Ohne gutes Werkzeug geht es nicht:

SMON, Teil 3

Der Maschinensprache-Monitor geht langsam seiner Vollendung entgegen. In diesem Teil kommen drei interessante Befehle hinzu, die vor allem bei der Fehlersuche sehr hilfreich sind.

Sicherlich haben Sie sich beim letzten Mal gewundert, wie es möglich ist, daß so viele neue Befehle in so wenig Programm stecken können. »Schuld« daran ist das SMON-Konzept. Wir haben im ersten Teil bereits alle Ein- und Ausgaberoutinen untergebracht. Alle Erweiterungen können nun darauf aufbauen und werden dementsprechend kürzer. Wir haben Ihnen sogar noch einen Befehl verschwiegen, der beim letzten Mal schon vorhanden war. Vielleicht haben Sie versehentlich einmal »B« eingegeben, und SMON hat mit einem Fragezeichen reagiert und damit gezeigt, daß er mit »B« etwas anzufangen weiß.

Im heutigen Teil 3 unserer SMON-Serie wollen wir Ihnen drei weitere Befehle vorstellen: BASIC-DATA, KONTROLLE und FIND. Es sind diesmal nur drei neue Befehle, nicht weil wir Sie für den nächsten Artikel über »SMON« auf die Folter spannen wollen, sondern weil wir der Meinung sind, daß ein so umfassender Befehl wie »FIND« schon eine Menge an Beispielen braucht, um verstanden zu werden.

BASIC-DATA

B (ANFADR ENDADR) wandelt das Maschinenprogramm von ANFADR bis ENDADR-1 in Basic-DATA-Zeilen um.

```
B 4000 4020
```

Unser Testprogramm (Sie erinnern sich doch noch an unser kleines Programm aus 11/84?) wird in DATA-Werte umgerechnet und dann mit Zeilennummer 32000 beginnend im Basic-Speicher abgelegt. Ein im Speicher befindliches Basic-Programm (zum Beispiel ein Basic-Lader) mit kleineren Zeilennummern kann dann diese DATA-Zeilen benutzen.

Wenn Sie das Testprogramm wie oben beschrieben umgewandelt haben, verlassen Sie nun mit »X« den SMON und überzeugen sich mit »LIST« von der Ausführung. Dann können Sie folgendes eingeben:

```
10 FOR I=16384 TO 16415 : READ D :POKE I,D : NEXT
```

In Verbindung mit den oben erzeugten DATA-Zeilen (und RUN!) hätten Sie wieder das ursprüngliche Maschinenprogramm im Speicher. Falls Sie dieses Beispiel durchführen wollen, denken Sie bitte daran, daß Sie nach Erstellung der DATAs, das Originalprogramm zum Beispiel mit OCCUPY (O 4000 4020 AA) überschreiben, damit Sie die richtige Ausführung überprüfen können. Der BRK-Befehl am Ende des Testprogramms bewirkt einen Sprung zum SMON zurück. Wollen Sie ein Maschinenprogramm von Basic aus starten und auch wieder dorthin zurückgelangen, muß der letzte Befehl ein RTS sein. Probieren Sie es aus, indem Sie das Basic-Programm um 20 SYS 16384 erweitern.

KONTROLLE

K (ANFADR ENDADR) listet die ASCII-Zeichen im gewünschten Bereich. Es werden jeweils 32 Zeichen pro Zeile ausgegeben, so daß man sich einen schnellen Überblick über Texte oder Tabellen verschaffen kann.

Beispiel:

```
K 4000
```

listet die ersten 32 Zeichen unseres Programms. Die weitere Ausgabe ist genau wie beim Disassemblieren durch Druck auf SPACE oder RETURN möglich. Auch hier können Sie wie bei den anderen Bildschirm-Ausgabebefehlen Änderungen durch einfaches Überschreiben vornehmen (natürlich nicht im ROM und nur mit ASCII-Zeichen!).

