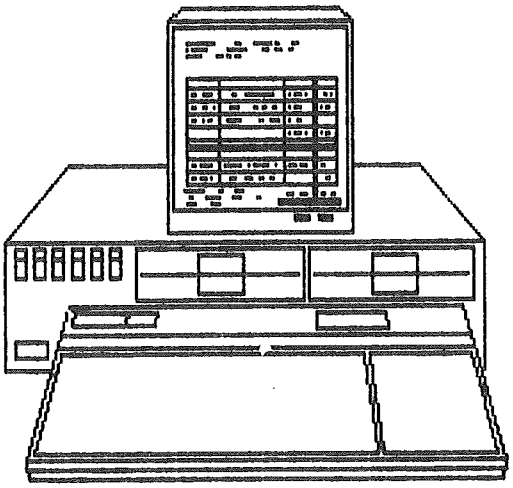
SVI JOURNAL

Die Zeitschrift des Spectra Video Club Austria



Spectra Video Club Austria auf der HE

Hobby Elektronik

Wiener Messepalast

15.-18. Nov. 1984

NOVEMBER 1984

INHALT

Heft 5/84 **S 15.-**

Liebes Clubmitglied!

Wie in jedem Heft haben wir auch in unserer fünften Ausgabe einige Neuigkeiten! Zum Ersten wären hier unsere Rätsel. Wir geben es ja zu! Kreuzworträtsel, in denen Flüsse und Städte in Brasilien und Peru gefragt werden, gehören eigentlich nicht in eine Computerzeitschrift.

Daher haben wir uns entschlossen, unsere Rätselecke neu zu überarbeiten. Nach einer Folge Abwesenheit kommen nun wieder neue "Knobeleien", die aber völlig anders aufgebaut sind. In jedem Heft werden BASIC- oder Maschincode-Programme abgebildet, die fehlerhaft sind. Oft sind es banale, oft aber auch knifflige Fehler, die das Programm gewaltsam stoppen. Nun sollte man herausfinden, was an diesem Programm falsch ist.

Sollten Sie ein Programm haben, daß ganz und gar nicht laufen will, dann schicken Sie uns diese Entwicklung. Ebenso sind Programme willkommen, die einen kniffligen Fehler hatten, der aber schon vom Programmierer gefunden wurde. Die einzigen Bedingungen, die wir stellen, sind folgende: Das Programm sollte genau beschrieben sein und, was in dem Fall eines fehlerhaften Programms besonders wichtig ist, die Gedankengänge des Programmierers geschildert werden. Denn es kann ohne weiteres sein, daß auch Logikfehler in der Entwicklung auftreten! Zuletzt soll das fehlerhafte Programm auch nicht all zu lang sein.

Um noch einmal alle Programme zusammenzufassen, die uns ausgehändigt werden können. Wir nehmen gerne Beschreibungen von langen "professionellen" Entwicklungen an, um diese in unserer Programmecke als Bedienungsanleitung und kleinerem "Testbericht" abzudrucken.

Kleine Programme, die lehrreich sind, oder Programmteile, die in größere Programme eingebaut nützlich sein können, werden als gelistete und dokumentierte Programme verarbeitet. Zuletzt sind nun auch fehlerhafte Entwicklungen willkommen, um unsere Rätselecke zu füllen! (oder wollen Sie wieder brasilianische Flüsse aus dem Atlas suchen?)

Über unsere Aktivitäten auf der "Hobby-Elektronik" informieren Sie diesmal die Seiten in der Mitte unserer Ausgabe. Neben genauer Information, wo wir zu finden sind, geben wir auch bekannt, was nun entgültig von uns auf der Messe verbrochen wird.

Ihr SVi-Journal-Chefredakteur Gerhard Fally!

```
*****************************************
*                                       *
*  Die nächsten Clubabende:             *
*                                       *
*   Mi, dem 21. November  1984, ab 19 Uhr *
*                                       *
*   Für Dezember und die Weihnachtsferien *
*   gibt es eine Sonderregelung:        *
*                                       *
*   Sa, dem  1. Dezember  1984, ab 17 Uhr *
*   Mi, dem 12. Dezember  1984, ab 19 Uhr *
*   Mi, dem  2. Jänner    1985, ab 19 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.                         *
*                                       *
*****************************************
```

```
*****************************************
*                                       *
*    SUBROUTINENBIBLIOTHEK IN PLANUNG !  *
*                                       *
*****************************************
```

In Vorbereitung ist eine Clubbibliothek, welche die von unseren Mitgliedern erstellten Subprogramme verwalten soll, um jedem die Möglichkeit zu geben, auf dieses Programmreservoir zuzugreifen.

Die Idee ist folgende: Viele unserer Mitglieder erstellen eigene Programme. In diesen Programmen werden mehr oder weniger Subroutinen verwendet. Subroutinen, die teilweise auch von anderen verwendet werden könnten, da sie so geschrieben sind, daß man sie mit geringem Aufwand auch für andere Zwecke einsetzen kann. Wie oft ist es Ihnen denn schon passiert, daß Sie bewaffnet mit einem Programm, das Ihnen viel Schweiß und Mühe gekostet hat im Club erscheinen, und dann hier "draufkommen", daß ein anderes Clubmitglied schon ein ähnliches geschrieben hat?

Oder Sie wollen Ihr Programm mit wenigen Tricks so verfeinern, daß es so perfekt wie eine im Handel erhältliche Entwicklung aussieht. Sie wissen, daß es sicher jemanden im Club gibt, der Ihre Probleme in Form von Kurzprogrammen (oder Unterprogrammen) bereits gelöst hat; nur leider wissen Sie nicht, an wen Sie sich wenden sollen!

Bei all diesen oder ähnlichen Problemen soll unsere SUBROUTINENBIBLIOTHEK Abhilfe schaffen. Wir wollen:

- alle Mitglieder auffordern, nützliche Subroutinen oder Programmteile an uns weiterzugeben.
- diese Programme verwalten und ordnen.
- die so entstehende Bibliothek allen Mitgliedern zur Verfügung stellen.

Selbstverständlich wird dieses Wissen nur an Clubmitglieder weitergeleitet, es braucht niemand Angst zu haben, daß seine mühsame erarbeitetes Werk hinter seinem Rücken vermarktet wird.

Das war die Idee, die dem Ganzen zugrunde liegt. Jetzt kommen wir zur Durchführung. Wir brauchen mindestens zwei Leute, die sich auch um diese Bibliothek kümmern. Es soll ja an jedem Clubabend die Möglichkeit bestehen, auf die Bibliothek zuzugreifen. Und - das ist das Wesentliche - jede Bibliothek ist nur so gut, wie ihr Archivar. Es müssen sich also Engagierte melden, die vorhaben, diesen Job einige Zeit zu machen, sonst haben wir in kurzer Zeit den schönsten "Palawatsch" beisammen.

Jetzt kommt unsere Bitte an Sie: Wenn Sie diese Idee gut finden, so unterstützen Sie uns mit nützlichen Programmen. Und wenn Sie diese Idee ausgezeichnet finden, so melden Sie sich am nächsten Clubabend bei uns, damit wir diese Idee in die Tat umsetzen können.

Wichtig ist vielleicht noch, daß die Programme gut dokumentiert werden, damit man gleich weiß, welche Variablen auszutauschen sind, wenn man das Programm einbauen will.

****

ACHTUNG, (NOCH) NICHT-MITGLIEDER!!

Ab Anfang November gilt die Anmeldung für unseren Spectra Video Club Austria auch schon für nächstes Jahr!

Spectravideo-BASIC

Diesmal werden die "normalen" Anweisungen durchgenommen. Neben DEF FN lernen wir auch so wichtige Befehle wie DIM oder DATA kennen. Im USER-Manual sind diese Befehle unter 2.2 angegeben!

Der ursprüngliche Zweck von "CLEAR" ist es, soviel Speicher im Computer zu reservieren, wie für Strings benötigt wird. Die erste Zahl hinter dem Token kennzeichnet die Länge des geschätzten Platzbedarfs von alphanumerischen Variablen. Voreingestellt ist der Wert 200. Wenn man nun zum Beispiel 50 Strings zu je 10 Zeichen braucht, dann muß man mindestens einen Stringspeicher von 500 Zeichen reservieren (zum Beispiel "CLEAR 550"). Erstens können sich manche dieser Strings etwas verlängern, zweitens könnten noch irgendwo im Programm ein paar alphanumerische Variablen vorkommen.

Der zweite Wert gibt das Speicherende an, es ist jedoch nur in Spezialfällen notwendig, diesen Wert zu setzen. Wann immer man einen Speicherplatz braucht, der vor dem BASIC geschützt sein soll, verwendet man diesen Wert. Steht nun zum Beispiel "CLEAR 600, &HF000", so heißt dies, daß 600 Bytes Stringspeicher reserviert werden. Dieser Bereich erstreckt sich von Adresse &HEA00 bis &HEFFF. Würde &HF000 in der "CLEAR"-Anweisung geändert werden, dann würde sich der ganze Stringspeicher dementsprechend ändern.

CLEAR 600, &HE000 ergibt daher den Bereich &HDA00 bis &HE000, und so weiter.

Dies ist die erste Bedeutung, nämlich Speicherplatz zu reservieren. In Ladeprogrammen für Maschincode wird noch eine zweite Anwendung praktiziert. Um Maschincodeprogramme gegen den Stringspeicher abzusichern, werden die Strings unter das MC-Programm abgelegt. Strings "wachsen" nämlich von oben herab (von hohen Adressen zu niedrigen). Wenn man nun das Ende bei &HDFFF anlegt, dann kommt kein String nach &HExxx oder &HFxxx. In diesen Bereichen kann man dann seine MC-Programme ablegen. Es wird übrigens in diesem Fall "CLEAR 200, &HXXXX" verwendet. "&HXXXX" ist die Adresse des Speicherendes. Wieviel Speicher für Strings reserviert wird, ist ja unwichtig, weil mit diesem "CLEAR" nur ein Schutz für MC-Programme angestrebt wird.

Apropos Maschincodeprogramme! Mit den drei Befehlen "DATA", "READ" und "RESTORE" werden unter anderem auch in MC-Ladeprogrammen die Opcodes des MC-Programms bereitgehalten, um sie in den Speicher zu transportieren. Hinter dem "DATA" stehen allgemein irgendwelche Daten, sowohl numerischer als auch alphanumerischer Art, die darauf warten, mit "READ" einmal in einer Variable abgespeichert zu werden.

Es gilt folgendes für die Daten. Prinzipiell brauchen Strings keine Anführungszeichen (Daß numerische Daten keine brauchen, dürfte ja sowieso bekannt sein, Strings haben bis jetzt aber immer Anführungszeichen gebraucht!).

Es gibt hier allerdings Ausnahmefälle: Besteht ein String nur aus Zahlen, oder beinhaltet er einen Beistrich, dann muß er zwischen Anführungszeichen stehen. Im ersten Fall würde der Computer nämlich die Zifferzeichen ohne "Gänsefüßchen" automatisch als numerische Daten ansehen und nie als alphanumerische. Im zweiten Fall glaubt der Computer, wenn der String nicht durch Anführungszeichen eingegrenzt ist, daß der Beistrich in dem String ein Trennbeistrich ist und zwei einzelne Zeichenketten trennt. Aus der einen werden dann zwei alphanumerische Datenansammlungen. In "DATA"-Zeilen trennt der Beistrich nämlich zwei Daten voneinander.

Mit "READ" holt sich der Computer aus den "DATA"-Anweisungen die einzelnen Daten und speichert sie in den Variablen, die hinter dem Befehlswort "READ" angegeben sind (zum Beispiel "READ A$,B$,C). Dabei müssen sich allerdings die Typen der Daten mit den Variablentypen gleichen. Andernfalls schreibt der Computer einen "Syntax-Error" auf den Bildschirm (ACHTUNG!! Der Error wird für die "DATA"-Zeile, in welcher der Wert vorkommt, ausgegeben!! Bei der Fehlersuche ist also unbedingt zu beachten, daß ein "Syntax-Error" in einer "DATA"-Zeile ohne weiteres von einem falschen "READ" herrühren kann!).

Ein Beispiel soll dies verdeutlichen: Man will sieben alphanumerische und sieben numerische Daten abwechselnd in die Variablen A$(I) und B(I) laden. "DIM"-Anweisung braucht man bei sieben Feldern noch nicht!

```
10 FOR I=1TO7
20 READA$(I),B(I)
30 NEXT
40 DATA EMIL,5,ALEXANDER,7,GEORG,11,RUDOLF,3
50 DATA HUGO,6,ANTON,54,GERHARD,4
60 FOR I=1TO7
70 PRINT A$(I);B(I)
80 NEXT
```

An diesem Programm erkennt man, daß erstens die Positionierung der "DATA"-Zeilen völlig willkürlich erfolgen kann, denn auch wenn vor dem ersten "DATA" ein "READ" kommt, sucht sich der Computer die richtigen "DATA"-Zeilen. Zweitens behindert "DATA" den Programmablauf nicht. Trifft der Computer während des Ablaufs auf eine "DATA"-Zeile, dann überspringt er sie einfach.

Für den Computer sind alle "DATA"-Zeilen zusammengenommen ein "kontinuierlicher Daten-Strom". Das heißt, es ist völlig egal, ob man zum Beispiel fünf "DATA"-Zeilen für die benötigten Daten verwendet oder vier oder sechs. Allerdings gibt es Ausnahmesituationen, wo der Programmierer aus verschiedensten Gründen eine der Möglichkeiten verwenden muß. Einen dieser Fälle werden wir am Ende von "RESTORE" kennenlernen.

