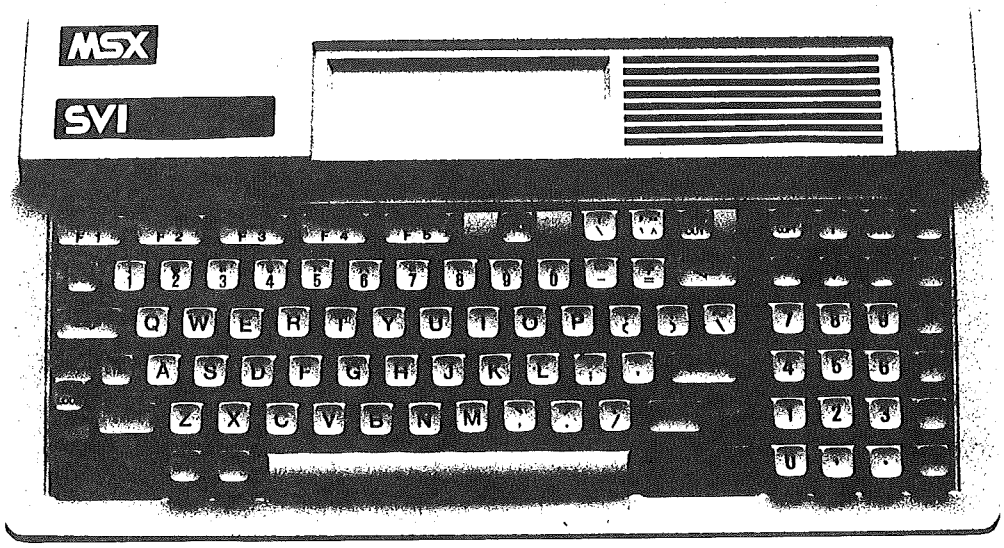
SVI JOURNAL

Die Zeitschrift des Spectra Video Club Austria



SVI·728

Testbericht

MSX

DEZEMBER 1984

INHALT

Heft 6/84 **S 15.-**

Liebes Clubmitglied!

Wir haben es doch tatsächlich geschafft, endlich als Club ganz offiziell bei einer Messe teilzunehmen. Die "Hobby-Elektronik" war ein voller Erfolg. Aus diesem Grund bringen wir auch eine Nachlese in diesem Heft, in dem Clubmitglieder, die nicht auf der Messe waren, einen kleinen Einblick bekommen können, wie es dort zugegangen ist. Von alleine stellt sich natürlich kein gutes Ergebnis ein. So halfen auch bei uns einige Leute dem Erfolg etwas nach. Dank gebührt allen jenen Mitgliedern, die ihre Freizeit opferten und sich am Stand platzierten, um Antworten auf alle möglichen Fragen zu geben und die Geräte zu bewachen. Es stellten sich so viele Personen zur Verfügung, daß rund um die (Messe)-Uhr ein volles Kontingent im Einsatz sein konnte.

Dank gebührt freilich auch jenen, die mit ihrer Software angerückt kamen und unseren Computern die nötige Intelligenz verliehen. Unter diesen fand sich übrigens auch ein SVi-Journal-Mitarbeiter, der den Sprachsynthesizer bearbeitete und ihn zum Sprechen brachte. Die Software samt Demo-Programm schrieb er zusätzlich noch für das sprechende "Kastl", sodaß es dann auf der Messe seine Weisheiten von sich geben konnte. Daß das allerdings nicht ohne leicht amerikanischen Akzent vonstatten ging, dürfte wohl klar sein. Verstanden hat man ihn aber, unseren Sprachsynthesizer!

Aber all dieser Einsatz unserer Clubmitglieder wäre umsonst gewesen, hätten wir keinen Platz zum Aufstellen unserer Geräte gehabt. Unsere Existenz auf der "Hobby-Elektronik" haben wir größtenteils der österreichischen Fachzeitschrift "itm praktiker" zu verdanken, die uns die Möglichkeit gab, quasi gratis als Messeausteller zu fungieren. Wir bekamen reichlichst Platz und konnten unsere mitgebrachten Plakate aufhängen. Wir hatten sogar soviel Freiraum, daß wir statt der drei vorgesehenen Computer vier aufstellen konnten.

Soweit die Teilnahme an der "Hobby-Elektronik"-Messe. Alles nähere steht in der oben erwähnten Nachlese im Inneren unserer Ausgabe.

Neu angekommen ist in Österreich der Spectravideo SVI-728. Dieser Computer ist dem MSX-Standard viel besser angepaßt, als der SVI-328. Dies war Anlaß genug für uns, diesen neuen Computer einmal etwas unter die Lupe zu nehmen. Welche Änderungen Spectravideo bei dem SVI-728 gegenüber dem SVI-328 durchgeführt hat, das steht in unserem Testbericht "SVI-728 kontra SVI-328". Dabei wollen wir natürlich keinen der beiden ausspielen, sondern lediglich die Unterschiede der beiden Modelle analysieren.

Die nächsten Clubabende:

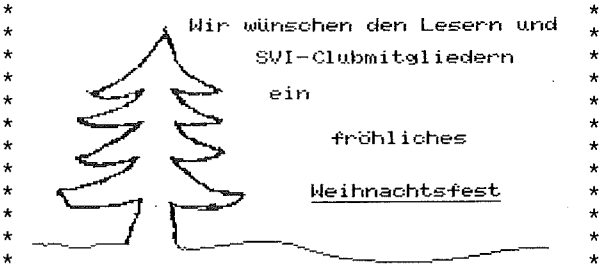
Mi, dem 2. Jänner 1985, ab 19 Uhr
Sa, dem 19. Jänner 1985, ab 17 Uhr
Mi, dem 30. Jänner 1985, ab 19 Uhr
Sa, dem 16. Februar 1985, ab 17 Uhr

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 Clubmitgliedern.

Es gibt übrigens seit Anfang November die Möglichkeit, sich auch schon für nächstes Jahr beim Club anzumelden. Das Gleiche gilt für die Abonnements unseres SVi-Journals.

Obwohl der Club erst im April dieses Jahres gegründet wurde, haben schon über 80 Spectravideo-Besitzer die Möglichkeit wahrgenommen, Mitglied zu werden. Natürlich gibt es auch Leute, die aus geographischen Gründen keine Clubmitglieder werden wollen, denn für einen SVI-User zum Beispiel in Innsbruck wäre es gelinde gesagt umständlich, bei unseren Clubabenden präsent zu sein. Aus diesem Grund bieten wir noch zusätzlich Abonnements für unsere Clubzeitschrift an. Davon haben immerhin 29 Leute Gebrauch gemacht.



Es ist kein Geheimnis mehr, daß in Österreich der SVI-Virus, auch oft MSX-Virus genannt, grassiert. So besitzen viele von diesem Virus befallene einen Spectravideo-Computer, suchen aber verzweifelt nach einem Echo in Form von anderen "Leidensgenossen". Sollten Sie so einen "armen" Menschen kennen, dann können Sie ihn "erlösen". Schenken Sie ihm doch einfach eine Clubmitgliedschaft oder SVi-Journal-Abonnement zu Weihnachten. Auch wenn diese Idee im ersten Moment seltsam anmutet! Bedenken Sie doch, um wie viel es besser ist, einem Club beizutreten, als sich haufenweise mit Büchern über Spectravideo herumzuschlagen. Ein Buch wird ja gerne als Weihnachtsgeschenk ausgewählt, warum nicht auch die viel nützlichere Clubmitgliedschaft.

Das Gleiche gilt natürlich für unsere Clubzeitschrift. Wir versuchen ja, ein möglichst reichhaltiges Programm über die Spectravideo-Computer anzubieten.

Daß wir ihnen dabei helfen, ihren Bekannten zu überraschen, ist klar. Sie kommen zu uns, und wir geben ihnen einen fertig ausgestellten Gutschein. Diesen Gutschein braucht der Beschenkte dann lediglich bei uns abzugeben. Schon ist er Mitglied oder SVi-Journal-Abonnent.

Bleibt noch eines für heuer übrig:

Das SVi-Journal-Team wünscht ihnen ein frohes Weihnachtsfest und einen guten Rutsch ins neue Jahr! Hoffentlich bleiben Sie uns ein treuer Leser!

Ihr SVi-Journal Chefredakteur Gerhard Fally!

Jetzt ist ein günstiger Zeitpunkt zur Bestellung eines Jahresabonnements. Bitte verwenden Sie den Abonnementbestellschein auf Seite 19.

Spectravideo-BASIC

Einige Befehle aus dem Kapitel 2.1 des USER-Manuals kennen wir schon. In dieser Folge lernen wir "ERASE" und "ERROR" kennen. In dem Zusammenhang stellen wir die Fehlertabelle der SVI-Computer einmal ganz richtig. Zuletzt schneiden wir noch den Befehl "FOR...NEXT"an.

Schon in der vorigen Folge wurde der Befehl "ERASE" erwähnt. Wir brauchen ihn, um von "DIM" erzeugte Felder wieder löschen zu können. Wie wir wissen, kann man keine Variable zweimal hintereinander, ohne "ERASE" dazwischen zu verwenden, dimensionieren. Welche Form hat diese Anweisung nun?

Die Syntax besteht lediglich aus dem Befehlswort und der Liste der Variablen, die gelöscht werden sollen. Demnach "vernichtet" "ERASE A,B" die beiden Felder A und B. Bedingung für diesen Befehl ist natürlich, daß die angegebene Variable vorher schon mit "DIM" Bekanntschaft gemacht hat. Ansonsten kann man sich an einer Fehlermeldung "Illegal function call" ergötzen.

Man wird es nicht glauben, aber es gibt beim Spectravideo einen Befehl, der auch dann immer einen Fehler generiert, wenn er richtig angewandt wird. Mit "ERROR" erzeugt sich der Bediener seinen Fehler selber. Hinter das Befehlswort wird einfach eine Zahl gesetzt. Diese Nummer ist der Code für die entsprechende Fehlermeldung. Die nachfolgende Tabelle soll endlich Klarheit schaffen, welche Zahl für welchen Error verantwortlich ist.

Man kann diesen Befehl übrigens auch im Direktmodus verwenden. Dann darf man aber nicht erschrecken, wenn zum Beispiel "Illegal Direct" (bedeutet, daß Befehl nicht im direkten Modus gebraucht werden darf) auf dem Bildschirm erscheint. Bei "ERROR 12" wird nämlich diese Fehlermeldung erzeugt, obwohl eigentlich kein Fehler vorliegt. Bleibt nur noch die Frage offen, wozu das Ganze?

