Ohne gutes Werkzeug geht es nicht: SMON

TEIL 1

In mehreren Teilen möchten wir Ihnen einen Maschinensprachmonitor vorstellen. Parallel zum Kursus über Assembler-Programmierung wird Schritt für Schritt ein Programm entstehen, das sich durchaus mit kommerziellen Monitoren messen kann.

Ich kann mich noch gut an unsere ersten Schritte in die Maschinensprache erinnern. Ausgerüstet mit einer Befehlsliste für den 6502 und einem in Basic geschriebenen »Mini-Monitor« entstanden Programme, die 3 und 5 addieren und das Ergebnis im Speicher ablegen konnten. Dazu mußten wir die Befehlcodes aus der Liste heraussuchen und dann in den Speicher »POKEn«. Jeder Sprung mußte von Hand ausgerechnet werden, jeder falsch herausgesuchte Befehl führte zum Programmabsturz. Der erste Disassembler — ein Programm zur Anzeige der Maschinenbefehle in Assemblersprache — war für uns die Offenbarung. Von nun an konnten wir Maschinenprogramme analysieren und daraus lernen. Zum Verständnis der Maschinensprache ist es nämlich noch weit mehr als bei anderen Sprachen wichtig, vorhandene Programme zu verstehen und sich dabei die wichtigsten Techniken anzueignen.

Mit der Zeit wuchsen unsere Ansprüche, ein Assembler mußte her, um die neugewonnenen Erkenntnisse auch auszuprobieren. Das war zuerst wieder ein Basic-Programm, langsam und wenig komfortabel, aber immerhin. Wir schrieben unsere ersten kleinen Routinen, vor allem, um vorhandene Maschinenprogramme unseren eigenen Wünschen anzupassen. Mit dem AMON für den VC 20 bekamen wir dann einen Monitor, der (fast) alle unsere Wünsche erfüllte. Als wir jedoch auf den C 64 umstiegen, mußten wir feststellen, daß es für diesen Computer nichts gab, das uns zufriedenstellen konnte. Der einzige Ausweg: Selbst programmieren. So entstand im Laufe eines Jahres SMON. Ursprünglich hatten wir nur vor, die Funktionen von AMON für den C 64 zu programmieren, aber dabei blieb es nicht. Immer neue Befehle und Routinen kamen hinzu, bis wir endlich zufrieden waren.

Was bietet SMON?

Zunächst ist alles enthalten, was zum »Standard« gehört: Memory-Dump, also die Anzeige des Speicherinhalts in Hexbytes, mit Änderungsmöglichkeiten, ein Disassembler mit Änderungsmöglichkeit sowie Routinen zum Laden, Abspeichern und Starten von Maschinenprogrammen. Darüber hinaus gibt es einen kleinen Direktassembler, der sogar Labels verarbeitet, Befehle zum Verschieben im Speicher mit und ohne Umrechnen der Adressen und Routinen zum Umrechnen von Hex-, Dezimal- und Binärzahlen. Der besondere Clou von SMON liegt aber zweifellos in seinen leistungsfähigen Suchroutinen und vor allem im Trace-Modus. Damit lassen sich Maschinenprogramme Schritt für Schritt abarbeiten und kontrollieren.

Dieser erste Teil umfaßt sämtliche Eingabe- und Ausgaberoutinen, die Registeranzeige, den Memory-Dump sowie Disassembler und Assembler. Damit steht Ihnen bereits ein lauffähiges Monitorprogramm mit den unten aufgeführten Befehlen zur Verfügung.

Der Monitor benötigt für alle Eingaben die hexadezimale Schreibweise, das heißt zu den Zahlen 1 bis 9 kommen noch die Buchstaben A (für dez. 10) bis F (für dez. 15) hinzu.

Bei der Eingabe von Adressen ist folgendes zu beachten: [ANFADR] bedeutet exakt die Startadresse, [ENDADR] bedeutet hierbei die erste Adresse hinter dem gewählten Bereich. Im Normalfall ist die Eingabe mit und ohne Leerzeichen zulässig. Beim Abweichen von dieser Regel wird darauf besonders verwiesen.

Assemblieren

A [ANFADR]
Assemblierung beginnt bei angegebener Adresse
Beispiel:
A 4000 Beginn bei Startadresse $4000

Nach Eingabe von »RETURN« erscheint auf dem Bildschirm die gewählte Adresse mit einem blinkenden Cursor. Die Befehle werden so eingegeben, wie sie der Disassembler zeigt: LDY #00 oder LDA 400E,Y und so weiter. »RETURN« schließt die Eingabe der Zeile ab. Bei fehlerhafter Eingabe springt der Cursor wieder in die Anfangsposition zurück. Ansonsten wird der Befehl disassembliert und nach Ausgabe der Hex-Bytes gelistet. Zur Korrektur vorhergehender Zeilen gehen Sie mit dem Cursor zur Anfangsposition (hinter die Adresse) zurück, schreiben den Befehl neu und gehen nach »RETURN« mit dem Cursor wieder in die letzte Zeile. Falls Ihnen bei Sprüngen (Branch-Befehl, JSR und JMP) die Zieladressen noch nicht bekannt sind, geben Sie einfach sogenannte »Label« ein.

Ein Label besteht aus dem Buchstaben »M« (für Marke) und einer zweistelligen Hex-Zahl von 01 bis 30.
Zum Beispiel: BCC M01

Wenn Sie die Zieladresse für diesen Sprung erreicht haben, dann kennzeichnen Sie diese mit eben dieser »Marke«.
Zum Beispiel: M01 LDY #00

Einzelne Bytes nimmt der Assembler an, indem Sie diese mit einem Punkt kennzeichnen: .00 oder .AB. In diesem Modus werden die Eingaben natürlich nicht disassembliert.

Nach Beendigung des Assemblierens geben Sie »F« ein. Danach sehen Sie alle Ihre Eingaben noch einmal aufgelistet und korrigieren bei Bedarf wie beim Disassembler (!) angegeben.

