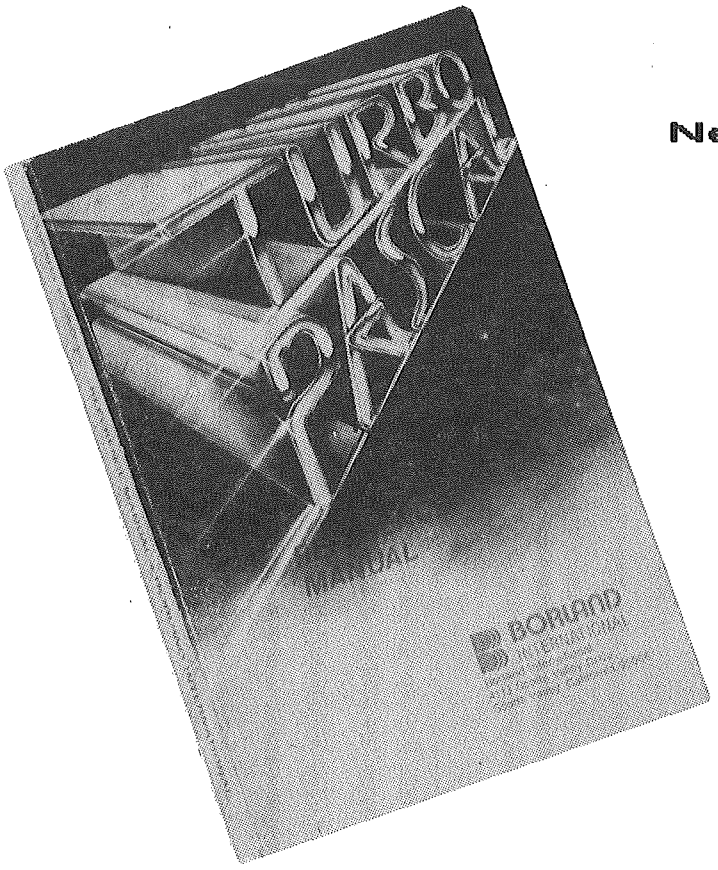
SVI JOURNAL

Die Zeitschrift des Spectra Video Club Austria



Neue Serien:

Turbo Pascal, von Anfang an

CP/M auf dem SVI-328

JÄNNER 1985

INHALT

Heft 1/85 **S 15.-**

Lieber SVI-Journal-Leser!

Sie werden schon bemerkt haben, wir stellten unsere Zeitschrift wieder etwas um. Das augenscheinlichste Merkmal ist die neue Aufmachung der Überschriften. Nicht nur, daß von nun an auch die bisher gesetzten Titel per Matrixdrucker erzeugt werden, wir haben sie auch noch mit zwei Linien umrahmt, um alles in einem "geordneten Rahmen" untergebracht zu wissen. Neben dieser teils optischen, teils technischen Retousche finden sich auch zwei weitere Serien in unserem Repertoire.

Kommen wir zuerst zum Turbo-Pascal. Da unser SVI-328 bekanntlich CP/M-fähig ist, gibt es schon seit längerer Zeit diese Sprache für die SVI-Computer. Grund genug für uns, einmal allen jenen Unterstützung zu bieten, die mit diesem Medium noch nie gearbeitet haben. So haben wir neben den Programmiersprachen Assembler und BASIC auch noch einen Turbo-Pascal-Kurs eingeführt.

Der Autor hat sich bemüht, einen sehr praxisnahen Lehrgang zu kreieren. Deshalb wird immer ein Teil Theorie und ein Teil "harte" Programmierpraxis sein. Schon in der ersten Folge wird ein kleines Programm erstellt. So bekommt der Leser von Anfang an ein direktes Gefühl fürs Programmieren. Im ersten Teil wird ein Quadratzahlenprogramm erstellt.

Man wird es uns zwar nicht glauben, aber es ist wirklich reiner Zufall, daß justament in der BASIC-Folge in dieser Ausgabe ebenfalls ein Quadratzahlenprogramm vorgestellt wird. Der Bediener hat jedoch so den Vergleich zwischen Pascal und BASIC. Natürlich sind die beiden Programme etwas unterschiedlich aufgebaut, da die Autoren verschiedene Zwecke mit ihren Entwicklungen verfolgten. Im BASIC wird es für die Erklärung von Schleifen gebraucht, im Pascal verwendet man es "nur" als Demo.

Doch nicht nur auf dem "Fremdsprachensektor" erneuerten wir unsere Zeitschrift. Viele verwenden das Betriebssystem CP/M, aber nur wenige wissen wirklich die Feinheiten dieses Systems (zu schätzen). Einige Anwender benützen das CP/M überhaupt nur als Grundlage für weitere Programme, etwa dem Turbo-Pascal-Compiler.

Für interessierte Bediener schaffen wir hier Abhilfe. Es startet eine Serie, welche das Betriebssystem CP/M beschreibt.

Nun wird man sich fragen, wo denn der Platz für zwei neue Serien herkommt. Zum Einen pausieren die TIPS & TRICKS, da der Block "Struktur des BASIC-Programmspeichers" zu Ende ist. Andererseits wurde der Platz für die Programme etwas beschnitten.

Sehen wir uns nun die Bilanz des Jahres 1984 an.

Zuerst klären wir den Umfang, den unser Club hat. 90 Mitglieder sind seit dem Bestehen unseres Clubs ab April 1984 zu uns gestoßen. Ebenso haben etwa 40 Personen unser "Abo" bestellt.

Unser Club hat schon 14 ordentliche Clubabende und zwei außerordentliche Clubanbende veranstaltet.

```
*****************************************
*                                       *
*  Die nächsten Clubabende:             *
*                                       *
*                                       *
*   Mi, dem 30. Jänner  1985, ab 19 Uhr *
*   Sa, dem 16. Februar 1985, ab 17 Uhr *
*   Mi, dem 27. Februar 1985, ab 19 Uhr *
*   Sa, dem 16. März    1985, ab 17 Uhr *
*                                       *
*  wie  immer  im Computer-Studio,      *
*  1040  Wien,  Paniglgasse  18-20.     *
*  Nichtmitglieder sind willkommen.     *
*  Ende jeweils  ca. 22 Uhr!            *
*                                       *
*  Aktivitäten  an  den  Clubabenden:   *
*  Arbeiten an Spectravideo-Systemen,   *
*  Informationsaustausch zwischen Club- *
*  mitgliedern,  Gelegenheit  zum  Aus- *
*  drucken von Programmlistings.        *
*                                       *
*  Telefonische Auskünfte über den Club *
*  und seine  Aktivitäten  erhalten Sie *
*  unter der Telefonnummer 65 88 93.    *
*                                       *
*****************************************
```

Die ordentlichen Clubtreffen sind allgemeinbekannt. Sie dienen zum Informationsaustausch, zum Videospielen und zu anderen Gelegenheiten, sich mit SVI-Bedienern zu unterhalten. Dabei sind wir im Computer-Studio der Firma Wehsner untergebracht, welche uns die Räumlichkeiten zur Verfügung stellt. Ebenso wird uns dort die Möglichkeit zur kostenlosen Benützung von Computer und Peripherie gegeben.

Sehen wir uns die außerordentlichen Clubabende an. Wir veranstalteten zwei dieser Art. Der erste fand am 10. Oktober statt und behandelte das Thema "Floppy-Disks und ihre Anwendungen". Über die SVI-Computer samt Peripherie wurde dann am 7. November vorgetragen. Damit dürfte der Sinn dieser Type von Clubabenden klar sein. Wir wollen für ernste Interessenten auch Vorträge gestalten, von denen unsere Clubmitglieder definitiv von anderen Clubmitgliedern lernen können.

Last not least, kommt auch noch der Umfang unserer Clubzeitschrift vor den Vorhang. Wir haben schon sechs Nummern glücklich herausgebracht, die Novembernummer hatte dabei schon eine Auflage von 750 Stück.

Ihr SVI-Journal-Chefredakteur Gerhard Fally!

Ihr SVI-Journal-Abonnement garantiert Ihnen, über alle SVI-Entwicklungen stets auf dem Laufenden zu sein.

Das Jahresabonnement 1985 (12 Hefte) kostet S 150,- (inklusive Postversand).

Übrigens:Für Mitglieder des Spectra Video Club Austria ist das SVI-Journal gratis.

Bestellungen richten Sie bitte an: Spectra Video Club Austria, c/o. Computer-Studio, 1040 Wien, Paniglgasse 18-20. Tel. 65 88 93.

Nachdem wir uns beim letzten Mal mit der FOR..NEXT-Schleife beschäftigt haben, jedoch mit der Beschreibung nicht ganz fertig geworden sind, setzen wir diesmal wieder mit dieser Anweisung fort.

Wir lernten, wie eine Schleife im Programm entsteht, und daß der FOR...NEXT-Befehl hier eine entscheidende Rolle spielt. Wir sehen uns nun drei Möglichkeiten an, wie wir diese Schleife nutzbringend anwenden können.

Der eigentliche Sinn unserer Schleife ist es, einen Programmteil beliebig oft zu wiederholen. Wir lernten so eine Anwendung schon bei unserem ersten Beispiel in der vorigen Folge kennen.

Die zweite Methode ist der ersten sehr ähnlich. Auch diesmal wird ein Programmteil mehrmals verwendet. Aber nun spielt auch die Laufvariable eine Rolle. Wir können sie nämlich auch in den Programmablauf einbauen. Wenn für jeden Schleifendurchlauf eine Zahl gebraucht wird, die um eins höher ist, als beim vorigen Mal, so eignet sich die Laufvariable sehr gut dazu. Wir wollen nun die Quadratzahlen der ersten fünf Zahlen erzeugen, mit Hilfe der Schleifenvariable.

```
10 FOR I=1 TO 5
20 PRINT I*I
30 NEXT I
```

Dieses Programm ergibt:

```
 1
 4
 9
16
25
```

Die dritte und letzte wichtige Möglichkeit macht sich den Effekt zu Nutze, daß jeder Befehl Zeit kostet, wenn er ausgeführt wird. Der Vorteil des FOR...NEXT-Befehls besteht darin, daß man die Zeit, die dieser Komplex dauert, genau einstellen kann. Man verändert ganz einfach die Anzahl der Schleifendurchgänge. So kann man leicht durch einfaches Experimentieren die genaue gewünschte Verzögerungszeit einstellen. "FOR I=1 TO 2000: NEXT I" erzeugt eine mäßig lange Verzögerung.

Der interessierte Leser darf sich selber ausrechnen, wie lang diese Wartezeiten werden können, wenn man nur die Zahlen des Anfangs- und des Endwertes ändert. Der gesamte Zahlenbereich des Computers umfaßt 10 128. Bei einer Verarbeitungszeit von ungefähr 1.5 Millisekunden pro Schleifendurchlauf ergibt sich eine Zeitdauer von 4.75*10 117 Jahren. Sollte jemandem wieder Erwarten diese Zeitdauer nicht genügen, so kann er eine Schleife mit einer zweiten verschachteln. Bei dieser Technik "umrahmt" eine Schleife mit einer Laufvariable eine andere mit einer zweiten Variable. Wie so eine Schleife aussieht, zeigt das untenstehende Programm.

```
10 FOR I=1 TO 20          R
20 FOR T=1 TO 10          I
30 ...                    C
    .                     H
    .                     T
    .                     I
400 NEXT T:NEXT I         G
```

Überkreuzen dürfen sich die beiden Schleifen jedoch nicht. Ein falsches Beispiel wäre:

```
10 FOR I=1 TO 20          F
20 FOR T=1 TO 10          A
30 ...                    L
    .                     S
    .                     C
    .                     H
400 NEXT I:NEXT T
```

Zuletzt lernen wir noch den Anhang STEP kennen. Mit ihm können wir die Schrittweite verstellen, mit der die Schleifenvariable erhöht wird. Voreingestellt ist ja Eins, aber man darf mit diesem Zusatz ohne Weiteres auch 5 oder 7.5 verwenden. Dann nimmt der Computer den Anfangswert, und erhöht die Schleifenvariable solange, bis der letzte Wert errechnet wird, der unter der Grenze des Endwertes liegt. Bei untenstehender Schleife wird bis 6 gezählt. 7 ist die Grenze, 8 wäre erst der nächste errechnete Wert (8 ist größer als 7).

```
10 FOR I=2TO7STEP2
20 PRINT I
30 NEXT I
```

Wenn man in die Zahl hinter STEP negative Werte einsetzt, zählt die Schleife selbstverständlich auch rückwärts.

Das Problem, Schleifen nicht über Kreuz zu verschachteln, haben uns die Programmierer des MSX-BASIC-Interpreters leicht gemacht. Wir können die Variable hinter dem NEXT-Befehl weglassen. Dann sucht sich der Computer automatisch die richtige Variable. Es können nur noch "richtige" Schachtelungen existieren.

Neben den FOR...NEXT-Schleifen sind die IF...THEN-Befehle ebenfalls sehr wichtig. Sie dienen dazu, zwei arithmetische Ausdrücke oder Zeichenketten zu vergleichen. Sehen wir uns dazu die einzelnen Teile dieser Anweisung an.

IF 'Ausdruck' THEN 'Ja-Zweig' ELSE 'Nein-Zweig'

Der 'Ausdruck' beherbergt die beiden Ausdrücke und den Vergleichsoperator. Dieser kann sein: <, >, =, <=, >= oder <>.

Wie Sie sehen, sind dies die "klassischen" Gleichheits- und Ungleichheitszeichen der Mathematik, also ganz einfach zu merken.

Wenn dieser Vergleich nun positiv ausfällt, daß heißt, die in 'Ausdruck' gestellte Bedingung ist erfüllt, dann wird der 'Ja-Zweig' hinter THEN ausgeführt. In diesem Zweig dürfen beliebige Befehle stehen. Ebenso viel Freiheit läßt der 'Nein-Zweig' hinter ELSE. Diesen Weg wählt der Computer, wenn die Bedingung nicht erfüllt ist.

Wie schon erwähnt, dürfen sowohl Strings als auch Zahlen verglichen werden. Es dürfte ja klar sein, wie die Regeln für das Vergleichen von Zahlen lauten (etwa: 3 < 7 stimmt, da 3 kleiner 7 ist). Bei Strings werden die ASCII-Codes überprüft. Zuerst nimmt sich der Computer die ersten Zeichen zum Vergleich vor. Wenn diese kein eindeutiges Ergebnis erzeugen, werden die zweiten

Zeichen verglichen und so weiter. Haben beide Strings genau die gleichen Zeichenketten, ist jedoch einer kürzer als der andere (etwa 'Maier' und 'Maierin'), so gilt, daß der kürzere String auch der kleinere ist.

Dadurch wird es übrigens auch möglich, Strings nach dem Alphabet zu ordnen (A<B, weil 65<66; 65 ist ASCII-Code für A, 66 ASCII-Code für B).

Doch nicht nur diese Spezialität kann der IF...THEN-Befehl vorzeigen. Man darf als Ausdruck auch mehrere Bedingungen verknüpfen. Man will zum Beispiel nur dann einen Programmteil ausführen, wenn sowohl N den Wert 4 als auch S den Wert 35 hat. Dies wird so im BASIC realisiert: IF N=4 AND S=35 THEN...

Dem versierten Elektroniker wird es bei "AND" einen Stich in sein Herz gegeben haben, den gleichen Ausdruck verwenden auch die Digitaltechniker. In der Tat ist hier eine enge Verbindung gegeben. Unser AND im BASIC verhält sich streng analog zu dem logischen AND in der Digitaltechnik.

Es gibt jedoch nicht nur AND, sondern auch OR, XOR, NOT, IMP und EQV. Sehen wir uns an, wie wir diese verschiedenen "Verknüpfungen" auf ihre Wirkungsweise überprüfen können. Elektroniker mögen nun dezent weghören, wenn wir die Wahrheitstabellen für logische Verknüpfungen etwas "verunstalten".

Um die einzelnen Funktionen leicht zu begreifen, müssen wir uns vorstellen, daß der Computer intern durch eine Eins oder durch eine Null entscheidet, ob eine Bedingung erfüllt ist, oder nicht. Wir sagen "erfüllt" oder "nicht erfüllt", der Computer merkt sich dies durch "1" oder "0". Steht in unserem Mikro nach einem Vergleich Eins, wird der THEN-Zweig ausgeführt, bei Null der ELSE-Zweig. Wir werden nun eine Tabelle aufstellen, wie wir unsere vorherige Bedingung, "sowohl N=4 als auch S=35" in die Form mit Einsen und Nullen bringen können.

