Tester

In diesem Fachgebiet wird ein weiteres Programm des Betriebssystems vorgestellt, der Tester.Dieses Programm, das in anderen Computern und deren Betriebssystemen auch "Debugger" (sinngemäß "Fehlersucher") genannt wird, enthält eine ganze Reihe von Kommandos. Man kann mit ihnen Speicherinhalte anzeigen, ändern, verschieben und vergleichen, ferner rechnen, Programme austesten, aber auch Speicherinhalte auf andere Rechner übertragen, auf Cassette oder Diskette abspeichern und von dort laden.

Sie haben zunächst gesehen, wie Sie Ihren Computer als Textsystem benutzen können, Jetz folgt der nächste Schritt: mit Hilfe des Testers läßt sich nachschauen, wie diese Texte im Computer gespeichert sind.

Speicherausdruck

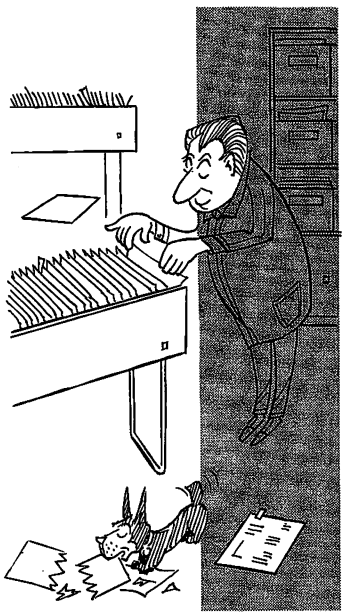
Der Tester wird mit dem Kommando TEST aufgerufen, er meldet sich mit einem Stern (*) und wartet auf die Eingabe eines Kommandos. Bei einer falschen Eingabe reagiert der Tester mit einem Fragezeichen (?) als Fehlermeldung.

Ein Tester-Kommando besteht immer aus einem oder mehreren Buchstaben und wird mit der Taste CR bzw. Ret abgeschlossen. Vollziehen Sie das in der Praxis nach, rufen Sie den Tester auf und geben dann ,wenn der Stern auf dem Bildschirm steht, als erstes Tester-Kommando D ein:

```
*d
0100  C3 03 84 00 01 C3 F0 54 1A E6 7F C9 00 00 00 00 C....CpT.f.I....
0110  C3 E9 9E E6 7F C9 00 00 FF FF 00 00 FF FF 00 00 Ci.f.I..........
0120  FF FF 00 00 FF FF 00 00 FF FF 00 00 FF FF 00 00 ................
0130  FF FF 00 00 FF FF 00 00 C3 03 84 00 FF FF 00 00 ........C.......
0140  C3 99 EA C3 4F EB 00 00 00 00 00 00 EA B0 ED C5 C.jCOk......j0mE
0150  FF FF 00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 ................
0160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
*
```

Dieser Ausdruck (dump, Speicherausdruck) muß jetzt auf dem Bildschirm stehen. Dabei sind es sicherlich andere Zeichen wie bei unserem Beispiel, aber die grundsätzliche Anordnung ist die gleiche. Am linken Rand steht eine Adresse, danach der Inhalt von 16 Speicherstellen, deren erste eben die linksstehende Adresse hat. Der Inhalt der Speicherzellen ist (wie auch die Adresse links) sedezimal

dargestellt. Am rechten Rand steht der Inhalt der 16 Speicherzellen noch einmal als scheinbar sinnlose Zeichenfolge. Erst wenn Sie den Textspeicher erwischen, erkennen Sie dessen Inhalt wieder.



Nach dem, was über die Speicheraufteilung auf Seite A gesagt wurde, ist der Textspeicher im Adressenbereich von etwa 5400 aufwärts zu finden. Das Dump-Kommando wird in diesem Fall um die gewünschte Adresse als Parameter ergänzt. Geben Sie also ein D 5400. Halt!, eine Voraussetzung muß selbstverständlich gegeben sein: es muß ein Text im Textspeicher stehen. Falls dies von den vorhergehenden Versuchen mit dem Texteditor nicht mehr der Fall ist, müssen Sie Ihren Text von der Cassette wieder laden. Verlassen Sie in diesem Fall den Tester mit CTRL-C (das muß am Anfang einer Zeile, also gleich nach dem Stern, stehen), rufen Sie mit EDIT den Editor auf und - den Rest können Sie inzwischen sicher ganz gut. Danach wird der Editor mit ESC E verlassen und im Tester fortgefahren.

Geben Sie nach D 5400 einfach D ohne Parameter ein, die Inhalte der nächsten 9 mal 16 Speicherzellen werden angezeigt. Auf diese Weise können Sie den Speicher "durchblättern", bis rechts im Dump vertrauter Text auftaucht. Probieren Sie noch eben eine weitere Spielart des Dump-Kommandos aus. Um einen bestimmten Speicherbereich anzuschauen, kann man diesen mit zwei Adressen als Parameter nach D angeben, beispielsweise D 5400,5600. Auf keinen Fall darf das Komma zwischen den beiden Parametern vergessen werden.

ASCII

Wenn Sie sich den Speicherausdruck genau ansehen und die Darstellungen vergleichen, dann werden Sie ein Zuordnung zwischen Buchstaben und sedezimal dargestellten Zahlen erkennen. Ein grosses A entspricht zum Beispiel der Zahl 41H (wenn es zur Identifikation oder Hervorhebung notwendig ist,wird eine sedezimal dargestellte Zahl durch ein angehängtes H gekennzeichnet.

Eine solche Zuordnung zwischen verschiedenen Zeichengruppen, im vorliegenden Fall also zwischen Buchstaben und Zahlen, die der Computer sedezimal ausgibt, nennt man einen Code.

Alle Mikroprozessor-Systeme verwenden für die Zuordnung von Zahlen zu Buchstaben den sogenannten ASCII-Code (american standard code for information interchange,amerikanischer Standardcode für Informationsaustausch). Dabei

ist der gebräuchliche Ausdruck ASCII-Code nicht ganz korrekt, weil das Wort Code ja schon in der Abkürzung vorkommt.

