```
15 '**  ---- STRUBS.4.QP ---      **

45050 ' ** MARKEN-TABELLE:
45060 £INIT: MM'AX'=99:DIM MA$(MM),MP%(MM):MP=0
45069 '
45120 '
45130 ' ** LOOP-TABELLE:
45131 ' *LO(...,0)=ZNR.LOOP
45132 ' *LO(...,1)=ZNR. ZUGEHOERIGES ELOOP
45135 L'OOP'M'AX'=140:DIM LO'OP'%(LM,1):L'OOP'P'OINTER'=0
45138 '
45140 ' ** IF-TABELLE:
45145 IM'AX'=270:DIM IX(IM):IP=0
45149 '
45188 '
45189 ' ** STACK:
45190 SM'AX'=60:DIM S'TACK'%(SM):SP'TR'=0
45200 '

READY.
```

Bild 1. Tabellen

Str

Ein Precompiler für Basic-Programme (Teil 4)

In der heutigen letzten Folge wollen wir einige Teile des Programmes Strubs genauer ansehen und untersuchen, wie man das Programm um zusätzliche Funktionen erweitern kann. Dabei werden wir sehen, daß ein solches Übersetzungsprogramm auch für ganz andere Aufgaben eingesetzt werden kann.

In den vorausgehenden Folgen wurde bereits erwähnt, daß die strukturierte Programmierung vor allem Vorteile in bezug auf Wartung, Änderungen und Erweiterbarkeit von Programmen bietet. Dies gilt auch für das Programm Strubs. Um in den Genuß dieser Vorteile zu gelangen, ist allerdings der Zugang zum Quellprogramm erforderlich. Wenn Sie sich das in Heft 5 abgedruckte Objektprogramm ansehen, werden Sie feststellen, daß es auch nicht viel aussagekräftiger als ein unkommentiertes Assemblerlisting ist. Wenn Sie an der Entwicklung eigener Programmerweiterungen interessiert sind, sollten Sie sich deshalb beim Verlag das Quellprogramm besorgen. Da ich hier davon ausgehen muß, daß die meisten Leser das Quellprogramm nicht besitzen, lohnt es sich gar nicht erst, systematisch die einzelnen Programmteile vorzustellen.

Statt dessen wollen wir nur die für Programmerweiterungen wichtigsten Programmelemente vorstellen und anhand einiger exemplarischer Erweiterungen, die auch, ohne sich weitere Gedanken zu machen, einfach eingetippt werden können, aufzeigen, wie man Erweiterungen implementieren kann und was dabei zu beachten ist. Aus dem gleichen Grund geben wir nur die Änderungen an, die im Objektprogramm vorzunehmen sind. Eine Anpassung an das Quellprogramm dürfte keine Probleme bereiten.

Achten Sie bei allen Programmänderungen darauf, daß das geänderte Programm abgespeichert wird, bevor es zum ersten Mal gestartet wird, da das Programm den Zeiger auf das Programmende verstellt. Sollte das Programm durch Erweiterungen so lang werden, daß es in den Editbereich hineinreicht, kann der Anfang des Editbereichs in Schritten zu 256 Byte nach oben verschoben werden, um Platz zu schaffen. Dazu ist in den Zeilen 70 bis 80 die Zahl 40 überall, wo sie auftaucht, durch eine größere Zahl (jeweils 4 für jedes Kilobyte) zu ersetzen (vgl. auch den Schluß der 3. Folge).

Die wichtigsten Programmelemente

Eine grobe Übersicht über den Aufbau des Programms haben wir bereits in der 2. Folge gegeben. Bevor wir uns nun mit einzelnen Erweiterungen beschäftigen, wollen wir zunächst einmal die wichtigsten Programmelemente vorstellen, die man für Änderungen und Erweiterungen des Programms benötigt. Wie bereits erwähnt, liest Strubs das Quellprogramm zweimal vom Anfang bis zum Ende durch. Um Zeit zu sparen, wird im 1. Lauf nur jeweils der Anfang einer Zeile untersucht. Deshalb müssen alle Befehle, die bereits im 1. Lauf zu behandeln sind, auch am Anfang einer Zeile stehen, während Befehle, die nur im 2. Lauf behandelt werden, überall stehen können. Ein Beispiel:

Die Definition von Marken muß am Zeilenanfang erfolgen, während der Aufruf von Marken an jeder Stel-

le erfolgen kann. Die Aufgabe des 1. Laufs besteht darin, verschiedene Tabellen anzulegen, mit deren Hilfe dann im 2. Lauf das endgültige Objektprogramm erzeugt wird.

