Strubs

Nun ist es endlich so weit! Wir gehen an die praktische Entwicklung von Programmen heran. Außerdem wird Strubs noch um ein kleines RENUMBER-Programm erweitert. Die Syntaxdiagramme der Basic-Erweiterungen werden das Arbeiten mit Strubs erleichtern.

Wenn Sie Quellprogramme schreiben, achten Sie vor allem auf folgendes: Jeder der neuen Befehle muß durch ein Ausrufezeichen gekennzeichnet werden, zum Beispiel !REPEAT, und benötigt — außer !DO — eine eigene Programmzeile. Folgende Konstruktion ist also zum Beispiel nicht erlaubt:

40 !REPEAT X=X+1 !UNTIL X>5

Marken beginnen grundsätzlich mit einem Pfund-Zeichen £.

Um die Korrektheit von Konstruktionen zu überprüfen, können Sie die Syntax-Diagramme in Bild 1 benutzen. Wenn sich ein Weg entlang der Linien finden läßt, der der Konstruktion entspricht, dann ist diese in Ordnung. Sie können sich aber auch an den verschiedenen Beispielen im Rahmen dieser Artikelfolge orientieren. Das beste Beispiel für ein Quellprogramm erhalten Sie, wenn Sie die Programmdiskette mit dem Programm Strubs bestellen (wird allerdings erst ab der nächsten Ausgabe angeboten). Diese Diskette enthält neben dem lauffähigen Objektprogramm, das in der letzten Ausgabe abgedruckt wurde, auch das ausführlich dokumentierte Quellprogramm von Strubs, dessen Abdruck aus Platzgründen nicht möglich ist.

Bei der Blockschachtelung ist darauf zu achten, daß sich verschiedene Blöcke nicht überschneiden dürfen und daß jeder Block korrekt abgeschlossen wird. Hierbei kann man sich immer das Beispiel der FOR-NEXT-Schleifen in Basic vor Augen halten. Vollkommen unmöglich ist beispielsweise folgende Konstruktion:

10 !REPEAT
20 : !WHILE ... !DO
30 !UNTIL ...
40 : !EWHILE

Nun wird es aber allmählich Zeit, mit der Praxis zu beginnen. Laden Sie das Programm Strubs in Ihren Computer und starten es mit »RUN«. Nun erscheint ein Menü. Geben Sie hier »E« ein, um in den Editbereich zu gelangen (siehe dazu die erste Folge). Der Computer meldet sich mit »READY«, das heißt also, Sie befinden sich jetzt im Direktmodus. Hier können Sie nun (fast) so arbeiten, als sei Strubs gar nicht vorhanden. Geben Sie zunächst »NEW« ein. Jetzt können Sie das kleine Programm aus Bild 2 eintippen und wie sonst gewohnt mit »SAVE "RENUMBER.QP",8« abspeichern.

Mit diesem »QP« hat es folgende Bewandnis: Bei Compilern ist es allgemein üblich, die verschiedenen Files, die zu den einzelnen Phasen der Übersetzung gehören, einheitlich zu kennzeichnen, der Austro-Compiler, arbeitet zum Beispiel mit den Files »name«, »p/name«, »z/name« und »c/name«. Um Quellprogramme und Objektprogramme auseinanderhalten zu können, sollten Sie sich entsprechend von Anfang an daran gewöhnen, dem Programme immer ein »QP« für Quellprogramm beziehungsweise ein »OP« für Objektprogramm hinzuzufügen. Nun kann das Programm übersetzt werden. Geben Sie ein »!RETURN« und es erscheint das Menü von Strubs. Die Übersetzung wird mit »U« gewählt. Strubs fragt nun nach dem Namen für das Objektprogramm. Geben Sie ein: RENUMBER.OP. Da das übersetzte Programm direkt auf Diskette geschrieben wird, achten Sie darauf, daß die Floppy eingeschaltet ist. Nun erscheint auf dem Schirm die Meldung »1. Lauf«, gefolgt von der Ausgabe der Blockstruktur. Nach Beendigung des 2. Laufs sollte die Meldung »0 FEHLER« erscheinen. Ist dies der Fall, dann können sie mit »E« wieder in den Edit-Bereich gelangen. Hier steht immer noch das Quellprogramm. Um sich das übersetzte Programm anzusehen, laden Sie es mit »LOAD "RENUMBER.OP",8«. Es sollte mit dem Listing in Bild 2 übereinstimmen. Aber starten sie das Programm nicht.

