Georg J. Praml

Variablen-Liste mit AIM und CBM

Wenn umfangreiche Basic-Programme erweitert werden müssen, kann es durch die Wahl neuer Variablen zu Komplikationen kommen: Werden schon einmal benützte Variablennamen irrtümlich erneut verwendet, sind logische Fehler oder falsche Ergebnisse die Folge.

```
OFF2          PASS 1
OFF2          PASS 2
0000 OUT              =$E9BC        ;($FFD2,WRT)
0000 CRLF             =$E9FO        ;($FDDO,CRLF)
0000 BLANK            =$E83E        ;($FDCD,SPACE)
0000 VARB             =$75          ;($2A)
0000                  *=$A9         ;($5E)
00A9 LOC              *=*+2
00AB END              *=*+2
00AD INTF             *=*+1
00AE STRF             *=*+1
00AF VARF             *=*+1
00BO                  *=$112
0112          4C400F  JMP VARLST    ;F3-KEY(AIM/PC ONLY)
0115                  *=$F40        ;(E.G.$5000)
OF40 VARLST
OF40          A001    LDY £1
OF42          84AF    STY VARF
OF44          ;1ST PASS: VARIABLES
OF44          20F0E9  JSR CRLF
OF47          A956    LDA £'V
OF49          20BCE9  JSR OUT       ;PRINT "V:"
OF4C          A93A    LDA £':
OF4E          20BCE9  JSR OUT
OF51          A203    LDX £3        ;TRANSFER 4 LOC'S
OF53 C        20F0E9  JSR CRLF
OF56          A003    LDY £3
OF58 Z1       B575    LDA VARB,X
OF5A          99A900  STA LOC,Y
OF5D          CA      DEX
OF5E          88      DEY
OF5F          10F7    BPL Z1
OF61 B        A5A9    LDA LOC       ;THIS PASS READY?
OF63          C5AB    CMP END
OF65          D024    BNE Z3
OF67          A5AA    LDA LOC+1
OF69          C5AC    CMP END+1
OF6B          D01E    BNE Z3
OF6D          A5AF    LDA VARF      ;YES
OF6F          D004    BNE Z2        ;WERE IT VARIABLES?
OF71          20F0E9  JSR CRLF      ;NO,ARRAYS
OF74          60      RTS           ;COMPLETE
OF75 Z2       C6AF    DEC VARF      ;MAKE ARRAYS NOW
OF77          A205    LDX £5        ;TRANSFER OTHER 4 LOC'S
```

Bild 1. Assembler-Listing des Programms zum Auflisten von verwendeten Variablen, hier in der Adressenbelegung von AIM-65 und PC-100 mit den Angaben für CBM-3001-Rechner in Klammern

Dieses Problem stellte sich bei einer geplanten Programmergänzung für ein Lungenfunktionsmeßgerät, das für arbeitsmedizinische Vorsorgeuntersuchungen routinemäßig verwendet wird. Nach Abschluß der Messung eines Probanden sollten die sonst durch die nächste Messung im Arbeitsspeicher überschriebenen Ergebnisse auf eine Magnetbandkassette gespeichert werden. Das Ergänzungsprogramm erfordert lokale Variable, die aber nicht schon im Hauptprogramm vorkommen dürfen. Das Hauptprogramm hatte in diesem Fall einen Umfang von ca. 16 KByte; das Listing enthielt etwa 500 Zeilen, über 130 einfache und 14 indizierte Variable wurden verwendet.

Bei Programmen in dieser Größenordnung können die verwendeten Variablennamen nicht mehr mit ausreichender Sicherheit dem Listing entnommen werden; von Zufällen sollte man sich nicht abhängig machen (und eine korrekte Zusammenstellung von Variablen wurde noch selten gesehen!). Damit wird es notwendig, den von den Variablen belegten Arbeitsspeicherbereich zu untersuchen und die Variablennamen ausdrucken zu lassen.

Basic-Programme, die von Microsoft-Interpretern abgearbeitet werden, können durch das beschriebene Maschinenprogramm (*Bild 1*) auf die verwendeten Variablen-Namen überprüft werden. Im Text und im Programmlisting angegebene Speicherstellen beziehen sich auf AIM-65/PC-100; auf Unterschiede zum Commodore-Basic weist ein eigener Abschnitt hin (getestet auf CBM 3032).

Format und Speicherung von Variablen

Variablen-Namen im Basic-*Programm* können aus einem bis drei ASCII-Zeichen bestehen: ein Buchstabe (notwendig), ein Buchstabe oder eine Ziffer (nicht notwendig), Typbezeichnung („%" für ganze Zahlen, „$" für Stringvariable, keine für Gleitkommazahlen). Dies gilt sowohl für einfache als auch für indizierte Variable, siehe auch Musterprogramm (*Bild 2*).