Der ASCII-Code, den Sie auf Seite T1 abgedruckt finden, ist ein ziemlich einfacher Code, die Buchstaben sind grob gesagt durchnummeriert. Wenn Sie sich den Ausdruck des Textspeichers noch einmal ansehen, werden Sie auch Speicherstellen finden, die rechts keine Interpretation in Buchstaben haben, beispielsweise die beiden Zeichen 0DH und 0AH. Das sind sogenannte Steuerzeichen, die gebraucht werden, weil ein Text ja nicht nur aus einer Folge von Buchstaben besteht. Man muß auch wissen, wann eine neue Zeile oder Seite anfängt.

Die beiden eben erwähnte Steuerzeichen 0DH und 0AH markieren den Anfang einer neuen Zeile: 0DH entspricht dem ASCII-Steuerzeichen CR, das steht für carriage return= Wagenrücklauf, 0AH entspricht LF, das steht für line feed =Zeilenvorschub. Die beiden Zeichen bewirken bei einem Drucker, daß sich der Schreibschlitten nach links und das Papier um eine Zeile hoch bewegt. Beim Bildschirm wird der Cursor an den Anfang der Zeile und dann um eine Zeile tiefer gesetzt.

Wichtige andere Steuerzeichen sind 09H für die Tabulatorfunktion, 0CH für FF = form feed (Seitenvorschub) und 1AH für EOF = end of file (Ende der Datei bzw. des Textes).

Apropos Drucker: alles, was auf dem Bildschirm bei Ihrem Umgang mit dem Tester geschieht, können Sie sich auf einem angeschlossenen Drucker protokollieren lassen. In diesem Fall muß gleich am Anfang CTRL-P eingegeben werden. Gerade ein Speicherausdruck macht sich manchmal auf dem Papier für bestimmte Arbeiten besser als auf dem Bildschirm.

Speicherausdruck

Das Dump-Kommando birgt noch eine ganze Anzahl Möglichkeiten, deren Anwendungen vorerst noch nicht erkennbar sind, die wir aber an dieser Stelle der Vollständigkeit halber noch aufführen wollen.

Für alle Testerkommandos gilt, daß als Parameter Konstanten, Berechnungen oder Symbole eingegeben werden können. Mit den Symbolen befassen wir uns im eigentlichen Assembler, Konstanten beispielsweise in Form von Speicheradressen haben Sie schon benutzt und Berechnungen zeigen wir Ihnen noch kurz. Um Konstanten von Symbolen zu unter-

scheiden, müssen Konstanten mit einer Ziffer beginnen. Das ist wichtig bei den sedezimalen Speicheradressen. Wenn eine solche Adresse mit den Zahlen von zehn bis fünfzehn beginnt, also AH bis FH, dann muß eine 0 vorgestellt werden.

Daß das Betriebssystem beim Dump die Speicheradressen immer in sedezimaler Darstellung ausgibt, haben Sie bereits festgestellt. Dazu entsprechend ist die Zahlenbasis für die Eingabe von Konstanten auf 16 voreingestellt. Ander gesagt, Sie müssen in den Parametern des Dump-Kommandos ebenfalls dezimal dargestellte Zahlen verwenden. Das geht aber auch anders. Geben Sie das Kommando DEC ein. Danach folgt die Eingabe D 100. Jetzt werden ab Adresse 60 die üblichen 9 Zeilen ausgegeben.Sie haben als Parameter 100 (dezimal) eingeben,das entspricht 64H. Diese Speicherzelle liegt in der ersten ausgegebenen Zeile (die Ausgabe beginnt immer am Anfang von 16 Adressen).

Da sich gewohnheitsmäßig mit dezimal dargestellen Zahlen besser rechnen läßt, zeigen wir noch ein Beispiel für eine Berechnung im Parameter. Geben Sie ein: D 20*5 . Der Stern steht hier als Zeichen für Multiplikation. Wieder wird der Speicherbereich ab 64H = 100D = 20 mal 5 ausgeben. Wenn Sie nun, nachdem Sie hoffentlich ein wenig "gespielt" haben mit den möglichen Eingaben, das Kommando HEX eingeben, hat alles wieder seine alte Ordnung, nämlich Eingabe und Ausgabe der Speicheradressen erfolgen in sedezimaler Darstellung.

Zusammenfassung

Der "Tester" genannte Teil des Betriebssystems enthält eine Reihe von Kommandos, von denen das Dump-Kommando es ermöglicht, Speicherinhalte anzuschauen. Es wird in folgender Form eingegeben:

D /parameter/	ab Stelle /parameter/ 9 Zeilen ausgeben
D /parameter1/,/parameter2/	von /parameter1/ bis /parameter2/ ausgeben
D	ab letztem Dump 9 Zeilen ausgeben

Mit dem Kommando DEC kann auf die Eingabe von dezimal dargestellten Zahlen übergegangen werden, mit HEX wird zur voreingestellten sedezimalen Eingabe zurückgekehrt. Für die Zuordnung von Zahlen und Buchstaben bei der Datenverarbeitung benutzen alle Mikrocomputer den ASCII-Code (auf Seite T1 aufgelistet).

Fehlersuche im Programm 3 - Ausgabe des Textspeichers

Assemblieren Sie das neugeladene Programm und rufen Sie dann den Tester auf, er meldet sich mit einem Stern (vgl. Seite D1). Jetzt werden Sie nacheinander die Hilfsmittel in Gestalt verschiedener Kommandos ausprobieren, die der Tester zum Austesten von Programmen bereit hält. Sehen Sie sich zunächst einmal alle Registerinhalte der CPU an. Dies geschieht mit dem Kommando X (eXamine = prüfen). Auf dem Bildschirm wird angezeigt:

```
*x
START:
0100  11 22 01     LD   DE,0122  =MELDUNG
 s0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=0      PC=0100
```

Der Tester hat auf dem Bildschirm den Befehl disassembliert, auf den der Programmzähler (PC) gerade zeigt. Dazu sind die Inhalte aller Register angezeigt. Mit dem Kommando L (List) können Sie sich das Programm weiter disassembliert ausgeben lassen:

```
*l
START:
0100  11 22 01     LD   DE,0122  =MELDUNG
0103  CD 4F 01     CALL 014F  =STRING
0106  CD 44 01     CALL 0144  =NEUE_ZEILE
0109  0E F7        LD   C,F7  =TXTBEGF
010B  CD 05 00     CALL 0005  =SYSTEM
SCHLEIFE:
010E  7E           LD   A,(HL)
010F  23           INC  HL
0110  FE 1A        CP   1A  =EOF
0112  28 06        JR   Z,06 -> 011A  =ENDE
0114  5F           LD   E,A
0115  CD 3E 01     CALL 013E  =ZEICHEN
```