Kommen wir nun zum letzten Befehl der "Dreier-Gruppe". "RESTORE" stellt den "DATA"-Zeiger auf eine Zeile ein. Bei jedem "READ" holt sich der Computer nämlich, wie oben schon erwähnt, einen Wert aus einer "DATA"-Zeile. Welcher Wert das ist, bestimmt ein interner Zeiger. Und diesen Zeiger verstellt "RESTORE". "RESTORE 40" sorgt dafür, daß das nächste "READ" die "DATA"-Werte ab der BASIC-Zeile 40 verarbeitet. Hat Zeile 40 keine Daten, so sucht sich der Interpreter die nächste "DATA"-Zeile nach 40.

Mit "RESTORE" stellt man normalerweise in einem Programm den Zeiger immer dann auf eine "DATA"-Gruppe, wenn man sicher sein will, daß keine Daten von einer vorherigen nicht zu dieser Gruppe gehörigen Datensammlung verwendet werden. Es kann ja passieren, daß ein Programm aus verschiedensten Gründen absichtlich nicht alle "DATA"-Werte in den Computer holen soll. Dann weiß der Programmierer natürlich nicht, wieviele Werte tatsächlich verarbeitet wurden. Um dann einem eventuellen Fehler vorzubeugen, geht man auf Nummer sicher und stellt den Zeiger mit dem

oben genannten Befehl auf jeden Fall auf die neue Gruppe. Und in diesem Fall muß der Anfang der neuen Gruppe der Anfang einer "DATA"-Zeile sein.

Wir hatten zwar kurz zuvor gehört, daß alle "DATA"-Zeilen für den Computer ein kontinuierlicher Daten-Strom sind und es eigentlich egal ist, wieviele Zeilen verwendet werden. In diesem Fall ist es nicht egal, denn hier können wir mit "RESTORE" nur einen Zeilenanfang ansprechen. Deshalb müssen wir den Anfang dieser neuen Gruppe an den Anfang einer neuen "DATA"-Zeile stellen.

Man kann übrigens auch "RESTORE" ohne eine Zeilennummer verwenden. Dann wird der obligate Zeiger, wie kann es anders sein, auf die erste "DATA"-Anweisung gestellt. Dies kann dann recht nützlich sein, wenn man zum Beispiel alle Daten mehrmals braucht. Wenn wir unser Programm von vorhin in eine Schleife stecken, dann müssen wir in Zeile 90 ein "RESTORE" anschreiben.

Zuletzt noch die wichtigsten Fehlermeldungen: Kommt der Computer zu einem "READ", und es sind keine Daten mehr vorhanden, dann kommt eine Fehlermeldung namens "OUT OF DATA". Wenn "READ" einer Variable einen Datenwert anderen Typs zuweisen will, wehrt sich der Computer mit Syntax-Error (zum Beispiel: "DATA ha:READA" funktioniert nicht, weil A eine numerische Variable, "ha" aber ein alphanumerischer Datenwert ist.).

Es soll immer wieder Leute geben, die Formeln und Funktionsterme in der Form von "2*X^4+25*SQR(TAN(X-45)+COS(78)" erstellen. Um aber diese Funktionsterme nur einmal aufschreiben zu müssen, wenn man ihn öfters verwenden will, kann man ihn mit "DEF FN" definieren. Die Syntax dieses Befehls sieht zum Beispiel so aus:

```
DEF FNX (A,B) = 2*A+5*(SQR(B))
```

Anschließend wird dieser Term jedesmal mit "FNX (A,B)" aufgerufen (Dies kann in der Form "A=FNX(A,B)" oder "?FNX(A,B)" geschehen). A und B können durch andere Variablen ersetzt werden. Der Computer ersetzt dann die alten Buchstaben in der Formel durch die neuen. Auch konkrete Werte können eingesetzt werden!

Kommen wir nun zu den einzelnen Teilen der Anweisung. "DEF" bezeichnet "Definiere folgende Funktion". "FN" ist der erste Teil des Funktionsnamens, unter dem die Funktion später aufgerufen wird. Er muß sowohl bei der Definition als auch beim Aufruf immer vorhanden sein und bleibt immer gleich. Der zweite Teil des Namens (hier X) darf frei gewählt werden. In der Klammer folgen alle Variablen, die im Term gebraucht werden (als Term bezeichnet man einen sinnvollen mathematischen Ausdruck, also auch jede Formel). Diese Variablen können, wie oben erwähnt, beim Aufruf durch andere Namen ersetzt werden. Nach dieser Klammer wird der Funktionsterm aufgeschrieben. Das nachfolgende Programm ruft die Funktion mit verschiedenen Variablen auf.

```
5 REM KREISFUNKTION
10 DEF FNKR (X,R)=R^2-X^2
20 INPUT A,B
30 PRINT FNKR (A,B)
40 FORG=0TO 6.28 STEP.02
50 E=FNKR(G,5)
60 PRINT E
70 NEXT
```

Keine Untergruppe von "DEF FN" sind "DEFINT", "DEFDBL", "DEFSNG" und "DEFSTR"! Diese vier "DEF's" definieren Variablen auf eine gewisse Type. "INT" steht für INTeger (ganzzahlig), DBL für DouBLe (doppeltgenau), SNG für SiNGle (einfachgenau) und STR für STRing (alphanumerisch). Hinter dem Befehlswort kommen einige Buchstaben. Alle die Variablen, die mit diesem Buchstaben beginnen, werden auf diese Type eingestellt (zum Beispiel "DEFINT A,B": AA, AB, A1, A3, BA,.. sind alle ganzzahlig).

Wenn zum Beispiel "DEFSNG AB" steht, dann werden alle Variablen, die mit AB beginnen, einfach genau charakterisiert. Die Liste der Variablen kann beliebig lang sein. Ändern kann man die Type nur mit einem neuerlichen "DEF..." oder einem angehängten Typenzeichen (!,#,%,$).

Eine weitere nützliche Einrichtung ist "DIM". Wie der Name schon sagt, kann man mit dieser Anweisung Variablenfelder dimensionieren. Was sind Variablenfelder. Dazu können wir ein Gedankenexperiment machen.

Eine Variable stellt man sich normalerweise als "Kübel" oder Lade vor. Nun kann man mehrere Laden stapeln oder nebeneinander schlichten. Wenn man nun einen Schrank aus diesen Laden herstellt, dann hat man diese vielen einzelnen Laden in ein großes Ladenfeld zusammengefaßt. Und genau das tut "DIM". Unter gleichem Variablennamen gibt es mehrere "Behälter", zum Beispiel Variable A(1), A(2), A(3) und so weiter.

Es gibt hier allerdings Unterscheidungen. Zum Ersten sind hier die einfach dimensionierten Variablen. Man kann sich diese Reservierung wie eine Ladenreihe vorstellen, getreu Bild 1. Indem man die Zahl ändert, kann man wahlweise auf alle Laden zugreifen.

L1	L2	L3	L4

Bild 1: einfach dimensionierte Variablen

Ebenso funktioniert es bei den zweifach dimensionierten Variablen, die man sich als Fläche vorstellen kann. Hier kann man sowohl die Zahl hinter der "Lade", als auch die hinter der "Reihe" ändern. Im übrigen kann man sich das folgendermaßen merken. Einfach dimensionierte Variablen stellt man sich in der ersten Dimension vor (eine Linie). Zweifache Variablen sind als zweite Dimension (=Ebene) anzusehen und zu diesem Vergleich paßt auch noch das dreifach dimensionierte Feld, welches mit der dritten Dimension (=Raum) verglichen werden kann. In Bild zwei und drei sind die zweifachen und dreifachen Dimensionen bildlich dargestellt.

L1R1	L2R1	L3R1	L4R1
L1R2	L2R2	L3R2	L4R2
L1R3	L2R3	L3R3	L4R3
L1R4	L2R4	L3R4	L4R4
L1R5	L2R5	L3R5	L4R5

Bild 2: zweifach dimensionierte Variablen

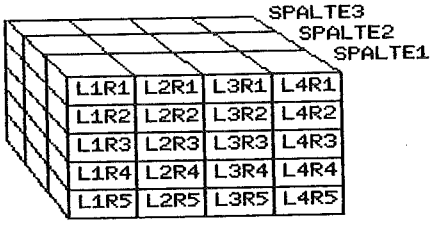


Bild 3: dreifach dimensionierte Variablen

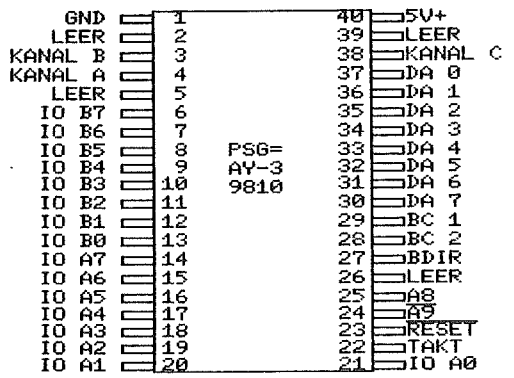
Der Tongeneratorbaustein AY-3-8910 ist nicht nur für die Musik- und Geräuschproduktion zuständig. Er besorgt auch die Abfrage der Joysticks und erledigt das Umschalten der Banken.

Die SVI-Computer enthalten den programmierbaren Tongenerator AY-3-8910 (PSG = Programmable Sound Generator) von General Instruments. Der hochintegrierte Schaltkreis ermöglicht die Herstellung vielfacher Klangeffekte. Nach der Initialisierung läuft der PSG ohne Steuerung durch den Prozessor weiter, wodurch sich die Möglichkeit zur Musik- und Geräuschuntermalung für Programme ergibt.

Der AY-3-8910 verfügt über drei unabhängig voneinander programmierbare Tongeneratoren (drei Kanäle A, B und C), einen Rauschgenerator, drei Mischer und einen Hüllkurvengenerator. Drei D/A-Converter produzieren die Analogsignale der drei Kanäle. Das PSG-Blockdiagramm befindet sich auf der folgenden Seite.

Der PSG ist ein registerorientiertes System. Seine Funktionsweise wird durch die Eingabe von Steuerwerten in die 16 Register programmiert, basierend auf "memory-mapped I/O", das heißt die Register werden wie Speicherzellen adressiert(Register-Array). Alle Register sind Schreib-/Leseregister und können daher auch abgefragt werden. Vom BASIC her erfolgt die Eingabe in die Register mit den Befehlen PLAY und SOUND.

Die Graphik zeigt die Anschlußbelegung des PSG. Die leeren Anschlüsse werden teilweise vom Hersteller zu Testzwecken verwendet.



DA7-DA0

Diese acht Leitungen stellen den 8 Bitbreiten bidirektionalen Bus dar, der Daten und Adressen vom Mikroprozessor zum PSG sendet und Daten vom PSG an den Mikroprozessor liefert. Im Data-Modus entsprechen die Leitungen DA7 bis DA0 den Bits B7 bis B0 des Registerfeldes (Register-Array). Im Adreßmodus bestimmen DA3 bis DA0 das Register, DA7

Spectravideo-BASIC

Fortsetzung von Seite 4

Darüber hinaus kann man Variablen auch viermal, fünfmal und mehr dimensionieren. Vergleiche gibt es höchstens noch bei der vierten, die man als Gebilde in vier Dimensionen ansehen kann. Nach Einstein ist ja die Zeit die vierte Dimension. Alles darüber sind rein rechnerische Gebilde.

Begrenzt wird das Ganze nur durch die Speicherkapazität des Computers und folgender Regel: Jede Dimension darf nicht größer als 255 Einheiten umfassen (Bei "DIM A(X)" darf X nie größer als 255 sein, bei "DIM A(X,Y)" darf weder X noch Y größer als 255 werden.).

Außerdem ist die Gesamtzahl der Dimensionen auf 32767 beschränkt. Um bei dem Beispiel in der Klammer zu bleiben, 20*30 darf nie größer als 32767 werden. Bei einer dreifachen Reservierung "DIM (X,Y,Z)" darf X*Y*Z nicht größer als 32767 werden.

Wozu braucht man nun diese Dimensionen. Es gibt ja genug Möglichkeiten, verschiedene Variablennamen zu bilden. Dazu braucht man ja keine Reservierung!

Zum Ersten sind die Namen auch auf ungefähr 3700 begrenzt und und zum zweiten würde es ziemlich unübersichtlich werden, mit A, B, C, D, DF, ED und so weiter zu rechnen. A(1), A(2), A(3) und so weiter ist viel übersichtlicher.

Der dritte und triftigste Grund ist folgender. Versuchen Sie doch einmal, bei einem "INPUT"-Befehl, zum Beispiel "INPUT A", die Variable hinter dem Befehlswort im Programmablauf zu ändern. Es geht nur sehr kompliziert mit "POKE". Man muß nämlich den ASCII-Code der neuen Variable in die Speicherstelle poken, wo die alte steht. Aber das nur nebenbei, wichtig ist, daß "INPUT A(I)" viel einfacher die einzelnen Behälter ändern kann. Man braucht lediglich den Wert der Variable I ändern. Je nachdem, welchen Wert I hat, wird nun die jeweilige Variable A(I) gefüllt.

Der Hauptgrund ist also die enorme Flexibilität, die mit der Dimensionierung erreicht wird.