Um dem Basic-Interpreter die Suche nach Variablen zu ermöglichen, werden die Namen der Variablen in umgesetzter Form jeweils am Beginn des zur Variablen gehörenden Speicherbereiches in zwei Bytes abgespeichert: 1. Byte = erster Buchstabe (ASCII), 2. Byte = zweites Zeichen (ASCII; Null, falls kein zweites Zeichen vorhanden). Für Gleitkommazahlen gilt dies exakt. Zur Unterscheidung der String-Variablen wird das

höchstwertige Bit im zweiten Byte gesetzt (MSB = 1); Ganzzahl-Variable zeichnen sich durch gesetztes MSB in *beiden* Bytes aus.
Einfache Variable enthalten den Wert in fünf (Gleitkommazahlen) oder zwei (ganze Zahlen) bzw. Informationen über Länge und Anfangsort eines Strings (Stringvariable) in drei Speicherstellen im Anschluß an die zwei Bytes des Variablennamens. Sie verwenden immer insgesamt sieben Bytes, auch wenn diese teilweise unbenützt sein sollten. Die einfachen Variablen folgen also dicht gepackt jeweils im Abstand von sieben Bytes aufeinander – in einer Reihenfolge, wie sie beim Ablauf des Programms angesprochen werden.
Indizierten Variablen (Matrizen oder Arrays) wird implizit oder beim Erkennen einer DIM-Anweisung ein Speicherbereich zugewiesen, der vom Typ der Variablen und von der Art der Indizierung abhängt. Nebenbei eröffnet diese Tatsache die Möglichkeit eventuell erheblicher Speicherplatz-Ersparnis, indem Arrays vom Ganzzahl- statt Gleitkomma-Typ definiert werden, wenn die Rechenoperationen dies erlauben – bei einfachen Variablen würde sich dagegen am benötigten Speicherplatz nichts ändern.
Für die indizierten Variablen ist die Suche nach Variablen-Namen anders geregelt: In den beiden auf den Namen folgenden Bytes steht der Offset zum Beginn der folgenden Variablen.
Die Grenzen der Speicherbereiche von einfachen und indizierten Variablen werden vom Interpreter in der Page Zero in Form von Zeigern festgehalten: $ 75-76 = Beginn der einfachen Variablen (zeigt auf 1. Buchstaben des Namens der 1. Variablen); $ 77-78 = Beginn der indizierten Variablen; $ 79-7A = Ende der indizierten Variablen + 1. In dem durch diese Zeiger limitierten Bereich befinden sich alle vom Programm benützten Variablen-Namen.
Der Speicherbereich für die Variablen schließt direkt an den Programmbereich an. Unmittelbar nach dem Laden (LOAD) eines Programms ist er noch nicht belegt. Erst beim Ablauf des Programms (RUN) wird den Variablen dynamisch Speicherplatz zugewiesen – die Variablen-Liste kann also erst nach Beendigung des Programmablaufes erstellt werden. Falls Programmteile nicht durchlaufen werden sollten – etwa aufgrund von Verzweigungen, erscheinen die in diesem Abschnitt verwendeten Variablen auch nicht im Variablen-Speicherbereich. Eine Ausnahme bilden nur die mit einer expliziten DIM-Anweisung definierten indizierten Variablen.