Probieren Sie einmal das folgende Beispiel:

A 4000

Der Assembler meldet sich mit: »4000« und einem blinkenden Cursor. Geben Sie nun ein (die Adressen erscheinen automatisch):

```
4000 LDY #00          4009 CPY #12
4002 LDA 400E,Y       400B BCC 4002
4005 JSR FFD2         400D BRK
4008 INY
```

Die folgenden Bytes werden wie beschrieben mit einem Punkt eingegeben. Sie werden nicht disassembliert.

```
400E .0D              4017 .54
400F .0D              4018 .20
4010 .53              4019 .53
4011 .4D              401A .55
4012 .4F              401B .50
4013 .4E              401C .45
4014 .20              401D .52
4015 .49              401E .0D
4016 .53              401F .0D
```

Drücken Sie anschließend »F«. Ihr Programm wird nochmal aufgelistet. Starten Sie es nun mit »G 4000«. Es erscheint ein Text auf dem Bildschirm — lassen Sie sich überraschen.

Disassemblieren

D [ANFADR,ENDADR]

disassembliert den Bereich von ANFADR bis ENDADR, wobei ENDADR nicht eingegeben werden muß. Wird keine Endadresse eingegeben, erscheint zunächst nur eine Zeile:

ADR	HEXBYTES	BEFEHL
4000	A000	LDY #00

Mit der SPACE-Taste wird der jeweils nächste Befehl in der gleichen Art und Weise gezeigt. Wünschen Sie eine fortlaufende Ausgabe, drücken Sie »RETURN«. Die Ausgabe wird dann so lange fortgesetzt, bis eine weitere Taste gedrückt wird oder bis ENDADR erreicht ist. Mit »RUN/STOP« springen Sie jederzeit in den Eingabemodus zurück.

Das Komma, das vor der Adresse auf dem Bildschirm erscheint, ist ein »hidden command« (verstecktes Kommando). Es braucht nicht eingegeben zu werden, da es automatisch beim Disassemblieren angezeigt wird. So ermöglicht es ein einfaches Ändern des Programms. Fahren Sie mit dem Cursor auf den zu ändernden Befehl und überschreiben Sie ihn mit dem neuen. Wenn Sie jetzt »RETURN« drücken, erkennt SMON das Komma als Befehl und führt ihn im Speicher aus. Achten Sie aber darauf, daß der neue Befehl die gleiche Länge (in Bytes) hat und füllen Sie gegebenenfalls mit »NOPs« auf. Zur Kontrolle können Sie den geänderten Bereich noch einmal disassemblieren.

Lassen Sie als Beispiel einmal das Programm (siehe Befehl »A«) ab 4000 disassemblieren (»D 4000 4011«). Ändern Sie nun den ersten Befehl auf LDY #01. Die Änderung zeigt sich daran, daß die HEX-Bytes automatisch den neuen Wert annehmen. Starten Sie nun das Programm nochmals mit »G 4000«. Jetzt erscheint der Text mit nur einer Zeile Abstand auf dem Bildschirm.

Starten eines Maschinenprogramms (Go)

G [ADRESSE]

startet ein Maschinenprogramm, das bei ADRESSE beginnt. Das Programm muß mit einem BRK-Befehl abgeschlossen werden, damit ein Rücksprung in SMON erfolgen kann. Wird nach »G« keine Adresse eingegeben, benutzt SMON die, die mit dem letzten BRK erreicht worden ist und bei der Register-Ausgabe als PC auftaucht. Mit dem »R«-Befehl (siehe unten) werden die Register vorher auf gewünschte Werte gesetzt.

Memory-Dump

M [ANFADR ENDADR]

gibt die HEX-Werte des Speichers sowie die zugehörigen ASCII-Zeichen aus. Auch hier kann auf die Eingabe einer Endadresse verzichtet werden. Die Steuerung der Ausgabe entspricht der beim Disassemblieren.

Beispiel:

M 4000 gibt die Inhalte der Speicherstellen $4000 bis $4007 aus. Weiter geht es wie beim Disassemblieren mit SPACE oder RETURN. Die Bytes können ebenfalls durch Überschreiben geändert werden, allerdings nicht die ASCII-Zeichen. Verantwortlich dafür ist der Doppelpunkt, der am Anfang jeder Zeile ausgegeben wird, ein weiterer »hidden command«. Wenn Ihre Änderung nicht durchgeführt werden kann, weil Sie zum Beispiel versuchen, ins ROM zu schreiben, wird ein »?« als Fehlermeldung ausgegeben.

Registeranzeige

R zeigt den gegenwärtigen Stand der wichtigsten 6510-Register an: Programmzähler (PC), Status-Register (SR), Akkumulator (AC), X-Register (XR), Y-Register (YR), Stackpointer (SP). Außerdem werden die einzelnen Flags des Status-Registers mit 1 für »gesetzt« und 0 für »nicht gesetzt« angezeigt. Durch Überschreiben werden die Inhalte auf einen gewünschten Wert gesetzt. Die Flags können allerdings nicht einzeln verändert werden, sondern nur durch Überschreiben des Wertes von SR.

Exit

X springt ins Basic zurück. Alle Basic-Pointer bleiben erhalten. Sie können also zum Beispiel direkt im Programm fortfahren, wenn Sie zwischendurch mit SMON einige Speicherstellen kontrolliert haben.

Probieren Sie alle bisher beschriebenen Befehle in Ruhe aus und machen Sie sich mit SMON vertraut. Arbeiten Sie auch parallel den Kurs über Assemblerprogrammierung in dieser Ausgabe durch. Alle Beispiele dort sind auf SMON abgestimmt.

Wir wollen jetzt einen Blick auf das Programm selbst werfen. Natürlich ist es unmöglich, den gesamten Quelltext umfassend zu beschreiben. Andererseits enthält SMON aber eine Reihe von Routinen, die in jedem Maschinenprogramm vorkommen. Wir werden im Rahmen dieser Serie versuchen, die wichtigsten zu erklären, damit Sie sie später in eigene Programme einbauen können.