N=4	S=35	N=4 und S=35
0	0	0
1	0	0
0	1	0
1	1	1

Nur wenn beide Bedingungen erfüllt sind, wird das Endergebnis ganz rechts auf Eins gesetzt. Diese Art von Tabelle nennt man Wahrheitstabelle. Sie wird vor allem in der Digitaltechnik verwendet. Hier sehen wir wieder, daß unser Computer auch "nur" ein Produkt dieses Teilgebietes der Elektronik ist.

Doch nicht nur für AND gibt es diese Wahrheitstabellen. Jeder der oben angeführten Operatoren hat eine zu bieten. Anhand der Wahrheitstabellen kann man sehen, wie das Gesamtergebnis in Abhängigkeit zu den Teilergebnissen aussieht.

OR:

0	0	0
0	1	1
1	0	1
1	1	1

NOT:

0	1
1	0

XOR:

0	0	0
0	1	1
1	0	1
1	1	0

IMP:

0	0	1
0	1	0
1	0	1
1	1	1

EQV:

0	0	1
0	1	0
1	0	0
1	1	1

Der interessierte Leser muß sich nun mit den Wahrheitstabellen näher auseinandersetzen, wenn er eine der Verknüpfungen verwenden will. Es würde den Rahmen des BASIC-Kurses sprengen, einen kompletten Digitaltechniklehrgang einzuschieben.

Die beiden linken Ziffern bezeichnen immer die beiden Teilbedingungen, die rechte Zahl das Endergebnis. NOT ist eine Ausnahme, es "invertiert" das Ergebnis nur. Aus Eins wird Null, aus Null wird Eins. Hier werden keine zwei Teilbedingungen verknüpft, sondern nur eine bearbeitet, daher nur eine Eingangs- und eine Ausgangskolonne.

Ein Beispiel für den IF...THEN-Befehl gibt es natürlich auch:

```
10 INPUT A$
20 INPUT B$
30 IF A$ B$ THENPRINTA$" IST KLEINER ALS "B$
ELSE A$" IST NICHT KLEINER ALS "B$
40 RUN
```

Nun kann man zwei Strings eingeben und sie danach vergleichen lassen.

Sehen wir uns ein weiteres "Feature" dieser Anweisung an: Das THEN kann gegen ein GOTO ausgewechselt werden, wenn als 'Ja-Zweig' nur ein Sprung vorgesehen ist. Man kann jeoch auch direkt hinter THEN einfach die Zeilennummer schreiben, zu der man springen will. Der Computer interpretiert dies dann als Sprung.

```
IF A=B THEN GOTO 20
IF A=B THEN 20
IF A=B GOTO 20
```

Die drei oben gedruckten Befehle bewirken das Gleiche. An diesen Beispielen sieht man auch, daß man ELSE weglassen darf.

Kommen wir zum nächsten Befehl. Sein Befehlswort könnte sich das Prädikat "Sinnlosestes Token" verdienen. Wir alle kennen nämlich seine Auswirkungen, doch wissen die wenigsten, daß zum Beispiel die Zuweisung X=35 vollständig LET X=35 lauten müßte. Kaum einer schreibt das LET vor die Zuweisung. Es ist auch nicht notwendig.

Der LET-Befehl ist das einzige Kommando im MSX-BASIC, bei dem man das Befehlswort weglassen darf. Die Funktion ist ja völlig identisch mit den "normalen" Zuweisungen. Das LET verbraucht eigentlich nur zusätzlichen Speicher. Der einzige vage Vorteil, den es hat, ist, daß manche Leute der Übersicht halber das Befehlswort verwenden.Wir verlieren aber darüber wenig Worte. Es dürfte jedem klar sein, wie X=35 arbeitet.

Wichtig ist, egal ob mit LET oder ohne, daß man einer Variablen einen Wert oder einen String zuweisen kann. Das '='-Zeichen fungiert hier als "Wertzuweisungsoperator". Er stellt die beiden Ausdrücke nicht gleich, sondern weist den rechtsstehenden Ausdruck der linksstehenden Variablen zu. Dadurch wird es im BASIC auch möglich, X=X+1 als sinnvolle Zuweisung zu verwenden. Jeder Mathematikprofesor würde einem an die Gurgel springen, wenn man dies als mathematischen Ausdruck verwenden wollte.

Assemblerfortsetzungskurs Nr.:007

Wir wollen gleich an das Ende der letzten Folge anschließen. Der "CALL"- und der dazugehörige "RET"-Befehl, wurden hoffentlich genau genug erklärt. Es gibt jedoch noch einen weiteren Befehl, der fast das Gleiche wie "CALL" bewirkt. Es ist dies das "RST"-Kommando. Mit diesem sind acht verschiedene Adressen erreichbar. Diese Einsprungadressen der "RST" (Restart)-Befehle liegen bei 00,08,10,18,...,38. Der Vorteil gegenüber dem CALL liegt darin, daß nur ein Byte an Speicher benötigt wird. Da aber nur acht verschiedene Unterprogramme aufrufbar sind, legt sich ein Programmierer dort nur seine wichtigsten Routinen an. Das Spectravideo-BASIC hat auf diesen acht verschiedenen Adressen die Bildschirmausgabe (RST 18), eine Routine zum Eliminieren von Leerzeichen, eine zum Einlesen eines Zeichen in eine Programmzeile (RST 10) und andere wichtige Unterprogramme liegen, die oft aufgerufen werden.

Bei dem Spectravideo kann der Benützer keine eigenen Routinen bei diesen Adressen ablegen, da sich dort das ROM befindet. Es ist jedoch möglich, dies als ein gevifter SVI-328-Programmierer zu umgehen, indem wir in die zweite Bank schalten und dort unsere Routinen bei diesen Adressen ablegt. Zu unserem Glück befinden sich in der zweiten Bank 32K RAM anstatt des BASIC-Interpreters. Wie man das genau macht, wird erst später erklärt.

Die Programme, die ich von nun an für diese Serie erstellen werde, werden Spectravideo-spezifisch sein, Programme, die man selber eintippen und testen kann. Da nicht alle Leser einen Assembler besitzen, werde ich auch die Opcodes der Befehle dazuschreiben. Dabei ist jedoch einiges zu beachten. Damit die eingetippten Programme nicht vom BASIC überschrieben werden, muß vor dem Eintippen ein CLEAR Kommando durchgeführt werden. Mit diesem Befehl kann, wie Sie aus dem Handbuch entnehmen können, der BASIC-Speicher eingeschränkt werden. Normalerweise ist für das BASIC Speicherplatz bis zur Adresse F500 frei. Darüber liegen Systemvariablen, die das BASIC verwendet.

Sie müssen nun das BASIC-Ende ein Byte unter die Adresse stellen, bei der das Maschinenprogramm beginnen soll. Beginnt ein Programm bei D000, so muß folgender CLEAR-Befehl eingegeben werden: CLEAR 200,&HCFFF. Die erste Zahl gibt an, wie viele Bytes für Strings freigehalten werden sollen (Normalerweise ist dieser Wert auch 200), und die zweite Zahl gibt die letzte mögliche Adresse an, welche das BASIC benützen darf. Der Speicherbereich von D000 bis F500 ist nun für das BASIC tabu und wird von ihm nicht mehr verändert.

In dem Source-File werden einige Befehle verwendet, die nur den Assembler betreffen und keine Z-80 Befehle sind. Diese werde ich nach dem ersten Programm erklären.

Das erste Programm soll ein kleiner bildschirmorientierter Editor werden. Dabei springen einige CALLs Unterprogramme des ROM-Interpreters an.

```
                AUSGABE:  EQU  18H     ;Ausgabe des Zei-
                                       ;chens im Akku
                                       ;zum momentanen
                                       ;Ausgabegerät
                TASTE:    EQU  403DH   ;Einlesen des
                                       ;ASCII-Codes der
                                       ;gedrückten Taste
                                       ;in den Akku
                PRINT:    EQU  6415H   ;Ausgeben des
                                       ;Zeichens auf den
                                       ;Drucker
                CLS:      EQU  377EH   ;Lösche den Bild-
                                       ;Bildschirm

                          ORG  D000H

CDH,7EH,37H               CALL CLS     ;Lösche den Bild-
                                       ;schirm
3EH,00                    LD   A,00H   ;Lösche Akku und
32H,3DH,D0H               LD   (DRUCKE?),A
                                       ;Druckerlaubnis
CDH,3DH,40H READ:
                          CALL TASTE   ;Lese Taste ein
FEH,10H                   CP    10H    ;Vergleiche ob
                                       ;CTRL-P gedrückt
                                       ;wurde
20H,12H                   JR   NZ,NO_CTRL-P
                                       ;Gehe weiter,wenn
                                       ;CTRL-P nicht ge-
                                       ;drückt wurde
3AH,3DH,D0H               LD    A,(DRUCKE?)
                                       ;Lade Akku aus
                                       ;der Speicherzel-
                                       ;le, die angibt,
                                       ;ob das Zeichen
                                       ;auf den Drucker
                                       ;geschickt werden
                                       ;soll oder nicht
FEH,00H                   CP   00H     ;Vergleiche Akku
                                       ;mit Null
28H,04H                   JR   Z,JA    ;Setze Druck-
                                       ;erlaubnis
3EH,00H                   LD   A,00H   ;Lösche Druck-
                                       ;erlaubnis
18H,02H                   JR   WEITER  ;Überspringe das
                                       ;Setzen der
                                       ;Druckerlaubnis
3EH,FFH JA:               LD   A,FFH   ;Setze Druck-
                                       ;erlaubnis
32H,3DH,D0H WEITER:       LD (DRUCKE?),A
                                       ;Schreibe den
                                       ;Wert der Druck-
                                       ;erlaubnis in das
                                       ;Byte, das die
                                       ;Druckerlaubnis
                                       ;beinhaltet
18H,E9H                   JR   READ    ;Lese wieder
                                       ;Taste ein
FEH,1BH NO_CTRL-P:        CP 1BH       ;Wurde ESC ge-
                                       ;drückt?
C8H                       RET  Z       ;Steige aus dem
                                       ;Maschinenpro-
                                       ;gramm aus
4FH                       LD   C,A     ;Speichere das zu
                                       ;druckende Zeich-
                                       ;en im Register C
3AH,3DH,D0H               LD   A,(DRUCKE?)
                                       ;Soll Zeichen
                                       ;auch zu Drucker
                                       ;gesendet werden?
FEH,00H                   CP   00H     ;Vergleiche Akku
                                       ;mit Null
28H,04H                   JR   Z,NO_PRINT
                                       ;Keine Ausgabe
                                       ;auf Drucker
79H                       LD   A,C     ;Der Akku bein-
                                       ;halted nun das
                                       ;zu druckende
                                       ;Zeichen
```

```
CDH,15H,64H     CALL PRINT    ;Sende Zeichen zu
                              ;Drucker
79H  NO_PRINT:  LD   A,C      ;Lade Akku mit
                              ;dem Zeichen
DFH             RST  AUSGABE;Gebe Zeichen auf
                              ;Bildschirm aus
FEH,0DH         CP   0DH      ;Wurde ENTER
                              ;gedrückt?
20H,D1H         JR   NZ,READ;Es wurde kein
                              ;ENTER gedrückt
3EH,0AH         LD   A,0AH    ;Lade Akku mit
                              ;dem ASCII-Code
                              ;von LINE FEED
DFH             RST  AUSGABE;und gib auf den
                              ;Bildschirm aus
18H,CCH         JR   READ     ;Beginne wieder
                              ;mit Schleife und
                              ;lese Taste ein
00H  DRUCKE?:    DB  00H      ;Reserviere ein
                              ;Byte für das
                              ;Label DRUCKE?
```

Was macht dieses Programm nun? Antwort: Es liest den ASCII-Code der gerade gedrückten Taste ein und zeigt diesen auch am Bildschirm an. Man kann durch gleichzeitiges Drücken der CTRL und der P-Taste die Druckerausgabe aktivieren. Danach wird jede gedrückte Taste auf den Bildschirm und auf den Drucker ausgegeben. Doch bevor ich auf das Programm genauer eingehen kann, müssen noch einige neue Befehle erklärt werden. Gleich am Beginn des Source-Files kommt der Befehl "EQU" vor. Dies ist ein Kommando, das nur vom Assembler gebraucht wird. Es weist einem Label einen konstanten Wert zu. In die linke Spalte ist der Name des Labels zu schreiben und rechts vom "EQU" der Wert, den das Label erhalten soll.

Diese Anweisung ist mit dem BASIC Kommando "LET" vergleichbar, mit dem man einer Variablen einen Wert zuweisen kann. In diesem Fall ist es jedoch nicht eine Variable, sondern eine Konstante, da ihr Wert nie verändert werden darf. Der Wert eines Labels bleibt immer gleich und wenn ihm ein Wert zugewiesen wird, so muß dieser mit einem Doppelpunkt enden. Da dies ein Assembler-Befehl und kein Z80-Code-Kommando ist, gibt es einige Assemblerversionen, die kein EQU akzeptieren. In der Regel wird dies jedoch nicht passieren.

In unserem Fall wurden den Adressen der wichtigsten Routinen außerhalb des eigentlichen Programmes Labels zugeordnet. Die Routine zum Ausgeben eines Zeichens, das sich im Akku befindet, liegt bei 18H. Diese Routine wurde mit dem Label "AUSGABE" gekennzeichnet. Wenn man diese Routine aufruft, wird kein Register verändert, auch nicht das der Flags.

Die Routine TASTE liest den ASCII-Code der momentan gedrückten Taste ein und speichert diesen im Akkumulator. Die Routine wird erst verlassen, wenn eine Taste gedrückt wurde.

Die Subroutine PRINT tut fast genauso das Gleiche wie das Unterprogramm AUSGABE, das Zeichen, welches sich im Akkumulator befindet, wird jedoch auf dem Drucker ausgegeben. CLS bewirkt das Gleiche wie der BASIC-Befehl CLS. Ruft man diese Routine auf, so wird der Bildschirm gelöscht. Der BASIC-Interpreter macht im Wesentlichen auch nichts anderes, wenn er auf den Befehl CLS stößt, als diese Routine aufzurufen.

Der nächste unbekannte Befehl ist die "ORG" (Origin; engl.: Ursprung)-Anweisung. Damit läßt sich angeben, wo der Assembler das assemblierte Programm ablegen soll. In unserem Fall wird das fertig assemblierte Maschinencodeprogramm bei D000H abgelegt. Alle Sprungadressen innerhalb eines Programmes werden relativ zu dieser Adresse berechnet. Weiters möchte ich noch darauf hinweisen, daß von nun an allen Hexzahlen doch ein "H" angefügt wird, um diese als Hexadezimalzahlen zu kennzeichnen. Obwohl in einer der ersten Ausgaben darauf hingewiesen wurde, daß alle Zahlen, denen nichts beigefügt wurde, Hexzahlen sind, ändere ich nun die Schreibweise, da viele Assembler-Versionen dieses "H" brauchen. Wir werden jetzt voll in die Praxis einsteigen, und da kann es lästig sein, wenn ein Assembler dieses "H" braucht, wir aber nicht angeben, wo es überall stehen soll. Eine Zahl ohne Zeichen wird nämlich als Dezimalzahl interpretiert.

Am Ende befindet sich noch ein neuer uns unbekannter Befehl: "DB" (Define Byte; engl.: Definiere Byte). DB ist wieder nur ein Assemblerkommando, welches bewirkt, daß die folgenden Zahlen, die durch Beistriche getrennt werden müssen, in den dortigen Speicher geschrieben werden. In diesem Fall könnte man das "DB 00" auch durch den Z-80 Befehl "NOP" (No operation; engl.: keine Operation) ersetzen. Dieser Befehl hat den Opcode 00 und veranlaßt den Mikroprozessor, nichts zu tun. Er verzögert nur die Programmausführungszeit und hat keinerlei Auswirkung auf die Register oder Flags. Doch wieder zurück zum DB! Will man mehrere Werte nacheinander in den Speicher geschrieben haben, so sind diese, wie schon erwähnt, durch Beistriche zu trennen. Will man zum Beispiel die Wert 15H,22H und D8H im Speicher stehen haben, so sieht der DB-Befehl folgendermaßen aus: DB 15H,22H,D8H.