Ohne Zweifel findet dieser Befehl dann Anerkennung, wenn es darum geht, einen Fehlerbehandlungszweig auf seine Richtigkeit zu testen. Es ist nämlich manchmal ziemlich schwierig, einen bestimmten Fehler zu erzeugen, um überhaupt in diese Routine zu kommen. Sicher, einen "Syntax-Error" erzeugt man fast blind, aber wie steht es mit "Internal Error" oder mit "String formula too complex"? Man kann sich natürlich herumspielen, bis man so eine Fehlermeldung angezeigt sieht, aber schneller geht dies eindeutig mit "ERROR".

Kommen wir nun zur Fehlertabelle:

1) NEXT without FOR
   Für die Variable im "NEXT" wurde kein "FOR" ausgeführt.

2) Syntax Error
   In der Zeile wurde eine Zeichenfolge benutzt, die kein Befehl ist, oder ein Befehl wurde mit einer falschen "Grammatik" verwendet (statt Beistrichen Semikolons oder umgekehrt und so weiter). Zum Beispiel: CIRCLE (34,45;2 (es fehlt eine Klammer hinter 45, statt des Semikolons gehört ein Beistrich)

3) RETURN without GOSUB
   Es wurde vor dem "RETURN" kein "GOSUB" durchgeführt.

4) Out of DATA
   Es ist für das "READ" keine "DATA"-Anweisung vorhanden oder die Daten reichen nicht aus, um alle Variablen im "READ" zu versorgen.

5) Illegal function call
   Ein Argument ist außerhalb des Bereichs oder hat den falschen Typ.

6) Overflow
   Das Ergebnis einer Berechnung ist größer oder kleiner, als der Zahlenbereich des Computers zuläßt.

7) Out of memory
   Das Programm oder die Daten verbrauchen mehr Speicherplatz, als vom Computer zur Verfügung gestellt werden kann. Es können jedoch auch zu viele "GOSUB"-Verschachtelungen, "FOR...TO"-Schleifen oder zu komplexe Ausdrücke vorhanden sein.

8) Undefined line number
   Es soll eine Zeilennummer verarbeitet werden, die nicht vorhanden ist, zum Beispiel im "GOSUB", "GOTO" und im "IF..THEN"-Befehl.

9) Suscript out of range
   Index ist entweder zu hoch definiert oder höher als mit "DIM" reserviert.

10) Redimensioned array
    Eine Feldvariable wurde neu dimensioniert, obwohl noch kein "ERASE" vorher verwendet wurde.

11) Division by zero
    Eine Zahl hätte durch Null dividiert werden sollen (ist mathematisch nicht sinnvoll).

12) Illegal direct
    Ein Befehl, der im direkten Modus verboten ist, wurde direkt verwendet.

13) Type mismatch
    Eine Variable wurde verwendet, obwohl sie einen falschen Typ hat (zum Beispiel String statt numerisch).

14) Out of string space
    Der Stringspeicher ist zu klein. Man muß ihn mit "CLEAR" erweitern, oder die verwendeten Strings kürzen.

15) String too long
    Es wurde versucht, einen String zu erzeugen, der länger als 255 Zeichen ist.

16) Stringformula too complex
    Zu komplizierter Stringausdruck (zu viele Klammern, Funktionen und so weiter).

17) Can't continue
    Ein Programm wurde nach dem letzten Stoppen geändert und ein "CONT" versucht.

18) Undefined user function
    Vor einem "USRx"-Aufruf wurde kein "DEFUSRx" ausgeführt.

19) Device I/O error
    Zwischen dem Computer und einem Peripherie-Gerät kommt es zu "Verständigungsschwierigkeiten" (fehlerhafte Datenübertragung).

20) Verify error
    Fehler beim Daten-Überprüfen eines Peripherie-Gerätes.

21) No RESUME
Eine Fehlerroutine wurde nicht mit "RESUME" abgeschlossen.

22) RESUME without error
Der Computer hat ein "RESUME" gefunden, obwohl kein Fehler vorhanden ist.

23) keine Fehlermeldung vorhanden, daher "Unprintable error".

24) Missing operand
Operand fehlt.

25) Line buffer overflow
Eingegebene Zeile hat zu viele Zeichen.

Von Code 26 bis Code 49 entsteht "Unprintable error", da keine ausdrücklichen Fehlermeldungen vorhanden sind.

50) FIELD overflow
Die Anzahl der Bytes im FIELD ist größer als die Satzlänge der Datei.

51) Internal error
Interner Fehler im Computer.

52) Bad file number
Unter dieser Nummer wurde keine Datei geöffnet oder die Nummer überschreitet die in MAX-Files festgelegte Zahl.

53) File not found
Auf der eingelegten Diskette existiert dieses File nicht.

54) File already open
File ist schon geöffnet.

55) Input past end
Die Datei hat noch keinen Inhalt, oder das "EOF", beziehungsweise das "LOF" wurde nicht beachtet.

56) Bad file name
Im Dateinamen sind falsche Zeichen oder die Form des Dateinamens ist falsch.

57) Direkt statement in file
In der soeben hereingeholten Datei (im ASCII-Code) ist eine direkte Anweisung zu finden.

58) Sequential after PUT
Nach "PUT" wurde eine sequentielle Operation durchgeführt.

59) Sequential I/O only
Man darf nur sequentielle Operationen durchführen.

60) File not OPEN
File noch nicht geöffnet.

Ab Code 61 sind die Fehlermeldungen des Disk-BASIC abgelegt. Diese Fehlermeldungen sind nur dann für den User vorhanden, wenn das Disk-BASIC verwendet wird, weil sie nicht im ROM abgelegt sind.

61) Bad allocation table
Die Namenstabelle der Dateien einer Diskette ist fehlerhaft, das Retten der Files ist nur durch Kopieren der Dateien auf eine andere Disk möglich, oder man kann die Namenstabelle auf der Disk reparieren.

62) Bad drive number
Falsche Laufwerknummer, bei SVI nur 1 oder 2 verwenden.

63) Bad track/sector
Es ist ein fehlerhafter Sektor vorhanden, oder eine falsche Eingabe liegt vor.

64) File still open
Datei ist noch offen.

65) Disk not mounted
Laufwerk ist noch nicht bereit.

66) Deleted record
Datensatz ist schon gelöscht, daher nicht mehr verwendbar.

67) File already exists
File existiert schon.

68) Disk full
Diskette hat keinen Speicherplatz mehr frei.

69) File write protected
Die Datei ist schreibgeschützt, ein Verändern der Datei mit dem Attribut "P" ist nicht mehr möglich.

70) Disk I/O error
Ein-/Ausgabefehler bei einer Diskettenoperation.

71) Disk offline
Angegebenes Laufwerk ist nicht angeschlossen.

72) Rename across disk
In der Anweisung "NAME" wurde in beiden Dateinamen eine Laufwerksnummer gefunden, dies ist verboten.

Ab Code 73 wird immer ein "Unprintable error" ausgegeben. 255 ist der höchste verwendbare Code, bei allen darüberliegenden Zahlen druckt der Computer einen (diesmal echten) "Syntax error" aus.

Wohl einer der wichtigsten Befehle, die es im BASIC gibt, ist der "FOR...NEXT"-Befehl. Diese Anweisung kann man für viele Dinge gebrauchen. Als erstes wollen wir die Frage klären, was diese Anweisung in ihrer Grundfunktion überhaupt macht.

Der "FOR...NEXT"-Befehl gliedert sich im rauhen Programmiereralltag in zwei Einzelbefehle. Als erstes wird immer zum Beispiel "FOR I=1 TO 4" geschrieben, danach erst "NEXT I". Zwischen diesen beiden Anweisungen können beliebig viele andere Befehle stehen. Das folgende einfache Programm soll uns dies verdeutlichen:

```
1 FOR I=1 TO 5
2 PRINT I
3 NEXT I
```

Wenn wir dieses Programm laufen lassen, dann erkennen wir die ersten fünf Ziffern unseres Zahlensystems Eins bis Fünf auf dem Bildschirm. In der Variablen I müssen also nacheinander die Werte von Eins bis Fünf eingespeichert werden. Der logische Schluß liegt nahe, daß diese Veränderung nur der "FOR...NEXT"-Befehl bewirken kann (weil sonst kein anderer Befehl im Programm steht).

Offensichtlich schreibt der Interpreter den Anfangswert, in unserem Fall 1, in die in der Anweisung angegebene Variable. So kommt es, daß beim nachfolgenden "PRINT" bei uns eine Eins angezeigt wird. Wenn der Computer das "NEXT" erreicht, springt er wieder zum "FOR" zurück, erhöht die Variable I um Eins und wir sehen eine Zwei. Dieses Spiel wird solange wiederholt, bis in der Variable I der Endwert 5 steht. Erst jetzt wird das "NEXT" ignoriert und der direkt hinter dem "NEXT" stehende Befehl bearbeitet. Vorher erhöht der Computer noch einmal die Variable um Eins. Wir können dies nachprüfen, indem wir im direkten Modus "PRINT I" eintippen. Wir sehen eine Sechs. Es stimmt!

In dieser Folge wird die Programmierung des Tongeneratorbausteins AY-3-8910 beschrieben. Dann wenden wir uns dem programmierbaren Ein-/Ausgabebaustein 8255 zu. Seine wichtigste Aufgabe ist die Tastaturabfrage.

Neben den bereits erwähnten Aufgaben des Tongeneratorbausteins ist natürlich die Ton- und Geräuscherzeugung die wichtigste. Das BASIC der Spectravideo-Computer kennt dazu drei Befehle: BEEP, PLAY und SOUND.

BEEP ist der einfachste Befehl. Er produziert einen kurzen Ton von voreingestellter Tonhöhe und Lautstärke. Es ist derselbe Ton, der eine Fehlermeldung begleitet. Will man ihn verlängern, so fügt man den BEEP-Befehl in eine FOR...NEXT-Schleife ein.

Der wichtigste Befehl, um Musik auf dem Spectravideo-Computer zu machen, ist PLAY. Mit den zugehörigen Makrobefehlen kann man die gewünschte Oktave festlegen, die Lautstärke, die Tonhöhe, Pausen usw. Eine sehr komfortable Art, um in BASIC Musik zu produzieren. Besonders angenehm ist die Möglichkeit, Musik im Hintergrund spielen zu lassen, also z.B. bewegte Grafik mit Musikbegleitung.