Zum besseren Verständnis werden solche Routinen so abgedruckt, wie wir sie im Assembler-Quelltext geschrieben haben. Sie enthalten daher anstelle absoluter Adressen Labels, deren Name — hoffentlich — etwas über den Sinn und Zweck aussagt. Parallel dazu sollten Sie sich diese Routinen von SMON disassemblieren lassen, damit Sie sehen, wie es denn nun fertig im Speicher aussieht.

Beginnen wir mit der Routine GETCHRERR. Das soll soviel bedeuten wie »Hole ein Zeichen und erzeuge eine Fehlermeldung, wenn keins eingegeben wurde«. Leider wäre so ein Label auch für den geduldigsten Assembler viel zu lang, daher die merkwürdige Abkürzung. Mit dieser Routine holen wir ein Zeichen von der Tastatur. Das erledigt die Betriebssystemroutine CHRIN. Um zu prüfen, ob überhaupt etwas eingegeben wurde, untersuchen wir das Zeichen. Handelt es sich um die »RETURN«-Taste ($0D), hat der Benutzer gar kein Zeichen eingegeben. Dies quittiert SMON mit einem »?« und dem Rücksprung in den Eingabemodus. So läßt sich — in gewissen Grenzen — kontrollieren, ob zu einem Befehl die richtigen Eingaben gemacht wurden. Geben Sie einmal den »D«-Befehl ohne Angabe einer Adresse ein, dann sehen Sie, was gemeint ist.

Alle Eingaberoutinen benutzen GETCHRERR, um Falscheingaben zu prüfen. Nehmen wir als Beispiel GETBYT. Diese soll ein Byte, also zwei ASCII-Zeichen 0 - F von der Tastatur holen und in ein Byte umwandeln. Das erste Zeichen wird darauf überprüft, ob es sich um ein »Space« oder ein Komma handelt. Trifft das zu, wird es einfach übergangen und das nächste Zeichen geholt. Der Benutzer kann also Leerzeichen und Komma benutzen, um seine Eingaben übersichtlicher zu machen, er muß aber nicht! Ist das Zeichen aber gültig, wird es von ASCHEX in eine Hexzahl gewandelt.

Dazu ein Beispiel:

Auf der Tastatur wurde 5B eingetippt. Zuerst wird jetzt die 5 (ASCII $35) mit $3A verglichen, um festzustellen, ob es sich um eine Zahl (0 - 9) oder einen Buchstaben (A - F) handelt. ASCII $35 ist eine Zahl, also wird nur die linke Hälfte ausmaskiert (AND #$0F). Ergebnis ist $05. Jetzt wird viermal nach links geschoben und das Ergebnis ($50) in $B4 zwischengespeichert. Nun ist das B (ASCII $42) an der Reihe. Da $42 größer ist als $3A werden diesmal 8 und das gesetzte Carry-Flag, also 9 addiert. Ergebnis ist $5B. Linke Hälfte ausmaskieren wie gehabt und eine OR-Verknüpfung mit dem gemerkten $50 ergibt $5B. Das war's.

Meistens aber braucht SMON zwei Bytes als Eingabe, zum Beispiel für Adressen. Mit dem, was wir schon haben, kein Problem: GETADR ruft einfach GETBYT zweimal hintereinander auf und legt das Ergebnis in zwei Speicherstellen in der Zeropage ab, die mit dem X-Register ausgewählt werden können. Brauchen wir mehr als eine Adreßeingabe, rufen wir einfach GETADR mehrmals auf. So etwas machen GET3ADR und GET2ADR. Bisweilen aber, zum Beispiel beim G-Befehl, darf eine Adresse eingegeben werden, es muß aber nicht sein. Deswegen prüft GETSTART, ob direkt nach dem »G« »RETURN« gedrückt wurde. Dies erledigt GETRET. Wenn ja, wird die Adresse benutzt, die in PCL und PCH steht. Das sind SMONs interne Programm-Counter. Ansonsten wird die eingetippte Adresse benutzt.

Sie sehen, wie aus einfachen Routinen immer kompliziertere Befehle zusammengesetzt werden. Und das ist das ganze Geheimnis, wenn Sie umfangreiche Programme schreiben: Gliedern Sie sich das Problem (hier eine benutzerfreundliche Eingabe) in kleine und kleinste Schritte auf, die Sie dann jeden für sich programmieren und austesten.

Werfen wir noch einen Blick auf die Art und Weise, wie SMON Befehle verarbeitet. In EXECUTE setzen wir zunächst den Stackpointer auf den Wert, den er beim letzten BRK erreicht hatte. Dann werden als erstes die »hidden commands« abgeprüft. Wir lesen dazu direkt vom Bildschirm. D3 enthält die Anfangsadresse der aktuellen Zeile im Speicher. Übrigens gibt es neben den bereits erwähnten noch weitere »hidden commands«, die in den späteren Folgen noch auftauchen werden. Liegt kein verstecktes Kommando vor, holen wir mit GETCHRERR ein Zeichen und merken es uns in COMMAND. Jetzt untersuchen wir, ob dieses Zeichen in der Befehlsliste (CMDTBL) steht. CMDTBL steht übrigens ab $C00B ganz oben im Speicher. Sie endet mit fünf Nullen für spätere Erweiterungen. Direkt dahinter stehen die Anfangsadressen der zugehörigen Routinen in der für den 6502 typischen Reihenfolge, Low-Byte zuerst, dann High-Byte. Sehen Sie sich das mit M C00B einmal an. Am Ende dieser Tabelle stehen nochmals 10 Nullen, denn zu jedem Byte in CMDTBL gehören ja zwei Adreßbytes in der Liste (CMDS). Wenn nun ein Kommando in CMDSEARCH gefunden wurde, wird CMDEXEC als Subroutine aufgerufen. CMDEXEC legt nun die zugehörigen Adreßbytes auf den Stack und führt dann einen RTS aus, der jetzt — nach der Stackmanipulation — zu dem gewünschten Befehl führt. Beachten Sie, daß RTS immer auf die um eins erhöhte Adresse springt, daher müssen Sie zu den Adressen in CMDS immer 1 addieren, wenn Sie den Anfang einer Routine suchen.