Mit L werden 11 Befehle disassembliert angezeigt, eine wiederholte Eingabe von L zeigt die nächsten 11 Befehle:

```
*l
0118  18 F4        JR   F4 -> 010E  =SCHLEIFE
ENDE: •
011A  CD 44 01     CALL 0144  =NEUE_ZEILE
011D  0E 00        LD   C,00  =WSTARTF
011F  CD 05 00     CALL 0005  =SYSTEM
```

```
MELDUNG:
0122  0D          DEC  C
0123  0A          LD   A,(BC)
0124  49          LD   C,C
0125  6E          LD   L,(HL)
0126  68          LD   L,B
0127  61          LD   H,C
0128  6C          LD   L,H
```

Beim ersten Versuch (Seite C9) hatte das Programm den Anfangstext richtig ausgegeben. Deshalb ist anzunehmen, daß der Fehler in der Schleife liegt. Mit einem weiteren Kommando des Testers lassen wir es vom Label START bis zum Label SCHLEIFE ablaufen. Dieses Kommando heißt G (Go=gehe, Start). Zu G kann eine Anfangsadresse sowie eine oder mehrere Unterbrechungsstellen angegeben werden.Die Adressen können wie bei allen Tester-Kommandos auch symbolisch angegeben werden, hier sieht das so aus:

D
6

```
*g start,schleife

Inhalt des Textspeichers:
SCHLEIFE:
010E  7E            LD   A,(HL)
 s0z0.0h0.1p0n0c0  A=F7  BC=0027  DE=0000  HL=44C0  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1      PC=010E
```

Das Programm ist beim Label SCHLEIFE wunschgemäß unterbrochen worden, der Tester zeigt den Registerstand zu diesem Zeitpunkt an. Im HL-Registerpaar soll zu diesem Zeitpunkt die Anfangsadresse des Textspeichers stehen. Das läßt sich mit dem Kommando D nachprüfen, wobei in Ihrem Computer nicht unbedingt genau die gleiche (44C0H) Adresse wie in unserem stehen muß, denn der Beginn des Textspeicher kann geringfügig anders liegen.

```
*d 44c0,44ff
44C0  0D 0A 0D 0A 20 09 54 49 54 4C 45 09 54 45 58 54 .... .TITLE.TEXT
44D0  53 50 45 49 43 48 45 52 20 2D 20 50 72 6F 67 72 SPEICHER - Progr
44E0  61 6D 6D 20 33 0D 0A 0D 0A 3B 2A 2A 2A 2A 2A 2A amm 3....;******
44F0  2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ****************
```

Das kurze Stück genügt, um den Anfang des Programms, das ja noch im Textspeicher steht, zu erkennen. Bis hierher scheint das Programm noch in Ordnung zu sein. Da man nicht weiß, an welcher Stelle der Fehler liegt, ist es angezeigt den Lauf der Dinge Schritt für Schritt zu verfolgen. Dafür gibt es das Kommando T (Trace=verfolgen,he-

rausfinden), das einen oder mehrere Programmschritte ausführen läßt und jeweils die Registerinhalte anzeigt.
Sehen Sie sich die nächsten 5 Programmschritte an:

```
*t 5
SCHLEIFE:
010E  7E           LD   A,(HL)
 s0z0.0h0.1p0n0c0  A=0D  BC=0027  DE=0000  HL=44C0  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=010F
010F  23           INC  HL
 s0z0.0h0.1p0n0c0  A=0D  BC=0027  DE=0000  HL=44C1  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0110
0110  FE 1A        CP   1A  =EOF
 s1z0.0h0.1p0n1c1  A=0D  BC=0027  DE=0000  HL=44C1  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0112
0112  28 06        JR   Z,06 -> 011A  =ENDE
 s1z0.0h0.1p0n1c1  A=0D  BC=0027  DE=0000  HL=44C1  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0114
0114  5F           LD   E,A
 s1z0.0h0.1p0n1c1  A=0D  BC=0027  DE=000D  HL=44C1  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0115
```

Bis hierher ist kein Fehler zu erkennen, also noch einmal
5 Schritte:

```
*t 5
0115  CD 3E 01     CALL 013E  =ZEICHEN
 s1z0.0h0.1p0n1c1  A=0D  BC=0027  DE=000D  HL=44C1  IX=0000  IY=0000  SP=00FE
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=013E
ZEICHEN:
013E  0E 02        LD   C,02  =CONAUSF
 s1z0.0h0.1p0n1c1  A=0D  BC=0002  DE=000D  HL=44C1  IX=0000  IY=0000  SP=00FE
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0140
0140  CD 05 00     CALL 0005  =SYSTEM
 s1z0.0h0.1p0n1c1  A=0D  BC=0002  DE=000D  HL=44C1  IX=0000  IY=0000  SP=00FC
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0005
SYSTEM:
0005  C3 3E 40     JP   403E
 s0z1.0h0.0p1n0c0  A=00  BC=00D5  DE=0000  HL=0000  IX=0000  IY=0000  SP=00FE
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0143
0143  C9           RET
 s0z1.0h0.0p1n0c0  A=00  BC=00D5  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=0118
```

Noch ein Schritt:

```
*t
0118  18 F4        JR   F4 -> 010E  =SCHLEIFE
 s0z1.0h0.0p1n0c0  A=00  BC=00D5  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1       PC=010E
```

Das erste Zeichen des Textspeichers (20H) wurde anscheinend richtig ausgegeben. Danach sollte beim Label SCHLEIFE das nächste Zeichen nach A geladen und ausgegeben werden:

```
*t
SCHLEIFE:
010E  7E            LD   A,(HL)
 s0z1.0h0.0p1n0c0  A=C3  BC=00D5  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=F4  IFF=1      PC=010F  *
```

D
8

Hier geht's nicht weiter, C3H ist kein Zeichen aus dem Textspeicher und das HL-Registerpaar zeigt nicht mehr auf den Textspeicher. Verfolgen Sie die Einzelschritte zurück: beim Aufruf der Systemfunktion zur Konsolausgabe ist der Inhalt des HL-Registerpaars verloren gegangen. Nachstehend noch einmal in übersichtlicher Form die Tester-Kommandos, die Sie bei der Fehlersuche im vorliegenden Programm kennengelernt haben:

X	alle Register ausgeben
L /parameter/	ab Stelle /parameter/ 11 Befehle listen
L /parameter1/,/parameter2/	von /parameter1/ bis /parameter2/ listen
L	ab letztem List 11 Befehle listen
G /parameter/	Programmstart an der Stelle /parameter/
G /parameter1/,/parameter2/	Lauf von /parameter1/ bis /paramter2/
G	Programmstart am aktuellen PC-Stand
T	1 Befehl ausführen, Register anzeigen
T /parameter/	/parameter/ Befehle ausführen

Zusammenfassung

Am Beispiel des Programms 3 wird der Gebrauch einiger Tester-Kommandos zur bei der Fehlersuche gezeigt.