Als Beispiel wollen wir einmal im Basic »herumpfuschen«: Das geht natürlich nicht so ohne weiteres, weil das Basic im ROM steht und damit nicht verändert werden kann. Tippen Sie bitte folgendes ein:

```
W A000 C000 A000
```

Auf den ersten Blick eine unsinnige Anweisung; der Speicher soll von A000 bis C000 nach A000 verschoben werden. Dieser Befehl entspricht exakt der Basic-Schleife

```
FOR I = 40960 TO 49152 : POKE I, PEEK (I) : NEXT
```

Nun ist es aber so, daß beim PEEK das ROM gelesen, beim POKE aber ins darunterliegende RAM geschrieben wird. Wir erreichen also, daß das Basic ins RAM kopiert wird. Jetzt müssen wir dafür sorgen, daß das Betriebssystem sein Basic aus dem RAM und nicht aus dem ROM holt. Zuständig dafür ist die Speicherstelle 0001. Geben Sie bitte »M 0001« ein, und überschreiben Sie die »37« mit »36«.

Es passiert gar nichts. Jetzt tritt unser K-Kommando in Aktion. Geben Sie ein: K A100 A360

Was Sie sehen, sind die Basic-Befehlswörter und -Meldungen. Schalten Sie mit SHIFT/CBM auf Kleinschrift, dann erkennen Sie, daß der jeweils letzte Buchstabe eines Befehlswortes groß geschrieben ist (Endekennung). Jetzt ändern Sie durch Überschreiben das »LIST« (A100) in »LUST« und »ERROR« (A360) in »FAELER«. (Bei »FAELER« müssen Sie ein Zeichen vor »ERROR« beginnen, sonst paßt es nicht.)

Verlassen Sie jetzt SMON mit »X« und geben Sie danach ein:
POKE 1,54
SMON schaltet nämlich beim »X«-Befehl immer auf das Basic-ROM zurück, daher müssen wir wieder auf unser geändertes Basic umschalten. Schreiben Sie nun einen Basic-Dreizeiler und versuchen Sie, diesen zu LISTen. Ergebnis? Versuchen Sie es jetzt einmal mit »LUST«. Ihrer weiteren Phantasie sind keine Grenzen mehr gesetzt...
Wie oben angesprochen stellt SMON eine Reihe verschiedener Suchroutinen zur Verfügung, die im Folgenden an vielen Beispielen beschrieben werden. Alle diese Befehle bestehen aus zwei Zeichen und beginnen mit dem Buchstaben »F«.

FIND

F (HEX-WERT(e), ANFADR ENDADR) sucht nach einzelnen HEX-Werten innerhalb eines bestimmten Bereichs. Das zweite Zeichen (hinter F) ist hier ein Leerzeichen und darf nicht weggelassen werden! Die Bereichsangabe kann wie bei allen folgenden Befehlen entfallen, dann wird der gesamte Speicher durchsucht.
Beispiel: Wir suchen alle Befehle LDY #01, also die Werte A0 01 im Bereich von $2000 bis $6000.

F A0 01, 2000 6000 (Die Leerzeichen zwischen den Hexbytes dürfen nicht weggelassen werden !). Es erscheinen alle Speicherstellen, die die gesuchten Bytes enthalten, also zum Beispiel 4000.

FA (Adresse, ANFADR ENDADR) sucht alle Befehle, die eine bestimmte Adresse als Operanden haben (absolut). Die Adresse braucht nicht vollständig angegeben zu werden, es kann das Jokerzeichen »*« benutzt werden.

1. Beispiel: Wir suchen alle JSR FFD2-Befehle im Bereich $2000 bis $6000.

FAFFD2,2000 6000

Es erscheinen alle Befehle disassembliert, die FFD2 im Operanden enthalten (also auch LDA FFD2 oder STA FFD2,Y...).

2. Beispiel: Wir suchen alle Befehle, die auf den Grafikbereich ($D000 bis $DFFF) zugreifen.

FA D* * *,2000 6000

Der Joker kann aber auch zum Beispiel zur Suche im Bereich $D000 bis $D0FF dienen: FAD0* *,2000 6000

FR (ADR, ANFADR ENDADR) sucht nach relativen Sprungzielen. Anders als bei absoluten Sprüngen (JMP, JSR) benutzen die Branch-Befehle eine relative Adressierung, also zum Beispiel »Verzweige 10 vor« oder »37 zurück«. Solche Sprünge lassen sich mit dem FA-Kommando nicht finden. Hier wird »FR« eingesetzt.
Beispiel: Gesucht werden alle Branch-Befehle, die die Adresse $4002 anspringen.