Alle Befehle in SMON enden mit einem RTS, springen also auf den JMP EXECUTE hinter CMDFOUND. Damit ist eine Endlosschleife geschlossen, die immer einen Befehl ausführt und anschließend wieder in die Eingabe zurückspringt. Beim nächsten Mal erfahren Sie etwas über LOAD, SAVE und die Umrechnung verschiedener Zahlensysteme.

(Dietrich Weineck/N. Mann/gk)

Hinweise zum Abtippen

Es ist mal wieder eine DATA-Wüste, die wir Ihnen zumuten, aber wenn Sie die erfolgreich hinter sich brachten, haben Sie schon mehr als die Hälfte vom gesamten SMON geschafft. Um Ihnen das Abtippen beziehungsweise die anschließende — fast unvermeidliche — Fehlersuche so einfach wie möglich zu machen, unterteilten wir das Gesamtprogramm in Blöcke zu je 256 Bytes, die jeweils eine eigene Prüfsumme haben. Wenn Sie sich vertippt haben, erscheint eine Fehlermeldung mit Angabe des Blocks, in dem sich der Fehler — höchstwahrscheinlich — befindet.

Vor dem ersten »RUN« sollten Sie aber unbedingt das Programm speichern, sonst kann Ihnen bei Fehlern der Computer abstürzen, und alle Mühe war umsonst.

Eins findet die Prüfsummenmethode allerdings nicht, nämlich zuviel eingegebene Nullen oder Kommas. Erhalten Sie aber keine Fehlermeldung und das Programm läuft trotzdem nicht, kontrollieren Sie als erstes, ob wirklich alle DATAs »aufgebraucht« sind. Dazu tippen Sie im Direktmodus PRINT A ein. Jetzt muß die letzte Zahl, also 197 erscheinen. Wenn nicht, haben Sie eine 0 oder ein Komma zu viel.

Wenn das Ladeprogramm endlich ohne Fehler bis zum READY durchläuft, können Sie SMON mit SYS 49152 starten. Die Bildschirmfarben ändern sich und es erscheint die Registeranzeige und in der nächsten Zeile ein Punkt mit blinkendem Cursor. Probieren Sie jetzt alle Kommandos durch. Hüten Sie sich aber vor allen anderen Kommandos. Die Fehlermeldung bei falschen Kommandos funktioniert noch nicht richtig!! Deshalb führen Falscheingaben in den meisten Fällen zum Programmabsturz. Das wird sich im Verlauf dieser Serie allerdings noch ändern.

Bevor Sie Ihren Computer aus dem Fenster werfen, noch ein Hinweis: Von der nächsten Folge ab wird SMON auch fix und fertig im Leserservice zu erhalten sein.

Und noch ein letzter Tip: Das Wichtigste, was ein angehender Maschinenprogrammierer braucht, ist ein Reset-Taster. (Bauanleitungen oder fertige Taster wurden schon oft im 64'er vorgestellt.) Sie werden es merken, wenn Sie mit sorgenzerfurchter Stirn, den Tränen nahe, vor Ihrem Bildschirm sitzen, kein freundlich blinkender Cursor weit und breit und RUN/STOP RESTORE auch dann nichts mehr bringt, wenn Sie die Tasten durch das Gehäuse durchdrücken. Verzweifeln Sie nicht, drücken Sie RESET, starten Sie SMON neu mit SYS 49152 und schon können Sie bis zum nächsten Absturz weiterarbeiten.....