Frage:

1. Welcher Unterschied besteht zwischen den Kommandos L und G?

(Die Antwort zu dieser Frage finden Sie auf Seite G3.)

Register setzen und ausgeben

Das Kommando X zur Ausgabe aller Registerinhalte der CPU haben Sie auf Seite D5 bereits kennengelernt. Nach dem, was auf den Seiten C36 bis C38 gesagt wurde, wissen Sie jetzt auch, daß die mit einem Apostroph beginnende zweite Zeile des Ausdrucks die Inhalte des zweiten Registersatzes angibt. Neu hingegen ist die Ergänzung des Kommandos mit einem Registernamen. Damit ist es möglich, den Inhalt eines einzelnen Registers auszugeben und vor allem zu ändern.

Mit dem Kommando X A wird beispielsweise der Inhalt des A-Registers ausgegeben. Nach den Inhalt steht ein Bindestrich, z.B. 00 - ,der Computer wartet auf die Eingabe eines neuen Inhalts. Das kann eine Zahl oder ein Rechenausdruck sein. Durch die anschließende Betätigung der CR-(Return) Taste wird der neue Inhalt abgespeichert. Wenn nichts geändert werden soll, genügt CR.

Beim Kommando X werden folgende Registerangaben als gültig erkannt:

Registersatz 1

Register:	A B C D E H L
Registerpaar:	AF BC DE HL

Registersatz 2

Register:	A' B' C' D' E' H' L'
Registerpaar:	AF' BC' DE' HL'

Sonstige Registerpaare: PC SP IX IY

Interrupt-Register: I IFF

Schreiben und assemblieren Sie ein kleines Programm, das lediglich aus sechs EX X-Befehlen besteht. Rufen Sie danach den Tester auf, damit Sie das vorher gesagte in der Praxis ausprobieren können.

Auf der nächsten Seite ist eine Tester-Sitzung mitprotokolliert, die Sie mit Ihrem Computer nachvollziehen sollten, wobei selbstverständlich immer der Rat gilt, außer den angeführten Beispielen noch ein wenig mehr mit dem System zu "spielen".

```
>test
*; geben Sie die Inhalte aller Register mit dem X-Kommando aus:
*X
0100  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=EC  IFF=0      PC=0100
*; die Register wurden vom System alle gelöscht.
*; Einzelne Register werden durch Angabe des Namens als Parameter gesetzt
*X B
00 - 13
*X H
00 - 0FF
*X L
00 - 01
*; Sie können auch ein ganzes Registerpaar auf einmal ändern:
*X DE
0000 - 1234
*; Sehen Sie sich den Registersatz wieder an:
*X
0100  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=EC  IFF=0      PC=0100
*; Sie können im Tester sogar Werte in den zweiten Registersatz laden,
*; auf den Programme an sich keinen direkten Zugriff haben, indem Sie einen
*; Apostroph an die Registerbezeichnung anhängen:
*X HL'
0000 - 89AB
*X C'
00 - 0FE
*; ...und sich das Ergebnis wieder ansehen:
*X
0100  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=00FE  DE=0000  HL=89AB  I=EC  IFF=0      PC=0100
*; führen Sie nun einige Einzelschritte aus und beobachten Sie, wie sich die
*; die Inhalte der Registersätze durch den EXX-Befehl vertauschen:
*T
0100  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=00FE  DE=0000  HL=89AB  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  I=EC  IFF=0      PC=0101
*T
0101  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=00FE  DE=0000  HL=89AB  I=EC  IFF=0      PC=0102
*T
0102  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=00FE  DE=0000  HL=89AB  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  I=EC  IFF=0      PC=0103
*T
0103  D9            EXX
 sOz0.0h0.0p0n0c0  A=00  BC=1300  DE=1234  HL=FF01  IX=0000  IY=0000  SP=0100
 'sOz0.0h0.0p0n0c0  A=00  BC=00FE  DE=0000  HL=89AB  I=EC  IFF=0      PC=0104
*
```

Rechnen im Tester

Wenn Sie sich die Liste der Z80-Befehle anschauen, die der Assembler erkennt (auf den letzten Seiten der Beschreibung des ZEAT-Betriebssystems), dann finden Sie bei vielen Befehlen im Operandenteil "<Ausdruck>". Damit sind Rechenausdrücke gemeint, unser Assembler verfügt nämlich über umfangreiche mathematische Fähigkeiten, die wir bis jetzt kaum benutzt haben. Auf den folgenden Seiten sollen Sie diese Möglichkeiten kennenlernen, in den folgenden Kapiteln werden sie dann angewendet.

Der Assembler arbeitet bei seinen Rechenfunktionen immer mit 16-Bit-Dualzahlen, wobei er allerdings vorzeichenlose Zahlen von solchen mit Vorzeichen im echten Komplement nicht unterscheidet. Das kann zu Fehlinterpretationen des Ergebnisses führen, ist aber bei Assemblern allgemein so üblich, weil man in Assemblerprogrammen sowohl Zahlen mit als auch ohne Vorzeichen benötigt. Außerdem erforderte eine eindeutige Darstellung mehr als 16 Bit, es ergäbe sich ein erheblich größerer Rechenaufwand bei geringem Nutzen.