FR4002,2000 6000

Natürlich können solche Befehle nur höchstens 128 Byte vom Sprungziel entfernt sein. Die Bereichsangabe ist hier also viel zu groß gewählt (SMON stört dies allerdings nicht). Der Einsatz des Jokers ist hier ebenfalls wie oben beschrieben möglich.

FT (ANFADR ENDADR) sucht Tabellen im angegebenen Bereich. SMON behandelt dabei alles, was sich nicht disassemblieren läßt, als Tabelle.
Beispiel: Wir suchen Tabellen oder Text im Bereich $2000 bis $6000.

FT 2000 6000

FZ (Adr, ANFADR ENDADR) sucht alle Befehle, die Zeropage-Adressen haben.
1. Beispiel: FZC5,2000 6000 findet alle Befehle, die C5 adressieren, also zum Beispiel BIT $C5, LDA (C5), Y etc.
2. Beispiel: FZF*,2000 6000 findet alle Befehle, die den Bereich zwischen $F0 und $FF adressieren.
3. Beispiel: FZ* *,2000 6000 findet sämtliche Befehle mit Zeropage-Adressierung.

FI (Operand, ANFADR ENDADR) sucht alle Befehle mit unmittelbarer Adressierung (immediate).
Beispiel: Gesucht werden Befehle, die zum Beispiel das Y-Register mit 01 laden. FI01,2000 6000 findet LDY #01 in Adresse $4000.

Sie sehen, SMON bietet eine Fülle von verschiedensten FIND-Routinen, mit denen alles gesucht und auch gefunden (!) werden kann. Zum Üben wollen wir ein großes Preisausschreiben veranstalten, bei dem Sie zumindest an Erfahrung gewinnen können! Hier sind die Aufgaben, die es zu lösen gilt:
1. Wie oft wird von SMON aus in das Betriebssystem gesprungen ($E000 — $FFFF)?
2. Welche Zeropage-Adressen benutzt SMON?
3. Wo wird die Hintergrundfarbe ($D021), wo die Schreibfarbe ($0286) gesetzt?
4. Wo sind im SMON die Tabellen untergebracht?
5. An einigen Stellen stehen Befehle, die die Register unmittelbar mit dem Highbyte des SMON-Speicherbereichs laden (dez. 49152 — 52208). Rechnen Sie die HEX-Werte aus und suchen Sie die Speicherstellen.
6. An zwei Stellen stehen aufeinanderfolgend fünf Nullen. Wo?
Notieren Sie Ihre Lösung bitte auf einem Zettel und werfen Sie diesen fort. Der Rechtsweg ist ausgeschlossen...

Der Lösung ein Stück näher...

Auch heute werden Sie nicht entlassen, ohne daß wir Ihnen einige Tips für eigene Assemblerprogramme mitgeben. Erinnern Sie sich noch an den in der letzten Ausgabe angesprochenen 16-Bit-Vergleich? Dieser wird im SMON zum Beispiel dazu benutzt, festzustellen, ob ein Programmteil weiter durchlaufen werden soll, oder ob das Ende erreicht ist. Das prüft SMON bei fast allen Befehlen, jüngstes Beispiel sind die FIND-Kommandos. Zur Erinnerung: Wir wollten zwei 16-Bit-Zahlen vergleichen, der Prozessor kann aber nur mit 8-Bit-Zahlen umgehen.

Wir brauchen dazu einen hochlaufenden Zähler (er heißt in unserem Beispiel »Programmzähler« PC und besteht aus Highbyte PCH und Lowbyte PCL) und einen End-Zeiger (ENDHI und ENDLO). Unser Programm dafür sah folgendermaßen aus:

```
LDA     PCL
CMP     ENDLO
LDA     PCH
SBC     ENDHI
```

Anschließend haben wir das Carry-Flag überprüft und festgestellt, daß schon bei Übereinstimmung beider Adressen dieses Flag gesetzt war. In unserem Falle würde also ein Beenden des Programmteils mit »BCS ENDE« zu einer »Unterschlagung« des letzten noch auszuführenden Befehls führen. Um hier einen Weg aus dem Dilemma zu finden, wollen wir uns das Verhalten der Zero- und Carry-Flagge im Status-Register einmal genauer ansehen. Und zwar in Abhängigkeit von PC und dem ENDE-Zeiger.