Kommen wir nun nach dem zugegeben langen Gedankenexperiment zu harten Realität zurück. Welche Syntax hat "DIM" nun genau?

"DIM <Variablenname> (X,...)" ist die allgemeine Form. Nach dem Befehlswort wird die gewünschte Variable eingetragen. In der Klammer steht die Anzahl der "Laden", die reserviert werden soll. Dabei sind die oben erwähnten Begrenzungen zu beachten.

Fehlermeldungen für "DIM" gibt es auch in Hülle und Fülle. Da gibt es zum Ersten einmal "Redimensioned Array". Dies passiert, wenn man eine schon dimensionierte Variable noch einmal dimensionieren will. Um dieser Fehlermeldung zu entgehen, kann man, bevor man ein Feld noch einmal reserviert, mit dem Befehl "ERASE" arbeiten (näheres über "ERASE" in der nächsten Folge.).

Weiters kann es einem passieren, daß man die Meldung "Suscript out of Range" an den Kopf geworfen bekommt. Dies passiert, wenn man zu wenig oder überhaupt keine Felder dimensioniert hat und versucht, diese zu füllen. Man kann übrigens bis zum Wert 10 ohne "DIM" arbeiten.

Soweit über den wichtigen Befehl "DIM" und seine Bedeutung. In der nächsten Folge werden wir uns dem Rest der allgemeinen Anweisungen annehmen. Außerdem wird noch ein klein wenig von den Eingabe-/Ausgabeanweisungen durchgenommen!

bis DA4 in Verbindung mit den Adreßeingängen $\overline{A9}$ und A8 bilden den höherwertigen Teil der Adresse (Chip Select).

$\overline{A9}$, A8

Diese zusätzlichen Adreßbits werden beim SVI nicht verwendet und sind mit Masse bzw. +5V verbunden.

$\overline{RESET}$

Zur Initialisierung des PSG wird logisch "0" an diesen Anschluß gelegt.

BDIR, BC1, BC2

Diese Bus-Kontrollsignale werden vom Z80A generiert und steuern alle Busoperationen des PSG. Die Dekodierung dieser Bussignale zeigt die folgende Übersicht:

BDIR	BC1	BC2	Funktion
0	1	0	inaktiv
0	1	1	Lesen vom PSG
1	1	0	Schreiben zum PSG
1	1	1	Latch Adresse

Kanäle A, B und C

Analogausgabe der drei Kanäle, 1 V Spitze-Spitze.

Input/Output A7-A0, B7-B0

Die beiden parallelen 8 Bit-Ports können zur Eingabe bzw. Ausgabe zu peripheren Anschlüssen dienen. Im Eingabemodus sind die Pins "high".

Port A dient der Abfrage der Joysticks, und zwar wie folgt:

IO A0	Joystick 1	0 = vorwärts
IO A1		0 = rückwärts
IO A2		0 = links
IO A3		0 = rechts
IO A4	Joystick 2	0 = vorwärts
IO A5		0 = rückwärts
IO A6		0 = links
IO A7		0 = rechts

Wenn die Bits gesetzt sind ("1"), bedeutet dies, daß kein Kontakt besteht, daß also keine Eingabe vom Joystick her besteht.

Port B erledigt das Bank-Switching:

IO B0	Bank 1 (ROM-Steckschacht)
IO B1	Bank 21
IO B2	Bank 22
IO B3	Bank 31
IO B4	Bank 32
IO B5	LED bei Caps Lock
IO B6	ROM enable 0 (BASIC ROM)
IO B7	ROM enable 1 (BASIC ROM)

Bei den Bits 0 bis 4 bedeutet "0" Bank ist eingeschaltet, "1" Bank ist weggeschaltet.

Bitte beachten Sie den Programmbeitrag über Bank-Switching in diesem Heft.

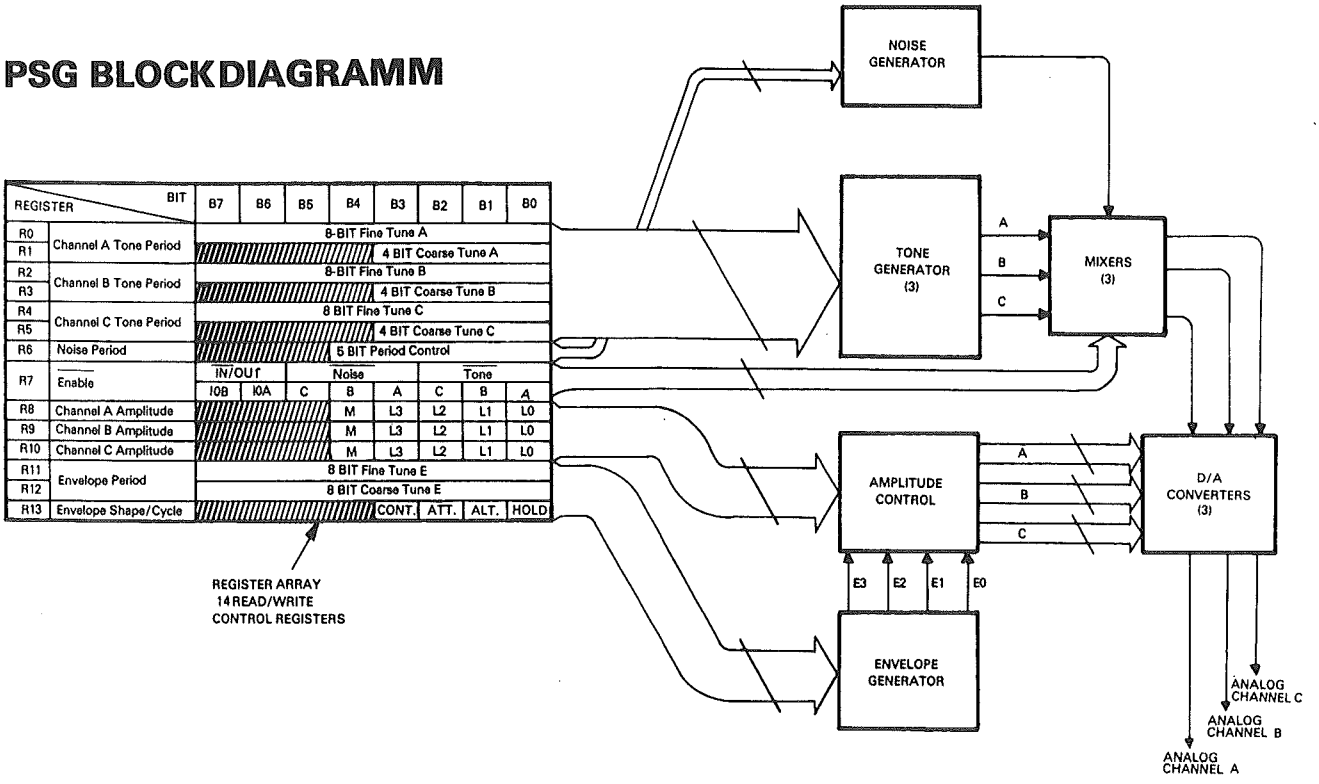
Bei den SVI-Computern ist das I/O-Port A auf Einlesen geschaltet, Port B als Ausgabe.

Das untenstehende PSG-Blockdiagramm gibt eine Übersicht über die Bedeutung der Register. Für die vorstehend angeführten Ports A und B sind die Register 16 und 17 zuständig, die im Blockdiagramm nicht dargestellt sind. Auch die Bedeutung der Bits B7 bis B0 im Register-Array ersieht man aus dem Diagramm.

Die Register 0 bis 5 liefern die Töne in drei getrennten Tonkanälen. Register 0 beinhaltet die 8 niederwertigen Bits der Zahl, die die Tonhöhe für Kanal A angibt, Register 1 enthält die vier höherwertigen Bits dieser Zahl. Register 2 und 3 sind für Kanal B zuständig, 4 und 5 für Kanal C. Diese Register liefern nur die Tonhöhe, und zwar bedeutet eine höhere Zahl einen tieferen Ton. Die Lautstärke wird mit den Registern 8 bis 10 eingestellt (zwischen 0 und 15, wobei 15 die größte Lautstärke darstellt).

In der kommenden Folge werden weiter die Möglichkeiten der Programmierung des Tongenerators beschrieben. Dann folgt die Behandlung des Parallelausgabebausteins 8255.

PSG BLOCKDIAGRAMM



Z 80 - Programmieren in Assembler

Assemblerfortsetzungskurs Nr.:005

Nun haben wir schon 16-Bit Additionen und Subtraktionen kennengelernt, jedoch wurde bis jetzt noch nie erwähnt, wie man nun mit dezimalen Zahlen rechnen kann. In der Assemblerprogrammierung gibt es eine Zahlendarstellung, mit der man recht einfach dezimale Zahlen darstellen kann. Das ist die sogenannte 'BCD'-Darstellung. BCD ist die Abkürzung von 'binary coded decimal', beziehungsweise heißt es auf Deutsch 'binär codiert'. Mit dieser Zahlendarstellung ist es möglich, in einem Byte zwei dezimale Ziffern als Zahlen im Wert von 0-99 dezimal darzustellen. Dabei wird einem Nibble (=4 Bits) von einem Byte eine Ziffer zugewiesen. Ein Nibble darf daher nie einen größeren Wert als 9 haben. Die dezimale Zahl '48' wird in einem Byte folgendermaßen dargestellt: '4 8'. Sieht man nun den hexadezimalen Wert des Bytes an, so ist zu erkennen, daß es den hexadezimalen Wert 48 hat. Das Byte mit dem hexadezimalen Wert '34' hat den dezimalen BCD-Wert 34.

Dies bedeutet, daß von einer hexadezimale Zahl die beiden Nibbles als zwei Stellen einer dezimalen Zahl angesehen werden. Nibbles, deren Werte grösser als 9 sind und daher hexadezimale Werte von "A" bis "F" besitzen, sind in der BCD-Darstellung falsch und dürfen daher nicht vorkommen.

Da der Z-80 ein sehr komfortabler Mikroprozessor ist, hat er auch einen Befehl, der das Rechnen mit solchen BCD-Zahlen erleichtert. Dieser Befehl ist der 'DAA' (dezimaler Abgleich) -Befehl. Er wird hinter einen Arithmetischen Rechenbefehl gestellt und gleicht nur den Akkumulator an. Addiert man nun zwei BCD Zahlen mit dem Akkumulator, so wird es öfters passieren, daß das Ergebnis keine BCD-Zahl ist und damit den Computer in die Irre eliten würde (es kämen garantiert falsche Ergebnisse heraus). Der DAA-Befehl hinter einer Subtraktion oder Addition bewirkt nun, daß das falsche Ergebnis, welches sich im Akkumulator befindet, richtig gestellt wird. Der Prozessor benützt dabei einige Flags, die gemäß dem Ausgang der Rechenoperation gesetzt waren.

Wir werden nun ein Programm erstellen, in welchem zwei BCD-Zahlen, die bei der Adresse C000 und C001 abgelegt sind, miteinander addiert werden. Das richtige BCD-Ergebnis wird in der Speicherzelle mit der Adresse C002 abgespeichert.

```
LD   A,(C000)      ; Lade erste BCD-
LD   C,A             Zahl in das Regi-
                     ster C.
LD   A,(C001)      ; Lade zweite BCD-
                     Zahl in den Akku-
                     mulator
ADD  A,C           ; Die beiden BCD-
                     Zahlen werden ad-
                     diert und das
                     'falsche' Ergeb-
                     nis im Akku abge-
                     legt
DAA                ; Das 'falsche' Er-
                     gebnis im Akkumu-
                     lator wird rich-
                     tig gestellt
LD   (C002),A      ; Das Ergebnis wird
                     abgespeichert
```

Was passiert nun, wenn man zwei Zahlen addiert, die eigentlich BCD-Zahlen sein sollten, aber ungültig sind, das heißt, daß ein Nibble einen größeren Wert als 9 enthält. Addiert man solche Zahlen, und versucht man diese dann nachher mit dem DAA-Befehl dezimal abzugleichen, so passiert gar nichts, außer, daß der Inhalt des Akkumulators zerstört wird. Das Gleiche passiert auch, wenn der DAA-Befehl eingesetzt wird, ohne daß eine ihm Rechenoperation vorausgegangen ist. In dem nun folgende Programm sollen die beiden dezimalen Zahlen 1735 und 4218 miteinander addiert werden. Das Ergebnis wird dann im Registerpaar HL stehen.

```
LD   D,35          ; Lade die nieder-
                     wertigen Ziffern
                     der ersten Zahl
                     nach D.
LD   A,18          ; Lade die nieder-
                     wertigen Ziffern
                     der zweiten Zahl
                     in den Akku.
ADD  A,D           ; Addiere die bei-
                     den BCD-Zahlen.
                     Der Akkumulator
                     enthält jetzt die
                     hexadezimale Zahl
                     '4D', welche eine
                     unzulässige BCD-
                     Zahl ist.
DAA                ; stelle das Ergeb-
                     niss im Akku
                     richtig. Der Akku
                     enthält den hexa-
                     dezimalen Wert 53
LD   L,A           ; Speichere die
                     niederwertigen
                     Ziffern nach L
LD   D,17          ; Lade nach D die
                     höherwertigen
                     Ziffern der
                     ersten Zahl.
LD   A,42          ; Lade in den Akku
                     die häherwertigen
                     Ziffern der Zahl.
ADC  A,D           ; Addiere die bei-
                     den Zahlen und
                     auch das Carry-
                     flag (beziehungs-
                     weise den Über-
                     trag der Addition
                     der niederwerti-
                     gen Ziffern.)
                     Akku = 59 oder 5F
DAA                ; Wenn der Akku den
                     Wert 59 enthält,
                     so wird der
                     Inhalt des Akkus
                     nicht verändert.
                     Enthält er jedoch
                     den Wert 5F, so
                     wird er auf 60
                     umgeändert.
LD   H,A           ; Speichere das
                     Ergebnis der Ad-
                     dition der höher-
                     wertigen Ziffern
                     in das H-Register
```

Wie Sie sicher schon bemerkt haben, ist das Programm so geschrieben, daß man sämtliche vierziffrige BCD-Zahlen addieren kann. Ansonsten wären der ADC- und der zweite DAA-Befehl unnötig.