Ein Beispiel für die mögliche Doppeldeutigkeit bei der Darstellung: es soll in einem Assmblerprogramm ein Registerpaar mit der positiven Zahl (ohne Vorzeichen) 60000 geladen werden. Der Assembler erzeugt ganz richtig die duale Darstellung 1110.1010.0110.0000 (0EA60H). Möchte man dagegen das Register mit der vorzeichenbehafteten Zahl -5536 laden, so erzeugt der Assembler genau die gleiche Bitfolge, die als Darstellung im echten Komplement ebenfalls völlig richtig ist. Es gibt in der Regel mit der Doppeldeutigkeit keine Probleme, solange man beachtet, daß sich negative Zahlen kleiner als -32768 (-128) mit einer 16-(8-)Bit-Komplementdarstellung nicht erfassen lassen. Besonders praktisch ist es, daß man auf die erwähnten Rechenfunktionen des Assemblers ebenso vom Tester aus zugreifen kann. Auch hier können alle Parametereingaben nicht nur als einfache Zahlkonstante, sondern in Form eines kompletten Rechenausdrucks erfolgen.

Darüberhinaus verfügt der Tester über ein Kommando, mit dem man ihn Rechenausdrücke an sich auswerten lassen kann. Dieses Kommando besteht aus einem Fragezeichen, gefolgt von einem oder mehreren durch Kommata getrennten Rechenausdrücken. Das ist genau das richtige Kommando, um sich die Rechenausdrücke anzuschauen. Sie werden damit ausgiebig spielen, um die Rechenausdrücke kennenzulernen und auch zu erfahren, wie praktisch das Rechnen mit Komplementdarstellungen im Grund ist.

Die Rechenfunktionen können zunächst selbstverständlich einzelne Zahleneingaben auswerten:

(Vollziehen Sie bitte alle Eingaben auf Ihrem System nach und womöglich noch ein paar andere Beispiele dazu - man kann nicht genug mit dem Computer spielen und dabei spielend lernen!)

```
*? 0,1,17,60000
0000  0001  0017   0:0000
```

Hoppla, da ist etwas schief gelaufen! Eigentlich sollten Dezimalzahlen eingegeben werden, der Computer hat diese aber als sedezimal dargestellte Zahlen interpretiert. Und die Zahl 60000H führt natürlich zu einem Überlauf. Mit dem Kommando DEC läßt sich im Tester auf die Eingabe dezimal dargestellter Zahlen umschalten:

```
*DEC
*? 0,1,17,60000
0000  0001  0011  EA60
```

Das sieht schon besser aus. Auch mit negativen Zahlen geht es:

```
*? -1,-17,-5536,-60000
FFFF  FFEF  EA60  15A0
```

Das letzte Ergebnis (15A0H) ist allerdings falsch, denn -60000 läßt sich mit 16 Bit garnicht darstellen. Jetzt soll aber auch einmal richtig gerechnet werden:

```
*? 2000+245, 100*37, 10000/3, 2-1, 1-2
08C5  0E74  0D05  0001  FFFF
```

oder etwas komplizierter:

```
*? (243+17)*(128-13)/(2*7-9)
175C
```

Rechnen Sie diesen Ausdruck einmal mit dem Taschenrechner nach, es kommt 5980 (=175CH) heraus, wenn alles richtig eingegeben wurde.

Mit der Rechenoperation NOT kann auch das unechte Komplement gebildet werden. Vergleichen Sie es mit dem echten Komplement, das mit einem Minuszeichen gebildet wird:

```
*? NOT 1, -1, NOT 2, -2
FFFE  FFFF  FFFD  FFFE
```

Das echte und das unechte Komplement unterscheiden sich tatsächlich um die Differenz Eins! Mit dem Kommando NOT

<ausdruck> wird also das unechte Komplement von <ausdruck> gebildet. Das Kürzel <ausdruck> bedeutet, wie schon gesagt, daß an dieser Stelle nicht nur eine Konstante (wie im vorliegenden Beispiel), sondern wiederum ein kompletter Rechenausdruck stehen darf.

Außer der Rechenoperation NOT sind noch einige andere Rechenoperationen möglich. Zunächst einmal kann mit

<ausdruck1> SHL <ausdruck2> bzw.
<ausdruck1> SHR <ausdruck2>

der <ausdruck1> um die im <ausdruck2> angegebenen Binärstellen nach links (SHL = Shift Left) bzw. rechts (SHR = Shift Right) geschoben werden. Diese Operation entspricht einer Multiplikation bzw. Division mit 2 hoch <ausdruck2>:

```
*? 1, 1 SHL 1, 1 SHL 2, 1 SHL 3, 1 SHL 4
0001  0002  0004  0008  0010
*?1280,1280 SHR 1, 1280 SHR 2, 1280 SHR 3, 1280 SHR 4
0500  0280  0140  00A0  0050
```

Achten Sie bei den Eingaben auf die exakte Schreibweise. Wenn die Leertasten zwischen den Zahlen und SHL bzw. SHR oder nach ? fehlen, gibt es eine Fehlermeldung. Dagegen ist es egal, ob z.B. SHL groß oder klein eingegeben wird (wie bei allen Eingaben des Betriebssystems) - hier im Heft wird es aus Gründen der Übersichtlichkeit immer groß angegeben.Mit den Rechenoperationen

HIGH <ausdruck> bzw.
LOW <ausdruck>

kann das HOB (High Order Byte) bzw. LOB (Low Order Byte) einer vierstelligen Sedezimalzahl, also die oberen bzw. unteren 8 Bit, gewonnen werden. Um beim Ausprobieren eine bessere Übersicht zu haben, wird mit dem Kommando HEX auf sedezimale Eingabe zurückgeschaltet:

```
*HEX
*? HIGH 1234, LOW 1234
0012  0034
```

Diese Rechenoperationen können einige Umwege über Maskierungen usw. sparen, wenn die entsprechenden Teile von 16-Bit-Zahlen gewünscht werden.

Als Ergänzung zur Division gibt es ferner die Modulo-Funktion MOD, die einen Rest bei einer Division bestimmt.

<ausdruck1> MOD <ausdruck2>

ergibt den Rest, der sich bei ganzzahliger Division von

<ausdruck1> durch <ausdruck2> ergibt:

```
*? 1234/100, 1234 MOD 100, 9876/1000, 9876 MOD 1000
0012  0034  0009  0876
```

Vergessen Sie bitte nicht, daß in diesem Beispiel die Zahlen in sedezimaler Darstellung eingegeben und, wie immer, ausgegeben wurden. Wenn Sie, wie in den vorhergehenden Beispielen, mit DEC auf dezimale Eingabe umschalten und 1234 durch 1oo teilen, dann steht das Ergebnis 12 und der Rest 34 in sedezimaler Darstellung (CH bzw. 22H) da.