Man kann mit diesen Makrobefehlen beispielsweise auch Stringvariable in einen Tonanweisungsstring einbauen, die Periode der Hüllkurve bzw. die Hüllkurve selbst verändern.

Mehr und bessere Toneffekte erzielt man allerdings mit dem SOUND-Befehl, wobei allerdings höhere Anforderungen an den Programmierer gestellt werden, denn man schreibt mit diesem Befehl direkt in die Register des Tongeneratorbausteins. Dem Befehl SOUND müssen zwei Werte, getrennt durch ein Komma, folgen. Der erste Wert stellt die Registernummer dar, der zweite den Wert, der in das Register geschrieben werden soll. Die Syntax des Befehls lautet also:

SOUND PSG-Registeradresse, Datenbyte

Für die Musik- und Geräuschproduktion sind die Register 0 bis 13 zuständig. Die folgende Aufstellung gibt eine Übersicht über die Funktion der Register. Es empfiehlt sich, auch das in der vorhergehenden Folge dargestellte PSG-Blockdiagramm heranzuziehen.

PSG-Register-adresse	Funktion
0	Ton Kanal A, Feineinstellg. 0-255
1	Ton Kanal A, Grobeinstellg. 0-15
2	Ton Kanal B, Feineinstellg. 0-255
3	Ton Kanal B, Grobeinstellg. 0-15
4	Ton Kanal C, Feineinstellg. 0-255
5	Ton Kanal C, Grobeinstellg. 0-15
6	Rauschgenerator 0-31
7	Mischer

B7	B6	B5	B4	B3	B2	B1	B0
unbe-nutzt		Rauschen aktivieren auf Kanal C	B	A	Ton aktivieren auf Kanal C	B	A

Die jeweilige Funktion wird durch Nullsetzen des entsprechenden Bits im Register 7 erzielt.

8	Lautstärke Kanal A 0-15 für Hüllkurven 16
9	Lautstärke Kanal B 0-15 für Hüllkurven 16
10	Lautstärke Kanal C 0-15 für Hüllkurven 16
11	Periode der Hüllkurve Fein 0-255
12	Periode der Hüllkurve Grob 0-255 (Bereich 0 bis 65535)
13	Festlegung der Hüllkurve

Wert	Form
0,1,2,3,9	
4,5,6,7	
8	
10	
11	
12	
13	
14	
15	

Der Befehl PLAY "S<n>" hat übrigens die gleiche Wirkung wie der Befehl SOUND 13,<n>.

Das folgende Beispiel soll die Verwendung des Befehls SOUND illustrieren:

```
10 FOR I = 1 TO 255
20 SOUND 7,254
30 SOUND 8,15
40 SOUND 0,I
50 NEXT
60 SOUND 8,0
```

Die produzierten Töne sollen nur auf Kanal A ausgegeben werden, das bedeutet, daß nur Bit 0 des PSG-Registers 7 auf Null gesetzt werden darf. Nun wird noch die Lautstärke für Kanal A voll aufgedreht (Register 8) und dann auf Kanal A der Ton ausgegeben.

Wie man sieht, lassen sich mit dem Tongeneratorbaustein AY-3-8910 viele Klangeffekte realisieren. Eine genaue Kenntnis seiner Register ist allerdings erforderlich.

Wenden wir uns nun dem programmierbaren Parallelbaustein 8255 zu.

Spectravideo-BASIC

Fortsetzung von Seite 4

Diese Art der Befehlskonfiguration nennt man eine Schleife. Der "FOR...NEXT"-Befehl bearbeitet den Programmteil zwischen den beiden Teil-Anweisungen so oft, bis er die Variable vom Anfangswert bis zum Endwert erhöht hat. Wenn der Anfangswert Eins ist, dann wird diese Schleife genauso oft durchgeführt, wie der Endwert hoch ist, in unserem Beispiel fünf mal.

Die restlichen Geheimnisse von "FOR...NEXT" werden in der nächsten Folge gelüftet. Aber nicht nur diese Anweisung lernen wir anwenden, sondern auch den mindestens ebenso wichtigen Befehl IF...THEN...ELSE" und einige andere.

Der 8255 wurde speziell für 8080- und Z80-Mikrocomputersysteme entwickelt. Er enthält drei 8 Bit-breite Ports (A, B und C) die unabhängig voneinander als Ein- oder Ausgänge betrieben werden können. Während die Ports A und B immer nur als 8 Bit-Ports verwendet werden können, besteht beim Port C die Möglichkeit, die unteren und oberen 4 Bit jeweils getrennt als Ausgang oder Eingang zu benutzen.

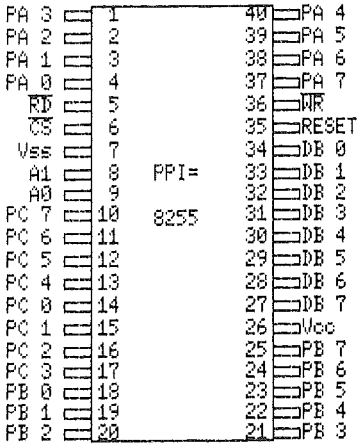
Vor der Datenein- bzw. Ausgabe muß der 8255 programmiert werden. Es muß festgelegt werden, welche Ports auf Ausgabe und welche auf Eingabe programmiert werden sollen. Dies geschieht mittels eines Befehlswortes, das mit der Adreßkombination A0=1 und A1=1 ausgegeben werden muß. Dieses Kontrollwort wird in das interne Befehlsregister geladen und legt damit die Betriebsart des 8255 fest.

Beim SVI-318/328 wird der 8255 zur Tastaturabfrage, für die Abfrage des Feuerknopfs der Joysticks, die Abfrage des Grafiktabletts und die Kontrolle des Kassettenrecorders verwendet. Die Ports werden wie folgt programmiert:

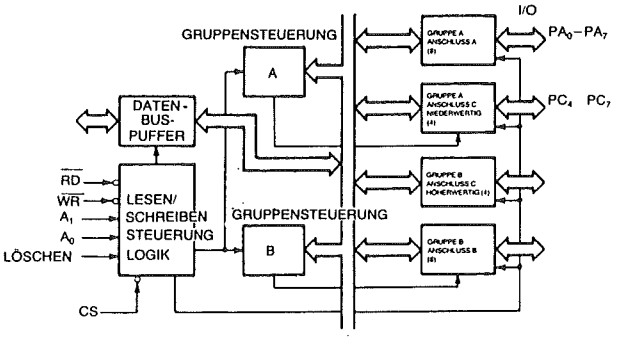
Port A: Eingabe. Bits 0 bis 3 für das Grafiktablett, Bits 4 und 5 für die Feuerknöpfe der Joysticks (0 = gedrückt, 1 nicht gedrückt), Bits 6 und 7 für Signale vom Kassettenrecorder(Bit 6 0=Ready, Bit 7 Read Data).

Port B: Eingabe. Tastaturabfrage.

Port C: Ausgabe. Bits 0 bis 3 geben BCD-Werte zur Abfrage der Tastaturmatrix aus.



Anschlußbelegung des 8255



Blockschaltbild des 8255

Bits 4 bis 6 geben Signale an den Kassettenrecorder aus (Bit 4 0=Motor ein, 1 = Motor aus, Bit 5 Write Data). Bit 7 wird zum Mischen der PSG-Tondaten verwendet.

Die Programmierung der Ports erfolgt durch die Systemsoftware. Das Kontrollwort bei der Initialisierung ergibt sich wie folgt:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	0

$\overline{CS}$	A1	A0	$\overline{RD}$	$\overline{WR}$	FUNKTION	
0	0	0	0	1	ANSCHLUSS A NACH DATENBUS	MPU
0	0	1	0	1	ANSCHLUSS B NACH DATENBUS	LESEN
0	1	0	0	1	ANSCHLUSS C NACH DATENBUS	(A,B,C)
0	0	0	1	0	DATENBUS NACH ANSCHLUSS A	MPU
0	0	1	1	0	DATENBUS NACH ANSCHLUSS B	
0	1	0	1	0	DATENBUS NACH ANSCHLUSS C	SCHREIBEN
0	1	1	1	0	DATENBUS NACH STEUERUNG	
0	1	1	0	1	ILLEGAL	
1	-	-	-	-	DATENBUS NACH DREIFACHSTATUS	(VERHINDERN)

Adressierung des 8255

Die wichtigste Aufgabe, die der 8255 bei den Spectravideo-Computern zu erfüllen hat, ist die Abfrage der Tastatur. Die Reihen der Tastaturmatrix werden mit 0 bis 10 abgefragt, ist eine Taste in der entsprechenden Reihe gedrückt, so wird in Port B der dieser Taste entsprechende Wert eingelesen.

	PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7
	)	!	@	#	$	%	^	&
0	0	1	2	3	4	5	6	7
	*	(	;	"	<	+	>	?
1	8	9	:	'	,	=	.	/
2	-	A	B	C	D	E	F	G
3	H	I	J	K	L	M	N	O
4	P	Q	R	S	T	U	V	W
				{	~	}	BS	
5	X	Y	Z	[	\	]	⇐	↑
6	SHIFT	CTRL	LGR	RGR	ESC	STOP	ENTER	←
	F6	F7	F8	F9	F10			
7	F1	F2	F3	F4	F5	CLS	INS	↓
8	SP	TAB	DEL	CAPS	SEL	PRINT		→
9	0	1	2	3	4	5	6	7
10	8	9	+	-	*	:	.	,

Die Reihen 9 und 10 betreffen den abgesetzten Zehnerblock.

Von BASIC her könnte die Abfrage einer gedrückten Taste wie folgt durchgeführt werden:

```
OUT &H96,R : T = INP (&H99)
```

Das Port C erreicht man über die I/O-Adresse &H96, Port B über &H99.

Auf diese Weise können übrigens auch die Tasten SELECT und PRINT, die sonst keine Funktion haben, in ein Programm eingebaut werden.