Doch bei vielen Programmen bzw. bei schwierigeren Problemstellungen ist es nicht leicht möglich, ein Programm zu schreiben, welches von Anfang bis Ende durchgearbeitet wird, ohne zu verzweigen. Der Z-80 besitzt so wie alle anderen Mikroprozessoren und Programmiersprachen die Möglichkeit der Programmverzweigung. Das geht mit 'JP' (engl."Jump"=springen). Hinter diesem Befehl wird eine 16-bit Adresse geschrieben. Stößt der Prozessor bei der Abarbeitung seines Programmes auf einen solchen JP-Befehl, so wird wie immer zuerst der Programmcounter auf den nächsten Befehl gestellt, jedoch

setzt der Computer nicht bei diesem fort, sondern bei dem Befehl, der bei der Adresse steht, die sich hinter dem JP-Befehl befindet. Um dies näher zu veranschaulichen werden wir ein kurzes Programm erstellen, wo dieser Befehl verdeutlicht wird.

```
D000:   LD      A,15
D002:   LD      E,A
D003:   JP      D400
D006:   LD      A,8D
D008:   ADD     A,E
D009:     .
          .
          .


D400:   LD      A,F4
D402:   SBC     A,E
D403:   LD      (E000),A
```

Dieses Programm ist ab D000 abgelegt, und ein anderes befindet sich bei D400. Was passiert nun, wenn der Computer das Programm ab D000 durchläuft? Zuerst bekommt der Akkumulator den Wert 15, dann das Register E den Wert des Akkumulators. Beim JP-Befehl setzt der Computer nicht mit dem Befehl LD A,8D fort, sondern mit dem Befehl, der sich bei der Adresse D400 befindet. Das heißt, das Programm tut nichts anderes als den Wert 15 von F4 zu subtrahieren und das Ergebnis nach E000 abzuspeichern. Der Prozessor kommt bei der Abarbeitung dieses Programmes natürlich nie zu den beiden Befehlen die sich nach dem JP-Befehl befinden.

Doch dieser Befehl alleine würde noch nicht allzuviel bringen, denn er bewirkt ja nur daß das Programm woanders fortgesetzt wird. Beim JP-Befehl gibt es die Möglichkeit, Bedingungen zu stellen, das heißt, daß der Computer nur dann verzweigt wenn die Bedingung wahr ist, ansonsten setzt er mit dem nächsten Befehl fort. Die Bedingungen können sich nur auf Flags beziehen. Da aber noch nicht alle Flags erklärt worden sind, will ich das jetzt nachholen.

Das Carry-Flag kennen wir ja schon. Es wird gesetzt, wenn bei der letzten Rechenoperation ein Übertrag aufgetreten ist.

Das nächste Flag ist genauso wichtig, wenn nicht noch wichtiger als das Carry-Flag. Dies ist das Zero (Z) - Flag. Dieses Flag signalisiert, ob das Ergebnis der letzten Rechenoperation gleich oder ungleich Null war. Durch die Abfrage dieses Flags lassen sich herrlich leicht Programmschleifen erstellen. Man weist einem Register einen Wert zu, der angibt, wie oft die Schleife durchlaufen werden soll. Dann erniedrigt man ihn nach dem Durchlaufen der Schleife um eins und arbeitet die Schleife wieder ab, wenn der Wert ungleich Null ist.

Subtraktions(N)-Flag: Dieses Flag ist eher unwichtig für den Programmierer. Ist der letzte arithmetische Befehl eine Subtraktion gewesen, so wird das Flag auf Eins gesetzt. Verwendet wird es von dem DAA-Befehl, der wissen muß, ob die letzte Rechenoperation eine Subtraktion oder Addition war.

Parität/Überlauf (P/V)-Flag: Das P/V-Flag erfüllt gleich mehrere Funktionen. Die erste Funktion ist die Anzeige auf gerade oder ungerade Parität. Die Parität gibt an, ob die Summe der gesetzten Bits einer Zahl, gerade oder ungerade sind. Ist sie ungerade, so ist das Paritäts-Bit auf Null gesetzt (ungerade Parität). Auf Eins wird es bei einer geraden Anzahl von gesetzten Bits (gerade Parität gesetzt. Dies wird meistens beim Übertragen von Daten im ASCII-Format verwendet, um zu wissen, ob die übertragenen Daten auch richtig angekommen sind Da das ASCII-Format nur sieben Bits benötigt, wird das Paritäts-Bit im achten Bit abgespeichert. Stellt man beim Erhalten eines Datenwertes fest, daß das Paritäts-Bit nicht mehr mit der Anzahl der gesetzten Bits übereinstimmt (,wenn zum Beispiel eines "umgefallen" ist), so weiß man, daß ein Übertragungsfehler vorliegt.

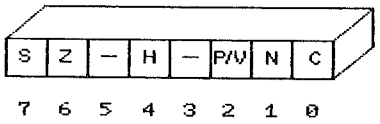
Die nächste Funktion des P/V-Flags ist die des Überlaufflags. Es zeigt an, wenn bei einer Addition oder Subtraktion von Zahlen im Einer-, Zweierkomplement oder in der vorzeichenbehafteten Zahlendarstellung eine Zahl außerhalb des Gültigkeitsbereiches auftritt. Das heißt, wenn das Ergebnis größer als 127 oder kleiner als -128 ist. Aber das Flag hat noch weitere Funktionen. Es wird von den Blocktransfer-Befehlen (LDDR, LDI, LDIR, LDD) und den Suchbefehlen (CPD, CPDR, CPI, CPIR) als Anzeiger für das Ende einer Schleife verwendet.

Die letzte Funktion dieses Flags wird bei den Spezialbefehlen LD A,I und LD A,R verwendet. Dabei erhält das P/V-Flag den Wert des Interrupt-Enable-Flip-Flops (IFF2). Das kann dazu verwendet werden um den Inhalt dieses Flip-Flops zu retten.

Halbübertrags (H)-Flag: mit diesem Flag wird angezeigt, ob ein übertrag von Bit drei nach Bit vier stattgefunden hat, beziehungsweise stellt es den übertrag vom unteren in den oberen Nibble dar. Diese Flag wird ebenfalls wie das Subtraktionsflag von dem Befehl DAA verwendet.

Vorzeichen (S (Sign))-Flag: Das Sign-Flag beinhaltet den Zustand des siebenten Bits. In den verschiedenen Zahlendarstellungen (Einer-, Zweierkomplement, vorzeichenbehaftete Zahlendarstellung) ist es damit möglich festzustellen, ob die Zahl negativ oder positiv ist.

Wie schon im zweiten Artikel erwähnt wurde, gibt es das Flag-Register. In diesem Register sind die eben genannten Flags enthalten. Die folgende Darstellung verdeutlicht, wie die einzelnen Flags im Flagregister angeordnet sind.



In diesem Flag-Register werden die Bits 5 und 3 nicht benützt.

Mit dem JP-Befehl kann man nun bedingt verzweigen, indem man die Flags auf ihren Zustand abfragt. Jedoch ist es leider nicht möglich, alle Flags direkt mit dem JP-Befehl abzufragen. Es lassen sich nur das Zero, Sign, P/V, und Carry, Flag abfragen. Die Bedingungen werden in der Mnemonic durch spezielle Buchstabenkombinationen symbolisiert:

```
 Z   =   Zero-Flag hat den Wert  1
NZ   =   Zero-Flag hat den Wert  0
 C   =   Carry     hat den Wert  1
NC   =   Carry     hat den Wert  0
PO   =   Ungerade Parität (P/V=0)
PE   =   gerade Parität   (P/V=1)
 P   =   Sign-Flag hat den Wert 0 (positiv)
 M   =   Sign-Flag hat den Wert 1 (negativ)
```

In der nächsten Ausgabe wollen wir die bedingten Sprünge anhand einiger simpler Programmbeispiele üben und verstehen lernen.

Vorerst fassen wir noch die Organisation des BASIC-Speichers zusammen, und widmen uns danach dem oft angekündigten Beispiel für für die praktische Anwendung.

Wie in den vorigen Folgen erwähnt, legt der BASIC-Interpreter die Programme und die dazugehörigen Variablen im User-RAM ab. Das BASIC-Programm wird immer mit einer Null in der Speicherzelle &H8000 begonnen. Danach findet sich die erste Zeile. Sie wird durch zwei Bytes eingeleitet, die auf die nächste Zeile zeigen. In Heft 1 auf Seite 13 sieht man diesen Zeiger, der dort auf die Adresse &h801A weist. Die nächsten beiden Bytes markieren die Zeilennummer. Erst danach fängt der eigentliche Inhalt der Zeile an.

Befehlswörter werden in einem eigenen Code abgelegt, der in Heft 3 ausführlich behandelt wurde. Ebenso kann man in Heft 2 über die Variablen nachlesen. Zu den Variablen sei folgendes gesagt: Prinzipiell legt der Computer den Variablennamen im ASCII-Code ab ("INPUT A" ergibt in den Speicherzellen "85 20 41"). Die Abhandlung über die Variablenzuweisungen in Folge 2 gilt nur für folgende Form: <Variable>=<Ausdruck>. Es gibt allerdings, wie oben erwähntes Beispiel beweist, auch Formen, wo nur eine Variable vorkommt, die keinen direkten Wert zugewiesen bekommt, zum Beispiel "INPUT A".

Der Wert von A wird dann direkt hinter dem Programm abgespeichert. Das ist wichtig für das Verständnis der Speicherstruktur. Der Computer legt alle Variablen, die gebraucht werden, direkt hinter dem Programm in folgender Codierung ab: Es wird zuerst nach den Typen unterschieden. Wie wir ja wissen, gibt es numerische und alphanumerische Variablen. Die alphanumerischen Variablen werden mit der Kennnummer 3 eingeleitet. Danach kommt der Variablenname, der aus zwei Bytes besteht. Der Hauptteil ist ein Zeiger, der auf eine bestimmte Stelle in den Stringspeicher zeigt, wo der Text der jeweiligen Variable abgelegt ist.

Will man sich also den Inhalt einer String-Variable ansehen, so muß man den 2-Byte langen Zeiger erkennen, und ab dieser Stelle den Text auslesen. Vor diesem Zeiger steht allerdings noch die Länge dieses Textes in einem Byte. Das Schema einer Stringvariable ist unten angegeben.

```
3 Variablenname Stringlänge Stringzeiger
zum Beispiel:
3 41 43         03          1D F2

AC$ ist 3 Buchstaben lang, Text ab &HF21D
```

Die numerischen Variablen sind noch einmal in einige Typen unterteilt.

Da sind zuerst die Integer-Variablen. Sie haben die Kennzahl 2. Danach kommt der Variablenname und zwei Bytes "Hex-Code". Dieser hexadezimale Code gibt den Wert der Variable an.

```
zum Beispiel "2 41 00 0A 00" bedeutet:
Integervariable A% hat den Wert 10
```

Die letzten beiden Unterteilungen sind einfachgenaue und doppeltgenaue numerische Variablen. Die ersteren haben die Kennziffer 4, die zweiten 8. Anschließend kommt wie immer der Variablenname. Die nächsten Punkte gelten für beide Variablen. Es wird zuerst einmal das Komma und das Vorzeichen bestimmt. In einem Byte wurde das folgendermaßen kombiniert. Das erste Bit kennzeichnet das Vorzeichen. 1 ist -, 0 ist +. Die restlichen sieben Bits sind für die Kommapunktbestimmung verantwortlich. Hier wird nach der Formel "Zahl=Punktposition + &H40" gerechnet.

Steht der Dezimal-Punkt also auf der ersten Stelle, so wird &H41 (binär 0100 0001) verwendet. Ist die Zahl obendrein noch negativ, dann ändert sich dieses Byte auf &HC1 (binär 1100 0001). Ist der Exponent negativ, was zum Beispiel bei 0.0001 der Fall ist, dann wird ebenso &H40 addiert, was bewirkt, daß das Byte selber nie negativ werden kann (weil kein Exponent größer als dezimal 64 werden darf.) Es ergibt sich aus dieser Formel folgende Bereichsaufteilung:

```
0-3F  Bereich von 0 bis 1
40-7F Bereich von 1 bis 10^62
80-BF Bereich von 0 bis -1
C0-FF Bereich von -1 bis -1*10^62
```

Die nächsten Bytes ergeben eine Ziffernfolge. Diese Ziffern werden dezimal abgelegt, und zwar zwei in einem Byte. Die Anzahl der Bytes ist für die doppeltgenauen Variablen 7, für die einfachgenauen 3. Zusammen mit dem Exponentenbyte bilden diese Bytes die Zahl.

```
zum Beispiel: "4 41 00 41 54 30 00" ergibt:
        A! hat den Wert 5.43
```

Soweit die Abspeicherung der Variablen hinter dem Programm! Wie oben erwähnt werden also die einzelnen Zeilen codiert abgelegt. Welchen Nutzen diese Erkenntnis hat, wird in der nächsten Folge geklärt.