Schließlich enthalten die mathematischen Fähigkeiten unseres Assemblers auch noch die bekannten logischen Verknüpfungen:

<ausdruck1> AND <ausdruck2>,
<ausdruck1> OR <ausdruck2> und
<ausdruck1> XOR <ausdruck2>,

mit denen die beiden Ausdrücke UND, ODER bzw. EXCLUSIV-ODER verknüpft werden:

```
*? 1234 AND OFF, 1234 OR OFF, 1234 XOR OFF
0034  12FF  12CB
```

(Es spielt übrigens keine Rolle, wenn Sie bei der Eingabe in gewohnter Weise mit dem nachgestellten H die sedezimale Darstellung kennzeichnen, der Assembler verdaut auch OFFH statt einfach OFF.) Wenn mehrere Rechenoperationen in einem Ausdruck vorkommen und die gewünschte Reihenfolge der Auswertung nicht durch Klammern festgelegt ist, dann werden die Operationen in dieser Reihenfolge durchgeführt:

```
LOW, HIGH
*, /, MOD, SHR, SHL
+, -
NOT
AND
OR, XOR
```

Alle beschriebenen Rechenoperationen stehen im Assembler und im Tester zur Verfügung. Wie Sie schon wissen, können dabei Zahlen natürlich auch in Form von Symbolen eingesetzt werden. Zusätzlich kann mit dem Zeichen $ der jeweilige momentane Stand des Programmzählers in eine Rechnung eingefügt werden. Außerdem können ASCII-Codes von einem oder zwei Buchstaben einfach in Anführungsstrichen eingesetzt werden:

```
*x
0100  C3 2A 01      JP   012A
 s0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  IX=0000  IY=0000  SP=0100
 's0z0.0h0.0p0n0c0  A=00  BC=0000  DE=0000  HL=0000  I=EC  IFF=0      PC=0100
*? $, $+100, 'A', 'AB', 'B'+80
0100  0200  0041  4142  00C2
```

Wenn Sie, wie wir Ihnen am Anfang dieses Kapitels geraten haben, ein wenig mit den Rechenausdrücken gespielt haben, kennen Sie jetzt die Fähigkeiten des Testers als "Taschenrechner" mit sedezimaler Ausgabe. Wozu diese Rechenfunktionen gut sind, ahnen Sie sicher und werden es bei den folgenden Programmierbeispielen sehen.

Die Hauptaufgabe des Testers besteht natürlich darin, Programme auszutesten und Fehler zu finden. Die dazu vorhandenen Tester-Kommandos für Einzelschritt und Programmablauf mit Unterbrechungen haben Sie teilweise schon kennengelernt. Der Tester verfügt aber noch über weitere nützliche Kommandos zu diesem Zweck, die bisher nicht verwendet wurden. Sie werden hier anschließend vorgestellt (während ab Seite H 7 eine komplette tabellarische Auflistung aller Tester-Kommandos steht).

Zunächst das schon bekannte Kommando für Einzelschritt:

```
T
T <ausdruck>
```

Es wird einer oder <ausdruck> Einzelschritte ausgeführt. Bei jedem Schritt wird der betreffende Befehl disassembliert und die Registerinhalte **nach** Ausführung des Schritts angezeigt. Die Kommandos

```
U
U <ausdruck>
```

haben die gleiche Wirkung wie das Kommando T, es erfolgt jedoch keine Disassemblierung und keine Ausgabe der Registerinhalte.Der Pogrammablauf in Einzelschritten kann jederzeit mit einer beliebigen Taste unterbrochen werden. Die Kommandos

```
G
G <ausdruck>
```

bewirken einen Programmstart beim aktuellen Stand des Programmzählers bzw. bei der Stelle <ausdruck>. Das Programm kann durch einen Warmstart ins Betriebssystem oder durch den Befehl RST 38H zum Tester zurückgelangen. Mit

```
G ,<ausdruck2>,<ausdruck3>,... ,<ausdruck9>
G <ausdruck1>,<ausdruck2>,<ausdruck3>,... ,<ausdruck8>
```

kann das Programm an der Stelle des Programmzählers bzw. der Stelle <ausdruck1> gestartet werden. Zusätzlich können bis zu 8 Unterbrechungsstellen gesetzt werden. Das Programm wird gestartet, stoppt beim Erreichen einer Unterbrechungsstelle und gibt die Registerinhalte aus. Wird die Ausgabe der Registerinhalte nicht gewünscht, so kann statt dessen die Form

GU ,<ausdruck2>,<ausdruck3>,... <ausdruck9>
GU <ausdruck1>,<ausdruck2>,<ausdruck3>,... <ausdruck8>

des Kommandos verwendet werden. Bei vielen in Schleifen ablaufenden Programmen ist eine Unterbrechung erst nach einer bestimmten Anzahl von Schleifendurchläufen erwünscht. Zu diesem Zweck kann jede Unterbrechungsstelle in der Form

...>,<ausdruck2a>;<ausdruck2b>,<...

definiert werden. Das Programm hält dann, wenn die Stelle <ausdruck2a> zum <ausdruck2b>-ten mal durchlaufen wird. Das Komma definiert also die Unterbrechungsstelle, das Semicolon die Anzahl der Durchläufe an dieser Stelle. Beim Kommando G werden die Registerinhalte bei jedem Durchlaufen einer Unterbrechungsstelle ausgegeben, beim Kommando GU entfallen alle Ausgaben.

Mit diesen Kommandos können auch umfangreichere Programme sehr schnell ausgetestet werden. Dabei ist es besonders angenehm, daß die Unterbrechungsstellen in Form von Symbolen angegeben werden können.

Zusammenfassung

Im Assembler und im Tester besteht die Möglichkeit, anstelle von Konstanten im Adreßteil von Befehlen eine ganze Reihe von Rechenausdrücken einzusetzen. Außerdem enthält der Tester neben den üblichen Kommandos wie TRACE usw. auch solche, die durch vielfache Möglichkeiten der Unterbrechungen das Austesten längerer Programme erleichtern.