Da man nicht immer nur einzelne Bytes in den Speicher zu schreiben hat, sondern auch ganze Adressen, gibt es das Kommando "DW" (Define Word; engl.: Definiere Wort). Es ist genauso wie DB zu handhaben, jedoch wird die angegebene Zahl als 16-Bit Zahl angesehen und daher zuerst ihr Lowbyte und dann ihr Highbyte in den Speicher geschrieben.

Kommen wir nun zu dem ersten neuen Z-80 Befehl. Gleich nach dem CALL "TASTE" erscheint der Befehl CP 10H. Der "CP" (Compare engl.: Vergleiche)-Befehl ist eine sehr wichtige und häufig verwendete Anweisung. Damit ist es möglich, den Akkumulator mit einem bestimmten Wert zu vergleichen. Sie ist fast identisch mit dem "SUB"-Befehl, der Inhalt des Akkumulators wird jedoch nicht verändert. Lediglich die Flags werden entsprechend dem Ausgang gesetzt. Ist der angegebene Wert gleich dem Inhalt des Akkumulators, so wird das Zero Flag gesetzt, da ja bei einer Subtraktion Null herauskommen würde. Genauso wird auch das Carry Flag entsprechend dem Ergebnis gesetzt.

Nachdem "CP" diesmal der einzige neue Befehl war, komme ich nun zur Programmbeschreibung. Am Beginn des Programmes wird der Bildschirm und die Druckerlaubnis gelöscht. Die Druckerlaubnis gibt an, ob die gedrückte Taste auch auf den Drucker ausgegeben werden soll oder nicht. Wenn ein Wert ungleich Null in der Speicherzelle steht, so wird alles auch auf den Drucker ausgegeben. Bei Null wird die Druckausgabe ganz einfach übersprungen. Nachdem nun eine Taste eingelesen wurde (CALL "TASTE"), wird abgefragt, ob CTRL und P gedrückt wurde. Wenn ja, so wird die Druckausgabe umgestellt. Ist das Byte in der Druckerlaubnis Null, so wird ein Wert ungleich Null (z.B. FFH) in die Druckerlaubnis geschrieben. War der Drucker vorher aktiviert, so wird Null hineingeschrieben und damit kein Zeichen auf den Drucker umgeleitet.

Danach wird wieder zu "READ" verzweigt und die nächste Taste eingelesen. Durch Betätigen einer anderen Taste als CTRL-P verzweigt

Fortsetzung auf Seite 8

Das CP/M-Betriebssystem am SVI-328

Das CP/M-Betriebssystem ist im Lieferumfang der Spectravideo-Computer enthalten. Es wird beim Kauf eines Diskettenlaufwerks auf einer Diskette samt ausführlicher Literatur(allerdings in englischer Sprache) mitgeliefert. Was es damit auf sich hat und was man alles damit anfangen kann, soll Ihnen die in diesem Heft beginnende Serie näherbringen.

1. Einleitung

Das Hauptbetriebssystem des SVI-328 ist der BASIC-Interpreter, der sich im Festspeicher (BASIC-ROM) befindet. Mit diesem Interpreter meldet sich der SVI-328 beim Einschalten. Bestandteil des BASIC ist das Laden bzw. Speichern von Programmen und Dateien auf KASSETTEN.

Will man diese Möglichkeiten (bei den im folgenden beschriebenen Betriebssystemen auch vieles mehr) auf DISKETTEN ausweiten, muß auf eines von zwei, von den Systemspuren einer Diskette zu ladenden Betriebssysteme zurückgegriffen werden. Die entsprechende Systemdiskette muß sich vor dem Einschalten des SVI-328 (der Super-Expander sollte schon eingeschaltet sein) im Laufwerk A befinden. Auf jeden Fall versucht die Initialisierungsroutine im BASIC eine "Boot-Routine" von Spur 0 einer eingelegten Diskette in den Speicher zu laden, um bei Erfolg dieser die Kontrolle über das weitere Geschehen zu übergeben.

Erst hier entscheidet sich, ob das Disk-BASIC zum schon vorhandenen BASIC hinzugeladen wird oder eben das CP/M-Betriebssystem zur Anwendung kommt. Bei Mißerfolg, sei es durch keine eingelegte oder mit keinem System versehene Diskette bleibt alles beim normalen BASIC.

Wie schon vorher erwähnt, werden im Falle des Disk-BASIC-Betriebs vor allem Unterprogramme zum schon vorhandenen BASIC-Interpreter geladen, der auch weiterhin die Systemkontrolle beibehält. Man kann hier eigentlich nur von einem erweiterten Betriebssystem sprechen.

Anders verhält es sich beim CP/M. Hier wird ein völlig neues und eigenständiges Betriebssystem geladen. Es soll aber nicht verschwiegen werden, daß für verschiedene Ein- und Ausgaberoutinen ins BASIC-ROM gesprungen wird, um schon vorhandene Unterprogramme zu nutzen. Die Kontrolle über das System selbst wird dabei von einem, vom BASIC-Interpreter unabhängigen Programmteil, übernommen.

Beide Betriebssysteme, sowohl das Disk-BASIC als auch das CP/M, gehen vom Vorhandensein eines Massenspeichers, den es zu verwalten gilt, aus. Bei den Computern von Spectravideo ist dieser eben die Diskette.

Das für BASIC-Programmierer wahrscheinlich geläufigere ist das Disk-BASIC. Dieses erlaubt, im Zusammenhang mit dem schon vorhandenen BASIC im ROM alle notwendigen Zugriffe auf das Diskettenlaufwerk durchzuführen. Darunter versteht man:

Laden, Speichern, Löschen von Programmen
Lesen und Beschreiben einzelner Sektoren
Öffnen und Schließen von Dateien usw.

Darüber hinaus wird auch der Betrieb der 80 Zeichen-Karte ermöglicht.

Dieses Betriebssystem ist speziell auf den SVI-328 zugeschnitten. Es bietet kaum Möglichkeit, und wenn, nur mit fundierten Kenntnissen des Systems, Programme von anderen Computern, vor allem Maschinencodeprogramme, am SVI-238 zum Laufen zu bringen.

Beim Betriebssystem CP/M ist genau das Gegenteil der Fall. Hier gibt es, was Ein-/Ausgabe- und Diskettenoperationen betrifft, eine Schnittstelle (für Spezialprogramme eigentlich zwei), die für alle CP/M-Rechner genormt ist und ein Betreiben von CP/M-Programmen auf den unterschiedlichsten Computern ermöglicht. Nicht übersehen darf man natürlich die unzähligen Aufzeichnungsformate, vor allem für Disketten, die diesen Vorteil schnell zunichte machen können. Aber die Erfahrung zeigt, wo ein Wille ist, ist auch ein Weg.

Da es sich um ein Betriebssystem handelt, welches mit vielen Rechnern arbeiten soll, wird auf spezielle Fähigkeiten, vor allem bei Heimcomputern (Farbgrafik, mehrere Tonkanäle), keine Rücksicht genommen. Deren Verwendung unter CP/M ist dadurch jedoch nicht ausgeschlossen. Sollen aber erstellte Programme auch auf anderen Rechnern laufen, weshalb sie auch ihren Namen "CP/M-Programm" tragen, muß darauf verzichtet werden. Die vom Betriebssystem bereitgestellten Routinen sind jedoch so vielseitig, daß dieser Nachteil für die Zeit des Arbeitens mit CP/M in Kauf genommen werden kann.

Außerdem besteht beim SVI-328, zum Unterschied von vielen Konkurrenten, die Möglichkeit, die CP/M-Ebene wieder zu verlassen, um sich, nur durch den Wechsel der Systemdiskette oder Weglassen einer solchen, dem ursprünglichen Betriebssystem, dem BASIC mit all seinen Grafik- und Tonmöglichkeiten, zu widmen.

Die folgenden Kapitel bieten vorerst allgemeine Information, der die Beschreibung der Schnittstellen und deren Anwendung folgt. Abschließend ist die Behandlung des hardwareabhängigen Teils des CP/M-Betriebssystems, des BIOS, geplant.

2. Allgemeine Information

2.1 CP/M-Versionen

Bei den Computern von Spectravideo kommen bis jetzt insgesamt drei sich in der Bezeichnung unterscheidende CP/M-Betriebssystemdisketten zum Einsatz:

Allen gemein ist die Bezeichnung "2.2x". Es handelt sich dabei, um die seit Ende 1979 nicht mehr geänderte Version des CP/M-Betriebssystems von DIGITAL RESEARCH, welches im Gegensatz zu den Vorgängerversionen, (1.3, 1.4 zuletzt 2.1) praktisch fehlerfrei ist. Nicht zuletzt deswegen ist es soweit verbreitet.

Die Unterschiede in den Versionsnummern 2.20, 2.22 und 2.23 beziehen sich ausschließlich auf den hardwareabhängigen Teil des CP/M, das BIOS, dessen Aufgabe es ist, eben den Betrieb des CP/M am SVI-328 zu ermöglichen.

Die Versionsnummer 2.22 beinhaltet im wesentlichen, auf die Version 2.20 aufbauend, zusätzlich:

A. Eine Initialisierung der Funktionstasten mit häufig benötigten CP/M-Befehlen.

B. Eine beim Kaltstart von CP/M stattfindende Überprüfung, ob eine 80 Zeichen-Karte vorhanden ist, um demnach gleich beim Starten des Systems das richtige Sichtgerät (Fernseher oder 80 Zeichen-Karte) einzustellen. Dem gewählten Ausgabekanal zufolge wird beim Anzeigen des Inhaltes einer Diskette, auf die Anzahl der Zeichen pro Zeile Rücksicht genommen, und bei Verwendung eines Fernsehers eine zweispaltige, bei der 80 Zeichen-Karte hingegen eine vierspaltige Darstellung gewählt.

C. Die Initialisierung der RS-232-Schnittstelle.

D. Für die Bedienung des Druckers wird aufgrund der wenigen, dafür notwendigen Maschinensprachbefehle nicht mehr ins BASIC-ROM gesprungen, sondern auf eine eigene, im Betriebssystem befindliche Routine zurückgegriffen.

Die Version 2.23 schließt, wiederum aufbauend auf 2.22, zusätzlich wie folgt ein:

A. Den Betrieb von doppelseitigen Diskettenstationen.

B. Durch das Setzen eines Bytes, Emulation von zwei verschiedenen Sichtgeräten mit unterschiedlichen Steuersequenzen, an der 80 Zeichen-Karte möglich. Voreingestellt ist die Emulierung des Videoterminals DEC VT-52 von Digital Equipment. Beim zweiten einzustellenden Sichtgerät handelt es sich wahrscheinlich um eines von Lear-Seagler, für dessen Vorhandensein ich noch keine Begründung gefunden habe.

2.2 Speicherbedarf und Speicherbänke

CP/M 2.20 fordert als kleinsten zusammenhängenden RAM-Speicher 20 KByte. Beim SVI-328 wird es mit 64 KByte RAM und somit im größten vorgesehenen Speicherausbau betrieben. Um dem CP/M praktisch den ganzen von der CPU adressierbaren Speicherraum in Form von RAM zur Verfügung zu stellen, ist es notwendig, das beim Einschalten des Grundgeräts auf Speicherstelle 0000H bis 7FFFH liegende 32 KByte BASIC-ROM wegzuschalten, um an dessen Stelle eben 32 KByte RAM zu platzieren. Das geschieht durch "BANK-SWITCHING" (Speicherbänke umschalten), unmittelbar nachdem die Initialisierungsroutine des BASIC-ROMs die Kontrolle an die Boot-Routine für CP/M übergeben hat.

Für die Dauer der Tastaturabfrage ,der Ausgabe von Zeichen auf den Fernseher (Videochip) und bei jedem, alle 20 Millisekunden stattfindenden Interrupt wird nach wie vor, nach Ablegen der Rücksprungparameter an einer Stelle außerhalb der zu schaltenden Bank, ins BASIC-ROM gesprungen.

Eine weitere, allerdings wesentlich kleinere Bank wird für die 80 Zeichen-Karte verwendet. Diese befindet sich von Speicherstelle F000H bis F7FFH, also 2 KByte, und dient dem Ablegen jener Zeichen, die unmittelbar darauffolgend auf dem Bildschirm erscheinen sollen.

So wie beim BASIC-ROM wird auch hier für die Zeit der Manipulation einfach die auf gleicher Adresse befindliche Bank in den ursprünglichen Speicherbereich eingeblendet.

Fortsetzung folgt

Fortsetzung von Seite 6

die CPU zu NO-CTRL-P. Hier wird abgefragt, ob die ESC-Taste gedrückt wurde. Wenn dem so ist, wird ins BASIC zurückgekehrt (RET Z), da dieses Programm mit einem CALL aufgerufen wird. Nachdem dies geschehen ist, wird der Tastencode zum Zwischenspeichern in das C-Register übertragen. Dies geschieht, weil der Inhalt des Akkumulators während der Abfrage, ob das Zeichen zum Drucker gesendet werden soll oder nicht, verändert wird.

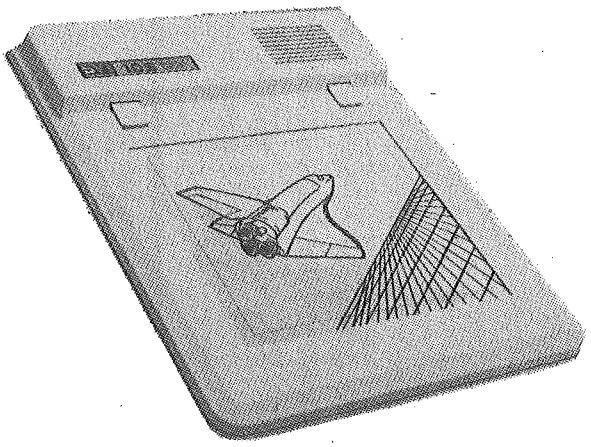
Falls der Inhalt von "DRUCKE?" ungleich Null ist, wird in die Druckerroutine (PRINT) verzweigt, in der das Zeichen zum Drucker gesendet wird. Bei NO-PRINT wird der Akkumulator mit dem aktuellen Zeichen geladen und auf den Bildschirm ausgegeben (RST AUSGABE). Da jedoch, wenn ENTER gedrückt und ausgegeben wurde, der Cursor nicht in die nächste Zeile kommt, sondern am linken Rand der Zeile stehen bleibt, muß man noch ein LINE FEED (Zeilenvorschub) einfügen, welches den Cursor in die nächste Zeile bringt. Bei der Druckerausgabe ist dies nicht notwendig, da die Routine dies selbst erkennt und nach einem ENTER ein LINE FEED nachsendet. Zuletzt verzweigt das Programm wieder zu READ, wo das Ganze von vorne beginnt.

Abschließend ist noch zu sagen, daß das Maschinenprogramm vom BASIC her durch den USR-Befehl aufrufbar ist. Durch den Befehl "DEFUSRx=nn", kann man zehn verschiedenen USR-Funktionen Adressen zuweisen. 'x' gibt eine Ziffer (0-9) an und 'nn' die Adresse, bei der das Maschinenprogramm beginnt. Am einfachsten aufzurufen ist es dann durch "PRINT USRx(0)". Das Argument in der Klammer ist bei diesem Maschinenprogramm total unwichtig. Ihm braucht keine Bedeutung geschenkt werden. Das oben abgebildete Programm ist nach dem Eintippen durch "DEFUSR0=&HD000:?USR0(0)" aufrufbar. "USR" ruft ein Maschincodeprogramm durch ein CALL-Kommando auf, und daher kehrt man durch ein RET wieder zurück. Nur durch Drücken der Taste ESC kommen Sie wieder ins BASIC zurück.

Vielleicht regt Sie dieses Programm an, eigene Entwicklungen zu erstellen, wenn auch nur auf dem Papier. In den kommenden Folgen werden neue Befehle größtenteils nur mehr in Programmen erklärt.

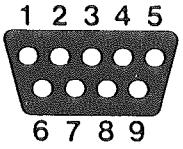
Nachdem wir uns in den letzten Folgen mit den Bausteinen auf der Hauptplatine unserer SVI-Computer befaßt haben, werden wir uns in diesem Heft mit dem Grafik-Tablett und mit Neuerscheinungen bei den Erweiterungskarten beschäftigen.