Fortsetzung folgt!

Spectra Video Club Austria

Der "Spectra Video Club Austria" stellt sich vor:

Aktivitäten:

- regelmäßige Clubabende zum Austausch von Information
- Möglichkeit zum kostenlosen Gebrauch von SVi-Computern und deren Peripherie (Drucker, Diskettenstationen, u.s.w.)
- außerordentliche Clubabende und Vorträge über Themen rund um Soft- und Hardware der Spectravideo-Computer
- Verbilligter Bezug von Spectravideo-Computern und deren Peripherie
- regelmäßiger und kostenloser Bezug unserer Clubzeitschrift, dem SVi-Journal

Über das SVi-Journal:

- Es dient vor allem der Kommunikation zwischen den Clubmitgliedern
- Es bietet die Möglichkeit, durch Kurse Programmiersprachen zu erlernen
- Es wird Einblick in die Hardware des Computers gegeben
- In jeder Ausgabe finden sich nützliche Programmteile
- Jedes Clubmitglied bezieht die Zeitschrift kostenlos
- Es erscheint monatlich (immer um die Monatsmitte)
- Obwohl das SVi-Journal erst fünf Monate existiert, ist es schon international anerkannt
- Es gibt die Möglichkeit, die Zeitschrift zu abonnieren

  jährlich S 150,-
  halbjährlich S 80,-

Ich interessiere mich für die Mitgliedschaft beim "Spectra Video Club Austria" und wünsche die Zusendung näherer Informationen und der Anmeldungsunterlagen.

Name .....

Adresse .....

Bitte in unserem Clublokal, dem Computer-Studio der Firma Wehsner oder auf der Hobby-Elektronik im Messepalast im "Action-Center" abgeben (Lageplan auf der folgenden Seite!) oder per Post an "Spectra Video Club Austria",
p.A. Computer-Studio, 1040 Wien, Paniglgasse 18-20 senden.

Die Abgabe dieses Coupons ist völlig unverbindlich und kostenlos.

Unsere Präsenz auf der **Hobby Elektronik**

Wir sind in der Halle U im 1. Stock zu finden!

Dort haben wir im "Action-Center" der österreichischen Fachzeitschrift "itm praktiker" folgendes für Sie vorbereitet:

- Programmierwettbewerb

  In einer Zeitspanne von 20 Minuten soll ein kurzes Programm entworfen werden. Gewertet wird das effektvollste und schönste Ergebnis. Die Prämierung erfolgt jeweils am Ende eines Messetages! SVi-Besitzer sind aus Fairness nicht zum Wettbewerb zugelassen.

  Preise für den Wettbewerb:
  1.Preis: ein Buch "BASIC-Kompendium" von Werner Chmel
  2.Preis: drei Disketten der Marke "DATALIFE" oder
  sechs Kassetten der Marke "DATA-TRACK"
  3.Preis: ein Jahresabonnement des SVi-Journals

- Demonstration der Graphikfähigkeit des SVi-328

  Mit einigen Graphiken und einem Zeichenprogramm kann man die enormen Fähigkeiten des Spectravideo auf dem Gebiet der "Zeichenkunst" bewundern.

- Demonstration der Synthesizerqualitäten des SVi-328

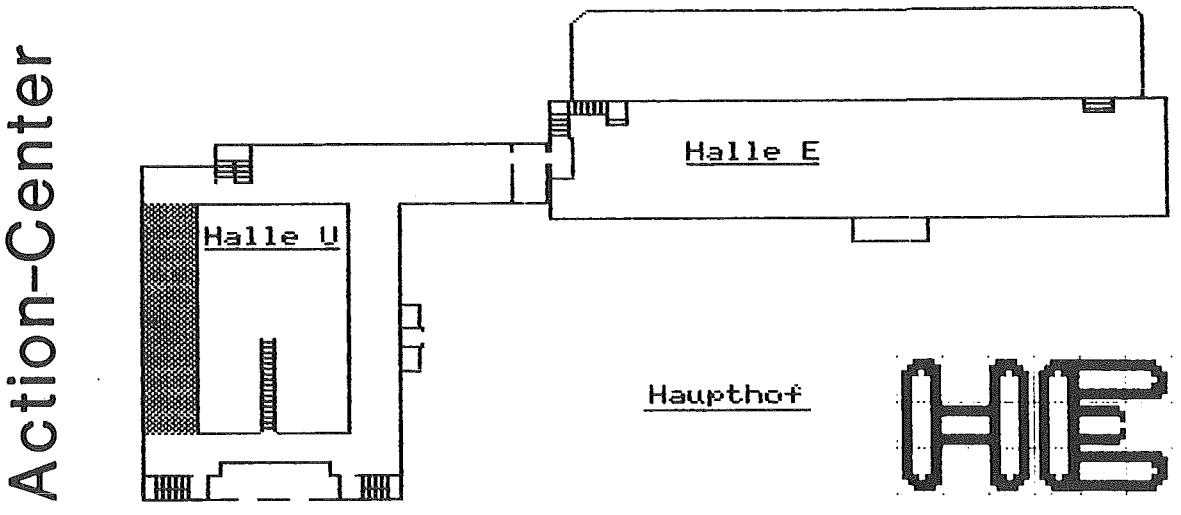
  Neben einer Sammlung musikalischer Werke wird ein Programm zur Simulation von Musikinstrumenten gezeigt.

- Hardware-Zusatz "Sprachsynthesizer"

  Man braucht einfach einen Satz eingeben
  und der Computer spricht mit Ihnen.

- Möglichkeit zum Testen des umfangreichen Befehlssatzes von 208 BASIC-Befehlen (Clubmitglieder stehen Ihnen gerne Rede und Antwort bei Fragen).

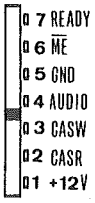
ACHTUNG!! Der Mitgliedsbeitrag gilt ab Anfang November schon für nächstes Jahr! NÜTZEN SIE DIE CHANCE !! Und werden Sie Mitglied mit allen Vorteilen, die Ihnen der "SVCA" bietet.



Cassetteninterface für SVI-3x8

Bis jetzt konnten an die SVI-Computer nur die Cassettenrecorder von Spectravideo angeschlossen werden. Mit dieser einfachen Schaltung kann man nun jeden handelsüblichen Cassettenrecorder anschließen. Dadurch kann man sich eine Menge Geld ersparen, da die Materialkosten nur ungefähr 250 Schilling betragen.

Der Aufbau der Schaltung ist im großen und ganzen einfach, doch können sich einige kleine Probleme ergeben. Als ich die Schaltung das erste Mal aufbaute, wußte ich nicht, wie die einzelnen Anschlüsse am SVI-328 lokalisiert sind, doch im Computer-Studio konnte man mir weiterhelfen. Hier ist nun ein Diagramm der Anschlüsse am SVI-328.

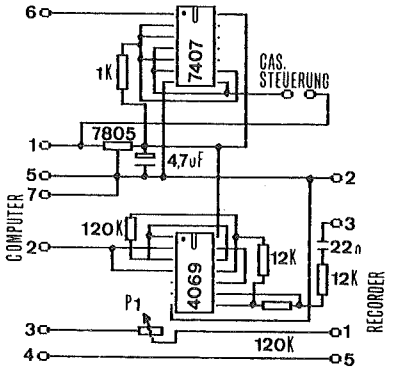


Als Orientierungshilfe beim Identifizieren der Anschlüsse dient der Teilstrich.

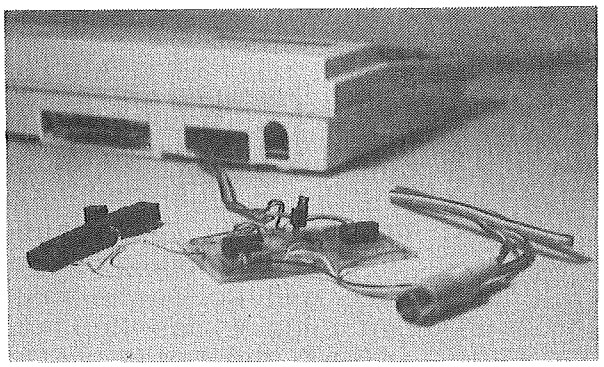
Die Anschlüsse des Interfaces zum Cassettenrecorder haben folgende Bedeutung:

1: Aufnahme linker Kanal
2: GND
3: Wiedergabe linker Kanal
5: Wiedergabe rechter Kanal

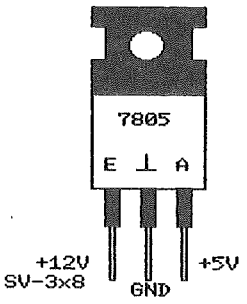
Der Anschluß 5 muß nicht unbedingt angeschlossen werden (das ist die 2. Tonspur des Recorders, falls ein Stereogerät vorhanden ist).



Als nächstes Problem ergab sich die geeignete Wahl des Potentiometers P1 für die Austeuerung bei SAVE. Dieser Widerstand ist vom Recorder abhängig. Ich wählte ein lineares 10k-Schiebepotentiometer. Mit dieser Wahl hatte ich keine Probleme. Zur Sicherheit sollte aber ein 100k-Poti verwendet werden. Dadurch sollten sich keine Probleme ergeben. Eine andere Schwierigkeit war die Beschaffung der Bauteile. Ich hatte zwar Glück, und bekam die zwei ICs, doch den Stecker des Interface zum SVI-3x8 bekam ich nicht so problemlos. Doch bei der Firma Printtechnik in der Stumpergasse konnte ich ihn kaufen. Allerdings war er sehr teuer, da es zwar ein 6-poligen, doch keinen 7-poligen gab. So war ich gezwungen einen 21-poligen zu kaufen, nämlich den Busstecker des C64. Dieser Stecker kostete aber rund 100 Schilling. Sonst waren alle Bauteile problemlos zu beschaffen. Wer den IC 4069 nicht bekommt, kann auch den 74C04 verwenden.



Der Aufbau der Schaltung selbst ist problemlos. Die kleine Platine kann man sich selber ätzen. Dann werden zuerst die Drahtbrücken und die Kabel eingelötet. Nachher lötet man die IC-Sockel ein, danach Widerstände und Kondensatoren. Aufpassen auf die Polung des Elkos! Erst jetzt wird der Spannungsregler 7805 eingelötet. Die richtige Polung ist aus der Skizze ersichtlich. Wenn alle Lötarbeiten abgeschlossen sind, werden die beiden ICs richtig in die Sockeln gesteckt. Die richtige Lage ist aus dem Schaltbild ersichtlich. Bevor man das Interface in Betrieb nimmt, sollte man noch einmal alles überprüfen.



Wenn man sich vergewissert hat, daß alles stimmt, wird das Interface angeschlossen. Die Stecker müssen richtig angelötet und eingesteckt sein, besonders der Stecker zum Computer. Die Fernsteuerung an den Cassettenrecorder muß nicht unbedingt angeschlossen sein. Wenn man alles noch einmal überprüft hat, schaltet man den Computer an. Zur Überprüfung ob das Interface funktioniert, nimmt man am besten eine Cassette, auf die man vorher etwas mit dem SVI-Recorder aufgenommen hat. Wenn es nicht sofort klappt, nur nicht den Mut verlieren. Man muß oft ein bißchen mit der Lautstärke und dem Klang experimentieren. Wenn der Ladetest funktioniert hat, nimmt man ein Programm auf. Hier muß man eventuell mit der Aussteuerung (P1) experimentieren. Man sollte auch versuchen, ein mit dem Interface geSAVEtes Programm mit dem SVI-Recorder zu laden. Man muß meistens dann die Aussteuerung verändern. Doch es sollten sich keine großen Probleme ergeben. Wer sich nicht zutraut, das Interface zu bauen, oder keine Lust dazu hat, kann sich an mich oder an das Computer-Studio wenden.

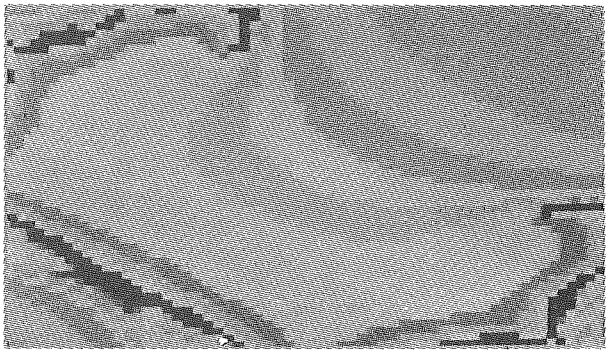
Und nun viel Spaß beim Aufbau der Schaltung.