Listing 1. Der DATA-Lader für SMON — Teil 1

```
10 REM ************************
20 REM *                      *
30 REM *      SMON TEIL 1     *
40 REM * VON N.MANN & D.WEINECK *
50 REM *     FLEETRADE 40     *
60 REM *     2800 BREMEN      *
70 REM *   TEL. 0421 / 493090 *
80 REM *                      *
90 REM ************************
100 FORI=0TO8:READA:PR(I)=A:NEXT
110 SA=49152:I=0
120 PA=SA+256*I:CH=0
130 FORJ=0TO255:READA:POKEPA+J,A:CH=CH+A
:NEXT
140 IFCH<>PR(I)THEN190
150 I=I+1:IFI<8THEN120
160 PA=PA+256:CH=0
170 FORJ=0TO60:READA:POKEPA+J,A:CH=CH+A:
NEXT
180 IFCH=PR(I)THENEND
190 PRINT"FEHLER IN BLOCK"I+1:END
191 REM
192 REM *** BLOCKPRUEFSUMMEN ***
195 DATA20921,25604,31944,33700,36302,34
378,34305,34639,7819
200 REM
210 REM *** BLOCK 1 ***
220 REM
230 DATA169,20,141,22,3,169,194,141,23,3
,0,39,35,36,37,44,58,59,61,63,65,66
240 DATA67,68,70,71,73,75,76,77,79,80,82
,83,84,86,87,88,0,0,0,0,0,218,202,45
250 DATA201,7,201,27,201,251,198,28,196,
181,195,244,202,153,200,208,198,107
260 DATA201,60,202,92,197,16,203,226,195
,67,200,182,202,77,200,248,195,192
270 DATA201,60,200,133,195,77,200,240,20
3,66,202,210,201,109,195,0,0,0,0,0
280 DATA0,0,0,0,0,255,255,1,0,65,90,73,8
2,84,128,32,64,16,0,2,1,1,2,0,145,145
290 DATA13,83,217,49,55,50,13,0,125,76,1
25,201,13,13,32,32,80,67,32,32,83,82
300 DATA32,65,67,32,88,82,32,89,82,32,83
,80,32,32,78,86,45,66,68,73,90,67,0
310 DATA2,4,1,44,0,44,89,41,88,157,31,25
5,28,28,31,31,31,28,223,28,31,223,255
320 DATA255,3,31,128,9,32,12,4,16,1,17,2
0,150,28,25,148,190,108,3,19,1,2,2
330 DATA3,3,2,2,2,2,2,2,3,3,2,3,3,3,2,0,
64,64,128,128,32,16,37,38,33,34,129
340 DATA130,33,130,132,8,8,231,231,231,2
31
350 REM
360 REM *** BLOCK 2 ***
370 REM
380 DATA227,227,227,227,227,227,227,227,
227,227,231,167,231,231,243,243,247
390 DATA223,38,70,6,102,65,129,225,1,160
,162,161,193,33,97,132,134,230,198
400 DATA224,192,36,76,32,144,176,240,48,
208,16,80,112,120,0,24,216,88,184,202
410 DATA136,232,200,234,72,8,104,40,64,9
6,170,168,186,138,154,152,56,248,137
420 DATA156,158,178,42,74,10,106,79,35,1
47,179,243,51,211,19,83,115,82,76,65
430 DATA82,69,83,83,79,76,76,76,67,65,65
,83,83,73,68,67,67,66,74,74,66,66,66
440 DATA66,66,66,66,66,83,66,67,67,67,67
,68,68,73,73,78,80,80,80,80,82,82,84
450 DATA84,84,84,84,84,83,83,79,83,83,79
,79,84,66,82,68,68,68,77,78,68,84,84
460 DATA78,69,80,80,73,77,83,67,67,69,77
,78,80,86,86,69,82,76,76,76,76,69,69
470 DATA78,78,79,72,72,76,76,84,84,65,65
,83,88,88,89,69,69,76,82,76,82,82,65
480 DATA67,65,89,88,65,80,68,67,89,88,67
,67,88,89,84,80,82,67,83,81,73,69,76
490 DATA67,83,73,75,67,68,73,86,88,89,88
,89,80,65,80,65,80,73,83,88,89,88,65
500 REM
510 REM *** BLOCK 3 ***
520 REM
530 DATA83,65,67,68,8,132,129,34,33,38,3
2,128,3,32,28,20,20,16,4,12,216,169
540 DATA8,141,176,2,169,4,141,175,2,169,
6,141,32,208,141,33,208,169,3,141,134
550 DATA2,162,5,104,157,168,2,202,16,249
,173,169,2,208,3,206,168,2,206,169
560 DATA2,186,142,174,2,169,82,76,255,19
4,32,194,194,240,11,32,126,194,141
570 DATA169,2,165,252,141,168,2,96,162,1
64,32,128,194,32,128,194,208,28,32
580 DATA126,194,169,254,133,253,169,255,
133,254,32,194,194,208,12,141,119,2
590 DATA230,198,96,32,126,194,44,162,251
,32,141,194,149,1,32,154,194,149,0
600 DATA232,232,96,32,202,194,201,32,240
,249,201,44,240,245,208,3,32,202,194
610 DATA32,175,194,10,10,10,10,133,180,3
2,202,194,32,175,194,5,180,96,201,58
620 DATA144,2,105,8,41,15,96,32,202,194,
201,32,240,249,198,211,96,32,207,255
630 DATA198,211,201,13,96,32,207,255,201
,13,208,248,169,63,32,210,255,174,174
640 DATA2,154,162,0,134,198,32,81,195,16
1,209,201,39,240,17,201,58,240,13,201
650 DATA59,240,9,201,44,240,5,169,46,32,
210,255,32,202,194,201,46,240,249,133
660 REM
670 REM *** BLOCK 4 ***
680 REM
690 DATA172,41,127,162,32,221,10,192,240
,5,202,208,248,240,194,32,21,195,76
700 DATA214,194,138,10,170,232,189,41,19
2,72,202,189,41,192,72,96,165,252,32
710 DATA42,195,165,251,72,74,74,74,74,32
,53,195,104,41,15,201,10,144,2,105
720 DATA6,105,48,76,210,255,169,13,32,21
0,255,138,76,210,255,32,76,195,169
730 DATA32,76,210,255,169,13,76,210,255,
133,187,132,188,160,0,177,187,240,6
740 DATA32,210,255,200,208,246,96,230,25
1,208,2,230,252,96,169,14,141,134,2
750 DATA141,32,208,169,6,141,33,208,169,
55,133,1,174,174,2,154,76,116,164,160
760 DATA192.169.140,32,86,195,162,59,32,
64,195,173,168,2,133,252,173,169,2
770 DATA133,251,32,35,195,32,76,195,162,
251,189,175,1,32,42,195,32,76,195,232
780 DATA208,244,173,170,2,76,208,195,32,
78,194,162,251,32,202,194,32,154,194
790 DATA157,175,1,232,208,244,32,76,195,
189,170,2,76,208,195,133,170,169,32
800 DATA160,9,32,210,255,6,170,169,48,10
5,0,136,208,244,96,32,73,194,174,174
```

```
810 DATA2,154,162,250,189,174,1,72,232,2