Das Grafik-Tablett SVI-105 wird direkt an den Joystick-Anschluß des SVI-Computers angesteckt. Es hat eine Zeichenfläche von 186 x 158 mm und verfügt über eine Auflösung von 256 x 192 Bildpunkten. Das Gerät selbst hat die Dimensionen 345 x 280 mm und ist 30 mm hoch.



Die Datenübertragung erfolgt seriell und zwar wird ein Bit in 6 Mikrosekunden übertragen. Die Anschlußbelegung zeigt Ihnen die folgende Darstellung.

1	SCK	Serial Clock
2	SI	Serial Input
3	CS	Chip Select
4	-	No Connection
5	SENSE	Sense Signal
6	SO	Output Signal
7	+5V	Power Supply
8	GND	System Ground
9	EOC	End of Conversion



Das Grafik-Tablett wird von Spectravideo mit der passenden Software (auf Kassette) geliefert. Ein grafisch ansprechend gestaltetes Menü bietet die möglichen Funktionen an: zeichnen mit dünnem oder dickem Strich, wechseln der Hintergrundfarbe, radieren, Text schreiben, Rechtecke oder Kreise zeichnen und so weiter. Auch das Ausmalen einer Figur ist möglich. Anders als beim BASIC-Interpreter muß man dabei nicht die Farbe der Umrandung wählen, sondern es ist jede beliebige Farbe möglich.

Auch das Speichern von gezeichneten Bildern auf Kassette ist möglich. Man kann sie später zu einer weiteren Bearbeitung in den Speicher holen.

Wer sich nicht auf die Spectravideo-Software verlassen möchte und seine eigenen Vorstellungen eines geeigneten Programms hat, kann die Abfrage des Grafik-Tabletts auch im BASIC realisieren (PAD(0), PAD(1), PAD(2)), so wie dies auch beim in Heft 5/84 vorgestellten PAINTSTAR praktiziert wird.

Der Pünktliche wird aber die von einer Wiener Firma hergestellte Ergänzungskarte zum Spectravideo vorziehen. Es handelt sich um eine Hardware-Uhr. Aber nicht nur die Uhrzeit kann man damit einstellen, auch das Datum und den Wochentag. Und wecken lassen kann man sich damit auch. Sogar die Umstellung auf Sommerzeit ist vorgesehen.

Die Karte wird ganz einfach in einen der Expander-Slots gesteckt. Sie ist akkugepuffert und "merkt" sich so die einmal eingestellte Uhrzeit auch nach dem Abschalten des Computers. Sogar auf Urlaub kann man in der Zwischenzeit fahren, denn ungefähr ein Monat hält die Akkuladung an. Dann ist allerdings Zeit, daß man den Computer wieder einmal in Betrieb nimmt, um die Akkus aufzuladen.

Da die Karte 64 Byte speichern kann, die Uhr selbst jedoch nur 14 Byte zur Speicherung der einzelnen Funktionen benötigt, stehen dem Anwender 50 Byte zur dauerhaften Speicherung zur Verfügung (vielleicht für die zuletzt ausgestellte Rechnungsnummer oder sonst eine erinnernswerte Tatsache).

Auch hier gibt es die passende Software samt einer kurzen Beschreibung der Funktionsweise der Karte. Es ist sowohl für die Anwendung mit BASIC als auch unter CP/M vorgesorgt. Die Uhr-Platine kostet S 1.590,-.

Noch eine andere Karte ist neu auf dem Markt: die EPROM-Programmierkarte. Sie wird in Deutschland erzeugt, ist aber auch schon in Österreich zu haben. Sie kostet S 3.390,- und besteht aus einer Steckkarte, die in einen Expander-Slot gesteckt wird. Über ein Kabel ist ein kleines Gehäuse mit einem Textool-Sockel angeschlossen. Dieses kann man außerhalb des Expanders anordnen und kann so EPROMs wechseln, ohne den Expander zu öffnen. Der Stromverbrauch der Karte ist minimal, wenn sie nicht in Betrieb ist, so kann sie also immer im Expander bleiben.

Mit dem EPROM-Programmierer kann man EPROMs bis zum Typ 27256 programmieren. Die mitgelieferte Software stellt die Programmierfunktionen allerdings nur unter CP/M zur Verfügung. Auch die Einstellung der für die verschiedenen EPROM-Typen unterschiedlichen Programmierspannungen erfolgt durch die Software. Alles in allem eine recht brauchbare Karte für alle jene, die ihre erstellte Software gern in EPROMs brennen möchten.

Vom selben Hersteller wird demnächst auch eine I/O-Karte zur Verfügung stehen, die es dem Anwender ermöglichen soll, verschiedene Steuerungen mit dem SVI-Computer zu erledigen.

Schon einige Zeit gibt es auch eine preiswerte Experimentierplatine. Sie kann ebenfalls in den Expander-Slot gesteckt werden. Sie hat etwa halbes Europakartenformat und auf beiden Seiten ein Lötaugenraster zum Aufbauen einer eigenen Schaltung. Die Kosten sind minimal (S 78,-).

Das wären also die neuen Karten, die nicht direkt von Spectravideo kommen. Wir werden auch diese Entwicklungen im Auge behalten. In der nächsten Folge werden wir uns jedoch wieder den von Spectravideo angebotenen Peripheriegeräten widmen und den Floppy-Controller unter die Lupe nehmen.

Schon seit einiger Zeit macht ein neues Softwarepaket von sich reden: Turbo-Pascal. Inzwischen gibt es diesen Compiler auch schon für die Spectravideo-Computer. Mit seinem geringen Preis ist er für jeden verfügbar. In dieser Reihe soll eine Einführung in die Sprache Pascal gegeben werden und die Vorteile und Nachteile von Turbo-Pascal beleuchtet werden.

Zuerst beschreibt der Autor, wie man Turbo in Betrieb nimmt . Wenn man die Systemdiskette in Händen hält, ist es besser, sich sofort eine Arbeitskopie mit dem Programm PIP zu machen. Dieses wird auf der CP/M-Diskette mitgeliefert. Auf der Systemdisk befinden sich außer dem Compiler noch das Installationsprogramm und ein einfaches Tabellenkalkulationsprogramm (das ist ein Software-Paket, bei dem man ein elektronisches Blatt Papier vor sich liegen hat, auf dem man rechnen kann). Dieses Programm ist ein gutes Beispiel, wie man mit relativ wenig Aufwand auch komplexe Programme schreiben kann. Wenn man eine 80-Zeichenkarte sein eigen nennt, kann man das Installationsprogramm sofort löschen. Andernfalls muß man noch den Editor auf einen 40-Zeichen-Bildschirm anpassen. Dazu gibt man einfach "TINST" und "Enter" ein. Alles Übrige ergibt sich im Dialog mit dem Programm. Danach kann man es löschen.

Um nun Turbo zu starten, gibt man einfach "TURBO" ein. Der Compiler meldet sich mit der Frage "Error messages (Y/N) ?". Man ist vor die Wahl gestellt, entweder Fehlermeldungen im Klartext oder 1.2 KB mehr Speicherplatz zu haben. Meistens kommt es jedoch auf diese 1.2 KB nicht an, und so kann man sich den Luxus von Fehlermeldungen ohne weiteres leisten. Man tippt dazu ganz einfach "Y". Turbo quittiert das mit "Loading: TURBO.MSG". Danach befindet man sich im Hauptmenü. Als ersten Versuch kann man das Demoprogramm laufen lassen. Dazu muß man aber das Installationsprogramm vorher gelöscht haben (mit dem Befehl "ERA TINST*.*" + ENTER), um genügend Platz auf der Diskette zu haben.

Zuerst muß das Programm geladen werden. Dazu gibt man "M" ein. Turbo druckt darauf "Main file name:" aus. Man gibt "mc.pas" ein und drückt die Taste ENTER. Wenn Turbo wieder mit dem Bereitschaftszeichen " > " da ist, tippt man "O", dann "C" und zuletzt "Q" ein. Dadurch weist man den Compiler an, auf Diskette zu compilieren. Doch das wird später noch genauer beschrieben. Dann gibt man "C" ein. Dadurch wird das Programm compiliert. Das dauert einige Minuten. Wenn Turbo wieder auf eine Eingabe wartet, gibt man "Q" ein, um aus dem Programm auszusteigen. Auf der Disk kann man jetzt ein File "MC.COM" finden. Dies ist das fertige Programm. Mit "MC" und ENTER wird es aufgerufen.

Ich möchte nun ein wenig auf die Besonderheiten, Vorteile und Nachteile eingehen. Die erste Überraschung war, daß unser Turbo-Pascal sehr wenig Speicher braucht (TURBO.COM hat nur 32 KB). Im Vergleich mit anderen Pascal-Compilern ist das nicht viel (Pascal MT+ hat mehr als 100 KB). Außerdem compiliert Turbo sehr schnell, und ein "Linken" (Vorgang, der das compilierte Programm in ein lauffähiges übersetzt) wie bei anderen Compilern ist überhaupt nicht notwendig. Ein weiterer großer Vorteil ist, daß man nicht extra einen Texteditor verwenden muß, um ein Programm zu schreiben. Bei Turbo ist der Editor beim Compiler gleich dabei, und wenn ein Fehler auftritt, wird der Cursor, nachdem der Fehler gemeldet worden ist, gleich an die Stelle gesetzt, wo der Fehler aufgetreten ist. Dadurch benötigt man relativ wenig Zeit, ein Programm zu entwickeln. Probleme kann es bei Turbo geben, wenn man fertige Assemblerroutinen in einem Pascalprogramm verwenden will. Allgemein kann man sagen, daß Turbo eine der besten Sprachen für die SVI-Computer ist.

Welche Vorteile hat eigentlich Pascal gegenüber anderen Sprachen? Einer der größten Vorteile ist die gute Strukturierbarkeit von Pascal-Programmen. Wenn man zum Beispiel ein langes BASIC-Programm geschrieben hat, und ein halbes Jahr später einige Änderungen vornehmen will, kann man das Programm praktisch neu schreiben. In den meisten Fällen wird man sich nämlich kaum mehr an den Aufbau der Entwicklung erinnern können. Natürlich kann man mit REM-Zeilen das Programm lesbarer gestalten, aber das kostet wertvollen Speicherplatz.

Pascal ist aber so aufgebaut, daß man es leicht "lesen" kann. Es ist möglich, Pascalprogramme ohne ein einziges "GOTO" zu schreiben. In BASIC wird das kaum gelingen. Auch die Verwendung von Labels erleichtert das Verstehen eines Programmes. Labels sind Symbole, die im Pascal für die Ziele von Sprunganweisungen stehen. "GOTO ZEICHNEN" sagt viel mehr aus als "GOTO 110".

Pascalprogramme sind also logischer aufgebaut als BASIC-Programme. Ein weiterer Vorteil ist, daß mehr Möglichkeiten für Entscheidungen während des Programmablaufs vorhanden sind. BASIC kann nur mit "IF...THEN Pascal hat außer "IF...THEN...ELSE" noch "REPEAT...UNTIL", "CASE" und "WHILE". Ein Nachteil von Pascal ist, daß man alle verwendeten Variablen definieren muß, doch daran gewöhnt man sich schnell.

Nun aber genug Theorie. Ich will nun erklären, wie man ein Programm mit Turbo entwikkelt, und eines zur Berechnung von Quadratzahlen schreiben.

Wenn man im Hauptmenü angelangt ist, tippt man "W". Turbo verlangt jetzt einen Filenamen. Geben wir "TEST" ein. Turbo nimmt automatisch die Extension ".PAS" an, und ein File mit dem Namen "TEST.PAS" wird angelegt. Um nun in den Editiermodus zu gehen, gibt man "E" ein. Daraufhin meldet sich der Editor. Dieser Editor entspricht dem N-Modus von Wordstar und ist im Handbuch besprochen. Ich will hier nicht näher darauf eingehen. In Pascal muß man in der erste Zeile den Programmnamen definieren. Das geschieht durch das Statement "PROGRAM". Geben wir also ein:

```
PROGRAM test;
```

Wichtig ist der Strichpunkt am Ende. Er bezeichnet in Pascal immer "Ende der Zeile". Dann folgt der Variablendeklarationsteil Hier müssen alle im gesamten Programm benötigten Variablen vereinbart werden. Wir haben daher die Aufgabe zu überlegen, welche Variablen verwendet werden. Sicher wird eine

Variable nötig sein, in der die Eingabe abgelegt wird. Nennen wir sie "A". Das Quadrat der Zahl legen wir in "B" ab. Den Variablentyp muß man ebenfalls festlegen. Es gibt in Pascal mehrere Typen:

```
INTEGER: Ganzzahlen von -32785 bis +32785
REAL:Gleitkommazahlen  von 1E-38 bis  1E+38
     mit einer Genauigkeit von 11 Ziffern
BYTE:Wert eines Bytes von 0-255
CHAR:Ein einzelnes Zeichen, entspricht BYTE
STRING:Zeichenketten
BOOLEAN:Kann nur 2 Werte annehmen: TRUE
     oder FALSE, wird für logische
     Entscheidungen verwendet
```

Unsere Variablen müssen vom Typ REAL sein. INTEGER ist zu klein, da die Zahlen in diesem Variablentyp nicht größer als 32785 werden dürfen. Der Variablendeklarationsteil wird mit "VAR" eingeleitet. Wir müssen also schreiben:

```
VAR: a,b:REAL;
```

Groß- und Kleinschreibung ist egal. In Zukunft werde ich Pascalkommandos immer groß, alles andere klein schreiben. Wichtig ist wieder der Strichpunkt. Wenn wir drei REAL-, eine INTEGER-Variable und einen STRING mit maximal 30 Zeichen benötigen, müssen wir folgendes eintippen:

```
VAR: a,b,c:REAL;
     d:INTEGER;
     e:STRING(.30.);
```

Statt (. und .) kann man auch eckige Klammern verwenden. Die Reihenfolge innerhalb der Variablendeklaration ist übrigens egal. Man kann also auch INTEGER- vor REAL-Variablen deklarieren. Nun beginnen wir aber das Programm. Den Programmteil leitet man mit "BEGIN" ein. Nun muß man das Programm auffordern, eine Zahl einzugeben. Dazu benötigt man "WRITELN" und "READLN". "WRITELN" ist das Gegenstück zu "PRINT" und "READLN" zu "INPUT". Wir müssen also tippen:

```
BEGIN
  WRITELN('Bitte eine Zahl eingeben');
  READLN(a);
```

In Pascal wird statt dem " ein ' verwendet. Mit dem WRITELN-Befehl wird Turbo angewiesen, den Satz "Bitte eine Zahl eingeben" auf dem Bildschirm auszugeben. Es gibt auch den Befehl WRITE('String') der dem 'PRINT "String";' entspricht. Bei WRITELN wird automatisch der Cursor an den Beginn der nächsten Zeile gesetzt. Mit dem Befehl READLN wartet Turbo auf eine Eingabe, die mit ENTER abgeschlossen wird. Es gibt auch einen Befehl READ, bei dem der Cursor nicht in die nächste Zeile nach ENTER gesetzt wird (ähnlich dem INPUT$, kein Zusammenhang zum READ des BASIC!!).