Z 80 - Programmieren in Assembler

Assemblerfortsetzungskurs Nr.:006

Wie in der letzten Ausgabe versprochen wurde, werde ich einige Programmbeispiele für bedingte Sprünge bringen. Als Erstes werden wir als schönes Beispiel eine Multiplikation kennenlernen. Es soll die Zahl 3F mit 20 multipliziert werden. Dies soll so vor sich gehen, daß der Multiplikand so oft zu sich selber addiert wird, so groß der Wert des Multiplikators ist. Bei dem nun folgenden Programm wird das Ergebnis im Registerpaar HL abgelegt sein.

```
        LD   C,20       ; Lade C mit dem
                          Multiplikator.
        LD   E,3F       ; Lade den Multi-
                          plikand in das
                          Lowbyte des Re-
                          gisterpaars DE.
        LD   D,00       ; Lösche Highbyte
                          von DE. DE bein-
                          haltet jetzt den
                          Multiplikand.
        LD   HL,0000    ; Lösche HL.
        LD   A,C        ; Multiplikator
                          wird in den Akku
                          geladen.
loop:   ADD  HL,DE      ; Addiere zu HL den
                          Multiplikant.
        SUB  01         ; Erniedrige den
                          Akku um Eins.
        JP NZ loop      ; Wenn nach der
                          Subtraktion der
                          Wert im Akku un-
                          gleich Null war,
                          dann wiederhole
                          die Schleife und
                          springe zu 'loop'
```

In diesem Programm wird der Multiplikator als Schleifenzähler verwendet. Er wird bei jedem Durchlaufen der Schleife um eins erniedrigt. Danach steht im Zero-Flag, ob der Schleifenzähler (in diesem Fall der Akku) nach der Subtraktion den Wert Null enthält oder nicht. Ist der Schleifenzähler noch nicht auf Null, daß heißt, wenn der Wert des Akkus ungleich Null ist, dann wird zu der Adresse verzweigt, die das Label "loop" repräsentiert. In einer der ersten Folgen wurde schon berichtet, was ein Label ist. Doch wir wollen es hier noch einmal kurz erläutern.

Ein Label steht an Stelle eines beliebigen Wertes. Es kann am Anfang eines Assemblerprogrammes definiert werden, das heißt, dem Label kann irgendein Wert zugewiesen werden. Braucht man nun diesen Wert während eines Programmes, so ist es möglich dieses Label zu schreiben. Durch Labels lassen sich Programme sehr übersichtlich gestalten. Sie können jedoch auch vor Befehle gestellt werden. Dabei enthält das Label die Adresse, bei welcher der nachfolgende Befehl beginnt. Einige Assembler-Versionen erlauben die Verwendung von Labels nicht, da sie gleich die eingegebenen Befehle beim Drücken der Enter-Taste in den entsprechenden Maschinencode übersetzen. Diese Assembler werden "Line by Line-Assembler" genannt.

Auch in unserem Programm wird ein Label verwendet, nämlich vor dem Befehl "ADD HL,DE". Es ist nun nicht mehr nötig, von vornherein die Adresse zu wissen, ab welcher der Befehl beginnt. Der Befehl, mit dem sich Labels definieren lassen, lautet "EQU". In die linke Spalte wird das zu definierende Label geschrieben, in die Mitte "EQU" und daneben der Wert oder die Adresse, die es erhalten soll.

In den folgenden Programmen werde ich von nun an immer häufiger Labels statt Adressen verwenden, damit der Leser beim Ausprobieren eigene Adressen anstatt des Labels einsetzen kann, falls er die Programme mit einem Assembler ausprobieren will.

Doch die eben besprochenen "JP"-Befehle sind speicherspezifisch. Was heißt das nun? Man hat zum Beispiel ein Programm für den Speicherbereich C000 geschrieben. Will man jetzt dieses für C000 erstellte Programm in den Spicherbereich D000 verschieben, so sind alle Adressen hinter einem "JP"-Befehl auf den neuen Speicherbereich anzupassen, da diese Adressen absolut sind.

Nun besitzt der Z-80 aber auch Sprungbefehle, mit denen es möglich ist, relativ zu springen. Ein solcher Befehl wird "JR" ('Jump Relativ') abgekürzt. Hinter diesen Befehl wird nun im Zweierkomplement eine Zahl geschrieben, welche zum aktuellen Programmcounter dazuaddiert wird. Dabei lassen sich natürlich nur Sprünge im Bereich von -128 bis +127 Bytes ausführen. Der Vorteil liegt auf der Hand, man braucht keine absoluten Sprungadressen mehr zu ändern, weil sowieso relativ zum Befehlszähler gesprungen wird.

Ein weiterer Vorteil besteht darin, daß ein relativer Sprung nur zwei Bytes benötigt. Leider hat der relative Sprung auch Nachteile, die jedoch nicht sehr gravierend sind. So ist er zum Beispiel um zwei Taktzyklen langsamer als der "JP"-Befehl, in Sekunden ausgedrückt sind das aber nur ungefähr 0.5 Mikrosekunden.

Es ist ebenfalls möglich, bedingt zu springen, jedoch lassen sich mit dem "JR"-Befehl zwei Flags weniger abfragen als mit dem "JP"-Befehl. Es können nur das Zero- und das Carry-Flag getestet werden. Daher muß der "JP"-Befehl zu Hilfe gezogen werden, um das Parity- oder Sign-Flag zu prüfen. Da es nicht sehr leicht ist, sich die Sprungdistanz auszurechnen, ist hierbei die Verwendung von Labels angebracht. Bei den "Line by line-Assemblern" ist es nicht notwendig, die Entfernung im Zweierkomplement anzugeben. Man schreibt genauso wie beim "JP"-Befehl eine Adresse hinter den "JR"-Befehl. Der Assembler rechnet nun selbständig die Entfernung zu der angegebenen Adresse aus. Ist die Entfernung zu groß, so bekommt man eine Error-Meldung. Hat man nun ein Programm nur mit relativen Sprüngen erstellt, so kann man dieses Programm beliebig im Speicher verschieben, ohne Sprungadressen ändern zu müssen.

Es ist noch zu sagen, daß der Befehlszähler bereits die Adresse des nächsten Befehles nach dem "JR"-Kommando enthält, wenn das angegebene Byte im Zweierkomplement dazuaddiert wird. Genauer gesagt heißt das, daß sich die relative Entfernung auf den Beginn des nächsten Befehles bezieht. Wir wollen nun das vorhin erstellte Programm mit einem "JR"-Befehl und mit mehr Labels erstellen.

```
        LD   A,(multator)
        LD   E,A
        LD   A,(multkand)
        LD   D,00
        LD   HL,0000
loop:   ADD  HL,DE
        SUB  01
        JR NZ loop ; Opcode: 20 FB
```

Bei dieser Version stehen die Operanden in Speicherzellen, da es ja unsinnig ist, wenn

die Operanden variabel sind. Die Speicherzellen werden durch die Labels 'multator' und 'multkand' repräsentiert. Sie müssen natürlich am Beginn eines Programmes den Labels durch den Befehl "EQU" die Werte, beziehungsweise die Adressen zuweisen, bei welchen der Multiplikator und der Multiplikand abgelegt sind. Doch nun zum "JR"-Befehl. Der Opcode für den bedingten Sprung "JR NZ .." ist '20'. Danach folgt der Wert, um wieviele Bytes die CPU springen soll, wenn der Wert des Akkus ungleich Null ist. "ADD HL,DE" ist ein Byte lang, "SUB 01" zwei und "JR NZ loop" belegt ebenfalls zwei Bytes.

Damit wieder zum Befehl "ADD HL,DE" zurückgesprungen wird, muß insgesamt fünf Bytes nach hinten gesprungen werden. Minus Fünf sieht nun im Zweierkomplement so aus: FB. Daher benötigt man für 'jump relativ by not zero (="JR NZ") minus five bytes' folgende Bytekombination: '20' (Opcode für "JR NZ") und 'FB' (=-5).

In der folgenden Tabelle sind alle Befehle aufgelistet, die Flags verändern oder beeinflussen.

Beeinflussung des Übertragsflags (Carry) durch:

ADC A,s/ ADC HL,ss/ ADD A,s/ ADD DD,ss/ AND s/ CCF/ CP s/ DAA/ NEG/ OR s/ RLA/ RLCA/ RL m/ RLC m/ RR m/ RRC m/ RRA/ RRCA/ SBC A,s/ SBC HL,ss/ SCF/ SLA m/ SRA m/ SRL m/ SUB s

Beeinflussung des Subtraktionsflags:

auf "0" gesetzt durch ADD A,s/ ADC A,s/ AND s/ OR s/ XOR s/ INC s/ ADD DD,ss/ ADC HL,ss/ RLA/ RLCA/ RRA/ RRCA/ RL m/ RLC m/ RR m/ RRC m/ SLA m/ SRA m/ SRL m/ RLD/ RRD/ SCF/ CCF/ IN r,(C)/ LDI/ LDD/ LDIR/ LDDR/ LD A,I/ LD A,R/ BIT b,s.

auf "1" gesetzt durch SUB s/ SBC A,s/ CP s/ NEG/ DEC m/ SBC HL,ss/ CPL/ INI/ IND/ OUTI/ OUTD/ INIR/ INDR/ OTIR/ OTDR/ CPI/ CPIR/ CPD/ CPDR.

Beeinflussung des Paritäts/Überlaufs-Flags:

Das P-Flag durch AND s/ OR s/ XOR s/ RL m/ RLC m/ RR m/ RRC m/ SLA m/ SRA m/ SRL m/ RLD/ RRD/ DAA/ IN r,(C).

Das V-Flag durch ADD A,s/ ADC A,s/ SUB s/ SBC A,s/ CP s/ NEG/ INC s/ DEC m/ ADC HL,ss/ SBC HL,ss/ NEG.

Verwendet wird es intern durch LDIR/ LDDR/ LDI/ LDD/ CPI/ CPIR/ CPD/ CPDR.

Beeinflussung des Halbübertrags-Flags durch:

ADD A,r/ ADC A,s/ SUB s/ SBC A,s/ NEG/ AND s/ OR s/ XOR s/ INC s/ DEC m/ RLA/ RLCA/ RRA/ RRCA/ RL m/ RLC m/ RR m/ RRC m/ SLA m/ SR m/ SRL m/ RLD/ RRD/ DAA/ CPL/ SCF/ IN r,(C)/ LDI/ LDD/ LDIR/ LDDR/ LD A,I/ LD A,R/ BIT b,r/ CPI/ CPIR/ CPD/ CPDR/ CP s

Beeinflussung des Zero-Flags durch:

ADD A,s/ ADC A,s/ SUB s/ SBC A,s/ NEG/ AND s/ OR s/ XOR s/ INC s/ DEC m/ ADC HL,ss/ SBC HL,ss/ RL m/ RLC m/ RR m/ RRC m/ SLA m/ SRA m/ RLD/ RRD/ DAA/ IN r,(C)/ INI/ IND/ INIR/ INDR/ OUTI/ OUTD/ OTIR/ OTDR/ CPI/ CPIR/ CPD/ CPDR/ LD A,I/ LD A,R/ BIT b,s/ NEG/ CP s

Beeinflussung des Sign-Flags durch:

ADD A,s/ SUB s/ SBC A,s/ CP s/ NEG/ AND s/ OR s/ XOR s/ INC s/ DEC m/ ADC HL,ss/ SBC HL,ss/ RL m/ RLC m/ RR m/ RRC m/ SLA m/ SRA m/ SRL m/ RLD/ RRD/ DAA/ IN r,(C)/ CPI/ CPIR/ CPD/ CPDR/ LD A,I/ LD A,R/ NEG/ ADC A,s

Viele der in der obrigen Tabelle genannten Befehle sind noch unbekannt. Daher ist es ratsam sich diese Tabelle aufzuheben.