Frage:

1. Wieso bedeutet das Verschieben der Bits einer im Speicher binär dargestellten Zahl um 2 Stellen nach links eine Multiplikation mit 2 hoch 2?

(Die Antwort zu dieser Frage finden Sie auf Seite G 9.)

Datenübertragung im Tester

Auf den vorangegangenen Seiten des Fachgebiets MODEM haben Sie gesehen, daß zwar eine Datenübertragung mit Protokoll vorgesehen ist, dabei aber nur der Inhalt des Textspeichers gesendet oder die empfangenen Daten im Textspeicher abgelegt werden. Die im Textspeicher übliche ASCII-Codierung verbietet die Übertragung assemblierter Programme. Das ist auch in den meisten anderen Betriebssystemen der Fall.

Das ZEAT-Betriebssystem bietet hier zusätzliche Möglichkeiten. Es stehen nämlich auch im Tester Kommandos zur Verfügung, mit denen beliebige Speicherbereiche über das Telefonmodem ausgesendet und empfangen werden können. Damit können auch assemblierte Programme mit fehlerkorrigierendem Protokoll ohne Risiko übertragen werden. Die entsprechenden Tester-Kommandos lauten

RX
RX <ausdruck>

Es werden Daten vom Modem empfangen und ab der Adresse 100H bzw. der Adresse <ausdruck> abgelegt. Vor dem Empfang werden die Modem-Parameter (Sende- und Empfangs-Baudrate) abgefragt. Wie im Modem-Programm kann durch ein 0 vor den Modem-Parametern das Protokoll abgeschaltet werden. Das Programm empfängt dann bis zum Abbruch mit CTRL-X (zurück in den Tester) oder CTRL-C (Warmstart).

Bei einer Übertragung mit Protokoll werden als kleinste Einheit immer komplette Blöcke von 128 Bytes empfangen.

TX <ausdruck1>,<ausdruck2>

Aus dem Speicher werden die zwischen den Adressen <ausdruck1> und <ausdruck2> stehenden Daten über das Modem gesendet. Vor dem Senden werden ebenfalls die Modem-Parameter abgefragt, auch hier ist die Option 0 für eine Übertragung ohne Protokoll möglich.

Wie beim Empfang kann die Übertragung mit CTRL-X oder CTRL-C abgebrochen werden. Es werden immer komplette Blöcke von 128 Bytes übertragen. Wenn es beispielsweise einmal nur 15 Bytes sein sollen, dann wird trotzdem ein kompletter Block mit 128 Bytes über das Modem geschickt.

Speicheroperationen

Der geringe Umfang fertig übersetzter Maschinensprache-Programme gegenüber dem Assembler-Quelltext war ein Grund für die Gebühren-sparende Übertragung mit TX bzw. RX. Es liegt aber auch nahe, fertig entwickelte und ausgetestete Programme nicht nur als Quelltext, sondern ebenfalls in übersetzter Form als Datenblock auf einem Massenspeicher abzulegen.

Zu diesem Zweck verfügt der Tester über zwei zu RX und TX analoge Kommandos, mit denen ein Speicherbereich von Cassette oder Diskette eingelesen oder auf Cassette bzw. Diskette abgespeichert werden kann. Es sind dies die Kommandos RF für Lesen (Read File) und SF für Speichern (Save File).

RF
RF <ausdruck>

Es werden Daten vom Massenspeicher eingelesen und ab der Adresse 100H bzw. der Adresse <ausdruck> abgelegt. Vor dem Einlesen wird nach einem Dateinamen gefragt.

(Bei Diskettenbetrieb werden als kleinste Einheit immer komplette Blöcke von 128 Bytes eingelesen.)

SF
SF <ausdruck1>,<ausdruck2> STARTADRESSE+ENDADRESSE

Die Daten von der Adresse <ausdruck1> bis zur Adresse <ausdruck2> werden auf dem Massenspeicher abgelegt. Vor dem Abspeichern wird nach einem Dateinamen gefragt. Dieser kann aus maximal acht beliebigen Zeichen bestehen, wobei selbstverständlich Zeichen mit speziellen Funktionen wie z.B. *, ?, § nicht verwendet werden dürfen.

(Bei Diskettenbetrieb werden CP/M-gemäße Filenamen erwartet, außerdem werden immer komplette Blöcke von 128 Bytes abgespeichert.)

Es gibt übrigens noch drei weitere Möglichkeiten, im Tester Speicheroperationen auszuführen. Mit dem Kommando S (Set Memory) kann man sich den Inhalt einzelner Speicherstellen ansehen und verändern, mit F (Fill Memory) kann ein ganzer Speicherbereich mit einem bestimmten Inhalt gefüllt werden und schließlich mit M (Move Memory) kann ein ganzer Speicherbereich verschoben werden:

S
S <ausdruck>

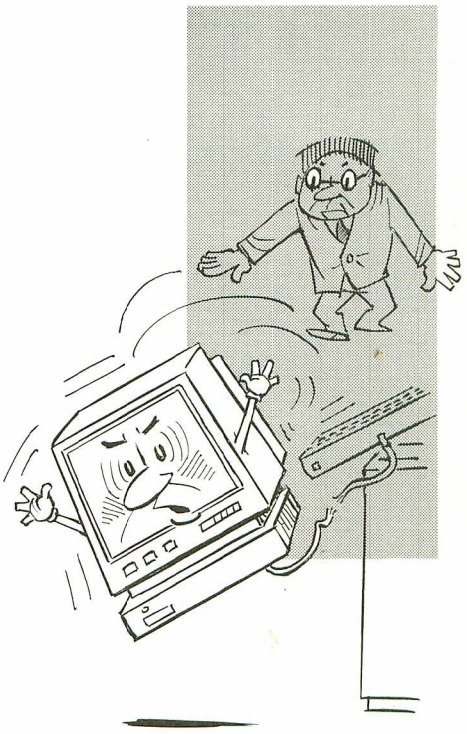
Der Inhalt der Speicherstelle, die nach dem letzten Kommando folgt, wird angezeigt (bzw. der Inhalt der Speicherstelle mit der Adresse <ausdruck>). Er kann mit der Taste CR (Return) unverändert quittiert werden oder es kann ein neuer Inhalt eingegeben werden. Dieser wird angezeigt und nach CR die Speicherzelle mit der nächsten Adresse angezeigt. Gibt man statt eines neuen Speicherinhalts einen Punkt (.) ein und danach CR, wird das Kommando abgebrochen.

