Harald Krake

ZETBUG – ein komfortabler Z-80-Monitor

ZETBUG ist ein leistungsfähiger System-Monitor zum Erstellen und Austesten von Z-80-Maschinenprogrammen. Die hier beschriebene Version läuft direkt auf dem derzeit wohl bekanntesten Z-80-Personal-Computer, dem TRS-80, ist aber grundsätzlich auch auf jedes andere Z-80-System übertragbar.

Geladen wird ZETBUG normalerweise von der Kassette, und zwar mit dem Level-II-System-Kommando:

```
>SYSTEM
*? ZETBUG
*? /
```

Anschließend meldet sich ZETBUG und druckt sein Promptsymbol #, das Zeichen, daß eine Befehlszeile eingegeben werden kann.

```
ZETBUG Z80-MONITOR V1.0
# –
```

Jede Befehlszeile hat die Form:

x aaaa bbbb cccc Enter

wobei x das aus einem ASCII-Zeichen bestehende Kommando, aaaa das erste maximal vierstellige Argument (hexadezimal), bbbb das zweite und cccc das dritte Argument bedeutet. Hinter jedem Argument muß mindestens ein Blank (Leerzeichen) stehen. Zwischen x und aaaa kann, aber muß keines stehen. Nach Drücken der Enter-Taste analysiert ZETBUG die eingegebene Zeile und führt das entsprechende Kommando aus. Wird die Eingabe nicht verstanden, so quittiert ZETBUG dies mit einem erstaunten WHAT? # –.

Im Gegensatz zu manchen anderen Monitorsystemen erfolgt der Dialog mit dem Benutzer daher nicht über ein „live keyboard", d. h. es wird nicht jede gedrückte Taste einzeln interpretiert, sondern im sogenannten „terminal mode", indem die eingegebene Zeile als Ganzes betrachtet wird. Das hat den Vorteil, daß jederzeit noch Korrekturen sowie Cursorpositionierungen möglich sind, bevor man den Input mit Enter abschließt. *Bild 1* zeigt das komplette Maschinencode-Listing.

ZETBUG-Kommandos

 # I
(Initialize Register Save Area (RSA) and Memory-map)

Die RSA ist ein bestimmter Speicherbereich im RAM, der dazu dient, die Z-80-CPU-Register (SP/PC/IX/IY/HL/DE/BC/AF/HL'/DE'/BC'/AF') zwischenzuspeichern, während ZETBUG aktiv ist, d. h. wenn er gerade einen Befehl abarbeitet oder einfach nur auf eine Eingabe wartet. Jedesmal, wenn ein Z-80-Maschinenprogramm zu ZETBUG zurückkehrt, werden sämtliche oben aufgeführten Register in die RSA „gerettet", damit sie nicht verlorengehen. Das umgekehrte Prinzip gilt für die Programmausführung (siehe E). Das I-Kommando setzt nun alle Register auf 0000 sowie den Stackpointer auf 5000 und initialisiert den ZETBUG-Arbeitsbereich, die Memory-map. Außerdem wird der Breakpoint (siehe B) auf 0000, d. h. in den ROM-Bereich gelegt.

 # T aaaa bbbb
(Tabulate memory on screen)

Dieses Kommando druckt den Speicherbereich ab der Adresse aaaa bis zur Adresse bbbb auf dem Bildschirm aus. Am Anfang jeder Zeile steht die jeweilige Speicheradresse, danach folgen 16 Datenbytes. Auf diese Weise kann man sich schnell einen Überblick verschaffen bzw. einen Hexdump erstellen.

Anmerkung:

Bei jedem der 16 ZETBUG-Kommandos können die Argumente weggelassen werden. An deren Stelle muß dann allerdings ein „:" stehen. Dieser veranlaßt, daß die zuletzt verwendeten Argumente, welche in der Memory-map zwischengespeichert sind, für den jeweiligen Befehl aktualisiert werden, ohne daß diese erneut eingegeben werden müssen. Fehlt der „:", dann wird für alle drei Argumente 0000 angenommen. Beispiel: # T: druckt den Speicherbereich ab der Adresse, die zuletzt als Argument 1 (aaaa) eingegeben wurde, bis zur Adresse bbbb, die zuletzt als Argument 2 eingegeben wurde.

 # M aaaa
(Modify and inspect memory)

Bewirkt Übergang in den Modify-Modus. ZETBUG meldet sich mit der Speicheradresse aaaa und dem dazugehörigen Byte. Im Anschluß an das Promptsymbol kann man dann das neue Datenbyte oder auch mehrere, getrennt durch ein Blank, für die nachfolgenden Speicherzellen eingeben. Nach dem Drücken der Enter-Taste werden die neuen Daten in den Speicher eingeschrieben und geprüft, ob nicht gerade im ROM-Bereich oder eine unbelegte Adresse modifiziert werden soll. Gegebenenfalls erfolgt dann die Fehlermeldung ROM ERROR xxxx dd # mit Übergang zur entsprechenden Datenzelle. Wird Enter ohne Neueingabe gedrückt, so geht ZETBUG einfach zur nächsten Speicherzelle weiter. Mit einem „." kann man den Modify-Modus jederzeit wieder verlassen.

Beispiel:

```
# M 4A00   Enter
4A00   5B   # 77   Enter
4A01   43   # ED 53 20 40   Enter
4A05   09   # Enter
4A06   C9   # C8.   Enter
# T:   Enter
4A00   77   ED 53 20 40 09 C8...
#
```

Anmerkung:

Bei Verlassen des Modify-Modus wird zusätzlich das Argument 2 (bbbb) auf die zuletzt bearbeitete Adresse gesetzt. Auf diese Weise kann z. B. durch # T: der gerade modifizierte Bereich ausgedruckt werden. Bei allen Kommandos kann auch über den Zeilenrand hinaus geschrieben werden.