08,249,104,168,104,170,104,64,32,100
820 DATA194,162,58,32,64
830 REM
840 REM *** BLOCK 5 ***
850 REM
860 DATA195,32,35,195,160,32,162,0,32,76
,195,161,251,32,42,195,161,251,32,57
870 DATA196,208,241,32,93,196,144,224,96
,32,126,194,160,32,162,0,32,202,194
880 DATA32,154,194,129,251,193,251,240,3
,76,209,194,32,57,196,208,236,96,201
890 DATA32,144,12,201,96,144,10,201,192,
144,4,201,219,144,4,169,46,41,63,41
900 DATA127,145,209,173,134,2,145,243,32
,103,195,200,192,40,96,32,111,196,76
910 DATA102,196,32,103,195,165,251,197,2
53,165,252,229,254,96,32,148,196,32
920 DATA134,196,240,14,32,134,196,240,25
1,201,32,208,5,141,119,2,230,198,96
930 DATA32,228,255,72,32,225,255,240,2,1
04,96,76,214,194,160,40,36,172,16,246
940 DATA132,200,132,208,169,255,32,195,2
55,169,255,133,184,133,185,173,175
950 DATA2,133,186,32,192,255,162,0,134,2
11,202,32,201,255,32,207,255,32,210
960 DATA255,201,13,208,246,32,204,255,16
9,145,76,210,255,160,0,177,251,36,170
970 DATA48,2,80,12,162,31,221,60,193,240
,47,202,224,21,208,246,162,4,221,73
980 DATA193,240,33,221,77,193,240,30,202
,208,243,162,56,221,17,193,240,20,202
990 DATA224,22,208,246,177,251,61,251
1000 REM
1010 REM *** BLOCK 6 ***
1020 REM
1030 DATA192,93,17,193,240,5,202,208,243
,162,0,134,173,138,240,15,162,17,177
1040 DATA251,61,181,192,93,198,192,240,3
,202,208,243,189,234,192,133,171,189
1050 DATA216,192,133,182,166,173,96,160,
1,177,251,170,200,177,251,160,16,196
1060 DATA171,208,7,32,74,197,160,3,208,2
,164,182,142,174,0,141,175,0,96,160
1070 DATA1,177,251,16,1,136,56,101,251,1
70,232,240,1,136,152,101,252,96,162
1080 DATA0,134,170,32,100,194,32,140,197
,165,173,201,22,240,12,201,47,240,8
1090 DATA201,33,240,4,201,48,208,13,32,8
1,195,162,35,169,45,32,210,255,202,208
1100 DATA250,32,93,196,144,217,96,162,44
,32,64,195,32,35,195,32,76,195,32,117
1110 DATA198,32,203,196,32,76,195,177,25
1,32,42,195,32,76,195,200,196,182,208
1120 DATA243,169,3,56,229,182,170,240,9,
32,73,195,32,76,195,202,208,247,169
1130 DATA32,32,210,255,160,0,166,173,208
,17,162,3,169,42,32,210,255,202,208
1140 DATA248,36,170,48,133,76,106,198,36
,170,80,41,169,8,36,171,240,35,177,251
1150 DATA41,252,133,173,200,177,251,10,1
68,185,60,3,141,174,0,200,185,60,3,141
1160 DATA175,0,32,190,198,164
1170 REM
1180 REM *** BLOCK 7 ***
1190 REM
1200 DATA182,32,147,198,32,203,196,189,9
1,193,32,210,255,189,147,193,32,210
1210 DATA255,189,203,193,32,210,255,169,
32,36,171,240,3,32,73,195,162,32,169
1220 DATA4,36,171,240,2,162,40,138,32,21
0,255,36,171,80,5,169,35,32,210,255
1230 DATA32,44,197,136,240,22,169,8,36,1
71,240,7,169,77,32,210,255,160,1,185
1240 DATA173,0,32,42,195,136,208,247,160
,3,185,172,192,36,171,240,9,185,175
1250 DATA192,190,178,192,32,66,195,136,2
08,237,165,182,32,103,195,56,233,1,208
1260 DATA248,96,164,211,169,32,145,209,2
00,192,40,144,249,96,228,171,208,4,5
1270 DATA173,133,173,96,185,173,0,145,25
1,209,251,208,4,136,16,244,96,104,104
1280 DATA96,208,28,138,5,171,133,171,169
,4,133,181,32,207,255,201,32,240,13
1290 DATA201,36,240,9,201,40,240,5,201,4
4,240,1,96,198,181,208,232,96,224,24
1300 DATA48,14,173,174,0,56,233,2,56,229
,251,141,174,0,160,64,96,32,126,194
1310 DATA133,253,165,252,133,254,32,81,1
95,32,228,198,48,251,16,246,169,0,133
1320 DATA211,32,76,195,32,35,195,32,76,1
95,32,207,255,169,1,133,211,162,128
1330 DATA208,5,162,128,142,177
1340 REM
1350 REM *** BLOCK 8 ***
1360 REM
1370 DATA2,134,170,32,126,194,169,37,133
,200,44,177,2,16,8,162,10,32,207,255
1380 DATA202,208,250,169,0,141,177,2,32,
161,198,201,70,208,22,70,170,104,104
1390 DATA162,2,181,250,72,181,252,149,25
0,104,149,252,202,208,243,76,100,197
1400 DATA201,46,208,17,32,154,194,160,0,
145,251,209,251,208,4,32,103,195,200
1410 DATA136,96,162,253,201,77,208,25,32
,154,194,160,0,201,63,176,239,10,168
1420 DATA165,251,153,60,3,165,252,200,15
3,60,3,32,161,198,149,169,224,253,208
1430 DATA4,169,7,133,183,232,208,240,162
,56,165,166,221,91,193,240,5,202,208
1440 DATA246,202,96,165,167,221,147,193,
208,244,165,168,221,203,193,208,237
1450 DATA189,17,193,133,173,32,161,198,1
60,0,224,32,16,9,201,32,208,8,189,77
1460 DATA193,133,173,76,49,200,160,8,201
,77,240,32,160,64,201,35,240,26,32,157
1470 DATA194,141,174,0,141,175,0,32,161,
198,160,32,201,48,144,27,201,71,176
1480 DATA23,160,128,198,211,32,161,198,3
2,157,194,141,174,0,32,161,198,192,8
1490 DATA240,3,32,190,198,132,171,162,1,
201,88,32,154,198,162,4,201,41,32,154
1500 DATA198,162,2,201,89,32,154,198
1510 REM
1520 REM *** BLOCK 9 ***
1530 REM
1540 DATA165,173,41,13,240,10,162,64,169
,8,32,129,198,169,24,44,169,28,162,130
1550 DATA32,129,198,160,8,165,173,201,32
,240,9,190,3,194,185,11,194,32,129,198
1560 DATA136,208,244,165,171,16,1,200,20
0,32,138,198,198,183,165,183,133,211
1570 DATA76,151,197

READY.
```