Doch so wie in allen anderen Programmiersprachen gibt es auch beim Z-80 die Möglichkeit, Unterprogramme zu verwenden. Unterprogramme sind Programmteile, die von verschiedenen Teilen eines Programmes mit Hilfe eines speziellen Befehles aufgerufen, das heißt angesprungen werden können. Nachdem dieses Unterprogramm abgearbeitet wurde, setzt die CPU mit dem Befehl fort, der nach dem Befehl steht, mit dem das Unterprogramm aufgerufen wurde. Der Befehl, mit dem Unterprogramme aufgerufen werden, ist der 'CALL'-(=engl.: rufen)-Befehl.

Jetzt tritt endlich der "Stackpointer" (engl.: Stapelzeiger) in Aktion. Dieser wird mit 'SP' abgekürzt. Trifft der Mikroprozessor während seiner Abarbeitung eines Programms auf einen 'CALL'-Befehl, so wird zuerst der momentane Wert des PC (Befehlszähler) gerettet. Dies geschieht dadurch, daß der 16-Bit lange PC auf den Stapel gelegt wird. Das heißt, daß die zwei Bytes in die Speicherzellen geschrieben werden, welche durch die Adresse im SP gekennzeichnet sind. Doch bevor der Prozessor dies vornimmt, wird der SP um zwei Bytes erniedrigt. Enthält der SP zum Beispiel die Adresse F100, und stößt der Mikroprozessor auf einen 'CALL'-Befehl, so wird wie erwähnt der SP erniedrigt. Er beinhaltet nun die Adresse F0FE (=F100-2). Danach wird das Lowbyte des PC nach (SP) geladen und das Highbyte nach (SP+1). Nach dem 'CALL'-Kommando steht nun genauso wie beim 'JP'-Befehl eine Adresse, zu welcher die CPU nach dem 'Retten' des Befehlszählers verzweigt. Es ist hierbei ebenfalls möglich, durch Abfrage von Flags bedingt Unterprogramme aufzurufen.

Ein Unterprogramm wird durch folgenden Befehl beendet: "RET" (=RETURN =engl.: Zurück). Bei diesem Befehl geschieht genau das Gegensätzliche wie beim CALL. Der Befehlszähler (PC) wird aus der Speicherzelle geladen, auf die der Stackpointer zeigt. Danach wird der Stackpointer noch um den Wert Zwei erhöht. Daraufhin setzt der Mikroprozessor bei jener Adresse fort, die nun im Befehlszähler (PC) steht.

Es lassen sich in einem Unterprogramm auch andere Unterprogramme aufrufen. Man kann sie beliebig oft verschachteln, nur muß auf den freien Speicherplatz geachtet werden. Wie zu sehen ist, wird der Stackpointer bei jedem "CALL" um den Wert Zwei erniedrigt und dann der Inhalt des PC hineingeschrieben. Man muß daher aufpassen, daß sich unter der Adresse, mit welcher der Stackpointer geladen wurde, genug Speicher frei ist. Endet zum Beispiel ein Programm bei der Adresse C000, und wird der Stackpointer mit der Adresse C005 geladen, so wird das Ende des Programmes überschrieben, wenn man mehrere Unterprogramme verschachtelt.

In der nächsten Folge werden Unterprogramme genauer erklärt und an Hand einiger Beispiele verständlich gemacht.

Tips & Tricks

Die interne Struktur des BASIC-Programmspeichers 6. Teil

Nachdem wir die Abspeicherung von Variablen hinter dem Programm kennengelernt haben, widmen wir uns nun dem praktischen Abschnitt unserer Serie. Wir üben zuerst anhand eines Programms unser Wissen und sehen uns danach die Bedeutung des Gelernten an.

Jeder von uns wird schon so ein Programm erstellt haben, wie unten angegeben. Daran ist nichts ungewöhnlich. Ungewöhnlich ist nur die Vorgangsweise, in der wir es nun betrachten wollen.

```
10 INPUT A:INPUT B$
20 B$=B$+LEFT("HALLO HALLO",A)
30 PRINT B$,SQR(A)
```

Dieses kleine Programm wird als "Hex-Dump" ausgedruckt. Dann sieht es folgendermaßen aus:

```
8000 00 0C 80 0A 00 85 41 3A
8008 85 42 24 00 2A 80 14 00
8010 42 24 F1 42 24 F3 FF 81
8018 28 22 48 41 4C 4C 4F 20
8020 48 41 4C 4C 4F 22 2C 41
8028 29 00 39 80 1E 00 91 20
8030 42 24 2C FF 87 28 41 29
8038 00 00 00 08 41 00 41 30
8040 00 00 00 00 00 03 42 00
```

Wir erkennen hier auf Adresse &H8000 den Anfangswert &H00. Danach folgen der Zeiger auf die nächste Zeile und die Zeilennummer der ersten Zeile. Nun fängt die erste eigentliche Programmzeile an. &H85 ist der Code für den "INPUT"-Befehl. Die ASCII-Codes für "A",":","B" und "$" folgen. Man kann daher deutlich "10 INPUTA:INPUTB$" erkennen. Die nächste Zeile ist genauso aufgebaut. Wieder stehen der Zeiger und die Nummer am Anfang. Danach folgen der eigentliche Inhalt. Interessant ist vielleicht, daß die beiden "HALLO"s direkt im ASCII-Code abgelegt sind. Die letzte Zeile ist ab &H8039 abgelegt.

Nach dem Programm sind die Variablen erkennbar. Wir sehen wieder das Schema, nach dem diese Speichereinheiten abgelegt sind. Zuerst wurde Zahl A abgelegt, danach die alphanumerische Variable B$. Von diesem String sind allerdings nur die Kennzahl und der Name "B" in unserem "Hex-Dump" sichtbar.

Welchen Nutzen hat nun dieses Wissen? Abgesehen davon, daß es immer gut ist, zu wissen, welche Tücken der Computer beim Abspeichern eines Programms aufweist, besteht ein großer Vorteil darin, "POKE" und "PEEK" mit den richtigen Adressen anzuwenden.

Die einzelnen Bediener verwenden ihr Wissen aber meist recht unterschiedlich. Während der eine sich spezialisiert, mit "NEW" gelöschte Programme aus den Speicherzellen zu kitzeln, speichert der andere mit Vorliebe Daten direkt im Speicher ab.

Wichtig ist auch noch bei gekoppelten BASIC-Maschincode-Programmen, daß die Werte, die bei jedem Wechsel der Programmiersprache übergeben werden, nicht just in dem Speicherbereich eingespeichert werden, wo der Computer seine Systemvariablen abgelegt hat. Denn dann würde man einen "richtigen" Absturz produziert haben.

Apropos Maschincode! Natürlich sollten die Programme selber auch nicht unbedingt bei &H8000 beginnen und das BASIC-Ladeprogramm während seiner Funktion killen. Wenn nämlich ab &H8000 eingespeichert wird, dann lädt das BASIC-Programm in seine eigenen Zeilen den Maschincode. Daß sich das Ladeprogramm nach kurzer Zeit nicht mehr auskennen und damit aussteigen wird, ist wohl klar!

Wie oben erwähnt, ist es aber dem Bediener überlassen, wie vielfältig er sein Wissen verwendet. Man kann zum Beispiel Programmzeilen ändern. Wenn man weiß, wie sie abgelegt sind, so kann man zum Beispiel ein "SQR" zum einem "TAN" umändern.

Es wird möglich, einen "Taschenrechner" ohne Schwierigkeiten zu programmieren. Der Bediener muß nur noch das eingeben, was er ausgerechnet haben will, der Computer speichert diese eingegebene Zeile in ein schon vorbereitetes Unterprogramm codiert ab (die Zeile muß aber vorher schon vorhanden sein!) und springt dann in dieses Unterprogramm. Dort wird die Zeile als "normale" BASIC-Zeile erkannt und mit Hilfe der Rechenfähigkeiten der Programmiersprache ausgerechnet. Nach dem Rücksprung aus dem Unterprogramm wird das Ergebnis angezeigt. Fertig!

Ebenso wird es möglich, hinter einem "INPUT"-Befehl die Variable beliebig zu ändern. Mit dem einfachen unten angeführten Programm können zum Beispiel hintereinander die Variablen A bis Z eingegeben werden.

```
10 FOR I=1TO26
20 POKE &H8024, I+64
30 INPUT A
40 NEXT
```

Das A hinter "INPUT" wird nach jedem Schleifendurchlauf geändert. Zuletzt steht ein Z hinter der Anweisung. Das Program hat sich geändert.

Nachteile hat das Ganze natürlich auch. Wenn man im Programm so ein "POKE" untergebracht hat, dann muß man sich nach jeder Programmänderung vergewissern, ob sich nicht die Adresse des zu wechselnden Bytes geändert hat. Dadurch wird das Programmieren natürlich mühsamer. Es gibt dann noch andere unwichtigere Nachteile.

Zusammenfassend kann man sagen: Es ist wichtig, die Speicherorganisation des Computers zu kennen, denn dadurch ist man in der Lage, in Ausnahmefällen, wenn das "normale" BASIC Schwächen zeigt, mit dem Befehl "POKE" auszuhelfen.

Der Spectravideo ist jedoch kein Computer, bei dem "POKE" eine Anweisung ist, die in jeder dritten Programmzeile vorkommt. Es soll ja noch immer Computer geben, bei denen man den Synthesizer und die Grafik mit "POKE"s aktivieren muß. Für unsereins undenkbar.