```
OF79          20F0E9 JSR CRLF
OF7C          20F0E9 JSR CRLF
OF7F          A941   LDA £'A
OF81          20BCE9 JSR OUT      ;PRINT "A:"
OF84          A93A   LDA £':
OF86          20BCE9 JSR OUT
OF89          D0C8   BNE C        ;BR ALWAYS
OF8B Z3       A000   LDY £0       ;INDEX FOR (IND),Y
OF8D          84AD   STY INTF     ;CLR BOTH FLAGS
OF8F          84AE   STY STRF
OF91          A204   LDX £4       ;CHARACTER CNTR
OF93          B1A9   LDA (LOC),Y  ;1ST CHAR OF VAR NAME
OF95          1004   BPL Z4       ;MSB SET?
OF97          E6AD   INC INTF     ;YES, IS INTEGER
OF99          C6AE   DEC STRF
OF9B Z4       297F   AND £$7F     ;STRIP MSB
OF9D          20BCE9 JSR OUT
OFA0          CA     DEX
OFA1          C8     INY
OFA2          B1A9   LDA (LOC),Y  ;2ND CHR OF VAR NAME
OFA4          1006   BPL Z5       ;MSB SET?
OFA6          E6AE   INC STRF     ;IS STRING OR INTEGER
OFA8          1002   BPL Z5       ;BR ALWAYS
OFAA E        90B5   BCC B        ;TRANSIT LANDING
OFAC Z5       297F   AND £$7F     ;STRIP MSB
OFAE          F004   BEQ Z6       ;2ND CHAR NULL?
OFB0          20BCE9 JSR OUT      ;NO,PRINT IT
OFB3          CA     DEX
OFB4 Z6       A5AD   LDA INTF     ;WAS IT INTEGER?
OFB6          F006   BEQ Z7
OFB8          A925   LDA £'%
OFBA          20BCE9 JSR OUT      ;YES,PRINT"%"
OFBD          CA     DEX
OFBE Z7       A5AE   LDA STRF     ;WAS IT STRING?
OFC0          F006   BEQ Z8
OFC2          A924   LDA £'$
OFC4          20BCE9 JSR OUT      ;YES,PRINT"$"
OFC7          CA     DEX
OFC8 Z8       203EE8 JSR BLANK    ;FILL WITH SPACES
OFCB          CA     DEX          ;UNTIL 4 CHAR'S
OFCC          D0FA   BNE Z8
OFCE          18     CLC          ;PREP FOR ADD
OFCF          A5AF   LDA VARF     ;ARE WE IN VARIABLES?
OFD1          F00C   BEQ NXTARY
OFD3 NXTVAR
OFD3          A907   LDA £7       ;YES,ADD £7 TO CURRENT
OFD5          65A9   ADC LOC      ;LOC FOR NEXT VAR
OFD7          85A9   STA LOC
OFD9          9014   BCC D
OFDB          E6AA   INC LOC+1
OFDD          D010   BNE D
OFDF NXTARY
OFDF          C8     INY          ;WE ARE IN ARRAYS
OFE0          B1A9   LDA (LOC),Y  ;ADD OFFSET TO
OFE2          65A9   ADC LOC      ;CURRENT LOC FOR
OFE4          48     PHA          ;NEXT ARRAY
OFE5          C8     INY
OFE6          B1A9   LDA (LOC),Y
OFE8          65AA   ADC LOC+1
OFEA          85AA   STA LOC+1
OFEC          68     PLA
OFED          85A9   STA LOC
OFEF D        18     CLC          ;PREP FOR BRANCH
OFF0          90B8   BCC E        ;BR ALWAYS
OFF2                 .END
OFF2           ERRORS= 0000
```