Jetzt übersetzen Sie einmal genauso verschiedene kleine Testprogramme — zum Beispiel die Beispiele aus der letzten Folge — und sehen sich die Ergebnisse an. Dabei werden Sie feststellen, daß einige Bedingungen im Objektprogramm in negierter Form erscheinen. Daß Basic keine boolschen Variablen kennt, hat eine wichtige Konsequenz: Beim Test, ob eine Variable ungleich 0 ist, darf man nicht — wie dies normalerweise häufig in Basic formuliert wird — beispielsweise schreiben

IF A THEN ...,

sondern muß bei jeder Bedingung die vollständige Form

IF A<>0 THEN ...

verwenden. Dies liegt daran, daß die Bedingung NOT(A) außer für —1 immer erfüllt ist.

Zweitens kann man sehen, daß in den Objektprogrammen manchmal neue Zeilen auftauchen, die das Quellprogramm nicht enthielt. Strubs generiert solche Zeilen als Sprungziele. Damit immer Platz für solche Zeilen ist, sollte der Abstand der Zeilennummern im Quellprogramm immer mindestens 2 betragen.

Ist bisher alles wie oben beschrieben verlaufen, dann können Sie sich freuen. Sind irgendwelche Fehler aufgetreten, dann vergleichen Sie noch einmal genau das Testpro-

ein Precompiler für Basicprogramme (Teil 3)