RAFAEL RAZIM

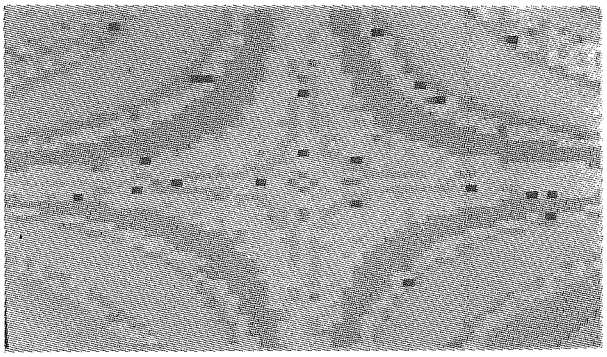
```
*****************************************
*                                       *
*     KOMPLEXE GRAFIKEN MIT SVI-3x8     *
*                                       *
*****************************************
```

Mit diesem kurzen Programm können sehr schöne Farbgrafiken erzeugt werden. Leider funktioniert dieses Programm nur mit SCREEN 2, da bei SCREEN 1 nur immer 8 Bildpunkte zusammen eine Farbe haben können. Dadurch würden sich unschöne Farbklekse auf dem Bildschirm ergeben. Außerdem würde ein Programmlauf in SCREEN 1 ungefähr acht Stunden dauern. Mit SCREEN 2 dauert die Erzeugung einer Grafik ungefähr eine halbe Stunde.

Nun zur Funktionsweise und dem Aufbau des Programmes: Aufgebaut wird die Grafik mit Hilfe der sogenannten komplexen Zahlen. Bei diesen Zahlen wird die Quadratwurzel von -1 (die ja normalerweise nicht erlaubt ist) definiert und 'i' genannt. Dadurch entstehen Zahlen der Form A+B*i. A ist der sogenannte Realteil, B der Imaginärteil der Zahl. Die Zahlen werden nicht wie bisher auf der Zahlengeraden, sondern in der sogenannten Zahlenebene aufgetragen. Dadurch stehen uns mehr Zahlen zur Verfügung.



```
10 REM *********************************
20 REM *                               *
30 REM *     KOMPLEXE GRAFIKEN         *
40 REM *                               *
50 REM *  WRITTEN BY RAFAEL RAZIM      *
60 REM *                               *
70 REM *    VERS 3.0 VOM 29.10.84      *
80 REM *                               *
90 REM *********************************
95 REM
100 ONKEYGOSUB380:KEYON
110 X=192:Y=256
120 'schritt 1
130   INPUTC,D
140   XI=-1:YI=-1:XA=1:YA=1
150   S=50
160   DX=(XA-XI)/(X-1)
170   DY=(YA-YI)/(Y-1)
180   SCREEN2
190 FOR N=0TO(X-1)STEP4
200   FOR M=0TO(Y-1)STEP4
210     'schritt 2
220       A=XI+N*DX
230       B=YI+M*DY
240       K=0
250     'schritt 3
260       A1=A*A-B*B-C
270       B1=2*A*B-D
280       K=K+1
290     'schritt 4
300       R=A*A+B*B
310       A=A1:B=B1
320       IF (R>S)AND(K<15)THENGOTO350
330       IF K=15THENK=0:GOTO350
340       GOTO250
350     'schritt 5
360       PSET(M,N),K
370   NEXT M
375 NEXTN
380 SAVE"1:BILD2",S
390 RETURN
400 END
```



Dieses Programm definiert nun eine einfache Folge von komplexen Zahlen, wie zum Beispiel in meinem Programm die Folge

X^2 + Y (in reellen Zahlen)

oder in komplexen Zahlen ausgedrückt:

(A+B*i)^2 - (C+D*i)

C+D*i ist ein komplexer Parameter, der zu Beginn des Programmes abgefragt wird (Zeile 130). A+B*i ist die laufende Variable. So kann der Computer allerdings nichts mit der Folge anfangen. Dazu muß man erst Gleichungen für den Realteil und für den Imaginärteil entwerfen. Darauf möchte ich aber hier nicht eingehen, sondern gleich das Ergebnis präsentieren:

Der Realteil ist A^2+B^2-C
der Imaginärteil ist 2*A*B-D

Der Computer setzt nun einen Anfangswert in die Gleichungen ein. Mit dem Ergebnis wird die Farbe des Punktes berechnet (Zeilen 300-330). Dann wird der Punkt geplottet. Die Anfangswerte werden erhöht (Zeilen 220,230). DX und DY sind die Schrittweite. Durch Verändern dieser Werte und der Startwerte (kann durch eine Zeile INPUT A,B gemacht werden) können auch Ausschnitte geplottet werden. Diese Werte setzt der Computer wieder in die Gleichungen ein und beginnt von Anfang an.

Der Aufbau des Programms im Einzelnen:

```
110 Festlegen  Auflösung des Bildschirms
120 Festlegen  Parameter (C,D)
 .             Grenzwerte (XI,XA,YI,YA)
 .             Schrittweite (DX,DY)
170
210 Die ersten Werte werden bestimmt
 .
230
260 Hier stehen die Formeln zur Berechnung
 .  der komplexen Werte
270
300 Berechnung des Farbwertes
 .
340
360 Plotten des Punktes in der Farbe K
380 Bild auf Disk speichern
```

Cassettenuser müssen nur die Zeile 380 auf 'CSAVE"BILD",S' ändern.

Andere Grafiken können erzeugt werden, indem man beispielsweise die Formeln in Zeilen 260 und 270 verändert. Der Phantasie sind keine Grenzen gesetzt. Einige Beispiele kann man auf dieser Seite bewundern. Wahrscheinlich kann man oft wiederkehrende Muster erkennen. Wenn jemand irgendeine Regelmäßigkeit beobachtet, so würde ich mich sehr freuen, wenn er sie im SVI Journal veröffentlicht.

Und nun viel Spaß und Freude beim Experimentieren.

RAFAEL RAZIM

BANK SWITCHING

Um die Computer der SVI-Serie CP/M-kompatibel zu machen, kann der Computer einen wesentlich größeren Speicherraum verwalten, als dies von dem Mikroprozessor vorgesehen ist. Der Z80 kann maximal 64 KB (=KByte) adressieren. Einen solchen Speicherraum nennt man auch "Bank". Der SVI-Computer kann, wenn er durch zwei 64 KB-Karten erweitert worden ist, vier solcher Banken ansprechen. Jede dieser Banken besteht aus 2 Seiten (engl. Pages) zu je 32 KB. Die Seite 1 erstreckt sich über den Bereich von 0000H bis 7FFFH und die Seite 2 von 8000H bis FFFFH (siehe auch Abbildung in Heft 1/Seite 8). Die Darstellung einer solchen Seite erfolgt nun üblicherweise in folgender Form: PAGE XY (abgekürzt PGE XY). X gibt an, um welche BANK es sich handelt, und Y um welche SEITE der Bank X. Ich werde von nun an ebenfalls diese Schreibweise verwenden.

Da das CP/M ein von 0000H bis FFFFH durchgehendes RAM benötigt, muß der Computer beim Laden des CP/M-Betriebssystems die ROM-Page wegschalten und die dazu parallel liegende PAGE 21 hinzuschalten. Das Wort "hinzuschalten" ist so zu verstehen, daß ab nun die Seite 1 der Bank 2 auf dem Adreßbus von 0000H bis 7FFFH liegt. Die PAGE 21 ist serienmäßig beim SV-328 vorhanden und muß beim SV-318 erst durch Zustecken einer 64 KB-Karte aktiviert werden. Für den praktischen Teil dieses Artikels setze ich voraus, daß Sie einen SV-328 oder einen erweiterten SV-318 Ihr Eigen nennen.

Das "Bank Switching" wird dem Computer durch den eingebauten Soundgenerator ermöglicht. Dieser spielt nicht nur Melodien, sondern hat zusätzlich die Ports für die Joysticks inne und ein Register 15, dessen Inhalt bestimmt, welche 2 Seiten gerade auf dem Adreßbus liegen. Es können dies durchaus Seiten von verschiedenen Banken sein wie z.B. beim CP/M. Dort wird der Bereich von 0000H bis 7FFFH von der PGE 21 und der Bereich von 8000H bis FFFFH von der PGE 02 gebildet (siehe Abbildung Heft 1/Seite 8). Die Bedeutung der Bits des Registers 15 haben Sie in der folgenden Auflistung angegeben:

Bit 0: Dieses Bit ist für die PGE 11 zuständig. Wenn es gesetzt ist, dann liegt die PGE 11 auf dem Adreßbus. Die PGE 11 ist identisch mit dem Cartridgeschacht und deshalb nur dann verfügbar, wenn ein Modul eingeschoben worden ist. Weiters kann in diese PGE nichts geschrieben werden, da dies schon hardwaremäßig nicht vorgesehen hat.

Bit 1: Dieses Bit entscheidet, ob die PGE 01 (Bit = 0) oder ob die PGE 21 (Bit = 1) auf dem Adreßbus liegt.

Bit 2: Dieses Bit entscheidet zwischen PGE 02 (Bit = 0) und PGE 22 (Bit = 1).

Bit 3: Dieses Bit ist zuständig für die PGE 31. Ist es gesetzt und das System durch eine 64 KB-Karte erweitert, liegt die PGE 31 auf dem Adreßbus.

Bit 4: Wie Bit 3 doch für PGE 32 zuständig.

Bit 5: Sicherlich werden einige von Ihnen die Bedeutung dieses Bits kennen. Es zeigt nämlich an, ob die LED (=Leuchtdiode) von "CAPS LOCK" eingeschaltet ist oder nicht.

Bit 6,7: Das ROM besteht aus 2 Teilhälften. Bit 6 legt fest, ob der Bereich von 0000H bis 3FFFH, und Bit 7 legt fest, ob 4000H bis 7FFFH vom ROM vorhanden ist. Da der BASIC-Interpreter alles andere als gleichmäßig aufgebaut ist, zahlt es sich kaum aus, nur 16 KB vom Interpreter aktiviert zu haben.

Sie müssen aber beim Umherschalten sehr vorsichtig sein. Sie dürfen niemals zwei parallel liegende Seiten gleichzeitig auf den Adreßbus schalten. Den Adreßbus stört es herzlich wenig, welche PAGE er nun adressiert, aber auf dem Datenbus kommen dann nur undefinierte Werte herein, die jedes Programm sofort zum Abstürzen bringen können. Auch dürfen Sie niemals die Seite wegschalten, in der sich Ihr laufendes Programm befindet. Das wäre ebenfalls tödlich für Ihr Programm.

Ich hoffe, die kleine Darstellung reicht aus, um zu erkennen, wie Sie eine Seite erreichen können. Um aber sämtliche Klarheiten zu beseitigen, habe ich ein kleines Programm geschrieben, daß Ihnen die Theorie in einem praktischem Beispiel näher bringen soll.

Vorher aber wieder ein wenig Theorie. Wenn Sie in der PGE 21 arbeiten wollen, müssen Sie bedenken, daß alle 20 Millisekunden ein Interrupt den Z80 erreicht und ihn zwingt, zu der Adresse 0038H zu springen, die Routine dort auszuführen und danach wieder zum Ausgangspunkt zurückzukehren. Da aber in meinem Beispiel die PGE 21 als Datenspeicher verwendet wird, kann man es dem Z80 nicht zumuten, dorthin zu springen. Deshalb müssen Sie immer, wenn Sie auf die PGE 21 umschalten, einen "DI" (Disable Interrupts) im Programm einbauen, um ein Abstürzen des Systems zu verhindern. Nachdem es aber "DI" im BASIC nicht gibt, dürfte es spätestens jetzt wohl jedem klar sein, daß man die PGE 21 nur in Maschinensprache ansprechen kann.

Eine Ausnahme bildet der "SWITCH"-Befehl, der zwischen der PGE 02 und der PGE 22 umschalten kann. Prinzipiell müssen Sie die Interrupts immer dann sperren, wenn Sie eine zum ROM (=PGE 01) parallel liegende Seite ansprechen.

Das Programm ermöglicht es Ihnen, zwei Bilder des hochauflösenden Graphikmodus in die PGE 21 zu speichern und sie dann von dort wieder in das Video-RAM zu laden.

Das Programm wird vom BASIC über "USR" aufgerufen. Weiters muß beim Aufruf des Programmes durch "USR" ein Parameter übergeben werden, der angibt, welches abgespeicherte Bild Sie ansprechen wollen, sei es nun, um es zu löschen, es zu laden, oder das aktuelle Bild abzuspeichern. Der Parameter kann nur den Wert 0 oder 1 annehmen und muß ganzzahlig sein. Ist er ungleich 0 oder 1, spuckt das Programm eine "Illegal function call " aus. Außerdem darf der Aufruf nur im hochauflösenden Graphikmodus erfolgen.

Es werden drei "USR"-Funktionen benötigt. "USR1(X)" gibt das Bild X wieder frei, das heißt, wenn Sie ein Bild abgespeichert haben, ist es automatisch durch den Speicherungsvorgang schreibgeschützt. "USR2(X)" speichert das aktuelle Bild unter der Nummer X in die PGE 21. Voraussetzung dafür ist, daß das Bild mit "USR1(X)" freigegeben wurde, da sich der Computer ansonsten mit einem "Illegal function call" bei Ihnen beschwert. "USR3(X)" lädt das Bild X in den aktuellen Bildspeicher. Dabei wird das aktuelle Bild zerstört.