Jede dieser Tabellen besteht aus einem oder mehreren Array(s), einer Variablen, deren zweiter Buchstabe ein »M« für »Maximal« ist und die Dimension, das heißt die maximale Zahl von Einträgen festlegt, und aus einer Variablen, deren zweiter Buchstabe ein »P« für »Pointer« ist und die auf den jeweils nächsten freien Listenplatz zeigt. Bei Speicherplatzproblemen brauchen nur die Werte der Dimensionsvariablen im Init-Teil geändert zu werden. Möchte man zum Beispiel mehr als 99 Marken (die jetzige Maximalzahl) benutzen, dann schreibt man in Zeile 45060 zum Beispiel »MM=150:...«.

Die Tabellen werden in den Zeilen 45050 bis 45200 definiert (Bild 1). Die Dimension des Stacks bestimmt die mögliche Schachtelungstiefe. Dazu kommen die Tabellen der neuen Befehle (Zeile 45260 bis 45274) und der Fehlermeldungen (Zeile 45480 bis 45514).

nächsten zu lesenden Zeichens. Im 2. Lauf wird zeilenweise das Objektprogramm erzeugt, wobei die jeweils aktuelle Zeile in der Variablen Z$ aufgebaut wird. Dabei enthalten die beiden ersten Zeichen von Z$ Low- und Highbyte der Zeilennummer (so wie sie später im Speicher steht), und das letzte Zeichen der fertigen Zeile besteht aus dem Zeichen CHR$(0).

Die relevanten Zeichencodes, auf die Strubs reagiert, werden in den Zeilen 45240 bis 45254 definiert (Bild 2). Die Variable ZA enthält die Adresse des Anfangs der Zeile, die gerade bearbeitet wird. In EA steht die Startadresse des Editbereichs.

Damit kommen wir zu den für Erweiterungen wichtigen Modulen von Strubs. Die Prozedur »NEXTCHAR« sucht ab Adresse NC das nächste relevante Zeichen des Quellprogrammtextes und liefert dessen Code in der Variablen C. Dabei werden Leerzeichen (Zeile 250) und Kommentare (Zeile 280-295) überlesen. Strings werden direkt in die Ausgabezeile Z$ übertragen (Zeile 350). Der Zeiger NC wird auf das nächste zu lesende Zeichen gesetzt. Die Prozedur »HOLNAME« Zeichen hinter dem Namen (das ist außer beim Blank das Trennzeichen), und NC zeigt auf das nächste Zeichen.

Die Prozedur »SCHREIBZEILE« (Zeile 550-580) generiert auf der Diskette aus den nacheinander eingegebenen Zeilen Z$ das zusammenhängende Objektprogramm und gibt die Nummer der aktuellen Zeile auf dem Bildschirm aus. Die Variable AA (Linkadresse) darf außerhalb dieser Routine nicht verändert werden!

Die Prozedur »ERROR« (Zeile 8050 bis 8099) erwartet als Eingabe einen Fehlercode ER. Dabei handelt es sich um den Index der Fehlermeldung in der Tabelle der Fehlermeldungen. Die Zeilennummer und die Fehlermeldung werden auf dem Bildschirm ausgegeben und zugleich in eine Fehlertabelle eingetragen, die man sich nach der Übersetzung auf Bildschirm oder Drucker ausgeben lassen kann. Zusätzlich wird die Fehlermeldung in die Ausgabezeile Z$ geschrieben, so daß sie auch im Objektprogramm erscheint. Die Übersetzung wird mit der folgenden Zeile fortgesetzt.

Die Prozedur »ABBRUCH« (Zeile 50000 bis 50030) sorgt für einen kontrollierten Abbruch der Übersetzung. Sie erwartet ebenfalls als Ein-

```
45240 ' ** RELEVANTE ZEICHENCODES **
45250 DP=ASC(":"):KO'MMENTAR'=ASC("'"):LA'BEL'=ASC("£"):NU$=CHR$(0):BL=ASC(" ")
45253 BE'FEHL'=ASC("!"):TE'XT("")'=34:G'O'T'O-CODE'$=CHR$(137)
45254 I'F'C'ODE'$=CHR$(139):TH'EN-CODE'=167:NO'T'$=CHR$(168):K'OM'M'A-CODE'=44
```