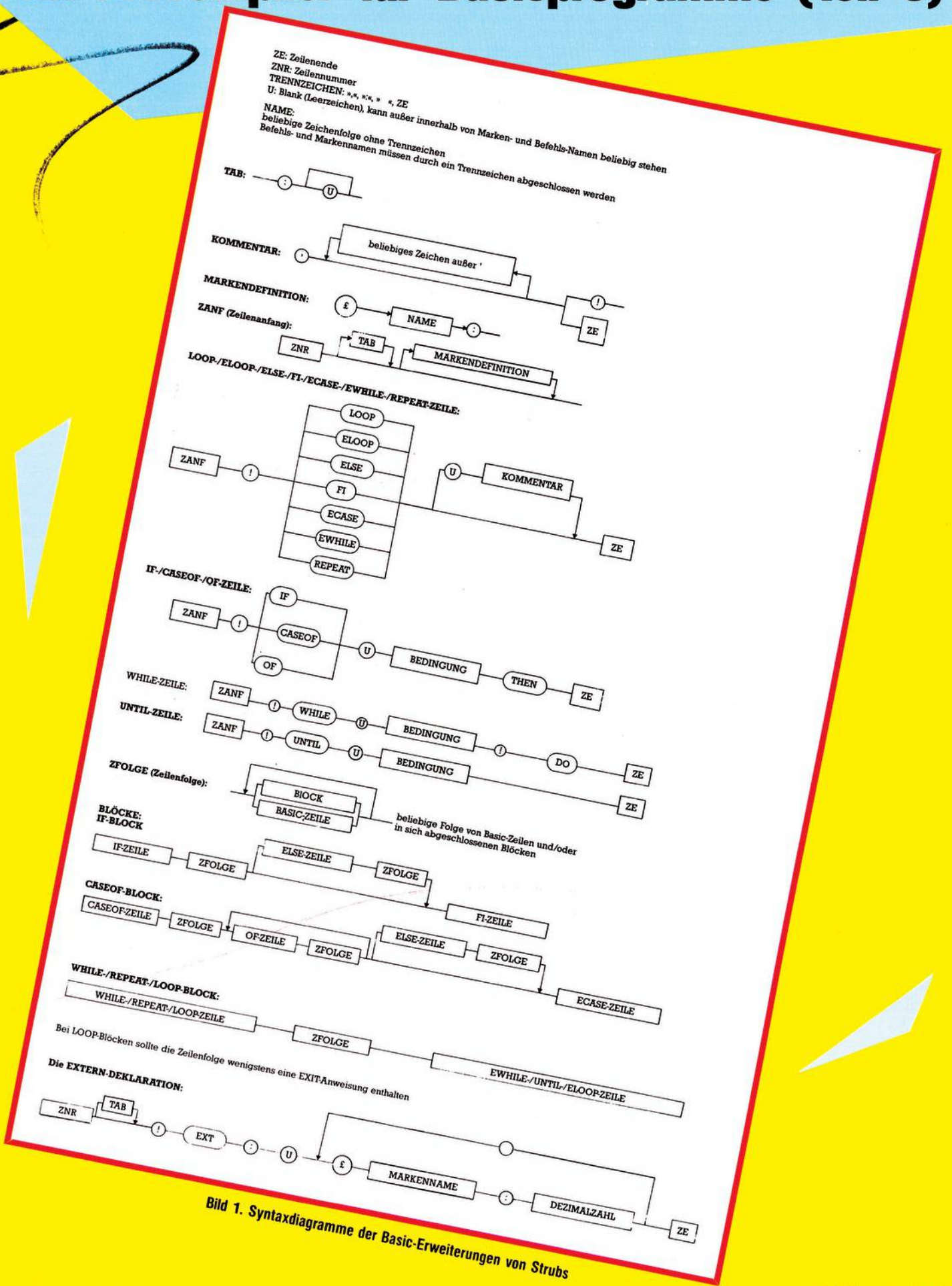

Bild 1. Syntaxdiagramme der Basic-Erweiterungen von Strubs

gramm mit dem Listing in Bild 2 und hoffen Sie, daß der Fehler hier liegt. Wenn Sie keine Abweichungen feststellen, dann haben Sie Pech — Sie haben das Programm Strubs falsch eingetippt.

Wie steht es aber mit Fehlern im Quellprogramm? Syntax-Fehler können sich auf drei verschiedene Arten bemerkbar machen: Vor allem Fehler, die nicht mit den neuen Befehlen zusammenhängen, führen wie gewohnt beim Lauf des Objektprogramms zu den bekannten Fehlermeldungen. Fehler in bezug auf die neuen Befehle quittiert Strubs mit Abbruch der Übersetzung, falls eine Fortsetzung nicht sinnvoll erscheint, oder mit Eintragung in eine Fehlerliste und gleichzeitiger Kennzeichnung der fehlerhaften Zeile im Objektprogramm. Die Fehlerliste kann man sich mit »F« ansehen.

Ein Abbruch der Übersetzung mit entsprechender Fehlermeldung am Bildschirm erfolgt vor allem bei Verstößen gegen die Blockstruktur (und bei Speicherplatzproblemen wie Stack-Overflow oder Listen voll). Bei Fehlern mit den Blöcken — zum Beispiel vor einem ELSE fehlt das IF oder zu einem WHILE fehlt das EWHILE etc. — gibt es ein Problem, das nicht nur bei Strubs, sondern generell bei allen Compilern auftaucht. Der Fehler wird nicht an der Stelle seines Auftretens bemerkt, sondern erst viel später. Die Zeilennummer bei einer Fehlermeldung wie »BLOCKSCHACHTELUNG: ANFANG FEHLT«, sagt also nichts weiter aus, als daß der Fehler erst hier bemerkt wurde. Um bei der Suche nach dem Fehler zu helfen, gibt Strubs aber während der Übersetzung ein Schema der Blockstruktur aus, mit dessen Hilfe sich solche Fehler leicht lokalisieren lassen.