Das nun folgende BASIC-Programm setzt voraus, daß sich das Maschinenprogramm ab D000H im Speicher befindet.

```
100 REM Demoprogramm fuer das
110 REM Maschinenprogramm zum
120 REM Speichern und Laden von
130 REM Bildern
140 REM
150 REM Einsprung fuer Bild freigeben
160 DEFUSR1=&HD000
170 REM Einsprung fuer Bild speichern
180 DEFUSR2=&HD003
190 REM Einsprung fuer Bild laden
200 DEFUSR3=&HD006
210 REM 2 verschiedene Bilder zeichnen,
220 REM dann abspeicheren und ab-
230 REM wechslend anzeigen.
240 SCREEN1,0:COLOR14,0,0:CLS
250 LINE (0,0)-(127,191),7,BF
260 A%=USR1(0):REM Bild 0 freigeben
270 A%=USR2(0):REM Bild abspeichern
280 CLS
290 LINE (128,0)-(255,191),8,BF
300 A%=USR1(1):REM Bild 1 freigeben
310 A%=USR2(1):REM Bild abspeichern
320 CLS
330 REM Anzeigen der Bilder
340 A%=USR3(0):REM Bild 0 laden
350 FOR A%=0 TO 1024:NEXT
360 A%=USR3(1):REM Bild 1 laden
370 FOR A%=0 TO 1024:NEXT
380 GOTO 340:REM Bild 0 wieder laden
```

Ich hoffe, das BASIC-Programm ist Ihnen in seinem Aufbau und in seiner Funktion klar. Und nun zum Maschinenprogramm. Ich habe wie üblich meine Kommentare in das Listing geschrieben.

ERABLD: Gib Bild Nummer X frei. Diese Routine ruft zuerst die Routine "INIT" auf. Danach errechnet sie aus der Bildnummer die Adresse in der Attributtabelle (am Ende des MC-Prgrammes) und löscht die Eintragung dort. Danach geht sie in die Routine "RETBAS" über, die auf den BASIC-Interpreter (=PGE 01) zurückschaltet und zurückspringt.

ERROR: Diese Routine wird immer dann angesprungen, wenn das Programm einen Fehler entdeckt (Parameter falschen Typs, Parameter zu groß, falscher "SCREEN"-Modus, Bild schreibgeschützt). Sie schaltet sicherheitshalber auf den BASIC-Interpreter um und springt zu dem Teil im Interpreter, der die Fehlermeldung 'Illegal function call" erzeugt.

SPBLD: Speichert das Bild unter der Bildnummer X in die PGE 21. Auch hier wird wieder "INIT" aufgerufen, dann der Transfer VDP zu PGE 21 durchgeführt und zu "RETBAS" zurückgesprungen.

LDBLD: Lädt das Bild Nummer X in das Video-RAM. "INIT", Transfer PGE 21 zu VDP und "RETBAS" kennzeichnen die Routine.

INIT: Eine äußerst wichtige Routine. Sie holt die Bildnummer in den Akku und überprüft, ob der Computer sich überhaupt im hochauflösenden Graphikmodus befindet. Weiters gibt sie im Register HL die Startadresse des Bildes X und der PGE 21 zurück und schaltet auf die PGE 21 um.

RAM: Diese Routine ist mit "ROM" die wichtigste Routine des Programmes. Sie zeigt, wie man mit den Befehlen "AND" und "OR" des Z80 die Pages umschalten kann.

SP: Tranferiert BC-Bytes vom DE nach HL. DE zeigt dabei in das Videoram un HL in die PGE 21.

LD: Transferiert BC-Bytes vom HL nach DE. DE zeigt dabei wieder in das Videoram und HL in die PGE 21.

BC, DE und HL sind Register des Z80.

```
100 REM Ladeprogramm zum Demoprogramm
110 REM des Artikels 'Bank Switching'
120 REM
130 CLEAR200,&HCFFF:DEFINTA,B:DEFSTRC
140 RESTORE:A=&HD000:B=0
150 READC:IFC="**"THEN180
160 B=B XOR (VAL("&H"+C))
170 POKEA,VAL("&H"+C):A=A+1:GOTO150
180 IFB=236THENNEW
190 PRINT"Ladefehler
200 END
210 DATA C3,22,D0,C3,2E,D0,CD,4D,D0,11
220 DATA 00,00,CD,72,D0,11,00,20,CD,72
230 DATA D0,CD,9E,D0,FB,AF,C9,CD,9E,D0
240 DATA FB,C3,9E,0F,CD,4D,D0,21,B8,D0
250 DATA 85,6F,36,00,18,E7,CD,4D,D0,E5
260 DATA 21,B8,D0,85,6F,7E,36,01,A7,E1
270 DATA C2,1B,D0,11,00,00,CD,89,D0,11
280 DATA 00,20,CD,89,D0,18,C8,FE,02,C2
290 DATA 1B,D0,23,23,7E,FE,02,D2,1B,D0
300 DATA 21,00,00,A7,28,03,21,00,40,F5
310 DATA 3A,3A,FE,FE,01,C2,1B,D0,F3,CD
320 DATA AB,D0,F1,C9,7B,D3,B1,7A,F6,40
330 DATA D3,81,E3,E3,01,00,18,7E,D3,80
340 DATA 23,0B,78,B1,20,F7,C9,7B,D3,81
350 DATA 7A,D3,81,E3,E3,01,00,18,DB,84
360 DATA 77,23,0B,78,B1,20,F7,C9,F5,3E
370 DATA 0F,D3,88,DB,90,F6,02,D3,8C,F1
380 DATA C9,F5,3E,0F,D3,88,DB,90,E6,FD
390 DATA D3,8C,F1,C9,01,01,**
```

Am Ende dieses Artikels finden Sie auch noch ein kleines Ladeprogramm für die hexadezimalen Opcodes des MC-Programmes. Nachdem dieses seine Arbeit getan hat, können Sie das obige BASIC-Programm laufen lassen. Wenn Sie das Programm unterbrechen, bleiben die Bilder aber trotzdem in der PGE 21 erhalten. Sie können so aus einem Programm aussteigen, ohne vielleicht Ihr mühevoll aufgebautes Kunstwerk zu verlieren. Wenn Sie dann mit "SCREEN 1:COLOR 14,0,0:CLS:CONT" fortsetzen, werden beide Bilder wieder unversehrt zum Vorschein kommen.

Zur Illustration des Gesagten wird hier noch ein Beispiel gezeigt!

```
100 SCREEN 1,0:COLOR 14,0,0:CLS
110 DEFUSR1=&HD000
120 DEFUSR2=&HD003
130 DEFUSR3=&HD006
140 DEFINTA-Z
150 A=0:F=15:GOSUB 220
160 A=USR1(0)+USR2(0)
170 CLS:A=1:F=15:GOSUB 220
180 A=USR1(1)+USR2(1)
190 CLS:DEFINTA
200 IF A THEN A=USR3(0) ELSE A=1+USR3(1)
210 GOTO 200
220 B=A
230 FOR X=0 TO 15
240 LINE (X*16,0)-(X*16,191),F
250 LINE (0,X*12)-(255,X*12),F
260 NEXT
270 FOR Y=0 TO 15
280 FOR X=0 TO 15
290 IF A THEN PAINT (X*16+8,Y*12+8),F
300 A=1-A
310 NEXT
320 A=1-B:B=1-B
330 NEXT
340 RETURN
```

```
      SV-328 Z80 ASSEMBLER
      VERSION 4.0
      29.09.1984
      PHILIPP OTT

 ASSEMBLERTEXT GEFUNDEN
DURCHLAUF NR: 1

 KEINE ERRORS
 108 ZEILEN

DURCHLAUF NR: 2
ADDR  OPCODES  SYMBOLE  BEF.  OPERAND        KOMMENTAR

               MODE:    EQU   OFE3AH        ;Enthaelt SCREEN-Modus

                        ORG   OD000H        ;Start bei &HD000
D000  C322D0             JP    ERABLD       ;Einsprung Bild loeschen
D003  C32ED0             JP    SPBLD        ;Einsprung Bild speichern

D006  CD4DD0   LDBLD:    CALL  INIT         ;Einsprung Bild laden
                                            ;Bildnummer und Adresse holen
D009  110000             LD    DE,0000H     ;&H1800 Bytes fuer Pixels
D00C  CD72D0             CALL  LD           ;transferieren, dann
D00F  110020             LD    DE,2000H     ;&H1800 Bytes fuer Farbcodes
D012  CD72D0             CALL  LD           ;transferieren

D015  CD9ED0   RETBAS:   CALL  ROM          ;Umschalten auf ROM,
D018  FB                 EI                 ;Interrupts freigeben,
D019  AF                 XOR   A            ;Akku loeschen und
D01A  C9                 RET                ;zurueck

D01B  CD9ED0   ERROR:    CALL  ROM          ;Sicherheitshalber ROM
D01E  FB                 EI                 ;herschalten, Sprung zu
D01F  C39E0F             JP    0F9EH        ;'Illegal function call'

D022  CD4DD0   ERABLD:   CALL  INIT         ;Bild Nummer in Akku holen
D025  21B8D0             LD    HL,ATTR      ;[HL] mit Anfang Attribut-
D028  85                 ADD   A,L          ;tabelle laden, Eintragung Nr.
D029  6F                 LD    L,A          ;[A] loeschen
D02A  3600               LD    (HL),00      ;
D02C  18E7               JR    RETBAS       ;Ruecksprung zu BASIC

D02E  CD4DD0   SPBLD:    CALL  INIT         ;Hole Seitennummer und
D031  E5                 PUSH  HL           ;Adresse, rette Adresse,
D032  21B8D0             LD    HL,ATTR      ;pruefe, ob Bild Nr. [A]
D035  85                 ADD   A,L          ;schreibgeschuetzt ist
D036  6F                 LD    L,A          ;
D037  7E                 LD    A,(HL)       ;
D038  3601               LD    (HL),1       ;Schuetze auf alle Faelle
D03A  A7                 AND   A            ;das Bild Nr. [A]
D03B  E1                 POP   HL           ;hole Adresse vom Stapel,
D03C  C21BD0             JP    NZ,ERROR     ;Error, wenn Bild geschuetzt
D03F  110000             LD    DE,0000      ;speichere &H1800 Bytes fuer
D042  CD89D0             CALL  SP           ;Pixel, speichere dann
D045  110020             LD    DE,2000H     ;&H1800 Bytes fuer die Farb-
D048  CD89D0             CALL  SP           ;codes, springe zurueck
D04B  18C8               JR    RETBAS       ;zu RETBAS

D04D  FE02     INIT:     CP    02           ;Ist Argument Integer ?
D04F  C21BD0             JP    NZ,ERROR     ;Error, wenn [A] ungleich 2
D052  23                 INC   HL           ;
D053  23                 INC   HL           ;
D054  7E                 LD    A,(HL)       ;Lies Lowbyte des Arguments
D055  FE02               CP    02           ;sieh nach, ob [A] 0 oder
D057  D21BD0             JP    NC,ERROR     ;1 ist, Error, wenn nicht
D05A  210000             LD    HL,0000      ;Errechne Adresse aus [A]
D05D  A7                 AND   A            ;[HL] enthaelt die Start-
D05E  2803               JR    Z,BLD0       ;adresse fuer das Bild [A]
D060  210040             LD    HL,4000H     ;Adresse fuer Bild 1

D063  F5       BLD0:     PUSH  AF           ;rette Bildnummer auf Stapel
D064  3A3AFE             LD    A,(MODE)     ;ueberpruefe auf SCREEN-Modus
D067  FE01               CP    01           ;1, wenn nicht dann springe
D069  C21BD0             JP    NZ,ERROR     ;zu Error
D06C  F3                 DI                 ;sperre Interrupts, schalte
D06D  CDABD0             CALL  RAM          ;auf PGE 21 um
D070  F1                 POP   AF           ;hole Bildnummer vom Stapel,
D071  C9                 RET                ;zurueck

D072  7B       LD:       LD    A,E          ;Schreibe [DE] als Adresse
D073  D381               OUT   (129),A      ;an den Videochip
D075  7A                 LD    A,D          ;
D076  F640               OR    64           ;Bit 5 muss beim Schreiben
D078  D381               OUT   (129),A      ;immer gesetzt sein
D07A  E3                 EX    (SP),HL      ;Zeitverzoegerung bis VDP
D07B  E3                 EX    (SP),HL      ;ready
D07C  010018             LD    BC,1800H     ;Transferiere [BC] Bytes
```

```
D07F  7E       LD1:    LD   A,(HL)          ;von der PGE 21 zum
D080  D380             OUT  (128),A         ;VDP
D082  23               INC  HL              ;Pointer erhoehen,
D083  0B               DEC  BC              ;Zaehler = Zaehler - 1
D084  78               LD   A,B             ;LD1 solange, bis
D085  B1               OR   C               ;Zaehler = 0
D086  20F7             JR   NZ,LD1          ;
D08B  C9               RET                  ;dann zurueck

D089  7B       SP:     LD   A,E             ;Schreibe Adresse [DE] zum
D08A  D381             OUT  (129),A         ;VDP, Bit 5 muss beim Lesen
D08C  7A               LD   A,D             ;geloescht sein
D08D  D381             OUT  (129),A         ;
D08F  E3               EX   (SP),HL         ;Zeitverzoegerung
D090  E3               EX   (SP),HL         ;
D091  010018           LD   BC,1800H        ;&H1800 Bytes transferieren,

D094  DB84     SP1:    IN   A,(132)         ;von dem VDP in
D096  77               LD   (HL),A          ;die PGE 21 laden
D097  23               INC  HL              ;erhoehe Pointer
D098  0B               DEC  BC              ;Zaehler = Zaehler - 1
D099  78               LD   A,B             ;SP1 solange Zaehler
D09A  B1               OR   C               ;ungleich 0
D09B  20F7             JR   NZ,SP1          ;
D09D  C9               RET                  ;zurueck

D09E  F5       ROM:    PUSH AF              ;rette [AF] auf Stapel,
D09F  3E0F             LD   A,15            ;waehle Register 15
D0A1  D388             OUT  (88H),A         ;des Soundgenerators
D0A3  DB90             IN   A,(90H)         ;lies Inhalt,
D0A5  F602             OR   2               ;setze Bit 1 fuer ROM
D0A7  D38C             OUT  (8CH),A         ;schreibe [A] in Reg. 15
D0A9  F1               POP  AF              ;hole [AF] vom Stapel
D0AA  C9               RET                  ;zurueck

D0AB  F5       RAM:    PUSH AF              ;rette [AF] auf Stapel,
D0AC  3E0F             LD   A,15            ;waehle Register 15
D0AE  D388             OUT  (88H),A         ;des Soundgenerators
D0B0  DB90             IN   A,(90H)         ;lies Inhalt, setze
D0B2  E6FD             AND  11111101B       ;Maske um Bit 1 zu loeschen
D0B4  D38C             OUT  (8CH),A         ;schreibe [A] in Reg. 15
D0B6  F1               POP  AF              ;hole [AF] vom Stapel
D0B7  C9               RET                  ;zurueck

D0B8  01       ATTR:   DEFB 1               ;Attributtabelle fuer den

D0B9  01               DEFB 1               ;Schreibschutz der Bilder
                                            ;1 = Schreibschutz,
                                            ;0 = freigegeben
```