Bild 2. Relevante Zeichencodes

```
10 REM **** LISTER-DEMO **************
30 PRINT"[illegible]TEST[illegible]"
20 REM WIRD ZU:
30 PRINT"<CD><CR><CU><CL><RON><ROF><HO>TEST<DEL><INS><WHT><RED><GRN><BLU><BLK><P
UR><YEL><CYN>"

READY.
```

Bild 3. Beispiellister

Dem schrittweisen Lesen des Quellprogramms dienen die Variablen C und NC. Die Variable C enthält den Code des jeweils zuletzt gelesenen Zeichens, wobei der Wert 0 ein Zeilenende markiert. Die Variable NC enthält die Adresse des

(Zeile 750-830) liest ab aktueller Adresse NC einen Namen (zum Beispiel Befehl, Label) und zwar bis eines der Trennzeichen »:«, »,«, Blank oder Zeilenende erscheint. Der Name wird in der Variablen T$ ausgegeben, C enthält das erste relevante

gabe den Fehlercode ER und gibt die entsprechende Fehlermeldung aus. Danach wird die Tabelle der bisher bemerkten Fehler ausgegeben, offene Files ordnungsgemäß geschlossen und Strubs neu gestartet.

Die Prozedur »WARTEN« (Zeile 49550 bis 49570) fordert den Benutzer auf, eine Taste zu drücken und wartet auf den Tastendruck.

Die Prozedur »INIT« (Zeile 45050 bis 45999) enthält die Definition der Variablen und Tabellen sowie die Interpretererweiterung.

Im »MENÜ« (Zeile 40050 bis 40495) können die verschiedenen Funktionen angewählt werden.

Die Prozeduren »BEFEHLE IM 1. LAUF« (Zeile 1550-2497) und »BEFEHLE IM 2. LAUF« (Zeile 2550-3640) werden von Strubs aufgerufen, sobald im Quellprogramm das Erkennungszeichen »!« für Befehle (Code in der Variablen BE) entdeckt wird. Sie holen den Namen des Befehls, suchen diesen in der Befehlstabelle und rufen entsprechend dem Index (+1) des Befehls in dieser Tabelle ein Unterprogramm auf. Falls der Befehl nicht in der Tabelle gefunden wird, wird eine entsprechende Fehlermeldung ausgegeben. Im 1. Lauf kommt noch die Ausgabe der Blockstruktur hinzu. Hierzu dient die Variable In (für Indentmodus). IN = 0 bedeutet, auf der gleichen Schachtelungsebene zu bleiben.

Damit haben wir nun das notwendige Wissen zusammen, um an dem Programm Strubs einige Änderungen und Erweiterungen vorzunehmen.

Andere Anwendungen

Bei den Programmtexten, die Strubs übersetzt, handelt es sich zwar um erweiterte Basicprogramme, aber nichtsdestoweniger um Basicprogramme. Deshalb ist es relativ einfach, Strubs auch zur Bearbeitung ganz normaler Basic-Programme einzusetzen. Zwei sinnvolle Möglichkeiten wollen wir im folgenden vorstellen.

1. Ein SPEED-UP-Programm, um normale Basicprogramme schneller zu machen.
2. Ein Programm, das besser lesbare Listings erstellt.

Dabei ist zu beachten, daß die Änderungen, die wir dazu vornehmen, nicht wie die Makro-Funktion eine Erweiterung des eigentlichen Programmes Strubs und seiner Funktion darstellen, sondern daß wir zwei völlig neue Programme mit völlig neuen Aufgaben erhalten. Deshalb sollten auch die erhaltenen Programme unter neuen Namen, beispielsweise »SPEED-UP« und »LISTER«, abgespeichert werden. Das Arbeiten mit diesen Programmen unterscheidet sich nicht von der Arbeit mit dem »normalen« Strubs-Programm.

```
10 REM ***** MAKROS BEISPIELE ********
20 !DMAKRO:STOPAN POKE 788,49
30 !DMAKRO:STOPAUS POKE 788,52
40 !DMAKRO:READJOY JS=PEEK(56320)
50 !DMAKRO:WARTE POKE 198,0:WAIT 198,1
60 !DMAKRO:CLOSEALL SYS 65511
70 !DMAKRO:SPRITEXPOS POKE 53248+2*
80 !DMAKRO:SPRITEYPOS POKE 53249+2*
90 !DMAKRO:SPRITEAN POKE 53269,PEEK(53269) OR 2^
95 '
96 '  ******** AUFRUFE:  **********
100 !M,STOPAUS:!M,CLOSEALL:!M,STOPAN
110 !M,READJOY:PRINT JS
120 !M,SPRITEAN 5:!M,WARTE
130 !M,SPRITEXPOS 5,240:!M,SPRITEYPOS 5,170
140 !M,FEHLER

READY.

10 REM*****MAKROSBEISPIELE********
100 POKE788,52:SYS65511:POKE788,49
110 JS=PEEK(56320):PRINTJS
120 POKE53269,PEEK(53269)OR215:POKE198,0:WAIT198,1
130 POKE53248+2*5,240:POKE53249+2*5,170
140 ***** ERR:UNDEFINIERTES MAKRO********

READY.
```