Arbeitsweise des Programms

Begonnen wird mit der Bearbeitung des Speicherbereiches der einfachen Variablen (VARF = 1). Die Zeiger für die Grenzen dieses Bereiches werden kopiert ($ 75-76 nach $ A9-AA, „LOC"; $ 77-78 nach $ AB-AC, „END") und „V": als Hinweis für die nun folgende Liste der einfachen Variablen ausgedruckt. Ist das Ende des Speicherbereiches noch nicht erreicht, untersucht das Programm die Variablen-Namen (siehe weiter unten). Wurden alle einfachen Variablen bereits bearbeitet (LOC = END), folgen die indizierten Variablen (VARF = 0). Die Zeiger für deren Speicherbereich werden wie vorher kopiert ($ 77-78 nach $ A9-AA, $ 79-7A nach $ AB-AC). Ausdruck von „A:" für die Liste der indizierten Variablen. Die Untersuchung der Variablen-Namen läuft nach dem gleichen Schema ab, bis das Ende des Speicherbereiches der indizierten Variablen erreicht ist (LOC = END); der Ausdruck ist beendet.
Untersuchung des Variablen-Namens: Ist das MSB im ersten Byte des Variablen-Namens gesetzt, handelt es sich um einen Ganzzahl-Typ (INTF = 1). In diesem Fall muß die Information über den String-Typ gesondert behandelt werden, weil auch das MSB des zweiten Bytes gesetzt sein wird (STRF = −1). Abschließend wird das Zeichen (ohne MSB) ausgedruckt. Ist das MSB des zweiten Bytes gesetzt, haben wir entweder eine String- oder eine Ganzzahl-Variable vor uns. Die String-Information wird um 1 erhöht (STRF : −1 ⧫ 0 falls Ganzzahl-Typ, 0 ⧫ 1 falls String-Typ). Das Zeichen wird wieder ohne MSB ausgedruckt; war das zweite Zeichen des Variablen-Namens nicht vorhanden, erfolgt kein Ausdruck (z. B. A, A%, A$).
Der Variablen-Name wird abschließend – wenn notwendig – durch den Ausdruck des Variablen-Typs ergänzt („%" oder „$") und durch Leerzeichen auf insgesamt 4 Zeichen gebracht. Damit ergibt sich von selbst die richtige Formatierung für den Thermodrucker von AIM-65 bzw. PC-100 mit fünf Variablen pro Zeile (*Bild 3*).
LOC zeigt auf die jeweils gerade bearbeitete Variable. Um zur nächsten Variablen zu kommen, muß 7 zum Wert des Zeigers addiert werden (NXTVAR), wenn einfache Variable behandelt werden; bewegt sich das Programm im Bereich der indizierten Variablen, wird auf den Zeiger der Offset addiert, der in den zwei auf den Namen folgenden Bytes zu finden ist (NXTARY).
Das Programm ist verschiebbar geschrieben – es enthält keine absoluten Sprünge außer jenen, die die ohnehin festen Monitor-Routinen ansprechen – und kann deshalb in beliebige Speicherbereiche gelegt werden. Da innerhalb des Programmes Sprünge auftreten, die den Bereich der relativen Adressierung überschreiten, wurde als Hilfslösung ein „Zwischenlandungspunkt" eingeführt (von Label D nach E und weiter nach B). Der eigentlich notwendige, die Verschiebbarkeit verhindernde absolute Sprung (JMP) wurde also durch zwei relative Sprünge ersetzt. An Speicherplatz werden $B2 (dez. 178) Bytes benötigt, dazu die Speicherstellen $A9-AE in der Page Zero (Gleitkomma-Akkumulator). Kollisionen mit der Benützung der Page Zero durch Basic treten nicht auf. Das Programm muß nicht notwendigerweise außerhalb des Basic-Speicherbereiches liegen: Sofern die String-Inhalte nach dem Ausdruck der Variablenliste nicht mehr gebraucht werden – wohl der übliche Fall – darf es auch innerhalb dieses Bereiches lokalisiert sein. Seine Anfangsadresse muß nur oberhalb der einfachen und indizierten Variablen stehen (Adresse des Programmanfangs ≥ Adresse in $ 79-7A bzw. $ 2E-2F bei Commodore).

```
LIST
100 DIM B(5),B%(5),B$(5)
110 DIM K6(5),K6%,K6$(5)
120 DIM YY(5),YY%(5),YY$(5)
130 A=1:A%=1:A$=""
140 Z5=5:Z5%=5:Z5$=""
150 ZZ=9:ZZ%=9:ZZ$=""
160 END
```

Bild 2. Musterprogramm in Basic, das alle prinzipiell möglichen Arten von Variablen-Namen und -Typen enthält

```
V:
A    A%   A$   Z5   Z5%
Z5$  ZZ   ZZ%  ZZ$

A:
B    B%   B$   K6   K6%
K6$  YY   YY%  YY$
```

Bild 3. Ausdruck der vom Musterprogramm in Bild 2 während dessen Ablauf benützten Variablen, erstellt auf einem AIM-65

```
SYS(20480)
V:
A    A%   A$   Z5   Z5%   Z5$   ZZ   ZZ%   ZZ$

A:
B    B%   B$   K6   K6%   K6$   YY   YY%   YY$
READY
```

Bild 4. Ausdruck wie in Bild 3, jedoch hier auf dem Bildschirm eines CBM 3032

Bedienung des Programms

1. Initialisieren von Basic (<5>). Eventuell, aber nicht notwendigerweise, Speicherbereich limitieren (MEMORY SIZE ?).
2. Das Basic-Programm laden, für das die Variablenliste erstellt werden soll. Bild 2 zeigt ein Musterprogramm, das alle möglichen Arten von Variablennamen und -Typen enthält.
3. Programm ablaufen lassen (RUN), um den Variablen-Bereich zu etablieren. Falls erforderlich, ist dabei darauf zu achten, das alle Verzweigungen des Programms durchlaufen werden.
4. Rücksprung zum Monitor (Escape).
5. Laden des Listing-Programmes; die Anfangsadresse des Programmes darf nicht innerhalb des Variablen-Bereiches von Basic zu liegen kommen.
6. Start des Listing-Programmes mit Taste F3 oder den Monitor-Befehlen <*> = F40 (bzw. der gewählten Anfangsadresse) und <G>/. Bild 3 zeigt den Ausdruck des Programms zum Auslisten der Basic-Variablen nach Ablauf des Musterprogrammes von Bild 2.

Abweichungen für CBM-Rechner

Wie schon in Bild 1 in Klammern angedeutet, muß für den CBM 3001 folgendes geändert werden: Zeiger für Grenzen der Speicherbereiche: $ 2A-2B = Beginn