Listing 1. Der DATA-Lader für SMON — Teil 1 (Schluß)

Listing 2. Der Assembler-Quelltext von SMON — Teil 1

```
;****************************
;*                          *
;*        S M O N           *
;*                          *
;* MASCHINENSPRACH-MONITOR  *
;*                          *
;*       T E I L  1         *
;*                          *
;*       AUGUST 1984        *
;*                          *
;*  BY N.MANN & D.WEINECK   *
;*    TEL. 0421 / 493090    *
;*                          *
;****************************
;
;
                .BA $C000
                .OS
                .CE
PCL             .DE $FB
PCH             .DE PCL+1
FLAG            .DE $AA
COMMAND         .DE $AC
;
BEFCODE         .DE $AD
ADRCODE         .DE $AB
BEFLEN          .DE $B6
LOPER           .DE $AE
HOPER           .DE LOPER+1
;
PCHSAVE         .DE $02A8
PCLSAVE         .DE PCHSAVE+1
SRSAVE          .DE PCHSAVE+2
AKSAVE          .DE PCHSAVE+3
XRSAVE          .DE PCHSAVE+4
YRSAVE          .DE PCHSAVE+5
SPSAVE          .DE PCHSAVE+6
;
PRINTNR         .DE PCHSAVE+7
IO.NR           .DE PCHSAVE+8
MEM             .DE PCHSAVE+9
;
TASTBUF         .DE $0277
COLOR           .DE $0286
BUF1            .DE $033C
BUF2            .DE BUF1+$30
BUF3            .DE BUF2+$30
BUF4            .DE BUF3+$30
;
READY           .DE $A474
BORDER          .DE $D020
BKGRND          .DE $D021
OPEN            .DE $FFC0
CLOSE           .DE $FFC3
CHKOUT          .DE $FFC9
CLRCHN          .DE $FFCC
CHRIN           .DE $FFCF
CHROUT          .DE $FFD2
STOPT           .DE $FFE1
GETIN           .DE $FFE4
;
;
SETBRK          LDA #L,BREAK
                STA $0316
                LDA #H,BREAK
                STA $0317
                BRK
;
CMDTBL          .BY $27
                .BY '#$%,!;=?A'
                .BY 'BCDFGIKL'
                .BY 'MOPRSTVWX'
                .BY $00 $00 $00 $00 $00
;
CMDS            .SE TICK-1
                .SE BEFDEC-1
                .SE BEFHEX-1
                .SE BEFBIN-1
                .SE KOMMA-1
                .SE COLON-1
                .SE SEMIS-1
                .SE COMP-1
                .SE ADDSUB-1
                .SE ASSEMBLER-1
                .SE BDATA-1
                .SE CONVERT-1
                .SE DISASSEM-1
                .SE FINDE-1
                .SE GO-1
                .SE IO.SET-1
                .SE KONTROLLE-1
                .SE LOADSAVE-1
                .SE MEMDUMP-1
                .SE OCCUPY-1
                .SE SETPRINTER-1
                .SE REGISTER-1
                .SE LOADSAVE-1
                .SE TRACE-1
                .SE VERSCHIEBE-1
                .SE WRITE-1
                .SE EXIT-1
                .DS 10
.
OFFSET          .BY $FF $FF $01 $00
;
FINDTAB         .BY 'AZIRT'
FINDFLG         .BY $80 $20 $40 $10 $00
FINDFLG1        .BY $02 $01 $01 $02 $00
;
SYS172          .BY $91 $91 $0D $53
                .BY  $D9 $31 $37 $32 $0D
;
DATATAB         .BY $00 $7D $4C
                .SE DATALOOP
;
REGHEAD         .BY $0D $0D $20 $20
                .BY 'PC  SR AC XR YR SP'
                .BY '  NV-BDIZC' $00
;
                .BA $C214
;
BREAK           CLD
                LDA #08
                STA IO.NR
                LDA #04
                STA PRINTNR
                LDA #06
                STA BORDER
                STA BKGRND
BREAK1          LDA #03
                STA COLOR
                LDX #$05
BREAK2          PLA
                STA PCHSAVE,X
                DEX
                BPL BREAK2
                LDA PCLSAVE
                BNE BREAK3
                DEC PCHSAVE
BREAK3          DEC PCLSAVE
                TSX
                STX SPSAVE
                LDA #'R'
                JMP CMDSTORE
;
GETSTART        JSR GETRET
                BEQ GETSTRTS
GETSTART1       JSR GETADR1
                STA PCLSAVE
                LDA *PCH
                STA PCHSAVE
GETSTRTS        RTS
;
GET3ADR         LDX #$A4
                JSR GETADR
                JSR GETADR
                BNE GETADR
;
GET1.2ADR       JSR GETADR1
                LDA #$FE
                STA *$FD
                LDA #$FF
                STA *$FE
                JSR GETRET
                BNE GETADR
                STA TASTBUF
                INC *$C6
                RTS
;
GET2ADR         JSR GETADR1
                .BY $2C
GETADR1         LDX #$FB
;
GETADR          JSR GETBYT
                STA *$01,X
                JSR GETBYT1
                STA *$00,X
                INX
                INX
                RTS
;
GETBYT          JSR GETCHRERR
                CMP #$20
                BEQ GETBYT
                CMP #$2C
                BEQ GETBYT
                BNE ASCHEX
;
GETBYT1         JSR GETCHRERR
ASCHEX          JSR ASCHEX1
                ASL A
                ASL A
                ASL A
                ASL A
                STA *$B4
                JSR GETCHRERR
                JSR ASCHEX1
                ORA *$B4
                RTS
ASCHEX1         CMP #$3A
                BCC ASCHEX2
                ADC #$08
ASCHEX2         AND #$0F
                RTS
;
SKIPSPACE       JSR GETCHRERR
                CMP #$20
                BEQ SKIPSPACE
                DEC *$D3
                RTS
;
GETRET          JSR CHRIN
                DEC *$D3
                CMP #$0D
GETBRTS         RTS
;
GETCHRERR       JSR CHRIN
                CMP #$0D
                BNE GETBRTS
ERROR           LDA #'?'
                JSR CHROUT
EXECUTE         LDX SPSAVE
                TXS
                LDX #$00
                STX *$C6
                JSR RETURN
                LDA ($D1,X)
                CMP #''
                BEQ EXEC1
                CMP #':'
                BEQ EXEC1
                CMP #';'
                BEQ EXEC1
                CMP #','
                BEQ EXEC1
                LDA #'.'
                JSR CHROUT
