MOVE #8,D3
0202B0 3E3C 000B      MOVE #!READ,D7
0202B4 4E41           TRAP #1
0202B6 41ED 00F4      LEA BUFFER(A5),A0
0202BA 4280           CLR.L D0
0202BC 3E3C 001D      MOVE #!WERT,D7
0202C0 4E41           TRAP #1
0202C2 2D00           MOVE.L D0,D6            * 'bis' RETTEN
0202C4 41ED 00F4      LEA BUFFER(A5),A0
0202C8 343C 0064      MOVE #100,D2
0202CC 323C 00B4      MOVE #180,D1
0202D0 303C 0021      MOVE #$21,D0
0202D4 363C 0008      MOVE #8,D3
0202D8 3E3C 000B      MOVE #!READ,D7
0202DC 4E41           TRAP #1
```

```
0202DE  41ED 00F4       LEA BUFFER(A5),A0
0202E2  4280            CLR.L D0
0202E4  3E3C 001D       MOVE #!WERT,D7
0202E8  4E41            TRAP #1
0202EA  2A00            MOVE.L D0,D5                    * 'nach' IN D5
0202EC  41FA FE9D       LEA PTXT&(PC),A0                * DANN MENUEZEILE AUSGEBEN
0202F0  343C 0032       MOVE #50,D2
0202F4  323C 000A       MOVE #10,D1
0202F8  303C 0021       MOVE #$21,D0
0202FC  3E3C 000A       MOVE #!WRITE,D7
020300  4E41            TRAP #1
020302  3E3C 000C       MOVE #!CI,D7                    * TASTATUR ABFRAGEN
020306  4E41            TRAP #1
020308  0C00 0066       CMP.B #'f',D0                   * FEHLER IN DER EINGABE ?
02030C  6700 FF0A       BEQ GETPAR                      * JA, DANN EINGABE NOCHMAL
020310  0C00 0077       CMP.B #'w',D0                   * 'w' GEDRUECKT ?
020314  6700 FF02       BEQ GETPAR                      * JA, DANN NOCH EIN EPROM PROGRAMMIEREN
020318  0C00 0073       CMP.B #'s',D0                   * PROGRAMMIERUNG STARTEN ?
02031C  6600 FDE0       BNE ENDE2                       * NEIN, DANN ZUM GRUNDPROGRAMM ZURUECK
020320  41FA FE92       LEA PTXT7(PC),A0
020324  303C 0021       MOVE #$21,D0
020328  0442 0014       SUB #20,D2
02032C  3E3C 000A       MOVE #!WRITE,D7
020330  4E41            TRAP #1
020332  2049            MOVEA.L A1,A0                   * VON IN A0
020334  2E06            MOVE.L D6,D7                    * BIS IN D7
020336  5247            ADDQ #1,D7                      * FUER ABFRAGE
020338  6000 FCEC       BRA PRGLP                       * UND PROGRAMMIEREN
02033C                  ENDEA:
02033C                  END.
```

Es wird softwaremäßig getriggert, wobei der zeitliche Abstand der Triggerimpulse durch Zählen der Vertikal-Synchronimpulse der Grafikbaugruppe ermittelt wird. Dieses Signal tritt alle 20 ms auf.

Das Monoflop wird nun nach jedem dritten Synchronimpuls neu getriggert. Dadurch entsteht die bereits erwähnte Programmierzeit von 60 ms pro Byte. Da das Monoflop auf eine feste Impulsdauer von 50 ms abgeglichen ist, für den hier verwendeten Algorithmus aber eine wesentlich geringere Impulsdauer benötigt wird und eine Hardware-Änderung vermieden werden sollte, wurde für die Impulserzeugung ein anderer Weg gesucht: Zur Erzeugung der Programmierimpulse wird das Monoflop ständig nachgetriggert. Das hat zur Folge, daß die Programmierspannung ständig am EPROM anliegt. Dies ist auch beim Intel-Algorithmus der Fall und hat keinerlei Nachteile.

Nach jedem Programmierimpuls wird der Inhalt des EPROMs geprüft und bei Übereinstimmung das nächste Byte programmiert. Auch hier werden Bytes mit $FF übersprungen.

Die Programmierimpulse entstehen durch den Wechsel zwischen Programmieren und Auslesen des Inhaltes und haben eine Länge von etwa 0,5 ms. Dadurch ergibt sich für ein 2764 eine Programmierzeit von etwa 30 Sekunden, was in etwa der Zeit entspricht, die beispielsweise der Böhm-Programmer erreicht.

### Die Bedienung

Vor der Programmierung muß der dem EPROM entsprechende DIL-Stecker in die Fassung gesetzt werden. Es können mit diesem Programm alle Intel-kompatiblen EPROMs vom Typ 2764 und 2732 programmiert werden. (Vorsicht: Manche von Texas Instruments hergestellten EPROMs lassen das Zurücklesen bei angelegter Programmierspannung nicht zu und sind deshalb für das hier propagierte Verfahren nicht geeignet!)
Das Programm kann aus der Bibliothek aufgerufen werden und meldet sich dann mit einem Menü: Es fragt dann nach Start- und Endadresse des Bereichs, der programmiert werden soll, und nach der Ablageadresse innerhalb des EPROMs. Sind die Eingaben erfolgt, so kann das EPROM in die Fassung gesetzt und der Programmiervorgang durch Drücken der Taste s gestartet werden. Hat man einen Fehler bei der Eingabe gemacht, so gelangt man mit f wieder zum Anfang der Eingaberoutine zurück. Die Taste w hat hier noch keine Bedeutung, bewirkt hier jedoch ebenfalls einen Rücksprung zur Eingaberoutine. Mit m gelangt man in das Grundprogramm zurück.

Da der Programmiervorgang sehr schnell vonstatten geht, wurde auf eine Ausgabe der gerade programmierten Adresse verzichtet; das Programm programmiert ohnehin schneller als man lesen kann! Nachdem die Programmierung abgeschlossen ist, meldet sich der Rechner mit einer OK-Meldung zurück. Sollte während der Programmierung ein Fehler auftreten, so wird eine Fehlermeldung ausgegeben. Man kann hier (wie auch nach erfolgreicher Programmierung) mit m in das Grundprogramm zurück.

Drückt man w, so kann ein weiteres EPROM programmiert werden; man gelangt dazu wieder in die Eingaberoutine zurück.

Sollte ein Fehler auftreten, so kann das mehrere Ursachen haben. Oft ist das EPROM nicht vollständig leer; dann muß es nochmals gelöscht werden. Ein EPROM, das nach 30 Minuten Löschzeit noch immer nicht leer ist, sollte im übrigen in den Papierkorb wandern!

**Spruch des Monats**

„Kein Programm ist unnütz – es kann immer noch als schlechtes Beispiel dienen."

Rainer Roßberg

# Symbolischer 8085-Assembler in Basic

Ein symbolischer Assembler erlaubt die Übersetzung eines in Mnemonics geschriebenen Quelltextes in ausführbare Maschinenbefehle und legt diese im Arbeitsspeicher ab. Das Programm eignet sich für Computer wie M-10 (Olivetti) und Tandy-100/200, aber auch für andere 8085-Rechner mit Basic-Interpreter.

Der Quelltext des zu assemblierenden Programms wird mit dem z. B. im M-10 von Olivetti eingebauten Texteditor erstellt. Die hierbei zu verwendenden mnemonischen Befehle entsprechen den leicht zu erlernenden Z80-Mnemonics, es werden jedoch lediglich diejenigen Befehle, die der 80C85-Prozessor ausführen kann, übersetzt. Der Assembler (*Bild 1*) erwartet folgendes Eingabeformat:

```
LABEL   OPCODE   OPERAND
;Kommentar
```

Die einzelnen Spalten werden durch TAB oder Space, Kommentare durch einen Strichpunkt getrennt. Numerische Werte im Operandenfeld können dezimal, hexadezimal (mit nachgestelltem „H"), als ASCII-Zeichen (in einfachen Anführungszeichen eingeschlossen) oder als Label angegeben werden. Außerdem ist es möglich, im Operandenfeld einen numerischen Wert als Summe oder Differenz zweier anderer Werte anzugeben. Im Quelltext sind nur Großbuchstaben zugelassen.

Die Assemblierung erfolgt in zwei Durchgängen (2-Paß-Assembler). Im ersten Durchgang werden in einer Tabelle die Werte aller Labels abgelegt. Im zweiten Durchgang erfolgt die eigentliche Übersetzung und die Ablage des Maschinenprogramms im Speicher. Dieser Speicherbereich muß daher vor der Übersetzung gegen Zugriff durch den Basic-Interpreter gesichert werden (im Regelfall: CLEAR1500,Startadresse). Sollte der Assembler auf einen für ihn erkennbaren Fehler treffen, so werden die Ursache sowie die fehlerhafte Zeile ausgegeben. Im Anschluß an die Übersetzung erfolgt die Ausgabe der Referenzliste.

Neben den eigentlichen Prozessorbefehlen werden noch folgende Pseudo-Anweisungen vom Assembler verarbeitet:

```
  ORG   Wert
```

Die ORG-Anweisung legt die Anfangsadresse des folgenden Programmabschnittes fest.