Bei Meldungen wie »zu viele Marken«, »zu viele WHILE/REPEAT« etc. empfiehlt es sich, das Programm in kleinere Teile zu zerlegen, diese getrennt zu übersetzen und anschließend wieder zusammenzufügen.

Wie man dazu vorgeht, behandeln wir weiter unten. Die entsprechenden Listen sind allerdings so großzügig dimensioniert, daß dieser Fall sehr selten eintreten wird.

Sollte während der Übersetzung aus irgendeinem Grund ein unkontrollierter Programmabbruch erfolgen (zum Beispiel OUT OF MEMORY ERROR), dann empfiehlt es sich mit »GOTO 5000« dafür zu sorgen, daß offene Disk-Files ordnungsgemäß geschlossen werden.

Die Beseitigung von Fehlern, die Strubs bei der Übersetzung entdeckt, gestaltet sich relativ einfach: Notieren Sie sich die Zeilennummern zu jedem Fehler und schalten in den Editbereich (mit »E«). Dort kann das Quellprogramm geändert werden, dann wird mit »!« und Wahl von »U« neu übersetzt. Da das Quellprogramm solange im Edit-Bereich bleibt, bis es durch »NEW« oder Laden eines anderen Programms gelöscht wird, kann dieser Vorgang solange wiederholt werden, bis der letzte Fehler beseitigt ist. Sobald die Übersetzung mit der Meldung »0 Fehler« beendet wird, geht es ans Testen des Objektprogramms.

Hierzu wird Strubs durch Eingabe von »S« verlassen. Dadurch wird ein Kaltstart ausgeführt, der die Interpretererweiterung abschaltet und den Speicher säubert. Nun laden Sie das Objektprogramm unter dem Namen, den Sie bei der Übersetzung angegeben haben und starten es mit RUN. Dieses Programm wird nun wie jedes normale Basic-Programm ausgetestet. Dazu können selbstverständlich auch Toolkits mit TRACE- und DUMP-Funktionen verwendet werden. Da die Zeilennummern denen des Quellprogramms entsprechen, empfiehlt es sich, ein Listing des Quellprogramms zur Hand zu haben. Fehler in der Programmlogik lassen sich damit leichter finden und beheben.

Die Bequemlichkeit, die Strubs dadurch bietet, daß Programmänderungen und Verbesserungen im Objektprogramm selbst vorgenommen und sofort ausgetestet werden können, erfordert auf der anderen Seite allerdings eine gewisse Disziplin, damit die Verbindung zum Quellprogramm nicht verloren geht. Jede vorgenommene Änderung sollte sorgfältig notiert und nicht zu viele Änderungen auf einmal vorgenommen werden. Dann wird wieder das Programm Strubs und das Quellprogramm (in den Editierbereich) geladen. Verbessern Sie das Quellprogramm entsprechend Ihren Notizen und übersetzen es erneut. Dieser Vorgang wird solange wiederholt, bis das Ergebnis zufriedenstellend ist.

```
54990 '******** RENUMBER ***************
55000 £RENUMBER: PRINT"□         ***** RENUMBER ******"
55010 ZA=40*256+1:INPUT"ØØ STARTNR.";ZN:INPUT " SCHRITTWEITE";SW
55020 !LOOP
55030 :  IFPEEK(ZA+1)=0THEN !EXIT 'FERTIG
55040 :  H%=ZN/256:POKE ZA+2,ZN-H%*256:POKE ZA+3,H%:ZA=PEEK(ZA)+256*PEEK(ZA+1)
55050 :  ZN=ZN+SW
55060 !ELOOP
55070 RETURN

READY.
```

Bild 2. RENUMBER-Quellprogramm für Strubs

```
55000 PRINT"□         ***** RENUMBER ******"
55010 ZA=40*256+1:INPUT"ØØ STARTNR.";ZN:INPUT" SCHRITTWEITE";SW
55020 :
55030 IFPEEK(ZA+1)=0THEN55061
55040 H%=ZN/256:POKEZA+2,ZN-H%*256:POKEZA+3,H%:ZA=PEEK(ZA)+256*PEEK(ZA+1)
55050 ZN=ZN+SW
55060 GOTO55020
55061 :
55070 RETURN

READY.
```