EXEC1           JSR GETCHRERR
                CMP #'.'
                BEQ EXEC1
CMDSTORE        STA *COMMAND
                AND #$7F
                LDX #CMDS-CMDTBL
CMDSEARCH       CMP CMDTBL-1,X
                BEQ CMDFOUND
                DEX
                BNE CMDSEARCH
                BEQ ERROR
CMDFOUND        JSR CMDEXEC
                JMP EXECUTE
;
CMDEXEC         TXA
                ASL A
                TAX
                INX
                LDA CMDS-2,X
                PHA
                DEX
                LDA CMDS-2,X
                PHA
                RTS
;
HEXOUT          LDA *PCH
                JSR HEXOUT1
                LDA *PCL
HEXOUT1         PHA
                LSR A
                LSR A
                LSR A
                LSR A
                JSR HEXOUT2
                PLA
                AND #$0F
HEXOUT2         CMP #$0A
                BCC HEXOUT3
                ADC #$06
HEXOUT3         ADC #$30
                JMP CHROUT
;
CHARRET         LDA #$0D
CHARR1          JSR CHROUT
                TXA
                JMP CHROUT
;
SPACE2          JSR SPACE
SPACE           LDA #$20
                JMP CHROUT
;
RETURN          LDA #$0D
                JMP CHROUT
;
PRINT           STA *$BB
                STY *$BC
                LDY #$00
PRINT1          LDA (*$BB),Y
                BEQ PRINT2
                JSR CHROUT
                INY
                BNE PRINT1
PRINT2          RTS
;
PCINC           INC *PCL
                BNE PCRTS
                INC *PCH
PCRTS           RTS
;
;**********
EXIT            LDA #$0E
                STA COLOR
                STA BORDER
                LDA #$06
                STA BKGRND
                LDA #$37
                STA *$01
                LDX SPSAVE
                TXS
                JMP READY
;
;**********
REGISTER        LDY #H,REGHEAD
                LDA #L,REGHEAD
                JSR PRINT
                LDX #';'
                JSR CHARRET
REGISTER1       LDA PCHSAVE
                STA *PCH
                LDA PCLSAVE
                STA *PCL
                JSR HEXOUT
                JSR SPACE
                LDX #$FB
REGISTER2       LDA SRSAVE-$FB,X
                JSR HEXOUT1
                JSR SPACE
                INX
                BNE REGISTER2
                LDA SRSAVE
                JMP CHANGBIN
;
;**********
SEMIS           JSR GETSTART1
                LDX #$FB
SEMIS1          JSR GETCHRERR
                JSR GETBYT1
                STA SRSAVE-$FB,X
                INX
                BNE SEMIS1
                JSR SPACE
                LDA SRSAVE,X
                JMP CHANGBIN
;
CHANGBIN        STA *FLAG
                LDA #$20
                LDY #$09
CHANGB1         JSR CHROUT
                ASL *FLAG
                LDA #$30
                ADC #$00
                DEY
                BNE CHANGB1
                RTS
;
;**********
GO              JSR GETSTART
GO1             LDX SPSAVE
                TXS
                LDX #$FA
GO2             LDA PCHSAVE-$FA,X
                PHA
                INX
                BNE GO2
                PLA
                TAY
                PLA
                TAX
                PLA
                RTI
;**********
MEMDUMP         JSR GET1.2ADR
MEMDUMP1        LDX #':'
                JSR CHARRET
                JSR HEXOUT
                LDY #32
                LDX #$00
MEMDUMP2        JSR SPACE
                LDA (PCL,X)
                JSR HEXOUT1
                LDA (PCL,X)
                JSR ASCII
                BNE MEMDUMP2
                JSR CONTIN
                BCC MEMDUMP1
                RTS
;**********
COLON           JSR GETADR1
                LDY #32
                LDX #$00
COLON1          JSR GETCHRERR
                JSR GETBYT1
                STA (PCL,X)
                CMP (PCL,X)
                BEQ COLON2
                JMP ERROR
COLON2          JSR ASCII
                BNE COLON1
                RTS
;
ASCII           CMP #$20
                BCC ASCII1
                CMP #$60
                BCC ASCII2
                CMP #$C0
                BCC ASCII1
                CMP #$DB
                BCC ASCII3
ASCII1          LDA #'.'
ASCII2          AND #$3F
ASCII3          AND #$7F
                STA ($D1),Y
                LDA COLOR
                STA ($F3),Y
                JSR PCINC
                INY
                CPY #40
                RTS
;
CONTIN          JSR TASTE
                JMP CMPEND1
;
CMPEND          JSR PCINC
CMPEND1         LDA *PCL
                CMP *$FD
                LDA *PCH
                SBC *$FE
                RTS
;
TASTE           JSR PRINTER1
TASTE1          JSR SCANKEY
                BEQ TASTRTS
TASTE2          JSR SCANKEY
                BEQ TASTE2
                CMP #$20
                BNE TASTRTS
                STA TASTBUF
                INC *$C6
TASTRTS         RTS
;
SCANKEY         JSR GETIN
                PHA
                JSR STOPT
                BEQ STOP
                PLA
SCANRTS         RTS
STOP            JMP EXECUTE
;
PRINTER1        LDY #40
PRINTER         BIT *COMMAND
                BPL SCANRTS
                STY *$C8
                STY *$D0
                LDA #$FF
                JSR CLOSE
                LDA #$FF
                STA *$B8
                STA *$B9
                LDA PRINTNR
                STA *$BA
                JSR OPEN
                LDX #$00
                STX *$D3
                DEX
                JSR CHKOUT
PRLOOP          JSR CHRIN
                JSR CHROUT
                CMP #$0D
                BNE PRLOOP
                JSR CLRCHN
                LDA #$91
                JMP CHROUT
;
;
;
;
.
;****************************
;*                          *
;*        S M O N           *
;*                          *
;* MASCHINENSPRACH-MONITOR  *
;*                          *
```