Bild 4. Beispiele Makros

Schnelleres Basic

Zunächst wollen wir Strubs so ändern, daß es normale Basicprogramme in Programme übersetzt, die keine Leerzeichen und Kommentare mehr enthalten und dadurch schneller ablaufen. Wie Sie sich erinnern werden, benutzt Strubs für Kommentare, die gelöscht werden sollen, ein eigenes Zeichen »'«. Kommentare, die mit REM gekennzeichnet werden, bleiben im Objektprogramm erhalten. Da Strubs bereits alle Blanks entfernt (außer in Strings), brauchen wir nur noch dafür zu sorgen, daß Strubs auf das REM-Token reagiert wie bisher auf das Kommentarzeichen »'«. Die relevanten Zeichencodes, auf die Strubs reagiert, werden in den Zeilen 45250 bis 45254 definiert (Bild 2). Wir brauchen nur in Zeile 45250 das KO = ASC(»'«) durch KO = 143 (143 ist das REM-Token) ersetzen und schon ist das Speed-Up-Programm fertig. Genauso können Sie die Erkennungszeichen für Label und die neuen Befehle ändern. Dies ist, um Kon-

```
0 ********* SPEED-UP ****************
45250 DP=ASC(":"):KO=143:LA=ASC("£"):NU$=CHR$(0):BL=ASC(" ")

READY.

0 ****** LISTER  ********************
350 Z$=Z$+C$(C):NC=NC+1:C=PEEK(NC):IF C AND C<>TE THEN 350
45300 DIM C$(255):FOR I=0 TO 255:C$(I)=CHR$(I):NEXT
45310 C$(17)="<CDO>":C$(29)="<CR>":C$(145)="<CU>":C$(157)="<CL>"
45312 C$(18)="<RON>":C$(146)="<ROF>":C$(19)="<HO>":C$(147)="<CLR>"
45314 C$(20)="<DEL>":C$(148)="<INS>"
45316 C$(5)="<WHT>":C$(28)="<RED>":C$(30)="<GRN>":C$(31)="<BLU>"
45318 C$(144)="<BLK>":C$(156)="<PUR>":C$(158)="<YEL>":C$(159)="<CYN>"

READY.

0 ********** MAKROS ****************
1571 IF I>14 THEN ON I-14 GOSUB 2350,2360
2350 IF NP>NM THEN ER=10:GOTO 50000
2355 Z$="":GOSUB 750:NA$(NP,0)=T$
2360 Z$=Z$+CHR$(C):GOSUB 250:IF C<>0 THEN 2360
2370 NA$(NP,1)=Z$:NP=NP+1:IN=0
2375 RETURN
2380 IN=0:RETURN
2571 IF I>14 THEN ON I-14 GOSUB 3700,3750
3700 Z$="":C=0:RETURN
3750 GOSUB 750
3755 FOR I=0 TO NP:IF NA$(I,0)<>T$ THEN NEXT
3760 IF I>=NP THEN ER=11: GOTO 8050
3765 Z$=Z$+NA$(I,1):RETURN
45155 NM=40:DIM NA$(NM,1):NP=0
45265 BM=15:DIM BE$(BM)
45275 DATA DMAKRO,M
45480 EM=11:DIM ERZ(30,1):EP=0:DIM ER$(EM)
45500 FOR I=0 TO EM:READ ER$(I):NEXT
45515 DATA "ZU VIELE MAKROS","UNDEFINIERTES MAKRO"
```

Bild 5. Die besprochenen Erweiterungen auf einen Blick

flikte zu vermeiden, für den Fall sinnvoll, daß Sie mit Strubs Programme für Interpretererweiterungen übersetzen, die ihrerseits »!« oder das Pfundzeichen als Erkennungszeichen für ihre neuen Befehle benutzen.