F <ausdruck1>,<ausdruck2>,ausdruck3>

Der Speicherbereich zwischen den Adressen <ausdruck1> und <ausdruck2> wird mit <ausdruck3> gefüllt. Dieses Kommando haben Sie auf Seite B 29 benützt, um einen Speicherbereich mit Nullen zu füllen, damit beim Speicherausdruck nach einer Operation keine alten Speicherinhalte irritieren.

M <ausdruck1>,<ausdruck2>,<ausdruck3>

Der Speicherbereich von der Adresse <ausdruck1> bis zur Adresse <ausdruck2> wird zur Adresse <ausdruck3> verschoben.



Wir raten auch hier wieder: Ausprobieren! Und dabei werden Sie vielleicht feststellen, daß bei allen drei Kommandos eine Fehlermeldung (I) erfolgt, wenn Sie versuchen, in eine geschützte Systemadresse zu schreiben. Damit wird verhindert, daß Sie das System aus Versehen zum "Absturz" bringen.

Falls Sie nun schon ein richtiger Profi sind und ganz bewußt an einer geschützten Systemadresse etwas ändern wollen, können Sie das dem System mit dem Kommando

PROFI ON

mitteilen. Die Systemadressen werden dann nicht mehr vor dem Überschreiben im Tester oder Assembler geschützt. Der Zustand dieser Funktion kann abgefragt werden mit

PROFI

Mit dem Kommando

PROFI OFF

werden die Systemadressen wieder geschützt. Die PROFI-Kommandos können nicht nur innerhalb des Testers, sondern auch von der Ebene der Systemkommandos aus benutzt werden.

Unnötig zu sagen, daß man dieses Kommando, wenn überhaupt, mit größter Vorsicht verwenden sollte.

Zusammenfassung

Im Tester ist es möglich, Speicherinhalte über das Telefonmodem zu übertragen oder in den Speicher einzulesen. Damit können auch assemblierte Programme übertragen werden, und zwar mit Fehlerprotokoll. Der Tester enthält außerdem Kommandos, mit denen die Inhalte von Speicherstellen und -bereichen geändert und verschoben werden können. Mit den PROFI-Kommandos ist es möglich, gezielte Eingriffe in den geschützten Systemadressen vorzunehmen.

Frage:

1. Die Editor-Kommandos ESC-S und ESC-A bewirken im Grunde das gleiche wie die Tester-Kommandos SF und RF. Welcher wichtige Unterschied bestimmt den jeweiligen Anwendungsbereich?

(Die Antwort zu dieser Frage finden Sie auf Seite G 13.)

EPROM-Programmierung

Wenn Sie an Ihrem Computer ein EPROM-Programmiergerät angeschlossen haben, können Sie einige Tester-Kommandos benutzen und EPROMS "brennen", wie es im Laborjargon heißt.

Wir gehen davon aus, daß Sie im Speicher ein Programm stehen haben, das in ein EPROM geschrieben werden soll, um beispielsweise in einem kleinen Einplatinensystem als festes Programm für eine bestimmte Aufgabe zu dienen.

Nachdem der richtige Anschluß entsprechend der Hardware-Beschreibung erfolgt ist, wird der Tester aufgerufen und danach das Kommando

```
PROG /ausdruck1/,/ausdruck2/,/ausdruck3/
```

aufgerufen. Angenommen, Ihr Programm ist 2047 Byte lang und steht im Speicher von der Adresse 0100H bis zur Adresse 08FFH, dann steht folgender Ausdruck auf dem Bildschirm:

```
test
*
PROG 0100,08FF,00
CHECK +++++++++++++++
0000 ERROR(S)
PROGRAMM ++++++++++++++++
VERIFY +++++++++++++++++
0000 ERROR(S)
*
```

Die Anfangsadresse des Programms steht als /ausdruck1/, die Endadresse als /ausdruck2/ nach dem Kommando. In /ausdruck3/ steht die EPROM-Adresse, ab der das Programm eingeschrieben wird. Mit CHECK wird geprüft, ob das eingelegte EPROM leer ist, danach wird gebrannt (PROGRAMM). Anschließend überprüft (VERIFY) und ausgegeben, ob eventuell Fehler aufgetreten sind.

In gleicher Weise wie beim Assemblieren zeigt das Betriebssystem während alle dieser Läufe auf dem Bildschirm mit Pluszeichen an, daß etwas geschieht (damit nicht bei langen Programmen der Eindruck entsteht, das System sei ausgestiegen oder der Computer eingeschlafen.

Ein Pluszeichen steht dabei für 128 Byte.Unser Programmbeispiel ist 2047 Byte lang, das sind 16 mal 128 Byte, also stehen 16 Pluszeichen nach jedem Durchlauf auf dem Bildschirm.

Mit dem Kommando

VER /ausdruck1/,/ausdruck2/,/ausdruck3/

läßt sich gegebenenfalls noch einmal prüfen, ob das Programmiergerät auch richtig gearbeitet hat. Es wird nämlich der Inhalt des EPROM von EPROM-Adresse /ausdruck1/ bis EPROM-Adresse /ausdruck2/ mit dem Speicherbereich ab Adresse /ausdruck3/ verglichen. Auf dem Bildschirm steht danach:

```
VER 00,07FF,0100
++++++++++++++++
0000 ERROR(S)
```

Wenn Sie wissen wollen, was in einem EPROM drin steht, dann kann gelesen werden mit dem Kommando

READ /ausdruck1/,/ausdruck2/,/ausdruck3/

Im angeführten Beispiel sieht das so aus:

```
READ 00,07FF,0100
++++++++++++++++
```

Der Inhalt des EPROM im Programmiergerät wird ab der EPROM-Adresse 00 bis zur EPROM-Adresse 07FF in den Speicherbereich ab der Adresse 0100 eingelesen.

Zusammenfassung

Im Tester stehen 3 Kommandos zur Verfügung, mit denen EPROM im angeschlossenen Programmiergerät programmiert, überprüft und gelesen werden können. Dabei werden jeweils 128 bearbeitete Byte mit einem Pluszeichen auf dem Bildschirm quittiert.