Bild 3. RENUMBER-Objektprogramm für Strubs

Dieser soeben beschriebene Ablauf kann allerdings in den meisten Fällen vereinfacht werden: Bis auf zwei Ausnahmen können Objektprogramme auch direkt im Editbereich getestet werden. Damit entfällt die Notwendigkeit, Strubs für jede Übersetzung neu zu laden. Nach der Übersetzung wird mit »E« der Editbereich gewählt, dort das Objektprogramm geladen und getestet. Anschließend wird wieder das Quellprogramm in den Editbereich geladen, verbessert und mit »!« und »U« neu übersetzt und so weiter.

Bei Programmen, die nicht im Editbereich getestet werden können, handelt es sich 1. um Programme, die an einer festgelegten Stelle im Speicher stehen müssen. Strubs selbst ist solch ein Programm. Es muß unbedingt am normalen Basic-Anfang stehen. Solche Programme sind allerdings ziemlich selten. Häufiger dagegen findet sich der 2. Fall: Programme, die den Speicherbereich von 700 bis 800

verändern. Hier steht die in Folge 1 erwähnte Interpreter-Erweiterung von Strubs. Dadurch sind vor allem Programme betroffen, die in diesem Bereich Sprites oder Maschinenprogramme benutzen.

Kommen wir noch einmal auf das Schreiben und Editieren von Quellprogrammen zurück. Bisher haben wir nur davon gesprochen, daß die Programmtexte im Editbereich editiert wurden. Diese Methode hat insbesondere bei der Entwicklung umfangreicher Programme einen Nachteil: Da Strubs selbst mit einer Interpreter-Erweiterung arbeitet, kann man nicht gleichzeitig andere Interpreter-Erweiterungen — zum Beispiel Toolkits oder das DOS — benutzen. Möchte man auf Befehle wie MERGE, AUTO, FIND etc. nicht verzichten, dann kann man die Quellprogramme vollkommen unabhängig von Strubs entwickeln und erst anschließend das fertige Quellprogramm in den Editbereich laden.

```
40050 PRINT"□";"  ****************"
40052 PRINT "  * -- STRUBS  -- *"
40053 PRINT "  *  PRECOMPILER  *"
40055 PRINT "  * BITTE WAEHLEN *"
40058 PRINT "  ****************"
40060 PRINT"■■■■EDIT"



45600 I=0:READW
45610 POKE704+I,W:I=I+1:READW:IFW<256THEN45610
45620 DATA32,115,0,8,201,33,240,4,40,76,231,199
45630 DATA169,18,133,44,169,138,76,231,199,999
```

Bild 4. Diese Änderungen sind an Strubs vorzunehmen, um es auf einen VC 20 mit mindestens 16 KByte Speichererweiterung lauffähig zu machen (zusätzlich zu den Änderungen in Ausgabe 5, Seite 117).

Es zeigt sich, daß die meisten Beschränkungen bei der Arbeit mit Strubs ihren einzigen Grund in der kleinen Interpreter-Erweiterung haben. Wie bereits in der ersten Folge erwähnt, besteht der einzige Sinn dieser Erweiterung darin, das Starten von Strubs vom Editbereich aus dadurch bequemer zu gestalten, daß die Befehlsfolge
POKE 44,8: RUN
durch Eingabe von »!« abgekürzt werden kann. Wenn Sie bereit sind, diese Befehlsfolge jedesmal von Hand einzugeben, können Sie auf die Erweiterung verzichten, indem Sie im Programm die Zeilen 45600 bis 45680 einfach weglassen. Damit fallen dann die oben erwähnten Beschränkungen weg, das heißt die unter Fall 2 erwähnten Programme können im Editbereich getestet werden und Strubs kann zusammen mit einer Programmierhilfe benutzt werden. Aber editieren Sie keine Quellprogramme unter Simons Basic. Dazu sind weitere Anpassungen erforderlich, auf die wir in der nächsten Folge näher eingehen. Insbesondere wenn ein Programm aus fertigen Modulen zusammengesetzt werden soll, sind solche Programmierhilfen erforderlich.