Listings

Wollen Sie im »64'er« eigene Programme veröffentlichen? Dann können Sie den Lesern viel Ärger ersparen, wenn Sie das Listing vorher mit dem Programm »LISTER« aufbereiten. »LISTER« übersetzt Basic-Programme in Programmtexte, in denen die schwer entzifferbaren Steuer- und Grafikzeichen innerhalb von Strings durch lesbare Worte »⟨CDOWN⟩« oder »⟨HOME⟩« ersetzt sind (Bild 3).

Dazu ändern wir eine Zeile innerhalb der Prozedur »NEXTCHAR«. In Zeile 350 werden gelesene Zeichen mit dem ASCII-Code C innerhalb von Strings direkt in die Ausgabezeile Z$ übertragen. Wenn wir nun in Zeile 350 Z$=Z$+CHR$(C) durch Z$=Z$+C$(C) ersetzen, dann können wir ein Array C$(255) definieren, das in jedem ASCII-Wert den String enthält, der dafür im Objektprogramm erscheinen soll. Die Definition dieses Arrays gehört in das Modul »INITIALISIERUNG«:

45300 DIM C$(255):FOR I=0 TO 255:C$(I)=CHR$(I):NEXT

Damit haben wir zugleich unser Array mit den normalen Werten vorbesetzt. Jetzt bleiben nur noch die Ersetzungen:

45310
C$(17)="⟨CDOWN⟩":C$(19)="⟨HOME⟩"
45312
C$(28)="⟨ROT⟩":C$(31)="⟨BLAU⟩"
... usw.

Hier können Sie nun jedem Zeichen ein beliebiges Wort zuordnen: Den ASCII-Code der einzelnen Zeichen finden Sie im C-64 Handbuch auf S. 135 oder Sie können ihn einfach durch Eingabe von

PRINT ASC("X")

feststellen, wobei »X« für das interessierende Zeichen steht. Bei sehr vielen Zeichen innerhalb eines Strings kann es allerdings vorkommen, daß die Zeilen zu lang werden. Deshalb sollten die Worte möglichst kurz gewählt werden.

Makros

An einem etwas umfangreicheren Beispiel wollen wir nun zeigen, wie man neue Strubs-Befehle implementiert und wie man die Prozeduren von Strubs benutzen kann. Dies soll am Beispiel einer Makro-Funktion demonstriert werden.

Makros, vor allem von Assemblern her bekannt, stellen so etwas wie Abkürzungen für kurze Programmausschnitte dar. Dadurch verringert sich die Tipparbeit und vor allem werden die Quellprogramme übersichtlicher.

In der Makro-Definition wird ein Makro-Name definiert und diesem ein Programmstück zugeordnet. Überall, wo nun im Quellprogramm ein Makro aufgerufen wird, erscheint im Objektprogramm an dieser Stelle das entsprechende Programmstück. Ein einmal definiertes Makro kann wie ein Label beliebig oft aufgerufen werden.

Für die Definition eines Makros wollen wir den Befehl »!DMAKRO« und für den Aufruf eines Makros den Befehl »!M« wählen. Ein Beispiel mag die Wirkungsweise der neuen Befehle demonstrieren:

10 !DMAKRO:NAME SYS 833:X=PEEK(878)
...
200 PRINT X:!M,NAME:PRINT X

Die Definitionszeile 10 wird gelöscht, da sie nur für die Übersetzung notwendige Informationen enthält. Die Zeile 200 mit dem Makro-Aufruf sieht im Objektprogramm folgendermaßen aus:

200 PRINTX:SYS833:X=PEEK(878):PRINTX

Einige Beispiele für Makros und deren korrekte Benutzung sowie das sich ergebende Objektprogramm zeigt Bild 4. Vor allem ist darauf zu achten, daß Makronamen wie alle Befehls- und Labelnamen mit einem der Trennzeichen abgeschlossen werden müssen. Insbesondere darf bei der Makrodefinition und beim Aufruf mit nachfolgenden Parametern (Spritemakros in Zeile 120 und 130) nicht das Blank hinter dem Makronamen vergessen werden! Jede Makrodefinition benötigt eine eigene Zeile. Eine Übergabe von

Parametern an ein Makro ist nicht möglich. Achten Sie bei der Arbeit mit Makros darauf, daß die entstehenden Zeilen des Objektprogramms nicht zu lang werden. Zeilen, die länger als 80 Zeichen sind, lassen sich nicht mehr editieren. Zeilen, die länger als 256 Zeichen werden, führen zum unkontrollierten Abbruch der Übersetzung mit »String too long error«. In diesem Fall kann man mit »GOTO 50000« die Nummer der verantwortlichen Zeile erfahren und offene Files schließen.