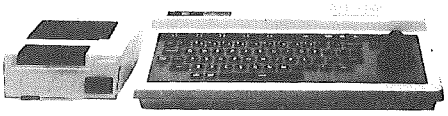
```
KEINE ERRORS
PRGRAMMBEGINN : D000
PROGRAMMENDE  : D0BA
PROGRAMMLAENGE:  186 BYTES


SYMBOLE:
MODE      : FE3A ,  65082      ; LDBLD     : D006 ,  53254
RETBAS    : D015 ,  53269      ; ERROR     : D01B ,  53275
ERABLD    : D022 ,  53282      ; SPBLD     : D02E ,  53294
INIT      : D04D ,  53325      ; BLDO      : D063 ,  53347
LD        : D072 ,  53362      ; LD1       : D07F ,  53375
SP        : D089 ,  53385      ; SP1       : D094 ,  53396
ROM       : D09E ,  53406      ; RAM       : D0AB ,  53419
ATTR      : D0B8 ,  53432

ENDE DER ASSEMBLIERUNG
```

Die Super-Computer.

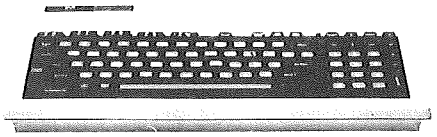
SVI



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 7.990,—**

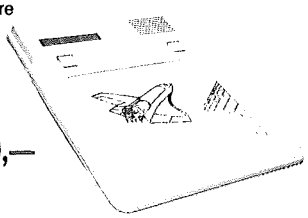
Super-Expander SVI-605, ein eingebautes Diskettenlaufwerk (160 K), Centroniks-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 14.990,—**

Super-Expander SVI-605 A, zwei eingebaute Diskettenlaufwerke (je 160 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 22.590,—**

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 29.990,—**

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 35.690,—**

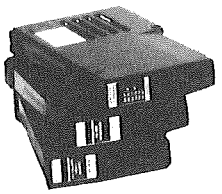
Grafik-Tablett SVI-105, 186 x 158 mm Zeichenfläche, Kassette mit Anwender-Software inkl. **öS 2.590,—**



Erweiterungskarten für SVI-605, A, B

SVI-803 16 K-Speicher-Erweiterung (für SVI-318) **öS 1.190,—**

SVI-805 RS 232, serielle Schnittstelle **öS 2.490,—**



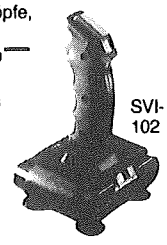
SVI-806 80-Zeichenkarte **öS 3.890,—**

SVI 807 64-K-RAM Speichererweiterung **öS 3.590,—**

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 7.690,—**

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:

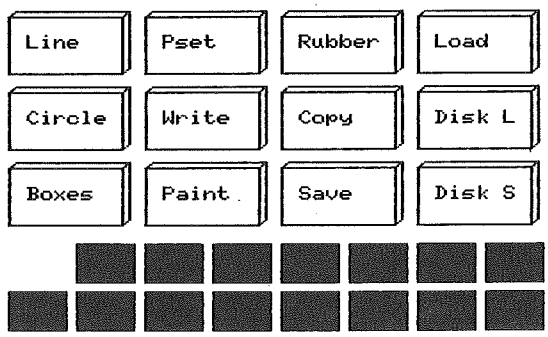
Großhandel, Facheinzelhandel **BASTL-COMPUTER-SYSTEME** **2700 Wr. Neustadt,** Hauptplatz 5 Tel. (0 26 22) 22 7 20 - 59 80 Telex 16 522	**DAHMS-PRAKTIKER-ELEKTRONIK** Pilgramgasse 11 **1050 Wien** Tel. (02 22) 54 34 21	**ESV ELEKTROTECHNISCHER SERVICE MBH.** Bayerhamerstraße 19—21 **5020 Salzburg** Tel. (06 62) 74 7 51	**SCHILLER MICRO-COM-BOUTIQUE** Fasangasse 21 **1030 Wien** Tel. (02 22) 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. (02 22) 65 73 42 Telex 132 827	**EDV-STUDIO PORSCH** Kinderspittalgasse 13 **1090 Wien** Tel. (02 22) 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. (02 22) 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             *
*                                       *
*****************************************
```

PAINTSTAR

Vielleicht kennen sie einige Grafikprogramme, die auf bekannten Personal-Computern laufen. Für den SVI-PC gibt es jetzt auch so ein Grafik-Softwaresystem. Nach Start des Programms erscheint ein Menüfeld (siehe folgende Abbildung). Nun kann man eines der vielen Felder mit dem Cursor ansteuern. Befindet man sich auf dem gewünschten Funktionsfeld, drückt man ganz einfach die Space-Taste und "ESC". Die "ESC"-Taste schaltet nämlich immer zwischen dem gezeichneten Bild und dem Menüfeld hin und her.



Welche Funktionen bieten sich nun an? Die erste und einfachste ist das ganz normale Zeichnen. Dies wird im "PSET"-Modus erreicht. Im Bild fährt man mit dem Cursor, der in Form einer Hand zu sehen ist, auf dem Schirm herum und zeichnet so die gewünschten Linien.

Da man manchmal Teile auch wieder löschen möchte, gibt es die "RUBBER"-Funktion. Sie radiert den Punkt am Bildschirm weg, der sich auf der Cursorposition befindet.

Sogar einen Zirkel und ein Lineal hat man mit diesem Programm zur Hand. Ersteres wird mit "CIRCLE" erreicht. Hier markiert man den Mittelpunkt des Kreises mit einem Druck auf die Space-Taste, wenn sich der Cursor auf dem gewünschten Point steht. Nun stellt man wieder mit dem Cursor den Radius ein und schon ist ein Kreis gezeichnet. Ähnlich muß man auch bei "LINE" den Anfangs- und Endpunkt der Linie markieren.

Mit "BOXES" wird es möglich, ganze Vierecke entstehen zu lassen. "PAINT" malt eine vorher definierte Fläche mit der gewählten Farbe aus (Markierungslinien müssen ausnahmslos die gleiche Farbe haben!). Mit Leichtigkeit kann auch am Bildschirm geschrieben werden. "WRITE" erfüllt diese Aufgabe.

Das Markieren und Definieren von Punkten geschieht immer mit der Space-Taste. Um einen anderen Modus einstellen zu können, drückt man immer zuerst die "ESC"-Taste. Dann folgen die oben erwähnten Schritte.

Selbstverständlich kann das Programm über Diskette wie Cassette gefahren werden. Dabei ist "SAVE" und "LOAD" für das Arbeiten mit Cassette verantwortlich. Die Disketten werden mit "DISK L" (wie Laden) und "DISK S" (wie Speichern) in Bewegung gesetzt. Die ganze Zeit über werden sie von einer kleinen Hand begleitet, über welche die einzelnen Parameter der Funktionen eingegeben werden können. Wenn Sie einen Drucker der Marke Epson, möglichst Type RX-80, besitzen, kann ganz leicht die vorher erstellte Grafik aufs Papier gebracht werden. "COPY" macht sich hier sehr nützlich und erspart dem Bediener jegliche Müh.

Bei komfortabler Nutzung des Diskettenlaufwerks wird das Directory ausgegeben und nach dem Namen gefragt. Jedoch die Krönung des Ganzen wird durch Verwendung des SVI-105 (Grafiktablet) erreicht, ist aber nicht nötig, da sowohl mit als auch ohne Grafiktablet ganz einfach Grafiken erstellt werden können. Wer sich aber doch dafür entscheidet, sein PAINTSTAR um eine wunderbare periphere Einheit zu erweitern, wird sicherlich nicht enttäuscht sein. Außerdem ist zu einer wirklich sinnvollen Anwendung einer Grafiksoftware eine so komfortable Eingabehilfe wie ein Grafiktablet unbedingt vonnöten. Auch im letzten SVI-Journal ist auf Seite 18 eine Zeichnung abgedruckt worden, die mit diesem Programm erstellt wurde.

Fazit: Diese Entwicklung ist sehr gut durchdacht, bis ins Kleinste ausgearbeitet und kann eine große Hilfe für ernsthafte Anwender sein.

```
*****************************************
*                                       *
*              RÄTSELECKE               *
*                                       *
*****************************************
```

Ob Sie es glauben oder nicht! Auch wir waren mit den ersten Rätseln nicht sonderlich glücklich. Grund genug für uns, unsere Rubrik einmal gründlich zu überarbeiten. Nach einer Folge Absens präsentiert sich nun unsere Rätselecke in neuem Kleid. Da wir glauben, daß südamerikanische Flüsse wenig in einer Computerzeitschrift zu suchen haben (aber Kreuzworträtsel nur aus Computerausdrücken sehr schwer zu erstellen sind), hat man sich entschlossen, folgende Art von Rätsel zu verwenden: Man nehme ganz einfach ein fehlerhaftes Programm, drucke den Gedankengang des Entwicklers aus und frage nach dem Fehler! Diese Art ist deshalb ungemein nützlich, weil es erstens viel mehr "kaputte" als lauffähige Programme gibt (Themenmangel gibt es bei dieser Form hoffentlich nicht), und zweitens Anfänger von den Fehlern anderer lernen können.

Das erste Programm soll lediglich als Einstimmung dienen, die Fehler sind ganz einfach zu finden! Dieses Programm wurde übrigens rein aus dem Zweck konstruiert, die Fehler sind absichtlich drinnen!

Aus diesem Grund wäre es unfair, die Gedankengänge des Entwicklers hier wiederzugeben, weil diese ja schon sämtliche Fehler kennen!

Man soll ganz einfach annehmen, daß dieses Programm ein Feld von numerischen Variablen im Computer verarbeiten soll. Danach werden diese dimensionierten Variablen ausgedruckt. Das Ganze soll sich beliebige Male wiederholen lassen. Sie werden merken, daß nicht nur ein Fehler im Programm steckt!

```
10 DIM S (12)
20 FORT=0TO12
30 INPUT S(T):NEXT
40 FORT=0TO13
50 PRINT A(F):NEXT
60 GOTO 10
```

Viel Spaß beim Fehlersuchen! Im übrigen können Sie es uns mitteilen, wenn Sie den oder die Fehler gefunden haben! Das Programm ist erst dann gänzlich gelöst, wenn das Programm auch richtig läuft. Die bloße Ausmerzung der Fehlermeldung gilt nicht als Lösung. Im nächsten Heft kommt dann neben einem neuen Rätsel die Auflösung!

Wir sind Spezialisten für SVI-Computer

Sie erhalten daher bei uns immer die aktuellsten und ausführlichsten Informationen über diese leistungsstarken Computer.

Wir haben auch immer die aktuellsten Preise!

Rufen Sie uns einfach an, wenn Sie den Kauf eines SVI-Computers vorhaben. Wir senden Ihnen Ihre Bestellung gerne per Nachnahme.

Super-Expander SVI-605 und SVI-605A

Super-Expander SVI-605B
mit zwei doppelseitigen Laufwerken mit je 320 KByte, mit Disk-Controller und CP/M 2.2, mit Centronics-Schnittstelle und mit Softwarepaket (Wordstar, Mailmerge, CalcStar, DataStar, ReportStar).

Neu: EPROM-Programmierkarte (bis 27128)
I/O-Karte
Hardware-Uhr
Experimentierplatine

Alle Erweiterungen und Peripheriegeräte sind prompt lieferbar.

COBOL, FORTRAN, LISP, C

HE 15.–18. NOVEMBER 1984
WIENER MESSEPALAST

Halle E Stand 215

Turbo Pascal 2.0

angepaßt auf SVI-328 prompt lieferbar, mit Texteditor und ausführlichem Handbuch in deutscher Sprache nur S 1.890,-

TOOL BOX mit deutschem Handbuch S 1.890,-

Druckerwochen

Wir liefern weiterhin Markendrucker zu besonders günstigen Preisen:

EPSON BROTHER SILVER-REED STAR

Computer-Studio

PANIGLGASSE 18 · A-1040 WIEN · TEL.(0222) 65 88 93

IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Rudolf Bolek, Philipp Ott, Rafael Razim, Heinz Schmid, 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