```
Label   EQU   Wert
```

Die EQU-Anweisung ordnet einem Label einen Wert zu.

```
  DEFB   Wert
```

Die DEFB-Anweisung legt ein Byte im Speicher ab.

```
50 CLS:CLEAR1500
60 PRINT"M10 ASSEMBLER
70 PRINT"(C) 1984 Rainer Roßberg
80 PRINT
100 DEFSNGA-Z
110 DIMNA(150),NA$(150)
120 NA=1
150 INPUT"Quellfile  ";QS
160 OPENQ$FORINPUTAS1
200 PRINT:PRINT"Pass 1
210 IFNA$="END"OREOF(1)THEN300
220 B0=0:B1=B0:B1$=""
230 GOSUB500
240 IFNZ=0ANDBZ<>0THENNZ=BZ
250 IFNA$=""THEN290
260 NA$(NA)=NA$
270 NA(NA)=BZ
280 NA=NA+1
290 BZ=BZ+B0
295 GOTO210
300 PRINT"Pass 2
305 NA$="":BZ=0:CLOSE#1:OPENQ$FORINPUTAS1
310 IFEOF(1)ORNA$="END"THEN450
315 B0=0:B1=0:NZ=NA:B1$="":GOSUB500:NA=NZ
320 IFB0=0THEN380
330 IFINSTR(FU$,"DEF")THENBZ=BZ-1:B0=B0+1ELSE
    POKEBZ,B1
335 IFFU$="DEFT"THENFORI=1TOB0-1:POKEBZ+I,ASC
    (MID$(B1$,I,1)):NEXTI:GOT
340 IFB1$<>""THENGOSUB3000
350 IFB0>1THENPOKEBZ+1,B1
355 IFB0=2ANDB2<>0THENBEEP:PRINT"Operand Error : "DZ$
360 IFB0=3THENPOKEBZ+2,B2
380 IFINSTR(FU$,"DEF")THENBZ=BZ+1:B0=B0-1
400 BZ=BZ+B0
410 GOTO310
450 PRINT"Referenz-Tabelle:
454 PRINT"Taste drücken
455 A$=INKEY$:IFA$=""THEN455
460 FORI=1TONA-1:PRINTNA$(I),NA(I):NEXT
490 END
500 LINEINPUT#1,DZ$
510 A=INSTR(DZ$,";")
520 IFA>0THENDZ$=LEFT$(DZ$,A-1)
530 DZ$=DZ$+" "
550 A=1:GOSUB5000
560 IFC>1THENNA$=LEFT$(DZ$,C-1)ELSENAS=""
570 A=C+1:GOSUB5000
580 IFC>ATHENFU$=MID$(DZ$,A,C-A)ELSEFU$=""
585 A=C+1:GOSUB5000
590 IFC>ATHENOP$=MID$(DZ$,A,C-A)ELSEOP$=""
610 IFFU$=""THENRETURN
630 IFFU$="ORG"THENB1$=OP$:GOSUB3000:BZ=B1+B2*256:RETURN
640 IFFU$="DEFB"THENB0=1:B1$=OP$:RETURN
650 IFFU$="DEFW"THENB1$=OP$:B0=2:RETURN
655 IFFU$="DEFT"THENB1$=RIGHT$(DZ$,LEN(DZ$)-A+1)
    :B0=LEN(B1$)-1:RETURN
660 IFFU$="DEFS"THENB0=VAL(OP$):RETURN
670 IFFU$="EQU"THENNA$(NA)=NA$:B1$=OP$:GOSUB3000
    :NA(NA)=B2*256+B1:NA=N A+1:NAS="":RETURN
700 REM Assembler sucht Prozessorcode
710 IFFU$="LD"THEN1500
```

**Bild 1. Listing des in Basic geschriebenen 8085-Assemblers für den M-10 von Olivetti**