Um die Übersetzung zu ermöglichen, muß im 1. Lauf eine Tabelle der Makronamen und der zugehörigen Programmausschnitte angelegt werden. Im 2. Lauf werden dann alle Aufrufe durch den zugehörigen Text ersetzt. Die Verteilung auf zwei Läufe bietet den Vorteil, daß ein Makro (ebenso wie Labels) auch schon vor der Definition aufgerufen werden kann.

Zur Implementation sind folgende Schritte erforderlich: Zunächst muß dem Übersetzungsprogramm mitgeteilt werden, daß es zwei neue Befehle gibt. Dann müssen wir die notwendige Tabelle definieren und auch entsprechende Fehlermeldungen vorsehen. Diese Erweiterungen gehören in den INIT-Teil.

Schließlich muß noch dafür gesorgt werden, daß Strubs weiß, wie es im 1. und 2. Lauf auf die neuen Befehle zu reagieren hat.

Die Befehlstabelle wird in Zeile 45265 definiert. Hier erhöhen wir die Zahl der Befehle um 2 und fügen dann noch eine DATA-Zeile mit den beiden neuen Befehlsnamen ein:

45265 BM=15:...
45275 DATA DMAKRO,M

Wählt man Befehlsnamen, die reservierte Basic-Worte enthalten, dann müssen die Tokens berücksichtigt werden (wie dies für IF in der Zeile 45271 geschieht). Für einen Befehl »DEFMAKRO« wäre zum Beispiel

BE$(14)=CHR$(150)+"MAKRO" zu setzen (150=DEF-Token).

Für die Tabelle wählen wir ein Array NA$(NM,1), da der Name M bereits für die Markentabelle vergeben ist. Die Dimension (..,0) soll die Namen und die Dimension (..,1) den zugehörigen Text aufnehmen.

45155 NM=40:DIM NA$(NM,1): NP=0

```
-------------------------------------------------------------------------------
CROSS REFERENCE MAP                        STRUBS.4.OP                    PAGE
-------------------------------------------------------------------------------

AA          560*   565    570   5120*  5130
AD()         555   1125   1574   1605   2100   2260   5052   5920   6100   6550
6655   8050   8060  45410  50008
B$          1565*  1575   1577   1579   1581   2685*  2693  48060* 48070  49060*
49070  49560*
BE          3470  45253*
BE$()       1550  45641  45265  45270* 45271*
BL           250    295    795    820  45250*
BM          1565  45641   2565  45265* 45270
C            260*   265    280*   290*   295    350*   795*   800    820   2420
2425   2470   2590   3010   3036*  3400*  3470   3495*  3610   4100
4108   4115   4130   4360   4380   5580*  5585   8090*
C$          8090*
DI          1170   1685   2100   2260   2640   2693   3030   3090   3490   3630
6100   8060  45220* 48140  49140
DP           795   5585  45250*
E           5090*  5095
E$          5090*  5095
EA            80*  5052   5120   5555   6550   8950
EM          8060  45480*
EP          5180   8060* 45480* 49050  49110  49120
ER          1160*  1565*  1600*  1605*  1640*  2010*  2040*  2100*  2160*  2275*
2410*  2425*  2565*  5143*  6050*  8050   8060   8080  50008
ER$()       8050   8080  45480  45500* 49140  50008
ER%()       8060* 45480  49140
F$          5070*  5080
GT$         2640   2685   3090  45253*
H           1125*  1170*  1180   2260*  2300   2425*  2647*  2648   3100*  3130
3190*  3200  48055* 48102* 48150  49055* 49102* 49150
H%           565*   570
I           1140*  1160   1170   1550   1565   1569*  1570   2275*  2290   2300
45641   2565   2570  45270* 45500* 45600* 45610* 45650* 48120* 48140
48150  49120* 49140  49150
I%()        2100*  2300*  3030   3090  45145
IC$         3010   3600  45254* 45271
IM         45145*
IN          1575   1577   1579   1581   1615*  1660*  1680*  2025*  2052*  2107*
2240*  2320*  2485*  3620*
IP          2010*  2050*  3030   3036*  3090   3140*  6560* 45145*
KM           795  45254*
KO           265    280   6585  45250*
L           2647*  2648   3100*  3130   3190*  3200
LA          4100   4360  45250*
LM          1605  45135*
LO%()       1605*  2640   2693   3490   3630  45135
LP          1605*  2595*  6560* 45135*
MA$()       1140   6100* 45060  48140
MA%()       1170   6100* 45060  48140
MM          2410   6050  45060*
MP          1140   1160   2410   2460*  6050   6100* 45060* 48050  48120
NC           250*   260    280*   290*   350*   370*   795    800*   820*  4080*
4115*  5580*
NO$         3010   3600  45254*
NU$         2640   3090   4115   8090  45250*
S%()        2050*  2100   2275   2595*  2640   2693   3490   3630  45190
SM          1600   2010   2160  45190*
SP          1600   1605*  1640*  2010*  2040   2050   2100   2160*  2275   2595*
2630*  2640   2693   3490   3620*  3630   5143   6560* 45190*
T$           750*   800*  1120   1140   1550  45641   6100
TA          1575   1577*  1579   1581*  5136*
TE           350  45253*
TH          3030   3480   3630  45254*
W          45600* 45610* 45650*
X          45410
Z$           350*   550    560    570   1180*  2590*  2640*  2648*  2685   2693*
3010*  3030*  3090*  3130*  3200*  3400*  3470*  3480*  3490*  3600*
3610*  3630*  4060*  4108*  4115*  4380*  5096*  5099   8080* 40160*
40170  40180  40195
Z1          6550*  6655*  6660
ZA           555   1125   1574   1605   2100   2260   2647   3100   3190   4060
4080   5555*  5570   5580   5920*  6100   6550*  6585   6655*  8050
8060  50008
```