Nach diesen Programmzeilen befindet sich die eingegebene Zahl in der Variablen A. Jetzt muß das Quadrat berechnet werden. Dazu gibt es in Pascal die Funktion SQR(x). Nicht wie im BASIC, ist SQR(x)=x*x. Das Ergebnis soll in die Variable B kommen. Man muß schreiben:

```
b:=SQR(a);
```

Wichtig ist das Zeichen := . In Pascal wird bei allen Wertzuweisungen := verwendet, nicht wie in Basic nur = . Das Zeichen = ist nur ein logischer Operator, der später behandelt wird. Nun muß das Ergebnis ausgegeben werden. Das geschieht durch foldende Zeile:

```
WRITELN(a,' zum Quadrat ist ',b);
```

Der Beistrich zwischen den einzelnen Argumenten in einem WRITELN oder WRITE Befehl bewirkt dasselbe wie ein Strichpunkt in BASIC. Das Programm ist jetzt fertig. Man muß nur mehr Pascal mitteilen, daß das Programm zu Ende ist. Dies geschieht so:

```
END.
```

Der Punkt nach dem END anstelle des Strichpunkts bezeichnet das Ende des Programms. Unser Programm sieht jetzt so aus:

```
PROGRAM test;
VAR a,b:REAL;
  BEGIN
    WRITELN('Bitte eine Zahl eingeben');
    READLN(a);
    b:=SQR(a);
    WRITELN(a,' zum Quadrat ist ',b);
  END.
```

Jetzt muß das Programm nur noch compiliert werden. Im Gegensatz zu BASIC übersetzt Pascal das Programm in echten Maschinencode. Dadurch wird das Programm bedeutend schneller. Man kann sich vorstellen, daß BASIC immer wenn es einen Befehl vorfindet, in einem "Lexikon" nachsieht, wie der Befehl in Maschinencode ausgeführt wird, und den Befehl ausführt. Dann aber vergißt BASIC die "Übersetzung" wieder, und muß danach wieder nachschauen. So arbeiten Interpreter. Turbo ist aber ein Compiler. Bevor das Programm gefahren werden kann, nimmt Pascal das "Lexikon" und übersetzt das Programm, "notiert" aber die übersetzte Fassung. Jetzt ist das Programm in eine für den SVI verständliche Fassung gebracht, und das "Lexikon" wird nicht mehr benötigt. Die Zeiten für das "Nachschauen" entfallen. Dadurch sind compilierte Programme viel schneller.

Um das Programm zu compilieren, muß man aus dem Editor aussteigen. Das wird mit CONTROL KD ausgeführt. Man tippt einfach die Tasten CTRL und K zusammen, und dann die Taste D. Nun gibt es wieder mehrere Möglichkeiten, ein Programm zu compilieren. Sie werden über ein eigenes Menü angewählt. Dieses Menü wird vom Hauptmenü, in dem wir uns gerade befinden, durch die Taste O abgerufen. Turbo bietet drei Möglichkeiten an, von denen aber nur zwei für unser Programm verwendbar sind:

```
Compilieren im Speicher (Taste M)
Compilieren auf Diskette (Taste C)
```

Kleinere Programme, wie das unsrige, gehen sich problemlos im Speicher aus. Wir drücken also M. Mit Q kommt man wieder in das Hauptmenü zurück. Dort angelangt, wird das Programm mit C compiliert. Das dauert wenige Sekunden. Anschließend kann es mit R gestartet werden. Um auf der Diskette ein ablauffähiges Programm zu erzeugen, so wie bei dem Demoprogramm muß im Optionenmenü (Taste O) C gedrückt werden. Mit Q zurück ins Hauptmenü, und mit C wird compiliert. Auf der Diskette befindet sich jetzt ein File TEST.COM. Dieses Programm ist jetzt auch ohne Turbo ausführbar, in dem man, wenn man sich im CP/M-Betriebssystem befindet, einfach den Namen unter ENTER eingibt.

Mit der Taste Q im Hauptmenü steigt man aus Turbo aus. Turbo fragt, ob das soeben im Speicher befindliche Programm abgespeichert werden soll. Es wird aber nur das nicht compilierte, nicht lauffähige Programm "der Nachwelt erhalten". Um ein compiliertes Programm abzuspeichern, muß man die Option C im Optionenmenü anwählen.

Soweit eine kleine Einführung in Turbo-Pascal. In der nächsten Folge werde ich beginnen, die Programmstrukturen, Unterprogramme und Funktionen in Pascal zu erklären. Ebenso kommt wieder ein kleines Programm an die Reihe.

RAFAEL RAZIM

Befehlserweiterungen ohne CMD

In den letzten Clubzeitschriften haben Sie erfahren, wie man den von Haus aus unbenutzbaren Befehl "CMD" doch verwenden kann. Um Erweiterungen herbeizuführen, wurden vom Interpreter anerkannte Befehlswörter verwendet. Doch was muß man machen, um zum Beispiel RECLAIM, XSWITCH oder PLAY OFF zu implementieren? Wenn Sie diese Wörter eingeben, wird der Computer Sie wahrscheinlich mit Fehlermeldungen eindecken. Um solche Wörter definieren zu können, muß man die Vorgänge beim Ausführen von BASIC-Befehlen genau kennen. Einigen von ihnen wird sicherlich schon die Sprungtabelle zwischen &HFE9E und &HFFE0 aufgefallen sein, die ohne Disk-BASIC mit lauter &HC9-Bytes gefüllt ist. &HC9 ist der Opcode für den Maschinsprachebefehl 'Return', der mit 'RET' abgekürzt wird. Wenn wir das Disk-BASIC in den Computer laden, werden in der Tabelle die Einsprünge für die Befehle zur Diskettenkommunikation abgelegt. Trotzdem bleibt auch bei Verwendung des Disk-BASIC noch immer mehr als die Hälfte dieser Sprungtabelle unbenutzt.

Der Interpreter springt öfters in die Tabelle, und kehrt meistens sofort wieder zurück, da dort &HC9 (RET) steht. Deshalb ist es an vielen Stellen des Interpreters möglich, den Programmablauf unseres BASIC-"Übersetzers" in andere Wege zu leiten. Bekanntlich gibt es im Interpreter einen Programmteil, welcher aus den BASIC-Tokens die zugehörigen Befehlsroutinen ermittelt. Auch in diesem Teil befindet sich ein Aufruf in die Sprungtabelle. So wie bei CMD kann man das RET nun durch einen Sprung in eine eigene Routine ersetzen. Das habe ich gemacht und dabei folgende Befehle erzeugt;

PLAY OFF: Es dürfte jedem klar sein, was diese Anweisung bewirkt. Da man im "normalen" BASIC eine musikalische Untermalung erst dann stoppen kann, wenn der Speicher des PSG leer ist, war es dringend notwendig, eine "Sofortunterbrechung" zu kreieren. Der Computer vergißt bei diesem Kommando alle restlichen Noten, die er noch zu spielen hat.

RECLAIM: Diese Erweiterung kann im wesentlichen das Gleiche, wie die in der letzten Clubzeitschrift beschriebende CMD-Erweiterung ("OLD für SVI-328"). Ein mit NEW gelöschtes Programm ist schon nach wenigen Kommandos nicht mehr zu restaurieren. Deshalb ist es fast unsinnig, eine Garantie auf solche Restaurierprogramme zu geben. Trotzdem ist der Autor der Meinung, daß das neue Programm gegenüber dem alten durch kleine Zusatzeffekte sicherer gemacht wurde. Sollten Sie nämlich versehentlich NEW eingegeben haben, so können Sie mit dem Befehl RECLAIM das Programm samt Variablen wieder zum Vorschein bringen. Sollte der Computer erkennen, daß zwischen dem letzten NEW und RECLAIM ein neues Programm im Speicher ist oder Variablen deklariert worden sind, so verändert RECLAIM nichts und liefert die Fehlermeldung "Illegal function call". Ansonsten wird das Programm wiederhergestellt.

Nebenbei: Sie werden sich vielleicht schon oft gefragt haben, warum der Computer bei Maschincodeprogrammen manchmal abstürzt. Entweder "verrennt" sich das Programm in einer Schleife, oder es verzweigt zu falschen Adressen und so weiter. Das der Computer dann oft neu startet, liegt daran, daß bei jedem NEW die Adresse 0000 auf den Stapel gelegt wird. Dabei kann der Computer schon bei ein, zwei "Returns" zuviel zu 0000 springen, ein vollständiger Neustart samt Initialisierung des Systems ist dann die Folge.

XCOPY: Dieser Befehl benützt die PAGE 21 oder die PAGE 31 des Computers als Speicher für den Inhalt der PAGE 02 ("normaler" Speicherbereich für BASIC-Programmierer). Auf Deutsch: Mit dem Befehl kopieren Sie den augenblicklichen Inhalt des Programmspeichers in eine andere Bank. Das ist ein guter Schutz vor einem ungewollten NEW, denn Sie können somit Ihr Programm nach jeder Änderung abspeichern. Mit dem Befehl XSWITCH vertauschen Sie dann ganz einfach den eventuell gelöschten Programmspeicher mit dem zuletzt kopierten und haben somit Ihr Programm wieder komplett.

XSWITCH (STOP): Wie schon erwähnt, vertauscht der Befehl den Inhalt von PAGE 02 mit PAGE 21 oder 31. Es gilt das Gleiche wie beim normalen SWITCH. Auch das STOP hat die gleiche Bedeutung wie das normale STOP. Die Systemvariable SWITCH wird dadurch nicht beeinflußt!

Im Anhang des Artikels finden Sie das dokumentierte Listing des Maschincodeprogrammes. Ich habe diesmal einen in Maschincode geschriebenen Assembler verwendet. Dieser druckt leider (noch) kein formatiertes Listing des Quellprogrammes aus. Deshalb fehlen dem Listing auch alle Adressen und Opcodes. Ich habe aber ein Hexdump des Programmes angefertigt, in dem alle nötigen Opcodes enthalten sind. Um das Maschincodeprogramm auch verwenden zu können, müssen Sie folgende Schritte genau einhalten:

1) Setzen Sie den Anfang von BASIC-Programmen auf &h8400. Dazu geben Sie folgende Zeile ein: POKE &HF54B,132:POKE &H8400,0:NEW
2) Poken Sie dann die Werte des Hexdumps in den Speicher.
3) Danach speichern Sie das Programm mittels BSAVE "",&H8000,&h827F,&H8200 ab. Das gilt sowohl für Disketten- als auch für Kassettenbenützer.

Sie haben nun ein Maschincodeprogramm zur Verfügung, welches Sie mit BLOAD "",R laden und starten können. Bitte vergessen Sie nicht das ',R'.

Das Programm erzeugt eine kurze Meldung. Wenn Sie nun mit XSWITCH in die andere Bank umschalten, erhalten Sie nochmals die Meldung (nur das erste Mal!).

Zum Maschincodeprogramm selbst:

Die lange Tabelle von "EQU"-Anweisungen mag vielleicht etwas verwirrend erscheinen, ist aber nur halb so schlimm. Die Systemvariablen und Einsprungadressen im ROM bekommen hier nur Namen zugewiesen. Danach folgt die Routine, die vom Interpreter angesprungen wird. Einerseits werden die Register gerettet. Andererseits überprüft der Interpreter, ob überhaupt einer der neuen Befehle eingegeben worden ist. Sollte dies nicht der Fall sein, dann springt das Programm mit unveränderten Registerwerten zurück. Praktisch ist somit dem Interpreter vorgegaukelt worden, daß er ein RET in der Sprungtabelle angesprungen hat. Dabei ist nichts besonderes passiert.

Wurde jedoch einer der neuen Befehle eingegeben, so stellt der Computer mittels drei 'POP BC'-Befehle den Stapel richtig und führt nachher den Befehl aus. Durch die drei POP-Anweisungen wird der Stapel genauso hergestellt, wie er bei einer interpretereigenen Routine wäre.

Die Initialisierungsroutine, die Sie im Listing nach der "ORG 8200H"-Anweisung finden, trägt den neuen BASIC-Beginn in die Systemvariable ein, macht aus dem "RET" bei

&hFF57 ein "JP NEUBEF" und löscht sowohl den Programmspeicher als auch die zweite Bank. Beiläufig schiebt sie noch eine kleine Meldung auf den Bildschirm und springt nach getaner Arbeit wieder in den Direktmodus.

```
HEX-Dump: Startadresse : 8000H
8000  F3 F5 E5 CD 43 80 FE C1  ....C...
8008  CA 5B 80 FE 52 CA 6B 80  .[..R.k.
8010  FE 58 CA DE 80 FE 94 CC  .X......
8018  1D 80 E1 F1 C9 3A FB FF  .....:..
8020  A7 C0 3C 32 FB FF 2A 4A  ..<2..*J
8028  F5 5E 23 56 ED 53 F3 FF  .^#V.S..
8030  2A EE F7 22 F5 FF 2A F0  *.."..*.
8038  F7 22 F7 FF 2A F2 F7 22  ."..*.."
8040  F9 FF C9 F5 E5 3A FB FF  .....:..
8048  A7 28 CF 2A EE F7 ED 5B  .(.*...[
8050  F2 F7 E7 28 C5 AF 32 FB  ...(..2.
8058  FF 18 BF D7 FE EB C2 1A  ........
8060  80 C1 C1 C1 E5 CD 66 40  ......f@
8068  E1 D7 C9 D7 FE 45 C2 1A  .....E..
8070  80 D7 FE 43 C2 1A 80 D7  ...C....
8078  FE 4C C2 1A 80 D7 FE 41  .L.....A
8080  C2 1A 80 D7 FE 49 C2 1A  .....I..
8088  80 D7 FE 4D C2 1A 80 D7  ...M....
8090  C1 C1 C1 F5 E5 F3 3A FB  ......:.
8098  FF A7 CA D5 80 2A 4A F5  .....*J.
80A0  7E 23 B6 3E 00 32 FB FF  .#.>.2..
80A8  C2 D5 80 2A EE F7 ED 5B  ...*...[
80B0  F2 F7 E7 C2 D5 80 2A 4A  ......*J
80B8  F5 ED 5B F3 FF 73 23 72  ..[..s#r
80C0  2A F5 FF 22 EE F7 2A F7  *.."..*.
80C8  FF 22 F0 F7 2A F9 FF 22  ."..*.."
80D0  F2 F7 C3 1A 80 E1 AF 32  .......2
80D8  FB FF F1 C3 9E 0F D7 FE  ........
80E0  C9 28 05 FE D6 C2 1A 80  .(......
80E8  C1 C1 C1 FE D6 F5 D7 F1  ........
80F0  28 01 7E FE 90 37 20 02  (.~..7..
80F8  D7 AF E5 ED 73 76 FE F5  ....sv..
8100  F3 3E 0F D3 88 DB 90 0E  .>......
8108  FD CB 57 20 02 0E F7 A1  ..W.....
8110  4F DB 90 08 21 EB 33 18  O...!.3.
8118  0D 1A CB BA 12 CB FA 13  ........
8120  10 F7 08 D3 8C 08 5E 23  ......^#
8128  56 23 46 23 79 D3 8C 7A  V#F#y..z
8130  B3 20 E6 F1 38 05 3E 03  .....8.>.
8138  32 2B 7E 01 00 80 21 FF  2+....!.
8140  FF CB BC 11 FF FF FE D6  ........
8148  20 05 EB ED B8 18 07 1A  ........
8150  ED A8 02 EA 4F 81 ED 7B  ....O..{
8158  76 FE 08 D3 8C CD 50 00  v.....P.
8160  E1 FB C9 00 00 00 00 00  ........
8168  00 00 00 00 00 00 00 00  ........
8170  00 00 00 00 00 00 00 00  ........
8178  00 00 00 00 00 00 00 D3  ........
8180  FF FF FF FF FF FF FF FF  ........
8188  FF FF FF FF FF FF FF FF  ........
8190  FF FF FF FF FF FF FF FF  ........
8198  FF FF FF FF FF FF FF FF  ........
81A0  FF FF FF FF FF FF FF FF  ........
81A8  FF FF FF FF FF FF FF FF  ........
81B0  FF FF FF FF FF FF FF FF  ........
81B8  FF FF FF FF FF FF FF FF  ........
81C0  00 00 00 00 00 00 00 00  ........
81C8  00 00 00 00 00 00 00 00  ........
81D0  00 00 00 00 00 00 00 00  ........
81D8  00 00 00 00 00 00 00 00  ........
81E0  00 00 00 00 00 00 00 00  ........
81E8  00 00 00 00 00 00 00 00  ........
81F0  00 00 00 00 00 00 00 00  ........
81F8  00 00 00 00 00 00 00 F5  ........
8200  F3 21 57 FF 36 C3 21 00  .!W.6.!.
8208  80 22 58 FF 21 69 81 22  ."X.!i."
8210  4A F5 2B 22 E4 FD CD B5  J.+"....
8218  65 2A E4 FD AF 32 FB FF  e*...2..
8220  77 23 22 4A F5 CD 57 65  w#"J..We
8228  21 45 82 CD 7D 69 21 EA  !E..}i!.
8230  7F CD 7D 69 3E D6 CD EB  ..}i>...
8238  80 CD 74 64 21 AF 09 E5  ..td!...
8240  37 F5 C3 5A 7C 45 78 74  7..Z|Ext
8248  65 6E 73 69 6F 6E 20 56  ension.V
8250  65 72 73 69 6F 6E 20 31  ersion.1
8258  2E 32 20 20 31 39 38 35  .2..1985
8260  07 0A 0D 00 00 00 00 00  ........
8268  00 00 00 00 00 00 00 00  ........
8270  00 00 00 00 00 00 00 00  ........
8278  00 00 00 00 00 00 00 00  ........
8280  00 00 00 00 00 00 00 00  ........
```