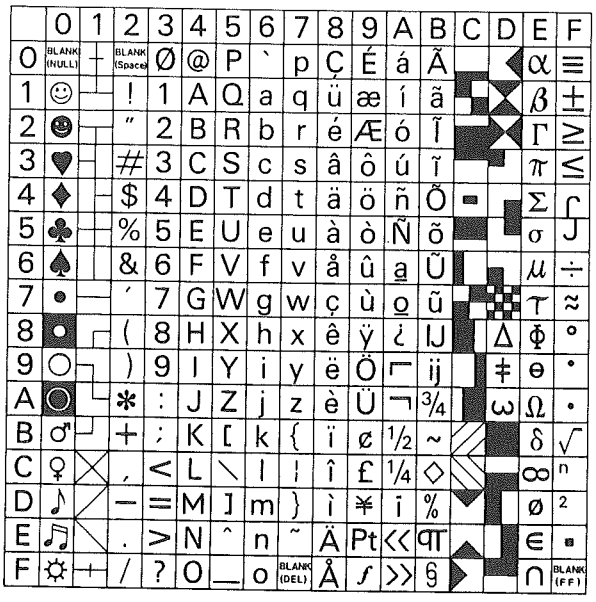
SVI-728 contra SVI-328

Seit kurzem geistert ein neuer SVI-Computer in den Medien herum. Der SVI-728 ist der erste "wirkliche" MSX-Computer von Spectravideo. Welchen Stellenwert er im Gegensatz zum SVI-328 einnimmt, welche Änderungen vorgenommen wurden, darüber mehr in diesem Artikel!

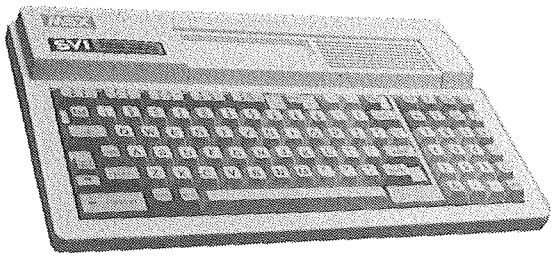
Vor etwa einem Monat wurde uns ein SVI-728 geschickt, mit der Seriennummer Zwei! Dies war Grund genug für uns, den neuen Spectravideo einem genauen Test zu unterziehen.

Wenn man nach langer Arbeit am SVI-328 den 728 in die Hände bekommt, dann muß man leider sofort bemerken, daß manche Tasten gegenüber der 3x8er-Serie geändert wurden. So wurde zum Beispiel der Doppelpunkt mit dem Strichpunkt vertauscht. Anstatt "Left"- und "Right Graph" wurde "Graph" und "Code" gewählt, warum, darüber später. Ebenso wanderte "CLS" und "Insert" auf die beiden Tasten "PRINT" und "SELECT". "PRINT" verschwand und "SELECT" wurde nach rechts verschoben. An der alten Stelle der beiden geänderten Tasten prangen nun neue Belegungen. So könnte man noch einige Tastaturänderungen aufzählen. Hat man sich jedoch erst einmal an die neuen Tasten gewöhnt, lernt man auch diese Aufteilung zu schätzen.

Bleiben wir noch beim Tastaturfeld. Wofür ist nun die Code-Taste dar? Man merkt bald, daß sie ebenso wie "Graph" Zeichen generiert. Mit der neuen Aufteilung haben wir viel mehr Zeichen als vorher zur Verfügung. Es sind nun nämlich folgende Kombinationen möglich: "GRAPH", "GRAPH" + "SHIFT", "CODE" und "CODE" + "SHIFT". Obwohl nicht alle Tasten immer belegt sind, ergeben sich nun atemberaubende Möglichkeiten. Deutscher Zeichensatz kann angewendet werden (vorerst noch nicht für den Drucker verwendbar!), ein "PI"-Zeichen steht zur Verfügung und einige altgriechische Buchstaben sind neben den üblichen Graphikzeichen abrufbereit. Zusätzlich gibt es dann noch einige weitere Sonderzeichen in diesen verschiedenen Paletten.



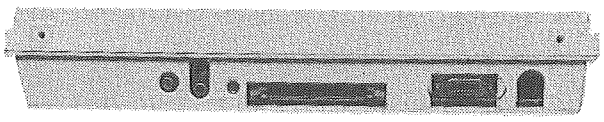
Zur Tastatur allgemein wäre noch zu bemerken, daß sie eine der Besten im MSX-Park ist. Nicht nur, daß sie eine robuste Schreibmaschinentastatur ist, sie hat auch noch einen getrennten Zehnerblock. Das zuletzt genannte Feature hat nämlich kein anderer MSX-Computer.



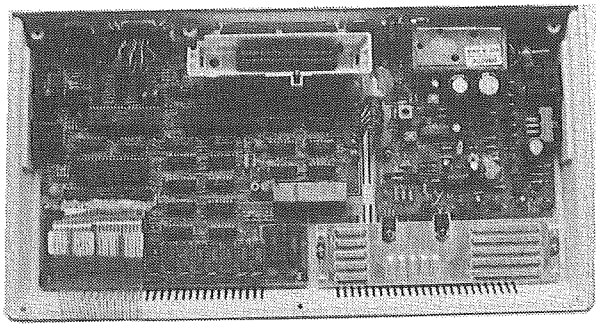
Als nächstes fällt dem SVI-328-Bediener auf, daß der Cartridge-Schacht in die Mitte gewandert ist und sich vergrößert hat. Dabei sind wir schon beim Kapitel Anschlußbelegungen. Auch hier hat sich einiges geändert.

Der längliche Schacht auf dem Computer ist nämlich nicht nur für die Cartridges zuständig, die Erweiterungskarten, welche ihren Platz früher im Expander fanden, werden nun ebenfalls in den Slot gesteckt. Das wären zum Beispiel die 80-Zeichenkarte oder eine Speichererweiterung. Auch die Floppies sind gewandert. Dort, wo der SVI-328 den Erweiterungsbus hatte, wurde nun der Anschluß für die Disketten gelegt. Der Floppy-Controller wurde in die Station verlegt, es ist somit leider nur noch eine Disk-Station anschließbar. Die hat dafür 320 KBytes Kapazität und bearbeitet wie gehabt 5,25 Zoll Disketten. Der Drucker ist an ein Interface anschließbar, welches sehr stark an die Centronics-Schnittstelle erinnert, mit ihr belegungsgleich ist, aber nicht die gleiche Form hat. Sie ist ebenso wie der Floppy-Anschluß an der Rückseite des Gerätes etwa an der Stelle, an welcher der SVI-328 das Kassetteninterface hatte.

Dieses wich der Drucker-Erweiterung und ist so links von ihm als runde Steckerleiste ausgeformt. Die Joystick-Anschlüsse sind lediglich auf die andere Seite des Gerätes gewandert. Dies hat seinen Grund darin, daß SVI-Bediener öfters beim Suchen nach dem Ein-/Ausschalter einen kleinen elektrischen Schlag bekamen, wenn sie über die Anschlüsse strichen. Einzig die Stromversorgung ist gleich geblieben, denn selbst die Bildschirmansteuerung hat sich geändert. Der SVI-728-User hat einen Audio- und Video-Anschluß für den Monitor und einen TV-Stecker. Zusätzlich ist noch ein Masse-Stecker ausgebildet.



Wie man an der Belegung der Schnittstellen gut erkennen kann, ist das Gerät ziemlich umsortiert worden. Dies kann man auch im Inneren sehen. Abgesehen davon, daß man das Drucker-Interface in den Computer verlegt hat, wurde auch dem PSG die Möglichkeit genommen, seine BANKS zu verwalten. Trotzdem sind noch immer vier BANKS vorhanden, nur werden sie anders gesteuert (vom 8255). Die Verwendung der Banken ist jedoch anders als beim SVI-318/328. Die BANK 0 beherbergt das BASIC-ROM, sonst nichts. Die zweite BANK bietet die 64 KBytes RAM und die dritte BANK ist ein Cartridge, das eventuell im Computer steckt. Die letzte BANK beherbergt unter anderem auch das Disk-Drive-ROM.



Sehen wir uns nun kurz den BASIC-Interpreter an. Auch er wurde umgemodelt. Da fast der gleiche Befehlssatz wie früher vorhanden ist, aber viel mehr Graphikzeichen abgelegt sind, liegt der Schluß nahe, daß viel dichter programmiert wurde. Denn das ROM bleibt 32 KBytes lang.

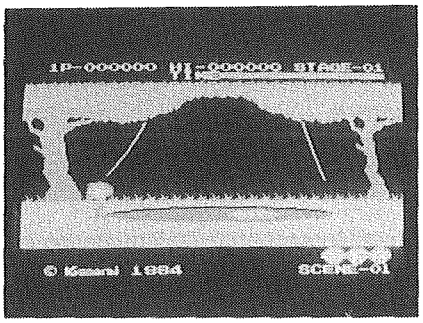
Ein weiteres Feature bietet das Video-Interface. Es ist nun möglich, in der Grundversion sowohl 40, 39 als auch 30 Zeichen pro Bildschirmzeile zu verwenden.

Wichtig sind auch noch die Erweiterungsmöglichkeiten. Dabei muß beachtet werden, daß kein Expander zur Verfügung steht. Eine Floppy-Disk kann ebenso wie ein Drucker unabhängig von anderen Erweiterungen existieren. Daß ein Kassettenrekorder auch immer dabei sein darf, ist ebenso klar, wie die zwei Joysticks und das Sichtgerät. Die restlichen Erweiterungen müssen aber nach einander verwendet werden. Eine Speichererweiterung zusammen mit einer 80-Zeichenkarte geht nicht. Von diesen Erweiterungskarten stehen also zur Wahl: eine 80-Zeichenkarte, eine 64 KByte-Erweiterung, eine RS-232 und ein Modem mit einer RS-232 (ist ebenso wie das Modem zum SVI-328 von der Post nicht erlaubt). Man kann aber sein Modem durch einen Akkustikkoppler mit einer RS-232 ersetzen.

Apropos Erweiterungen! Dadurch, daß der SVI-728 den MSX-Standard angehört, ist dem Bediener eine sehr große Palette von Video-Spielen offengelegt. Und diese Spiele haben ausgezeichnete Qualität. Wir hatten die Möglichkeit, zwei Spiele der Firma Kionami zu testen und glaubten uns in einen Zeichentrickfilm versetzt. Die Bewegung von Figuren war fließend, die Umgebung war nicht mit bunten Farbpunkten abqualifiziert, sondern erweckte den Eindruck einer fast naturgetreuen Landschaft. Abgesehen davon waren auch die Ideen originell. So handelte das eine von einer Leichtathletikmeisterschaft, während beim anderen durch ein unbekanntes Gebiet mit Abenteuern gewandert werden mußte.