Bild 6. Variablenliste

Damit können 41 Makros definiert werden. Indem wir die Zahl der Fehlermeldungen von 9 auf 11 erhöhen, erhalten wir die beiden neuen Fehlercodes 10 und 11 für »zu viele Makros« und »undefiniertes Makro«.

```
45480 EM=11:DIM...
45500 FOR I=0 TO EM:READ ...
45515 DATA "ZU VIELE MAKROS",
"UNDEFINIERTES MAKRO"
```

Nun müssen wir in die beiden Module »BEFEHLE IM 1. LAUF« beziehungsweise »BEFEHLE IM 2. LAUF« jeweils zwei Routinen für die neuen Befehle einfügen. Da die beiden Verteilerzeilen bereits voll sind, legen wir zwei neue Verteilerzeilen an, die dann aber auch gleich für 10 weitere neue Befehle Platz bieten:

```
1571 IF I>14 THEN ON I-14 GOSUB
2350,2380
```

für den 1. Lauf und

```
2571 IF I>14 THEN ON I-14 GOSUB
3700,3750
```

für den 2. Lauf.

Die Routine für »!DMAKRO« im 1. Lauf soll zunächst prüfen, ob noch Platz in unserer Makro-Tabelle ist und, falls nicht, mit entsprechendem Fehlercode die Abbruch-Routine anspringen:

```
2350 IF NP>NM THEN ER=10:
GOTO 50000
```

Jetzt können wir mit Hilfe der Prozedur »HOLNAME« den Makro-Namen lesen und in unserer Tabelle speichern:

```
2355  Z$=" ":GOSUB750:NA$(NP,0)
=T$
```

Nun übertragen wir den Rest der Definitionszeile mit Hilfe von »NEXTCHAR« nach Z$ (dadurch werden auch Strings mit übertragen. Als Ausgabezeile dient Z$ ja erst im 2. Lauf).

```
2360      Z$=Z$+CHR$(C):GOSUB
250:IF C<>0 THEN 2360
```

Nun brauchen wir nur noch den Text in die Tabelle aufzunehmen, den Zeiger zu erhöhen und den Indentmodus angeben.

```
2370 NA$(NP,1)=Z$:NP=NP+1:IN=
0
2375 RETURN
```

Der Aufruf eines Makros interessiert im 1. Lauf nicht, also:

```
2380 IN=0:RETURN
```