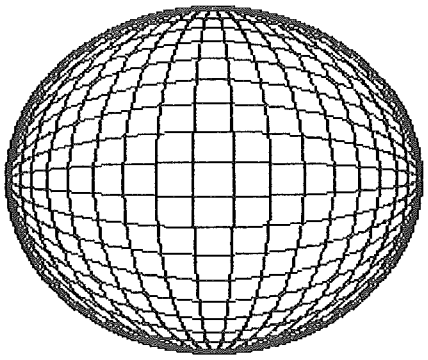
```
100 REM   org  8000h
101 ' ILLEGL ist die Einsprungsadresse
102 ' um den Fehler "Illegal function call"
103 ' zu erzeugen
104 REM illegl equ 0f9eh
105 ' NEWT? sind zwei ROM-Routinen, die
106 ' alle Zeiger wieder gerade biegen
107 ' und den Programmspeicher loeschen.
108 REM newt1  equ 6557h
109 REM newt2  equ 65b5h
110 ' SYSVAR ist der Anfang der Tabelle,
111 ' die Auskunft darueber gibt, welche
112 ' Systemvariablen kopiert werden
113 ' muessen.
114 REM sysvar equ 33ebh
115 ' PRINT und CRLF druckt einen Text bzw.
116 ' ein CR-LF aus.
117 REM print  equ 697dh
118 REM crlf   equ 6474h
119 ' INIT, Anfang des Textes fuer
120 ' "Initializing 2nd bank"
121 REM init   equ 7feah
122 ' Der Einsprung bei FREMSG zeigt an,
123 ' wieviel Speicher frei ist.
124 REM fremsg equ 7c5ah
125 ' CLRPSG loescht alle Zeiger auf
126 ' eventuelle noch zu spielende Noten
127 REM clrpsg equ 4066h
128 ' GRENZE = neuer BASIC-Anfang
129 REM grenze equ 8168h
130 ' BASBEG enthaelt den Zeiger auf die
131 ' erste Zeile, LOWMEM enthaelt das
132 ' "Mass aller Dinge", alles was unter
133 ' dem Inhalt von LOWMEM liegt, ist fuer
134 ' den Interpreter kein Programmspeicher
135 REM basbeg equ f54ah
136 REM lowmem equ fde4h
137 ' PRGEND zeigt auf die drei Nullen,
138 ' VAREND auf das Ende der einfachen
139 ' Variablen, BASEND auf das Ende des
140 ' Programmes einschliesslich aller
141 ' Variablen.
142 REM prgend equ f7eeh
143 REM varend equ f7f0h
144 REM basend equ f7f2h
145 ' STPKEY, wird von XSWITCHSTOP ver-
146 ' wendet, um das STOP einzuleiten.
147 ' STKPTR wird zum Abspeichern des
148 ' Stapelzeigers verwendet.
149 REM stpkey equ 7e2bh
150 REM stkptr equ fe76h
151 ' TOP und LEN dienen zum Umkopieren
152 REM top     equ ffffh
153 REM len     equ 8000h
154 ' EXT = Einsprungsadresse fuer den
155 ' Interpreter
156 REM ext     equ ff57h
157 ' FRSTLN enthaelt den Zeiger auf die
158 ' zweite Zeile des Programmes, ENDPRG,
159 ' ENDVAR und ENDBAS enthalten die Zeiger
160 ' wie sie vor dem letzten NEW waren
161 REM frstln equ top-12
162 REM endprg equ frstln+2
163 REM endvar equ frstln+4
164 REM endbas equ frstln+6
165 ' FLAG entaelt die Erlaubnis fuer
166 ' RECLAIM
167 REM flag    equ frstln+8
168 ' Die Anweisungen in den Zeilen
169 ' 175 bis 192 dienen dazu, die ersten
170 ' Zeichen der neuen Befehle zu er-
171 ' kennen. Das System ist einfach:
172 ' Ueberpruefe die Zeichen und
173 ' sind sie unbekannt, dann kehre
174 ' zurueck (=Label EX).
175 REM neubef
176 REM   di
177 REM   push af
178 REM   push hl
179 REM   call check
180 REM   cp   c1h
181 REM   jp   z,play
182 REM   cp   "R"
183 REM   jp   z,RE
184 REM   cp   "X"
185 REM   jp   z,XS
186 REM   cp   94h
187 REM   call z,new
188 REM ex
```

```
189 REM   pop  hl
190 REM   pop  af
191 REM   ret
192 ' Wenn Sie NEW eingegeben haben,
193 ' dann rettet die kleine folgende
194 ' Routine alle wichtigen Zeiger des
195 ' BASIC-Programmes in die eigens
196 ' dafuer vorgesehenen Variablen. Die
197 ' Variable FLAG verhindert, dass bei
198 ' einem neuerlichen NEW die Zeiger
199 ' nochmals abgespeichert werden.
200 REM new
201 REM   ld   a,(flag)
202 REM   and  a
203 REM   ret  nz
204 REM   inc  a
205 REM   ld   (flag),a
206 REM   ld   hl,(basbeg)
207 REM   ld   e,(hl)
208 REM   inc  hl
209 REM   ld   d,(hl)
210 REM   ld   (frstln),de
211 REM   ld   hl,(prgend)
212 REM   ld   (endprg),hl
213 REM   ld   hl,(varend)
214 REM   ld   (endvar),hl
215 REM   ld   hl,(basend)
216 REM   ld   (endbas),hl
217 REM   ret
218 ' Die Routine CHECK loescht immer dann
219 ' FLAG, wenn seit dem letzten NEW eine
220 ' Programmzeile eingegeben oder
221 ' eine Variable deklariert worden ist.
222 REM check
223 REM   push af
224 REM   push hl
225 REM   ld   a,(flag)
226 REM   and  a
227 REM   jr   z,ex
228 REM   ld   hl,(prgend)
229 REM   ld   de,(basend)
230 REM   rst  20h
231 REM   jr   z,ex
232 REM   xor  a
233 REM   ld   (flag),a
234 REM   jr   ex
235 ' Die folgenden paar Befehle bilden
236 ' den PLAYOFF-Befehl. Nachdem das
237 ' PLAY erkannt wurde, ueberprueft das
238 ' Programm nun auf ein OFF. Ist keines
239 ' vorhanden, springt das Programm zu
240 ' EX zurueck, ansonsten wird die Rou-
241 ' tine benuetzt, die auch von STOP
242 ' verwendet wird.
243 REM play
244 REM   rst  10h
245 REM   cp   ebh  ;BASIC-Code fuer OFF
246 REM   jp   nz,ex
247 ' Die drei folgenden POP BC dienen
248 ' dazu, den Stapel so herzustellen,
249 ' wie er bei einer Interpreter-eigenen
250 ' Routine waere.
251 REM   pop  bc
252 REM   pop  bc
253 REM   pop  bc
254 REM   push hl
255 REM   call clrpsg ;Loesche PSG-Reg.
256 REM   pop  hl
257 REM   rst  10h
258 REM   ret
259 ' Die folgenden Zeilen beinhalten die
260 ' Anweisungen um das Wort ECLAIM aus
261 ' der Zeile zu erkennen. Schon bei
262 ' der kleinsten Abweichung kehrt das
263 ' Programm zurueck.
264 REM RE
265 REM   rst  10h
266 REM   cp   "E"
267 REM   jp   nz,ex
268 REM   rst  10h
269 REM   cp   "C"
270 REM   jp   nz,ex
271 REM   rst  10h
272 REM   cp   "L"
273 REM   jp   nz,ex
274 REM   rst  10h
275 REM   cp   "A"
276 REM   jp   nz,ex
277 REM   rst  10h
278 REM   cp   "I"
279 REM   jp   nz,ex
280 REM   rst  10h
281 REM   cp   "M"
282 REM   jp   nz,ex
283 REM   rst  10h
284 REM   pop  bc
285 REM   pop  bc
286 REM   pop  bc
287 ' Die folgenden zwei Zeilen
288 ' retten den Akkumulator und den
289 ' Programmzeiger.
290 REM   push af
291 REM   push hl
292 REM   di
293 ' In den Zeilen 297 bis 322 wird
294 ' der Programmspeicher wieder restau-
295 ' riert, wenn a) das FLAG=1 und
296 ' b) der Programmspeicher leer ist.
297 REM   ld   a,(flag)
298 REM   and  a
299 REM   jp   z,error
300 REM   ld   hl,(basbeg)
301 REM   ld   a,(hl)
302 REM   inc  hl
303 REM   or   (hl)
304 REM   ld   a,0
305 REM   ld   (flag),a
306 REM   jp   nz,error
307 REM   ld   hl,(prgend)
308 REM   ld   de,(basend)
309 REM   rst  20h
310 REM   jp   nz,error
311 REM   ld   hl,(basbeg)
312 REM   ld   de,(frstln)
313 REM   ld   (hl),e
314 REM   inc  hl
315 REM   ld   (hl),d
316 REM   ld   hl,(endprg)
317 REM   ld   (prgend),hl
318 REM   ld   hl,(endvar)
319 REM   ld   (varend),hl
320 REM   ld   hl,(endbas)
321 REM   ld   (basend),hl
322 REM   jp   ex
323 REM error
324 REM   pop  hl
325 REM   xor  a
326 REM   ld   (flag),a
327 REM   pop  af
328 REM   jp   illegl
329 ' In der Zeile 331 erfolgt der Ein-
330 ' sprung sowohl fuer XSWITCH als auch fu
er XCOPY.
331 REM XS
332 REM   rst  10h
333 REM   cp   c9h
334 REM   jr   z,sw
335 REM   cp   d6h
336 REM   jp   nz,ex
337 REM sw
338 REM   pop  bc
339 REM   pop  bc
340 REM   pop  bc
341 REM la0
342 ' In den Zeilen 352 bis 357 wird das
343 ' Carryflag dem Befehl XSWITCH ange-
344 ' passt, und zwar bedeutet:
345 ' Carry=0:STOP und Carry=1:kein STOP
346 REM   cp   d6h
347 REM   push af
348 REM   rst  10h
349 REM   pop  af
350 REM   jr   z,la1
351 REM   ld   a,(hl)
352 REM la1
353 REM   cp   90h
354 REM   scf
355 REM   jr   nz,la2
356 REM   rst  10h
357 REM   xor  a
358 ' Die Routine ab LA2 rettet den Pro-
359 ' grammzeiger, den Stapelzeiger und
360 ' stellt fest, welche Bank angewaehlt
361 ' werden kann. Dann wird durch EX AF,AF'
362 ' der Inhalt von Register 15/PSG ge-
363 ' rettet, um damit immer auf das ROM
364 ' umschalten zu koennen.
365 REM la2
```

```
366 REM   push hl
367 REM   ld   (stkptr),sp
368 REM   push af
369 REM   di
370 REM   ld   a,15
371 REM   out  (88h),a
372 REM   in   a,(90h)
373 REM   ld   c,fdh
374 REM   bit  2,a
375 REM   jr   nz,la3
376 REM   ld   c,f7h
377 REM la3
378 REM   and  c
379 REM   ld   c,a
380 REM   in   a,(90h)
381 REM   ex   af,af'
382 ' Eine Tabelle im ROM ermoeglicht dem
383 ' Interpreter, auf die zweite Bank
384 ' obere Haelfte umzuschalten. Sie
385 ' enthaelt die Adressen der wich-
386 ' tigsten Systemvariablen. LA5 be-
387 ' sorgt dieses Umkopieren.
388 REM   ld   hl,sysvar
389 REM   jr   la4
390 REM la5
391 REM   ld   a,(de)
392 REM   res  7,d
393 REM   ld   (de),a
394 REM   set  7,d
395 REM   inc  de
396 REM   djnz la5
397 REM   ex   af,af'
398 REM   out  (8ch),a
399 REM   ex   af,af'
400 REM la4
401 ' Die Tabelle besteht aus 17 Ein-
402 ' traegen zu je drei Bytes.
403 ' Die zwei ersten Bytes geben die
404 ' Startadresse des Kopiervorgangs an,
405 ' und das dritte Byte die Anzahl der
406 ' zu kopierenden Bytes.
407 REM   ld   e,(hl)
408 REM   inc  hl
409 REM   ld   d,(hl)
410 REM   inc  hl
411 REM   ld   b,(hl)
412 REM   inc  hl
413 REM   ld   a,c
414 REM   out  (8ch),a
415 REM   ld   a,d
416 REM   or   e
417 REM   jr   nz,la5
418 ' Der Kopiervorgang ist abgeschlossen,
419 ' nun wird der Akkumulator vom Stapel
420 ' geholt und mittels des Carryflags
421 ' entschieden, ob STOP oder nicht.
422 REM   pop  af
423 REM   jr   c,la6
424 REM   ld   a,3
425 REM   ld   (stpkey),a
426 ' LA6 und LA7 kopiert bzw. ver-
427 ' tauscht die beiden Seiten.
428 REM la6
429 REM   ld   bc,len
430 REM   ld   hl,top
431 REM   res  7,h
432 REM   ld   de,top
433 REM   cp   d6h   ;COPY ?
434 REM   jr   nz,la7
435 REM   ex   de,hl
436 REM   lddr       ;LDDR kopiert kom-
437 REM   jr   la8   ;plett hinunter
438 REM la7          ;LA7 vertauscht
439 REM   ld   a,(de);solange BC<>0
440 REM   ldd
441 REM   ld   (bc),a
442 REM   jp   pe,la7
443 ' LA8 holt den Stapelzeiger wieder
444 ' aus seinem Versteck, holt den
445 ' Programmzeiger vom Stapel und
446 ' schaltet dann auf die ROM-Bank um.
447 REM la8
448 REM   ld   sp,(stkptr)
449 REM   ex   af,af'
450 REM   out  (8ch),a
451 REM   call 0050h
452 REM   pop  hl
453 REM   ei
454 REM   ret
```

```
455 ' Die folgenden Routine fuehrt die
456 ' Erweiterung herbei, loescht den
457 ' den Programmspeicher, kopiert ihn
458 ' in eine andere Bank um, erzeugt eine
459 ' kleine Meldung, legt die Adresse
460 ' vom Direktmodus auf den Stapel und
461 ' gibt dann mittels ROM-Routine ein
462 ' "..... Bytes free" von sich.
463 REM   org 8200h
464 REM   di
465 REM   ld   hl,ext
466 REM   ld   (hl),c3h
467 REM   ld   hl,neubef
468 REM   ld   (ext+1),hl
469 REM   ld   hl,grenze+1
470 REM   ld   (basbeg),hl
471 REM   dec  hl
472 REM   ld   (lowmem),hl
473 REM   call newt2
474 REM   ld   hl,(lowmem)
475 REM   xor  a
476 REM   ld   (flag),a
477 REM   ld   (hl),a
478 REM   inc  hl
479 REM   ld   (basbeg),hl
480 REM   call newt1
481 REM   ld   hl,msg
482 REM   call print
483 REM   ld   hl,init
484 REM   call print
485 REM   ld   a,d6h
486 REM   call la0
487 REM   call crlf
488 REM   ld   hl,9afh
489 REM   push hl
490 REM   scf
491 REM   push af
492 REM   jp   fremsg
493 REM msg
494 REM   defm "Extension Version 1.2  1985"
495 REM   defb 07,10,13,0
496 REM   end
497 ' Das war die ganze Hexerei, die Pro-
498 ' gramme brauchen nicht viel mehr als
499 ' 384 Bytes, die dafuer aber die
500 ' Speicherkapazitaet von weiteren
501 ' 29000 Bytes bringen.
```

****



```
10 CLS:INPUT"DICHTE";D:A=D:B=D
20 COLOR 15,0,0:SCREEN1
30 CIRCLE(128,96),91,15,,,A/D
40 CIRCLE(128,96),91,15,,,D/B
50 A=A-C:B=B-C:C=C+.2
60 IFA<0THEN70ELSEGOTO30
70 GOTO70'by R. B.
```

```
*****************************************
*                                       *
*            TÜRME VON HANOI            *
*                                       *
*****************************************
```

Schon die Chinesen kannten das Problem. Man hat einen Stapel von Scheiben verschiedener Größe auf einem Platz Nummer Eins. Die Scheiben sind der Größe nach geordnet, die kleinste oben. Nun soll man in möglichst wenigen Zügen die einzelnen Scheiben der Größe nach sortiert auf einen Platz Nummer Drei bringen. Zu Hilfe darf der Bediener Platz Nummer Zwei nehmen. Es ist immer nur der Transport einer Scheibe erlaubt. Wenn zwei Scheiben so übereinanderliegen, daß die größere über der kleineren liegt, dann hagelt es Schlechtpunkte.