Welche Konsequenzen bieten nun all die Veränderungen? Es ist klar, daß der neue SVI-728 dem MSX-Standard viel besser angepaßt ist, als der SVI-328. Man kann die Spiele des Philips MSX-8000 ebenso anstecken, wie die des Sony-Bit, um nur zwei Beispiele zu nennen.

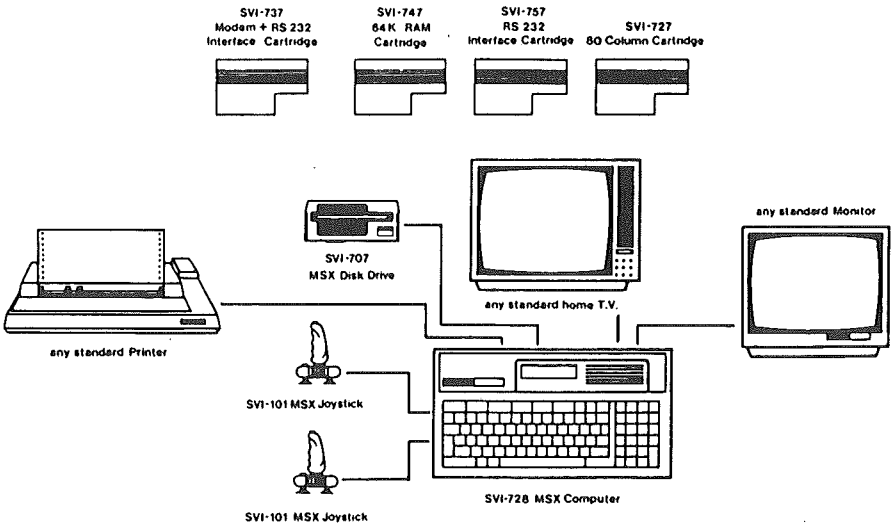
Die umfangreichere Anschlußbelegung läßt den Bediener zwar leichter und billiger Geräte anschließen, doch kann man am SVI-328 mehr Erweiterungen gleichzeitig verwenden.



Man wird bald erkennen können, daß Spectravideo mit dem SVI-728 einen sehr guten MSX-Computer geschaffen hat. Er ist wunderbar an den Standard gehalten und wird durch reichliche Software Spielernaturen sehr viel Freude bereiten. Dazu trägt natürlich auch die umfangreiche Zeichendarstellung bei. Es ist zusätzlich noch die Möglichkeit gegeben, auf Disketten und CP/M aufzurüsten, doch wird die volle Umstellung auf ein professionelles Gerät durch das Fehlen von zwei Floppy-Anschlüssen statt einem behindert.

Der SVI-328 hingegen bietet als Grundgerät ebenso Spielernaturen ein Angebot an Spielen, doch sind diese lang nicht so reichhaltig, wie für den SVI-728. Dafür hat jeder Geschäftsmann viel Freude, wen er zusammen mit dem Superexpander 605 B ein vollwertiges CP/M-kompatibles Gerät hat. Damit wird eindeutig klar, daß der SVI-328 für "ernsthafte" Anwender besser geeignet ist.

Der Preis für den SVI-728 beträgt bei uns derzeit 7990 Schilling, der SVI-328 ist für 6990 Schilling zu haben.



Hobby Elektronik-Nachlese

Wir haben ein Jubiläum zu feiern! Das erste Mal hat unser Club bei einer Messe ausgestellt. Die Möglichkeit dazu hat uns die österreichische Fachzeitschrift "itm praktiker" gegeben. Auf der "Hobby-Elektronik"-Messe bekamen wir einen Platz im sogenannten "Action-Center", welches die Redaktion des "itm praktiker" aufgestellt hat.

Mit vier Computern kreuzten wir auf der Messe auf, einer mehr als vorgesehen. Es stellte sich jedoch sehr bald heraus, daß alle vier genug zu tun hatten, mit nur drei hätten wir öfters einen Engpaß gehabt. Denn die zwei SVI-328 fanden anfangs fast ständig als CP/M-Vorführgeräte und Sprachsynthesizer-Ansteuerung Verwendung. Ein SVI-318 wurde für den Programmierwettbewerb eingesetzt und über den zweiten SVI-318 fielen die Video-Spieler her.

Apropos Video-Spiele! Neben altbewährten "Games" wie zum Beispiel das "Cross Force" stellten wir ein neues Video-Spiel namens "Sektor Alpha" vor. Der genaue Testbericht über diese neueste Entwicklung von Spectravideo folgt in einer der nächsten Folgen. Leider konnten wir in dieser Ausgabe noch keinen Bericht liefern, weil die Module innerhalb kürzester Zeit vergriffen waren. Es sei jedoch soviel verraten, auch im "Sektor Alpha" wird ordentlich geschossen, denn es hat zum Ziel, möglichst viele Flugobjekte vom Himmel zu holen.

Etwas ernster, aber mindestens genauso flott wie die Video-Spieler gingen die Messebesucher bei unserem Programmierwettbewerb ans Werk, hatten sie doch nur zwanzig Minuten Zeit, ein Programm zu schreiben. Am ersten Tag herrschte rege Teilnahme. So versuchte sich einer als Komponist. Ein anderer spielte sich mit unserem Befehl "LINE" und erzeugt so die wüstesten Muster. Den Trend zu Video-Spielen erfaßten zwei weitere Teilnehmer unseres Wettbewerbs. Sie heimsten die beiden ersten Preise ein.

Der eine entwarf ein Programm, daß eine Mondlandung simuliert. Dieser Teilnehmer muß übrigens ein Genie sein, denn ihm "floß" dieses Programm aus dem Handgelenk, als hätte er es auswendig gelernt (oder hat er es wirklich auswendig gelernt?). Egal, jedenfalls war er kein Spectravideo-Besitzer, und er schaffte es, innerhalb von zwanzig Minuten sein Programm auf den Computer anzupassen.

Nicht weniger "genial" erzeugte der andere sein Programm. Da dieses Programm sogar praktischen Wert hat, man kann es nämlich

öfters spielen, ohne die Lust zu verlieren, und die Ausarbeitung für zwanzig Minuten sehr sauber ist, bekam er den ersten Preis, das Buch "BASIC-Kompendium".

An den späteren Tagen war der Andrang beim Programmierwettbewerb etwas geringer, der Erfolg war aber auch in dieser Sparte gegeben.

Natürlich zeigten wir auch die "professionellen Seiten" unserer Geräte. Ein SVI-328 war eifrigst damit beschäftigt, immer wieder das Betriebssystem CP/M zu demonstrieren, und die dort anwesenden Clubmitglieder führten interessierten Messebesuchern unentwegt Programme a la "Wordstar" vor.

Etwas stiefmütterlich wurde der Synthesizer behandelt. Obwohl das dazu geschriebene Demo-Programm sehr gut war, wurde dem Sprachsynthesizer eher wenig Beachtung geschenkt. Einerseits ließ es sich nicht verhindern, daß das sprechende Kastl ohne amerikanischen Akzent "redete", und andererseits war es in der Umgebung unseres Standes doch relativ laut, wie man sich als eifriger Messebesucher lebhaft vorstellen kann.

Um möglichst vielen Leuten einen Einblick in unser Clubgeschehen zu geben, wurden Informationszettel verteilt, die man als regelmäßiger SVi-Journal-Leser kennen müßte. Sie waren nämlich in Ausgabe 5 in der Mitte des Heftes enthalten. Interessierten Besuchern wurde natürlich auch das SVi-Journal selber gegeben.

Da stellt sich nun die Frage, wenn alle Teile dieses Messebesuchs so erfolgreich abgelaufen sind, welchen Nutzen hat der Club aus dieser Messe gezogen. Um so etwas festzustellen, muß man immer die Kostenseite mit dem Nutzen gemeinsam betrachten. Bei uns sind die Kosten praktisch gleich null. Der Platz wurde uns von der Zeitschrift "itm praktiker" kostenlos zur Verfügung gestellt. Die Clubmitglieder halfen unentgeltlich und uns wurde nicht einmal ein Modul oder ein anderes Utensil gestohlen. (Andere Messeaussteller werden ja öfters von besonders intelligenten Leuten heimgesucht, die glauben, der Messerabatt beträgt für sie hundert Prozent!)

Auf der anderen Seite kamen direkt nach der Messe zehn neue Mitglieder zu unserem Club. Einige Messebesucher, vor allem aus den Bundesländern, entschieden sich für das SVi-Journal-Abonnement. Neben diesen augenfälligen Erfolgen gitb es natürlich noch einen verdeckten, der erst später zum Zug kommen wird. Unser Club wird mit jeder Messe und so auch mit der "Hobby-Elektronik" immer bekannter.

```
*****************************************
*                                       *
*            OLD FÜR SVI-328            *
*                                       *
*****************************************
```

Wie schon in den vorigen Ausgaben, so stellt auch dieses Programm eine Befehlserweiterung für den SVI-328 dar. Dabei benutzt der Autor auch diesmal den Befehl "CMD".

Mit Hilfe der kleinen Maschinenroutine ist es möglich, mit "NEW" gelöschte Programme wiederzubeleben. Natürlich darf man zwischen dem "NEW" und dem "CMD" keine Variablen definieren, da sonst das Programm beschädigt wird.

Wenn man den Computer einschaltet, lädt man am besten gleich den kleinen BASIC-Loader und läßt ihn ablaufen. Damit ist der Befehl einsatzbereit. Wenn man nun ein Programm versehentlich gelöscht hat, tippt man einfach "CMD" ein, und das Programm ist wieder da. Um vollkommen sicher zu sein, speichert man das Programm am besten vor dem ersten Ablauf auf Diskette oder Cassette und lädt es wieder. Dann ist das gelöschte Programm wieder einsatzbereit.

Nun ein paar Sätze zum Aufbau des Programms. Nach einem "NEW" wird der Speicher nicht gelöscht, sondern es werden nur die drei ersten Bytes auf Null gesetzt und die Zeiger für das Programmende und für die Variablen werden verändert. Dadurch ist eine Umkehrung des "NEW"-Befehls relativ einfach.

In den Zeilen 3 bis 12 sucht der Computer nach drei hintereinander liegenden Nullen im Programm. Drei Nullen kennzeichnen jeweils das Programmende bei den Spectravideo-Computern. Wenn der Rechner die Nullen gefunden hat, restauriert er den Zeiger für das Programmende (F7EE), für die einfachen Variablen (F7F0) und für die dimensionierten Variablen (F7F2). Dann muß der Computer noch das Ende der 1. Zeile suchen und auch diesen Zeiger richtigstellen (Zeilen 16-19).