Im 2. Lauf soll die Definitionszeile gelöscht werden. Dazu löschen wir den Ausgabestring und weisen C den Code für Zeilenende zu:

```
3700 Z$=" ":C=0:RETURN
```

Beim Aufruf eines Makros mit »!M« holen wir zunächst den Namen des Makros mit »HOLNAME« und suchen ihn in der Tabelle:

```
3750 GOSUB 750
3755 FOR I=0 TO NP: IF NA$(I,0)
<>T$ THEN NEXT
```

Falls der Name nicht gefunden wird, erfolgt ein Sprung zur Error-Routine mit dem Code für »undefiniertes Makro«:

```
3760 IF I>NP THEN ER=11: GOTO
8050
```

Nun ist nur noch das definierte Programmstück in die Ausgabezeile zu übertragen:

```
3760 Z$=Z$+NA$(I,1):RETURN
```

Dadurch, daß diese Makro-Erweiterung Zeile für Zeile besprochen wurde, um zu zeigen, wie man die von Strubs vorgegebenen Prozeduren benutzen kann, ist vielleicht der Eindruck entstanden, eine solche Erweiterung sei relativ kompliziert. Wenn Sie sich aber das Ganze noch einmal genauer ansehen, können Sie feststellen, daß für die Implementation neuer Befehle im Prinzip nur drei Schritte erforderlich sind:

1. Eintrag der neuen Befehlsnamen in die Befehlstabelle
2. Einfügen der entsprechenden Routinen
3. Eintrag der Adressen dieser Routinen in die beiden Verteilerzeilen

Die ganze Arbeit des Suchens und Decodierens übernimmt Strubs automatisch.

Wie neue Funktionen (beispielsweise die Ausgabe der Makro-Tabelle) in das Menü aufgenommen werden können, haben sie bereits in der letzten Folge am Beispiel der RENUMBER-Funktion gesehen.

Eine Zusammenstellung der oben besprochenen Erweiterungen finden Sie in Bild 5.

Strubs und Interpretererweiterungen

Wollen Sie mit Strubs Programme für Interpretererweiterungen bearbeiten, dann sind einige weitere Dinge zu beachten. Entfernen Sie zunächst wie in Folge 3 beschrieben die Interpretererweiterung von Strubs.

Falls die Erweiterung, die Sie benutzen wollen, nicht in den Editor eingreift, sondern ihre neuen Befehle durch besondere Zeichen (meistens »!«) gekennzeichnet werden, dann ändern Sie wie bereits oben beschrieben die entsprechenden Erkennungszeichen, die Strubs benutzt.

Bei Erweiterungen wie Simon's Basic, die in den Editor eingreifen und die neuen Befehle wie der Basic-Interpreter durch eigene Tokens darstellen, ist es am einfachsten, den Strubs-Befehlen, deren Namen solche Befehle enthalten, neue Namen zu geben. Im Fall von Simon's Basic sind davon beispielsweise Strubs-Befehle wie »!REPEAT«, »!UNTIL« oder »!ELSE« etc. betroffen.

Dazu sind nur die Namen in den DATA-Zeilen 45272 bis 45274 zu ändern. Sie können die betroffenen Strubs-Befehle aber auch wie oben am Beispiel von »DEFMAKRO« beschrieben aus den Tokens zusammensetzen. Dabei ist aber zu berücksichtigen, daß die Tokens von Simon's Basic aus zwei Zeichen und nicht wie die normalen Tokens aus nur einem Zeichen bestehen.

Eine Liste der von Strubs benutzten Variablen bietet Bild 6. Dabei kennzeichnet das Zeichen »*« Zeilennummern, in denen eine Wertzuweisung an die Variable erfolgt.

Zum Abschluß noch einige Berichtigungen zu den ersten Folgen: Die abgedruckte Programmversion wird nicht wie behauptet mit einem Kaltstart, sondern mit normalem »END« (Zeile 40190) beendet. Bei den in Teil 2 auf S. 121 angeführten Beispielen für Markendefinition und Externdeklaration haben sich Fehler eingeschlichen. Die korrekte Form entnehmen Sie bitte Bild 7.

(Matthias Törk)

```
10 £X-AUSGEBEN:
20 PRINT "X:";X
30 RETURN

READY.

20 REM VEREINBARUNG:
30 !EXT:£MAPRO:740,£PLOT:50000

READY.
```

Bild 7. Berichtigung zur Folge 2, Seite 121