Gefragt ist die Variante, welche die wenigsten Züge braucht. Natürlich hat unser Computer die Möglichkeit mit den wenigsten Zügen schon im vorhinein berechnet. Wir können dann versuchen, die Mindestzahl zu erreichen. Lustigerweise ist die Berechnung der Mindestanzahl von Zügen ganz leicht mit der Formel 2^X-1 zu finden. X ist die Anzahl der Scheiben.

Sehen wir uns nun kurz den Aufbau des Programms an. Den Programmteil ab Zeile 1940 können wir vernachlässigen, er ist programmtechnisch nicht sehr interessant. Hier wird nur die Spielerklärung erzeugt.

Interessanter wird es dann beim eigentlichen Spiel. In den Zeilen 1110 bis 1150 erkennen wir eine Formel, mit der wir leicht Text auch vertikal schreiben können. Ein String wird mit einer Schleife in einzelne Zeichen zerlegt und jedes Zeichen einzeln auf dem Bildschirm mittels LOCATE positioniert.

Danach wird die Anzahl der Mindestversuche berechnet. Nun muß der Computer nur noch dafür sorgen, daß die Eingaben des Bedieners richtig interpretiert und angezeigt werden.

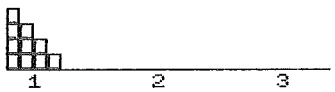
In den Zeilen 1340-1410 werden die Scheiben erzeugt. Was wir dann als Scheiben sehen, das sind in Wirklichkeit Strings (S$ dimensioniert).

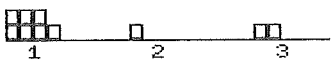
Die Zeilen 1420-1450 bilden die Scheiben am Bildschirm ab. Dies ist lediglich eine Schleife. Wichtiger ist die Routine zum Scheibentransport, die anschließt. Nach der Eingabe, von welchen Stapel wohin transportiert werden soll, wird geprüft, ob ein Fehler mit dieser Entscheidung erzeugt wurde. Ist dies der Fall, wird eine Fehlervariable um Eins erhöht.

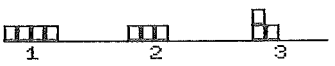
Ab Zeile 1570 wird dann der eigentliche Scheibentransport im Speicher durchgeführt. Dies wird gemacht, indem mit den Variablen hantiert wird, die die Anzahl der Türme pro Platz anzeigen. Ist man damit fertig, wird die Länge der übertragenen Scheibe bestimmt. Zu guter Letzt muß das Ganze nämlich noch angezeigt werden. Um Zeit und Speicher zu sparen, wird hier allerdings nicht der ganze Bildschirm neu aufgebaut, sondern nur der Transport der Scheibe nachgeahmt. Natürlich folgen Anzeige der bisher benötigten Versuche und der dabei erzeugten Fehler. Die Ausgabe erfolgt in den Zeilen 1840 bis 1880.

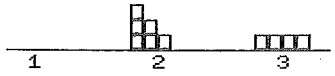
Die Zeilen 1880 bis 1920 sorgen dafür, daß der Computer erkennt, ob alle Scheiben auf ihrem richtigen Platz sind. Wenn dem so ist, wird gefragt, ob man noch einmal spielen möchte, oder ob einem dieses Spiel gereicht hat. Je nach dem, wie die Entscheidung ausgefallen, arbeitet der Computer das Programm noch einmal ab, oder er quittiert den Dienst.

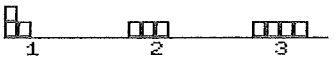
Damit sich unsere Leser ein Bild von den Vorgängen am Bildschirm machen können, drucken wir hier ein paar Hardcopys des Fernsehschirms in verschiedenen Phasen des Spieles aus.











SOFTKEYS - Änderungen leichtgemacht

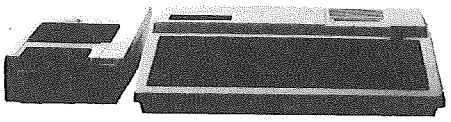
Nun haben wir von unserem jüngsten Clubmitglied auch ein Programm bekommen. Er hat sich sein Gehirn bei den Funktionstasten zermartert und ein kleines Programm geschrieben, mit dem man leicht diese nützlichen "Soft-Keys" ändern kann:

```
10 CLS
20 PRINT CHR$(27)+"x5"
30 PRINT TAB(3);"Aenderung der Funktionstast
en"
40 PRINT TAB(9);"von Thomas Moerth"
50 PRINT TAB(11);"Wohlsdorf 23"
60 PRINT TAB(12);"12.01.1985"
70 FOR Z=1 TO 3000:NEXT Z
80 CLS
90 PRINT "Welche Funktionstaste moechten Sie
":PRINT "aendern F";
100 INPUT A:IF A<1 OR A>10 THEN 90
110 CLS
120 PRINT TAB(10);"Aenderung von F"A
130 FOR Z=1 TO 1500:NEXT Z
140 CLS
150 PRINT "Soll es ein ENTER-Befehl werden";
160 INPUT B$
170 CLS
180 PRINT "Welchen Namen soll F"A"tragen?":P
RINT "Max. 7 Buchstaben";
190 INPUT C$
200 IF B$="JA" OR B$="ja" THEN 220
210 IF B$="NEIN" OR B$="nein" THEN 230
220 KEYA,C$+CHR$(13):GOTO240
230 KEYA,C$:GOTO240
240 CLS
250 PRINT "Moechten Sie noch eine Funktionst
aste":PRINT "aendern";
260 INPUT D$
270 CLS
280 IF D$="JA" OR D$="ja" THEN 90
290 PRINT CHR$(27)+"y5"
300 END
```

```
1000 ''''''''''''''''''''''''''''''''''
1010 '  T U E R M E  V O N  H A N O I  '
1020 ''''''''''''''''''''''''''''''''''
1030 'Copyright C.Gaganas & J.Gaganas  '
1040 '       40 column version         '
1050 ''''''''''''''''''''''''''''''''''
1060 CLS:SCREEN 0,0
1070 GOSUB 1940 ' Spielanleitung
1080 '
1090 DEFINT A-Z
1100 DIM S$(3,7),PT(3),TF(3)
1110 TXT$="TUERME VON HANOI"
1120 ' Vertikal Text
1130 FOR I=1 TO LEN(TXT$)
1140 LOCATE 36,I+2:PRINT MID$(TXT$,I,1)
1150 NEXT I
1160 ' Stack pointers  zu 0 setzen
1170 FOR K=1 TO 3
1180 PT(K)=0:TF(K)=0
1190 NEXT K
1200 LOCATE 1,22:PRINT "Sch?";
1210 LOCATE 10,22:PRINT "Fehl:";
1220 LOCATE 19,22:PRINT "Vers:";
1230 ' Eingabe der Anzahl der Scheiben
1240 LOCATE 7,22:N$=INPUT$(1):N=VAL(N$)
1250 IF N>7 OR N<3 THEN GOTO 1240 ELSE IF N>
0 THEN LOCATE 7,22:PRINT USING "#";N
1254 NV$="":VV$=""
1255 NV$="MINIMUM VERSUCHE"+CHR$(215)
1256 NV=2^N-1:VV$=STR$(NV)
1257 FOR I=1 TO LEN(NV$+VV$)
1258 LOCATE 38,I:PRINT MID$(NV$+VV$,I,1)
1259 NEXT I
1260 C1$="_":C2$=" _______"
1270 FOR I=1 TO 21 'Positionen Zeichnen
1280 LOCATE 1,I
1290 IF I=7 THEN PRINT C1$+" 1"+C2$; ELSE IF
 I=14 THEN PRINT C1$+" 2"+C2$; ELSE IF I=21
THEN PRINT C1$+" 3"+C2$;ELSE PRINT CHR$(197)
;
1300 NEXT I
1310 TF(1)=8-N:TF(2)=0:TF(3)=0
1320 PT(1)=7-N:PT(2)=7:PT(3)=7
1325 NS=NS+1:IF NS>1 THEN FOR I=0 TO 3: FOR
K=0 TO 7:S$(I,K)="":NEXT K: NEXT I
1330 K=N:PS=7
1340 FOR I=1 TO N 'Stapel von Scheiben
1350 FOR L=1 TO K
1360 S$(1,PS)=S$(1,PS)+CHR$(175)+CHR$(175)
1370 S$(0,PS)=S$(0,PS)+CHR$(32)+CHR$(32)
1380 NEXT L
1390 K=K-1
1400 PS=PS-1
1410 NEXT I
1420 ' Stapel auf Bildschirm printen
1430 FOR K=1 TO 7
1440 IF LEN(S$(1,K))<>0 THEN LOCATE 12,K:PRI
NT S$(1,K)
1450 NEXT K
1460 ' Abfrage fuer Scheibentransport
1470 LOCATE 28,22:PRINT "Von:";
1480 LOCATE 34,22:PRINT "Zu:";
1490 FEHL=0:VERS=0
1500 LOCATE 32,22:A$=INPUT$(1):V=VAL(A$):PRI
NT USING "#";V
1510 IF V<1 OR V>3 THEN GOTO 1500
1520 LOCATE 37,22:B$=INPUT$(1):Z=VAL(B$):PRI
NT USING "#";Z
1530 IF Z < 1 OR Z > 3 THEN GOTO 1520
1540 IF V=Z THEN FEHL=FEHL+1:VERS=VERS+1:LOC
ATE 15,22:PRINT USING "###";FEHL:LOCATE 24,2
2:PRINT USING "###";VERS:GOTO 1500
1550 VERS=VERS+1
1560 IF TF(V)=0 THEN FEHL=FEHL+1:VERS=VERS+1
:LOCATE 15,22:PRINT USING "###";FEHL:LOCATE
24,22:PRINT USING "###";VERS:GOTO 1500
1570 IF PT(Z)<7 THEN GOTO 1580 ELSE GOTO 168
0
1580 OBEN=LEN(S$(V,TF(V)))
1590 UNTEN=LEN(S$(Z,PT(Z)+1))
1600 IF OBEN>UNTEN THEN FEHL=FEHL+1:LOCATE 1
5,22:PRINT USING "###";FEHL:GOTO 1500
1610 S$(Z,PT(Z))=S$(V,TF(V))
1620 S$(V,TF(V))=""
1630 D=PT(Z):E=PT(V)+1:A=LEN(S$(Z,PT(Z)))
1640 '
1650 TF(V)=TF(V)+1
1660 IF TF(V)>7 THEN TF(V)=0:PT(V)=7 ELSE PT
(V)=PT(V)+1
1670 PT(Z)=PT(Z)-1:TF(Z)=TF(Z)-1:GOTO 1820
1680 S$(Z,PT(Z))=S$(V,TF(V))
1690 D=PT(Z):E=PT(V)+1
1700 S$(V,TF(V))=""
1710 A=LEN(S$(Z,PT(Z)))
1720 '
1730 IF TF(Z)=0 THEN GOTO 1740 ELSE GOTO 182
0
1740 TF(Z)=7
1750 PT(Z)=TF(Z)-1
1760 TF(V)=TF(V)+1
1770 IF TF(V)>7 THEN TF(V)=0:PT(V)=7:GOTO 18
20 ELSE PT(V)=PT(V)+1:GOTO 1820
1780 TF(V)=0
1790 PT(V)=7
1800 TF(Z)=TF(Z)-1
1810 PT(Z)=PT(Z)-1
1820 IF Z=1 THEN GRT=7 ELSE IF Z=2 THEN GRT=
14 ELSE IF Z=3 THEN GRT=21
1830 IF V=1 THEN FRM=7 ELSE IF V=2 THEN FRM=
14 ELSE IF V=3 THEN FRM=21
1840 ' Scheiben bewegen
1850 LOCATE 12,GRT-7+D:PRINT S$(Z,D);
1860 LOCATE 12,FRM-7+E:PRINT S$(0,PT(V));
1870 LOCATE 15,22:PRINT USING "###";FEHL;
1880 LOCATE 24,22:PRINT USING "###";VERS;
1890 IF PT(3)<>7-N THEN GOTO 1500
1900 LOCATE 1,22:PRINT CHR$(21)+CHR$(13);:LO
CATE 1,22:PRINT "Gratulation: Noch ein Spiel
?          ":PRINT " Wenn ja dann druecken Si
e 'J' oder 'j'";
1910 A$=INKEY$:IF A$="" THEN 1910 ELSE IF A$
="j" OR A$="J" THEN CLS:ERASE S$,TF,PT:GOTO
1100 ELSE END
1920 END
1930 ' Spielanleitug
1935 REM left grph + M ergibt ,
1940 PRINT:PRINT:PRINT:PRINT:PRINT
1950 PRINT ",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,":PRINT
1960 PRINT "  Willkommen bei TUERME VON HANO
I  ":PRINT
1970 PRINT ",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,":PRINT
1980 PRINT " Wissen Sie die Spielregeln ?":P
RINT
1990 PRINT " Wenn ja,<RETURN>, ansonsten <SP
ACE> ":PRINT
2000 C$=INKEY$:IF C$="" THEN GOTO 2000 ELSE
IF C$=CHR$(13) THEN CLS:RETURN ELSE IF C$="
" THEN 2010
2010 CLS
2020 PRINT "TUERME VON HANOI Kombinationspie
l."
2030 PRINT ",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,":PRINT
2040 PRINT "Es geht darum, einen Stapel, bes
tehend ":PRINT "aus 3 bis 7  Scheiben,  von
Position 1"
2050 PRINT "nach Position 3 zu bringen."
2060 PRINT:PRINT "Die  Anzahl  der Scheiben,
  zwischen 3":PRINT "und 7, waehlt der Spiel
er."
2070 PRINT:PRINT "Mehr Scheiben bedeuten ein
en  hoeheren":PRINT "Schwierigkeitsgrad."
2080 PRINT:PRINT "Je weniger Versuche Sie  b
rauchen, de-":PRINT "sto groesser ist  Ihr K
ombinationsver-  moegen."
2090 PRINT:PRINT "Versuchen Sie, fehlerfrei
zu bleiben.":PRINT:PRINT:PRINT "Taste fuer m
ehr"
2094 A$=INPUT$(1):CLS
2100 PRINT "Dabei sollten folgende Regeln ":
PRINT "beachtet werden:":PRINT
2110 PRINT CHR$(212);" Die Scheiben duerfen
nur einzeln":PRINT "  transportiert werden."
2120 PRINT CHR$(212);" Eine Scheibe darf nie
 ueber eine":PRINT "  kleinere gestapelt wer
den.":PRINT "  Sonst gibt es Fehlerpunkte."
2130 PRINT CHR$(212);" Als Hilfsablage kann
Position 2":PRINT "  benuetzt werden."
2140 PRINT CHR$(212);" Das Spiel ist beendet
,wenn alle ":PRINT "  Scheiben in Position 3
 gebracht wurden "
2150 PRINT
2160 PRINT "Taste fuer Spielbeginn. Viel Ver
gnuegen."
2170 U$=INPUT$(1):CLS:RETURN
2180 RETURN
```

Die Super-Computer.

