 # aaaa bbbb cccc

Neben den 16 beschriebenen ZETBUG-Kommandos gibt es noch diesen Blank- oder Leerbefehl. Er dient lediglich zum Abspeichern der Argumente, wenn im weiteren Verlauf mit der :–Option gearbeitet werden soll oder die Argumente aus einem anderen Grunde modifiziert werden müssen. Ohne Angabe von Argumenten (einfaches Drükken von ENTER) werden der Bildschirm um eine Zeile nach oben „gescrolled" und die Argumente auf 0000 gesetzt. Anwendung:

```
#     4000
# F:00   09   52   0D
471D   00 #
#
```

Hier bewirkt der Blankbefehl, daß die Datensuche erst ab der Adresse 4000 beginnen soll.

Erweiterung des Befehlssatzes

ZETBUG ist erweiterbar, denn er enthält keine Befehlstabelle, in der jedes Kommando katalogisiert ist, sondern „sucht" sich den vom Benutzer definierten Befehl im Speicher. Jedes Kommando hat an der Stelle, an der es von der ZETBUG-Hauptroutine angesprungen werden soll, seine eigene Codemarke. Diese hat die Form: 00 09 XX 0D ... routine ... C9, wobei XX der ASCII-Code des ZETBUG-Befehls ist. So lautet z. B. der Code für den R-Befehl: 00 09 52 0D.

Wird der Code in einem 4 KByte großen Bereich (ab 44C0) nicht gefunden, so druckt ZETBUG die Meldung WHAT? und verlangt eine neue Befehlszeile. Diese Methode erlaubt damit z. B. das unproblematische Hinzufügen zusätzlicher Kommandos, ohne ZETBUG abändern zu müssen. Allerdings muß beachtet werden, daß jede Kommandoroutine mit einem Z-80-Returnbefehl zu ZETBUG zurückkehren sollte (Achtung auf den Stack!).

ZETBUG auf anderen Systemen

Um ZETBUG auch auf anderen Systemen zu implementieren, muß man einige Veränderungen vornehmen. TRS-80/Level-2-spezifisch sind die INCH-Routine, die OUTCH-Routine und die Kommandos L,D,P. Zu beachten ist außerdem, daß die INLINE-Routine die Cursorposition (Level 2:4020) benötigt, um die Eingabezeile und deren Zeilenanfang zu errechnen. Die *Tabelle* zeigt die Adressen von ZETBUG.

Eingeben von ZETBUG in den Speicher

Um ZETBUG das erste Mal in den Speicher zu laden, kommt man leider nicht umhin, die 1,6 K Maschinencode von Hand einzutippen. Das geschieht mit dem BASIC-Eingabeprogramm (*Bild 2*). Hat man diese Fleißleistung dann glücklich vollbracht, so empfiehlt sich, alles nochmals nachzukontrollieren. Dann kann man ab 4C00 folgendes Maschinenprogramm eingeben:

```
21 B8 45 11 B8 43 01 FF 07 ED B0 C3 C9
44
```

Starten (im Level 2 BASIC):

```
>SYSTEM
*? / 19968  Enter
```

Sodann meldet sich ZETBUG. Jetzt als erstes auf Band sichern: # D 43B8 49FF 44C9 ZETBUG (besser mehrmals).

Tabelle: ZETBUG – Adressen und Unterprogramme (UP)

Memory-map	4020+4021	Cursorposition (Level II)
	ca. 4380 – 438F	ZETBUG Stack
	4390 – 4392	data
	4393+4394	start of input line (siehe INLINE)
	4395+4396	Argument 1
	4397+4398	Argument 2
	4399+439A	Argument 3
RSA	ab 439B in der Reihenfolge:	
Register	AF'/BC'/DE'/HL'/AF/BC/DE	
Save	HL/IX/IY/PC/SP	
Area	43B3+43B4	Breakpointadresse
	43B5–43B7	Opcodesequenz (3 Bytes)
ENTRY 44C9 (=17609 dez.)		ZETBUG-Entrypoint. Wird angesprungen, wenn ZETBUG aufgerufen werden soll (z. B. nach dem Laden von der Kassette).
RETURN 48B9 (C3 B9 48)		Adresse, die ein Benutzerprogramm anspringen muß, wenn es zu ZETBUG zurückkehren will. Keines der CPU-Register wird gerettet!
BREAK 4898 (CD 98 48)		Breakpointbehandlungsroutine (siehe B). Kann außerdem als Rücksprungadresse von einem Benutzerprogramm verwendet werden, wenn die CPU-Register in die RSA gerettet werden sollen.
LOAD 487E		Unterprogramm schreibt die RSA in die CPU zurück (außer SP- und PC-Register!)
SAVE 4867		UP rettet die CPU-Register (außer SP und PC) in die RSA.
INCH 43B8		UP liest ein ASCII-Zeichen von der Tastatur. Character in accu.
OUTCH 43C5		UP setzt ein Zeichen auf den Bildschirm und bewegt den Cursor (character in accu).
PRINT 43E1		Aufruf: CD E1 43 aa bb cc dd ... nn 00. UP druckt den ASCII-string aa bb ... nn auf den Bildschirm und bewegt den Cursor.
INLINE 43EE		UP liest eine Zeile vom Bildschirm (wickelt alle notwendigen INCH/OUTCH-Routinen ab) und errechnet den Zeilenanfang (start of line in 4393+4394). Der Zeilenanfang ist immer zwei Zeichen nach dem Promptsymbol.
INHEX 443F		UP liest eine 4-digit-Hexzahl vom Schirm in das HL-Register. Der Bufferpointer ist das DE-Register (wird aut. nachkorrigiert).
OUTHEX 4450		UP druckt den Accu im Hexformat aus.
OUTHL 4469		wie OUTHEX, jedoch HL-Register.

ZETBUG belegt den Speicherbereich von 43B8 bis 49FF.
Das Benutzerprogramm kann ab 4A00 stehen.
Der Stackpointer ist normalerweise auf 5000hex gesetzt, kann aber verändert werden (durch R/SP, siehe R).