Sie sehen in Listing 1, daß wir den Programmzähler (PCL/PCH) nach $FB/$FC und den Endezeiger (ENDLO/ENDHI) nach $FD/$FE schreiben. Dann springen wir die Routine an, die die Überprüfung auf erreichtes Ende im SMON vornimmt (CMPEND in $C466). Zum Abschluß sorgt der BRK-Befehl dafür, daß wir wieder im SMON landen. Schauen Sie sich die entsprechende Routine im SMON mit »D C466« an. Sie werden erkennen, daß sie den oben angesprochenen 16-Bit-Vergleich durchführt.

Speichern Sie jetzt dieses Programm mit »S "CMP-TEST" 4100 4112« ab und starten es mit »G4100«. Nachdem das Programm gelaufen ist, meldet es sich mit der Registeranzeige zurück. Achten Sie dabei vor allem auf die Statusregister-Anzeige rechts, uns interessieren die Werte für Z und C (Zero- und Carry-Flagge).

Tippen Sie bitte einmal folgendes Programm ein (mit »A 4100«):

```
4100    LDA     #00
4102    STA     FB      (=PCL)
4104    STA     FD      (=ENDLO)
4106    LDA     #C0
4108    STA     FC      (=PCH)
410A    LDA     #C1
410C    STA     FE      (=ENDHI)
410E    JSR     C466    (=CMPEND)
4111    BRK
```

Listing 1.

Wir wollen herausfinden, was passiert, wenn der Programmzähler (PC) kleiner, gleich oder größer als ENDE ist. Wenn Sie jetzt in Speicherstelle $4106 für PCH den Wert C1 einsetzen, können Sie den Vorgang wiederholen und die Änderung der Flaggen notieren. Anschließend setzen Sie C2 für PCH in $4106 ein. Tippen Sie »D 4100 4112«, gehen mit dem Cursor in die Zeile 4106 und überschreiben den Wert #C0 mit dem neuen Wert #C1 beziehungsweise #C2.

		PC<END	PC=END	PC>END
PC	FC/FB	C0/00	C1/00	C2/00
END	FE/FD	C1/00	C1/00	C1/00
		ZC	ZC	ZC
		00	11	01

So sollte Ihre Tabelle zum Schluß aussehen. Im ersten Fall (PC ist kleiner als END) ist das Carry-Flag gelöscht. Dann (PC ist gleich END) sind Z- und C-Flagge gesetzt, zum Schluß ist nur noch das C-Flag 1. Jetzt können wir unseren Vorstellungen entsprechend reagieren und mit den Branch-Befehlen verzweigen. Sehen Sie sich mit dem Disassembler auch einmal andere Routinen im SMON daraufhin an.

Hinweise zum Abtippen

Aus vielen Telefonanrufen ist uns klar geworden, daß der bisherige DATA-Lader oft die Fehlerquellen nicht genau genug aufzeigte. Mit diesem Lader dürften Sie beim Abtippen keine Schwierigkeiten mehr haben. Danach sollten Sie das Ladeprogramm auf alle Fälle auf Diskette oder Kassette abspeichern. Nach Eingabe von RUN muß der Lader bis zum READY durchlaufen. Um den neuen Teil an SMON anzukoppeln, müssen Sie jetzt den SMON vom letzten Mal mit »,8,1« laden und mit SYS 49152 starten. Jetzt können Sie mit

S "SMON $C000" C000 CBF1

das bis hier komplette Programm abspeichern. Natürlich müssen Diskettenbesitzer eine andere Diskette einlegen oder das alte SMON-Programm nach dem Laden (!) löschen. Und bis zum nächsten Mal: Üben, üben, üben!