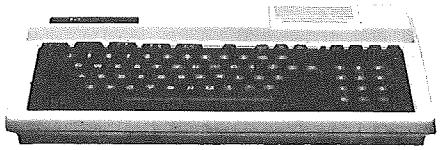
SVI



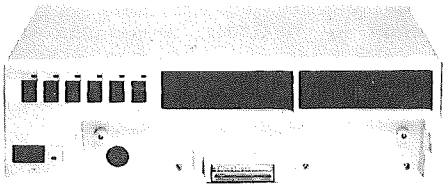
SVI-318 32 K RAM, erweiterbar bis **144 K RAM.** Erweitertes MICROSOFT-BASIC, integrierte Cursorsteuerung **öS 4.990,—**

SVI-904 Datenrecorder, 1800 Baud, Zählwerk, Laufwerksteuerung durch SVI-318 oder 328 inkl. 2 Spielkassetten **öS 990,—**

SVI-318-Set bestehend aus SVI-318 Basisgerät (32 K RAM, MICROSOFT-BASIC), SVI-904 Datenrecorder und Softwarepaket mit 5 Kassetten **öS 5.890,—**



SVI-328 32 K ROM, 80 K RAM, Erweitertes MICROSOFT-BASIC, Schreibmaschinentastatur, 10 Funktionstasten, 10er-Block **öS 6.990,—**



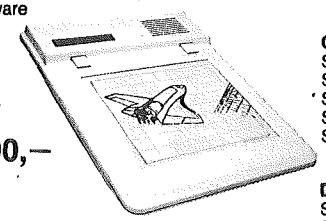
Super-Expander SVI-605, ein eingebautes Diskettenlaufwerk (160 K), Centroniks-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 12.990,—**

Super-Expander SVI-605 A, zwei eingebaute Diskettenlaufwerke (je 160 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 18.990,—**

Super-Expander SVI-605 B, mit Supersoftware-Paket, zwei eingebaute Diskettenlaufwerke (je 320 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 WordStar, Mailmerge, CalcStar ReportStar, DataStar **öS 25.990,—**

SVI-328 Pro Profisystem bestehend aus: Computer SVI-328, Super-Expander SVI-605 A (inkl. WordStar, Mailmerge, CalcStar, DataStar, ReportStar) Betriebssystem CP/M 2.2, 80-Zeichenkarte SVI-806, **öS 32.290,—**

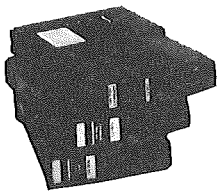
Grafik-Tablett SVI-105, 186 x 158 mm Zeichenfläche, Kassette mit Anwender-Software inkl. **öS 2.090,—**



Erweiterungskarten für SVI-605, A, B

SVI-803 16 K-Speicher-Erweiterung (für SVI-318) **öS 890,—**

SVI-805 RS 232, serielle Schnittstelle **öS 1.790,—**



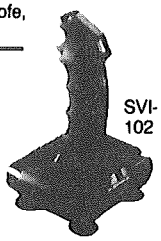
SVI-806 80-Zeichenkarte **öS 3.890,—**

SVI 807 64-K-RAM Speichererweiterung **öS 3.290,—**

Joystick SVI-101, zwei Feuerknöpfe, vier Saugfüße, ergonomischer Handgriff **öS 390,—**

Joystick SVI-102, automatisches Dauerfeuer, zwei Feuerknöpfe, vier Saugfüße **öS 490,—**

(Joystick SVI-101 und SVI-102 auch für Atari und Commodore geeignet)



SVI-102

Die Software

		öS
Kassettensoftware		
SVI-K 110	Einführung in das SVI-Basic inkl. 40seitigem Handbuch	390,—
SVI-K 115	SVI-Dateiverwaltung	290,—
SVI-K 122	SVI-Text	390,—
SVI-K 129	SVI-Termin	290,—
SVI-K 146	Disassembler	590,—
SVI-K 147	Maschinen-Code-Monitor	590,—
SVI-K 148	SVI-Spritegenerator	290,—
SVI-K 149	SVI-Zeichengenerator	290,—
SVI-K 179	Old-Mac-Farmer	390,—
SVI-K 180	Tetra Horror	390,—
SVI-K 181	Tele Bunny	390,—
SVI-K 182	Turboat	390,—
SVI-K 183	SASA	390,—
SVI-K 184	NINJA	390,—
SVI-K 185	Kung-Fu-Master	390,—
SVI-K 188	Armoured Assault	290,—
SVI-K 189	Spectron	290,—
Cartridgesoftware		
SVI-C 220	Sector Alpha	790,—
SVI-C 232	Frantic-Freddy	790,—
SVI-C 236	Music-Mentor	990,—
SVI-C 237	Super-Cross-Force	790,—
SVI-C 291	Flipper-Slipper	790,—
Diskettensoftware		
SVI-D 310	Einf. in das SVI-Basic	590,—
SVI-D 315	SVI-Dateiverwaltung	390,—
SVI-D 322	SVI-Text	590,—
SVI-D 334	SVI-Lager	390,—
SVI-D 348	SVI-Toolkit I (SVI-Spritegenerator u. SVI-Zeichengenerator)	590,—
SVI-D 349	SVI-Toolkit II (Disassembler und Maschinen-Code-Monitor)	1.190,—
SVI-D 359	LISP 80	1.690,—
SVI-D 360	C-Compiler	1.690,—
SVI-D 361	Turbo-PASCAL (Version 2.0)	2.390,—
SVI-D 318	Nevada-FORTRAN (Compiler)	1.390,—
SVI-D 382	Nevada-COBOL (Compiler)	1.390,—
SVI-D 383	Nevada-PILOT (Interpreter)	1.390,—
SVI-D 384	Nevada-EDIT (Editor)	1.390,—
SVI-D 388	Old-Mac-Farmer	390,—
SVI-D 389	Tetra Horror	390,—
SVI-D 390	Tele Bunny	390,—
SVI-D 391	Turboat	390,—
SVI-D 392	SASA	390,—
SVI-D 393	NINJA	390,—
SVI-D 394	Kung-Fu-Master	390,—

Druckeranschlußkabel SVI-205, 1,5 m, für parallele Schnittstelle **öS 590,—**

Diskettenlaufwerk SVI-905, 160 K, zur Erweiterung des Super-Expanders SVI-605 **öS 6.490,—**

Centronics-Interface SVI-802 mit Kabel 205 zum Anschluß an Mini-Expander SVI-602 **öS 3.080,—**

Preise incl. MWSt.

SVI-FACHBERATUNG UND VERKAUF BEI:

Großhandel, Facheinzelhandel **BASTL-COMPUTER-SYSTEME** **2700 Wr. Neustadt,** Hauptplatz 5 Tel. (0 26 22) 22 7 20 - 59 80 Telex 16 522	**DAHMS-PRAKTIKER-ELEKTRONIK** Pilgramgasse 11 **1050 Wien** Tel. (0 22 2) 54 34 21	**ESV ELEKTROTECHNISCHER SERVICE MBH.** Bayerhamerstraße 19—21 **5020 Salzburg** Tel. (0 66 2) 74 7 51	**SCHILLER MICRO-COM-BOUTIQUE** Fasangasse 21 **1030 Wien** Tel. (0 22 2) 78 35 99, 78 56 61	**TARGET ELECTRONIC** Bergstraße 6 **6900 Bregenz** Tel. (0 55 74) 23 7 18
BYTE COMPUTER Favoritenstraße 20 **1040 Wien** Tel. (0 22 2) 65 73 42 Telex 132 827	**EDV-STUDIO PORSCH** Kinderspittalgasse 13 **1090 Wien** Tel. (0 22 2) 42 63 44	**FEDCON ELEKTRONIK** Ing. Franz Krenn Kanalplatz 86 **9400 Wolfsberg** Tel. (0 43 52) 42 73	**TARGET ELECTRONIC** Reichsstraße 123a **6800 Feldkirch** Tel. (0 55 22) 21 5 29 Telex 52 300	**WEHSNER GMBH. COMPUTER-STUDIO** Paniglgasse 18—20 **1040 Wien** Tel. (0 22 2) 65 88 93, 65 78 08

GENERALVERTRETUNG FÜR ÖSTERREICH: MONACOR ELECTRONIC · VERTRIEBS-GESMBH. · 6800 FELDKIRCH · ☎ (0 55 22) 21 9 89 · TELEX 52 300 tarmo

SVi-Programmecke

SECTOR ALPHA

Wie schon bei Spectron, so muß man auch in "Sector Alpha" außerirdische Eindringlinge "vom Bildschirm entfernen". Doch sonst hat dieses Spiel wenig mit dem erstgenannten gemeinsam.

Bei Sector Alpha befindet man sich in einem bestimmten Abschnitt einer Landschaft und hat ein Visier zur Verfügung, um die UFOs abzuschießen. Man bewegt dabei dieses Visier mit den Cursor-Tasten solange, bis ein Raumschiff angepeilt ist. Danach drückt man die Space-Taste, um eine Rakete zu starten. Diese Rakete erkennt man an einem Lichtpunkt, der eine Zeit lang am Bildschirm sichtbar ist. Entweder, das UFO ist vernichtet, oder der Lichtpunkt verschwindet ohne Folgen. Am Anfang wird man seine liebe Not mit den Tasten haben. Die waagrechte Steuerung wird zwar richtig durch "Links" für eine Bewegung nach links und "Rechts" für eine Bewegung nach rechts erzeugt. Doch in der Senkrechten wird das ganze System unlogisch. Sorgt doch die Cursor-Taste "Hinauf" für eine Bewegung nach unten und umgekehrt. Man kann sich schwer vorstellen, daß dieser Effekt beabsichtigt war.

Den ganzen Sektor plus zwei Nebensektoren sieht man auf einer Leiste oberhalb des eigentlichen Gesichtsfeldes. Im großen unteren Teil werden UFOs, Minen und Landschaft naturgetreu gezeigt. Die drei Sektoren werden nur durch RADAR-Schirme erkenntlich gemacht. Auf dem mittleren Schirm sieht man auch die Positionen des Gesichtsfeldes, aller UFOs und deren Minen.

Das Spiel wird übrigens durch kontinuierliche Bewegung der Landschaft reizvoll gemacht. Selbst Raumstationen finden sich zwischen Bergen und Feldern.

Die UFOs können sich natürlich auch wehren. Sie setzen von Zeit zu Zeit eine Mine in der Luft aus. Diese wächst dann ziemlich schnell, bis sie explodiert. Sieht man sie dann noch am großen Bildschirm, so "fliegt" man ebenso "in die Luft".

Hat man alle Raumschiffe eines Sektors erledigt, dann wechselt man in den nächsten über. Dies wird gemacht, indem man an den Rand des Sektors fährt, und die Cursor-Taste, die man dazu verwendet hat, gedrückt läßt. Nach einer kurzen weiteren Zeit befindet man sich im Nachbarabschnitt.

Ziel des "Sector Alpha" ist es, möglichst viele UFOs abzuschießen. Trotz dieser einfachen Problemstellung ist es am Anfang schwer, mit diesem Spiel umzugehen. Die Raumschiffe fliegen schnell und werfen oft Minen ab. Dies soll jedoch keine schlechte Kritik sein. Es ist immer besser, man hat lange an einem Spiel zu kämpfen, als man schafft nach zwei Wochen schon über eine Million Punkte (solche "Games" gibt es auch!). Erleichtert wird das Computer-Spiel nur durch eine Art Zielautomatik. Befindet sich der Eindringling ganz in der Nähe des Visiers, so stellt sich dieses automatisch auf das UFO ein und folgt ihm.

Das Spiel wurde von Spectravideo erzeugt und ist unter anderem im Computer-Studio der Wehsner Ges.m.b.H. erhältlich. Man kann es derzeit nur in Modulform kaufen. Daher kostet es 790 Schilling.

AULÖSUNG DES RÄTSELS AUS HEFT 6/84

Nach der Eingabe von RUN und zwei Zahlen sehen wir ein "Syntax-Error in 50". Auf den ersten Blick können wir jedoch nichts Sonderbares erkennen. Erst bei genauerer Begutachtung dieser Zeile bemerken wir, daß IF T=SETHENPRINT... ein unbeabsichtigtes Befehlswort enthält, nämlich SET. Dadurch liest der Computer die Zeile so: IF T=SET HEN.... Da dies nicht zuläßig ist, gibt er einen Syntax-Error aus. Wir müssen daher zwischen SE und T ein Space einfügen. Dann haben wir diese Fehlerquelle beseitigt.

Doch schon beim zweiten RUN und einer Eingabe von ungleichen Zahlen erkennen wir einen Syntax-Error in Zeile 60. Wieder wurde ein Befehlswort in eine Wertzuweisung eingeschmuggelt. D=TORSE enthält TO. Daher können wir zwischen T und OR ein Blank einfügen. Wenn wir jedoch Speicherplatz sparen wollen, dann vertauschen wir ganz einfach die Operatoren. D=SEORT lautet eine der richtigen Anweisungen.

Beim nächsten RUN sehen wir beide Meldungen des Hauptprogramms vor uns. Da die Verknüpfung wohl kaum gleichzeitig nicht erfolgreich und doch erfolgreich sein kann, liegt der Schluß nahe, daß hier ein Fehler im Programm liegt. Wir finden ihn in Zeile 6. Nach der Meldung muß ein Sprung zurück zu Zeile 5 erfolgen. Sonst wird auch die zweite Meldung ausgegeben. Wir fügen in Zeile 6 ein "GOTO 5" ein. Nun sehen wir, daß bei ungleichen Zahlen das Programm einwandfrei arbeitet. Doch es wäre ein Trugschluß zu sagen, das Programm hätte nun keinen Fehler mehr. Wir müssen noch die Gleichheit von Zahlen testen. Hier sehen wir nach den nötigen Eingaben am Bildschirm:

```
Dieses Programm manipuliert Zahlen
? 7
? 7
Diese Zahlen sind gleich, daher nicht ve
rwendbar!:RETURN
 7 OR-verknüpft mit 7 ergibt 7
```

Wir brauchen einerseits ein Anführungszeichen zwischen "!" und ":". Andererseits ändern wir die Meldung so um, daß "verwendbar" als ganzes Wort in einer Zeile steht. Dadurch wurde die Verknüpfung zwar verhindert, doch bekommen wir die Meldung, daß die Verknüpfung erfolgreich war. Dies ändern wir durch Erweitern des ersten RETURN-Befehles mit der Zeilennummer 7.

Zuletzt wird noch in Zeile 7 selber das Hauptprogramm gegen das nachfolgende Unterprogramm abgesichert. Denn sonst würde der Computer vom Hauptprogramm in die Subroutine einsteigen, obwohl kein GOSUB verwendet wurde. Als letzte optische Korrektur richten wir noch die beiden Meldungen, ob die Verknüpfung erfolgreich oder nicht erfolgreich war, auf eine schöne Form.

```
5 GOSUB 10
6 PRINT "Die Verknüpfung zweier Zahlen war
 er-folgreich":GOTO 5
7 PRINT "Die Verknüpfung zweier ungleicher
Zah-len war nicht erfolgreich":GOTO 5
10 REM UNTERPROGRAMM
20 PRINT"Dieses Unterprogramm manipuliert Za
hlen
30 INPUT SE
40 INPUT T
50 IF T=SE THENPRINT "Diese Zahlen sind glei
ch, daher nicht verwendbar!":RETURN7
60 D=T OR SE:PRINTT"OR-verknüpft mit"SE"ergi
bt"D
70 RETURN
```

Was bietet Ihnen der Spectra Video Club Austria?

- regelmäßige Clubabende mit Gelegenheit zum Informationsaustausch
- Möglichkeit zum kostenlosen Arbeiten an SVI-Computern während der Clubtreffen und zum Ausdrucken von Programmlistings
- außerordentliche Clubabende mit Vorträgen über Themen rund um Hard- und Software der Spectravideo-Computer
- kostenloser Bezug der monatlich erscheinenden Clubzeitschrift SVI-JOURNAL
- verbilligte Angebote von Spectravideo-Produkten

Mitgliedsbeitrag: Jahresbeitrag S 500,-
für Schüler, Studenten, Lehrlinge S 250,-

Nähere Informationen beim
Spectra Video Club Austria
c/o Computer-Studio
1040 Wien, Paniglgasse 18-20
Telefon (0222) 65 88 93

IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Rudolf Bolek, Philipp Ott, Rafael Razim, Heinz Schmid, Stephan Traxler, Georg Wolfbauer

Medieninhaber (Verleger): Spectra Video Club Austria, p.A. Computer-Studio, A-1040 Wien, Paniglgasse 18-20, Tel (0222) 65 88 93

Hersteller: HTU-Wirtschaftsbetriebe Ges. m. b. H., 1040 Wien

Herausgeber: Spectra Video Club Austria, p.A. Computer-Studio, A-1040 Wien, Paniglgasse 18-20, Tel. 65 88 93

Erscheinungsweise: monatlich, jeweils zur Monatsmitte, Einzelheft S 15,-

Abonnementpreise:
jährlich S 150,-
halbjährlich S 80,-

Erscheinungsort Wien
Verlagspostamt 1040 Wien