Dieser Vorgang verläuft auf der Quellprogramm-Ebene aufgrund der Unabhängigkeit von Zeilennummern relativ einfach. Die einzelnen Programmteile werden in beliebiger Reihenfolge zusammengesetzt. Dazu kann ein Toolkit oder auch das kleine MERGE-Programm aus dem 64'er, Ausgabe 4/84, benutzt werden. Dabei können ruhig gleiche Zeilennummern auftreten und auch die Reihenfolge der Zeilennummern ist beliebig. Anschließend wird der fertige Programmtext mit Hilfe einer RENUMBER-Routine vernünftig durchnumeriert. Da Basic keine lokalen Variablen kennt, ist allerdings vor dem Zusammensetzen auf die Variablennamen zu achten. Um unerwünschte Seiteneffekte zu vermeiden, sind eventuell einige Umbenennungen vorzunehmen. Etwas aufwendiger gestaltet sich der Prozeß, wenn verschiedene Programmteile erst auf der Objekt-Ebene zusammengesetzt werden sollen. Hierbei ist darauf zu achten, daß sich die Bereiche der Zeilennummern nicht überschneiden. Weisen Sie jedem Programmteil einen bestimmten Zeilennummernbereich zu und verlegen diesen Teil gegebenenfalls mit RENUMBER in diesen Bereich. Anschließend werden nun in jedem Programmteil alle externen Routinen, die dieser Teil aufruft, mit Hilfe der EXTERN-Deklaration vereinbart (das sind die Routinen, die erst nach der Übersetzung angefügt werden). Das oben für die Variablennamen gesagte gilt hier entsprechend. Jetzt können die einzelnen Teile getrennt übersetzt und anschließend in der richtigen Reihenfolge verknüpft werden.

Falls Sie keine Erweiterung besitzen, dann können Sie Strubs um eine RENUMBER-Routine erweitern: Fügen Sie die Zeilen aus Bild 3 in das Objektprogramm von Strubs ein — und, falls Sie das Quellprogramm besitzen, dort entsprechend die Zeilen aus Bild 2. Um nun diese Routine anzubinden, müssen nur noch zwei Zeilen in das Menü eingefügt werden:
40110 PRINT "{CDOWN} {REV ON} R {REV OFF}ENUMBER"
und
40210 IF L$="R" THEN GOSUB 55000: GOTO 40050
beziehungsweise für das Quellprogramm:
40210 IF Z$="R" THEN GOSUB £RENUMBER: GOTO £MENUE

Diese Routine kann dann mit »R« aufgerufen werden, um ein Programm, das sich im Editbereich befindet, umzunumerieren. Das Programm Strubs arbeitet nicht mit der Datasette, sondern es benötigt eine Floppy. Damit der für Strubs unterhalb des Edit-Bereichs reservierte Platz nicht überschritten wird, ist darauf zu achten, daß beim Eintippen des Programms keine Blanks eingefügt werden.

Der Editbereich beginnt bei 40*256+1. Vor dem 1. Start von Strubs läßt sich mit 'PEEK(46) feststellen, ob Strubs diese Grenze nicht überschreitet (der Wert muß kleiner als 40 sein, im Originalprogramm liegt er bei 34).

Da Strubs den Zeiger für »Variablen-Anfang« heraufsetzt, sollte es immer von dem 1. Start abgespeichert werden (auch bei Veränderungen). Sollte man dies einmal vergessen, kann man durch »POKE 46,39:CLR« vor den Abspeichern Strubs in die richtigen Grenzen verweisen. (Mathias Törk)