(N. Mann/D. Weineck/gk)

Listing 2. Der dritte Teil des SMON als Basic-Lader

```
1 REM ********************************     <250>
2 REM *    ++++ SMON TEIL 3 ++++      *     <219>
3 REM *                               *     <230>
4 REM *  VON N.MANN UND D.WEINECK     *     <223>
5 REM *  FLEETRADE 40, 2800 BREMEN 1  *     <184>
6 REM *  TEL: 0421/ 493090            *     <055>
7 REM *       0421/ 231401            *     <011>
8 REM ********************************     <001>
9 :                                         <067>
10 DIM H(75) : FOR I=0 TO 9                 <088>
20 H(48+I)=I : H(65+I)=I+10 : NEXT          <250>
30 FOR I=51895 TO 52208:READ A$             <113>
40 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))   <063>
50 D=H(H)*16+H(L) : S=S+D : POKE I,D        <181>
60 A=A+1:IF A< 8 THEN NEXT : A=-1           <110>
65 PRINT "ZEILE:";1000+Z;                   <012>
70 READ V : Z=Z+1 : IF V=S THEN 85          <210>
80 PRINT"PRUEFSUMMENFEHLER!";999+Z:STOP     <015>
85 IF A<0 THEN END                          <043>
90 S=0 : A=0 : PRINT : NEXT : END           <053>
98 :                                        <156>
99 :                                        <157>
1000 DATA 20,64,C2,A2,27,20,40,C3, 818      <200>
1001 DATA 20,23,C3,A0,08,A2,00,20, 624      <182>
1002 DATA 4C,C3,A1,FB,20,39,C4,D0, 1176     <063>
1003 DATA F9,A2,00,20,5D,C4,F0,03, 975      <243>
1004 DATA 4C,BA,CA,60,20,7E,C2,A0, 1072     <068>
1005 DATA 03,20,CF,FF,88,D0,FA,20, 1123     <065>
1006 DATA CA,C2,C9,2E,F0,02,91,FB, 1281     <086>
1007 DATA C8,C0,20,90,F2,60,20,7A, 1060     <013>
1008 DATA C2,A2,00,A1,FB,C1,FD,D0, 1422     <083>
1009 DATA 0B,20,67,C3,E6,FD,D0,F3, 1275     <079>
1010 DATA E6,FE,D0,EF,20,4C,C3,4C, 1310     <104>
1011 DATA 23,C3,A9,FF,A2,04,95,FA, 1219     <083>
1012 DATA CA,D0,FB,20,CA,C2,A2,05, 1256     <091>
1013 DATA DD,6E,C0,F0,45,CA,D0,F8, 1490     <116>
1014 DATA 86,A9,20,B4,CB,E8,20,CF, 1189     <092>
1015 DATA FF,C9,20,F0,F3,C9,2C,D0, 1424     <102>
1016 DATA 03,20,7A,C2,20,51,C3,A4, 823      <224>
1017 DATA A9,B1,FB,20,D6,CB,D0,18, 1278     <100>
1018 DATA 88,10,F6,20,23,C3,20,4C, 768      <232>
1019 DATA C3,A4,D3,C0,24,90,09,20, 983      <244>
1020 DATA 94,C4,20,72,C4,20,51,C3, 994      <236>
1021 DATA 20,63,C4,90,DA,A0,27,4C, 964      <007>
1022 DATA 96,C4,BD,73,C0,85,A8,BD, 1332     <098>
1023 DATA 78,C0,85,A9,AA,F0,06,20, 1062     <055>
1024 DATA B4,CB,CA,D0,FA,20,7A,C2, 1391     <126>
1025 DATA 20,CB,C4,20,2C,C5,A5,A8, 1037     <079>
1026 DATA 24,AB,D0,09,A8,D0,21,A5, 998      <031>
1027 DATA AD,D0,1D,F0,0D,A4,A9,B9, 1181     <119>
1028 DATA AD,00,20,D6,CB,D0,11,88, 983      <027>
1029 DATA D0,F5,84,AA,20,8C,C5,20, 1156     <077>
1030 DATA 6F,C4,20,66,C4,90,D1,60, 1086     <056>
1031 DATA 20,6A,C6,F0,F5,20,C0,CB, 1248     <090>
1032 DATA 9D,CC,03,BD,3C,03,9D,6C, 881      <075>
1033 DATA 03,20,CA,C2,A0,0F,C9,2A, 849      <041>
1034 DATA D0,02,A0,00,20,AF,C2,9D, 928      <022>
1035 DATA 3C,03,98,9D,9C,03,60,85, 760      <008>
1036 DATA B4,4A,4A,4A,4A,59,6C,03, 676      <042>
1037 DATA 39,CC,03,29,0F,D0,0A,A5, 703      <031>
1038 DATA B4,59,3C,03,39,9C,03,29, 589      <014>
1039 DATA 0F,60, 111                        <093>
```