Da eventuell auch einige andere Zeiger korrigiert werden müssen, ist es besser das Programm kurz abzuspeichern und wieder einzuladen.

RAFAEL RAZIM

BASIC-Ladeprogramm:

```
10 FORF=&HD000TO&HD000+42
20 READA$
30 A=VAL("&H"+A$)
40 POKEF,A
50 NEXT
60 POKE&HFF9C,&HC3:POKE&HFF9D,&HD:POKE&HFF9E
,&HD0
1000 DATA F3,21,07,80,01,00,80,3E,00,CD,B1,B
E,C2,09,D0,23,BE,C2,09,D0,23,22,EE,F7,22,F0,
F7,22,F2,F7,21,06,80,01,00,80,ED,B1,2B,01,80
,FB,C9
```

```
ZEILE ADDR  OPCODES  SYMBOLE  BEF. OPERAND        KOMMENTAR

   1                         ORG  0D000H
   2  D000  F3               DI
   3  D001  210780           LD   HL,08007H      ;HL mit Programmanfang laden
   4  D004  010080           LD   BC,08000H      ;BC mit Laenge Speicher laden
   5  D007  3E00             LD   A,00H          ;A mit 0 =Programmende laden

   6  D009  EDB1     LOOP1:  CPIR                ;Nachpruefen,
   7  D00B  BE               CP   (HL)           ;ob 3 Nullen
   8  D00C  C209D0           JP   NZ,LOOP1       ;hintereinander
   9  D00F  23               INC  HL             ;vorkommen
  10  D010  BE               CP   (HL)
  11  D011  C209D0           JP   NZ,LOOP1
  12  D014  23               INC  HL
  13  D015  22EEF7           LD   (0F7EEH),HL    ;wenn ja,dann
  14  D018  22F0F7           LD   (0F7F0H),HL    ;Zeiger restaurieren
  15  D01B  22F2F7           LD   (0F7F2H),HL
  16  D01E  210680           LD   HL,08006H      ;Suchen nach Ende
  17  D021  010080           LD   BC,08000H      ;der 1.Zeile
  18  D024  EDB1             CPIR
  19  D026  220180           LD   (08001H),HL    ;Zeiger auf Ende einstellen
  20  D029  FB               EI
  21  D02A  C9               RET                 ;Ruecksprung ins Basic

KEINE ERRORS
PRGRAMMBEGINN : D000
PROGRAMMENDE  : D02B
PROGRAMMLAENGE:  43 BYTES

SYMBOLE:
LOOP1      : D009 ,  53257

ENDE DER ASSEMBLIERUNG
```

```
*******************************************
*                                         *
*                 VALUE                   *
*                                         *
*******************************************
```

Sicherlich werden einige von Ihnen den SINCLAIR ZX Spectrum kennen, der mit seinem leicht exotischen BASIC keinen Anklang bei mir findet. Allerdings ist es in seinem BASIC möglich, mit der BASIC-Funktion VAL den Wert eines Ausdrucks, der in einem String gespeichert ist, zu berechnen (z.B. liefert auf dem Spectrum PRINT VAL "4*8" den Wert 32). Diese Möglichkeit von VAL wird man vergebens auf allen gängigen MICROSOFT-BASIC Versionen suchen. Deshalb habe ich mich tiefer mit dieser Funktion befaßt. Aber das Ergebnis war entäuschend. Es ist nicht so ohne weiteres möglich, den VAL-Befehl abzufangen oder ihn zu verändern. Deshalb erschien es mir sinnvoller, eine USR-Funktion zu definieren, die dasselbe wie VAL(vom Spectrum) erfüllt.

Im Anhang dieses Artikels finden Sie zwei BASIC-Programme, die Sie laufen lassen müssen um die Erweiterung zu erreichen. Beide Programme sind sowohl auf dem SVI-328 als auch auf dem SVI-318 lauffähig, da sie sich den Gegebenheiten anpassen. Die Programme müssen getrennt bleiben und auch die Reihenfolge beim Ausführen darf sich nicht ändern. Nachdem beide Programme gelaufen sind, ist das Maschinenprogramm sicher im Speicher und die Funktion USR(STR) entspricht der Funktion VAL(STR) vom Spectrum.

Sie können also schreiben:

```
PRINT USR("4*8")    liefert 32
oder
PRINT USR("4*ATN(1)") liefert 3.1415..etc.
oder
PRINT USR("-1=(1=0)") liefert 0 (=falsch).
```

Das Maschinenprogramm besteht aus zwei wichtigen Teilen. Der Einsprung bei der ersten ROM-Adresse verwandelt den String, wie er ja auch von der Tastatur oder dem Editor kommen könnte, in eine BASIC-Zeile, der aber sowohl die Zeilennummer als auch Zeiger fehlen (die werden eventuell erst nachher dazugegeben). Der zweite Einsprung erfolgt beim BASIC-Akkumulator oder besser, seiner Recheneinheit und dort wird der Ausdruck berechnet. Wenn das Maschinenprogramm zurückkommt, übergibt es an das BASIC eine doppeltgenaue Zahl.

Allerdings kann es vorkommen, daß der BASIC-Interpreter mit diesem Dazwischenpfuschen nicht fertig wird und der Computer deshalb abstürzt (keine Angst, tritt nur selten und vereinzelt auf). Sie sollten darauf achten, daß Sie keine Fehler im String dem USR mitübergeben, besonders keinen, der Sie sonst mit einem 'Illegal function call' aufschrecken würde. Ansonsten kommen saubere und klare Fehlermeldungen zurück (z.B. liefert PRINT USR("1/0") ein klares 'Division by zero').

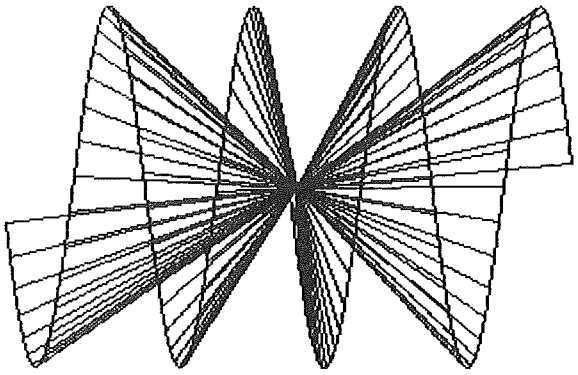
Das Maschinenprogramm liegt vollkommen sicher vor jedem destruktiven Charakter. Sie können es weder durch NEW noch durch CLEAR irgendwie angreifen. Nur die Definition können Sie mit DEFUSR verändern.

Diejenigen, welche nur mit einem Cassettenrecorder arbeiten, können die BASIC-Programme unverändert übernehmen. Auch die, welche mit Disketten arbeiten, müssen nicht auf meine Entwicklung verzichten. Sie ändern die CLOAD-Anweisung ganz einfach in ein LOAD"",R um und wenn Sie dann das erste Programm fahren, startet es automatisch das zweite Programm. Sie könnten sogar mit IPL den Aufruf der Programme auf Ihrer Systemdiskette verewigen und dadurch lästige Tipparbeit ersparen.

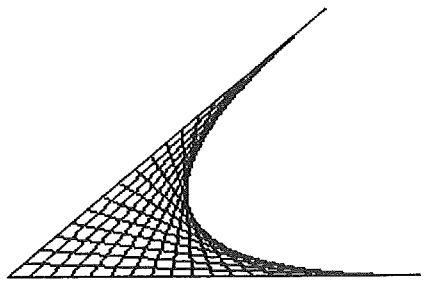
Philipp Ott

```
10 REM   Wert eines Stringaus-
11 REM   drucks berechnen.
12 REM   Autor: Philipp Ott
13 REM   Teil 1
14 POKE&HF54B,PEEK(&HF54B)OR1
15 POKE&HF54A,62:POKE(61+PEEK(&HF54B)*256),0
16 CLOAD
```

```
10 REM   Wert eines Stringaus-
11 REM   drucks berechnen.
12 REM   Autor: Philipp Ott
13 REM   Teil 2
14 DEFSNGA,B:DEFSTRZ:RESTORE
15 A=(PEEK(&HF54B)-1)*256:B=A
16 READZ:IFZ="*"THENDEFUSR=B:NEW
17 IFZ="++"THENZ=HEX$(PEEK(&HF54B)-1)
18 POKEA,VAL("&H"+Z):A=A+1:GOTO16
19 DATA FE,3,C2,5,9,23,23,5E,23,56,EB,4E,6,0
,23,5E,23,56,EB,C5,E5,D5,11,3A,++,ED,B0,EB,3
6,0,D1,E1,C1,11,0,0,21,3A,++,CD,44,B,21,4F,F
5,CD,CA,14,CD,65,57,3A,93,F7,2A,23,F9,C9,*
```



```
10 COLOR 0,0,0
20 SCREEN 1
30 FOR A=1 TO 250 STEP 2
40 T=T+.2:Y=96+96*SIN(T)
45 LINE-(A,Y)
46 COLOR 15,0,0
50 LINE(128,96)-(A,Y)
60 NEXTA
70 GOTO 70'by R. B.
```



```
1 COLOR 15,0,0
10 SCREEN1
20 X=20:Y=20
30 A=X+140:B=Y:C=X:D=Y+140
40 LINE(A,B)-(C,D),15
50 A=A-7:B=B+7:C=C+9
60 IF B>D THEN80
70 GOTO 40
80 GOTO 80'by R. B.
```

```
*****************************************
*                                       *
*          SPRITEGENERIERUNG            *
*                                       *
*****************************************
```

Wer kennt nicht die Sorgen bei Spriteprogrammen. Man will einige 16*16-Sprites definieren und muß dabei horrenden Speicherplatz verbrauchen. Eine Binär-Matrix, in der man die Form so eines Sprites sieht, braucht ungefähr 370 Bytes RAM. Dies ist eindeutig zu viel. Nun kann man zuerst mühsam die Form des Sprites erstellen, um dann die Matrix in Zahlenwerte umzuwandeln und die hart erarbeitete Matrix wegwerfen.

Eine weitaus einfachere Lösung ist es, ein eigenes Programm zur Generierung von Sprites zu verwenden. Man gibt einfach ein, ob man 8*8 oder 16*16 definieren will. Danach wird einem die Matrix präsentiert. Nun kann man mit den Cursor-Tasten, "1" und "0" die Gestalt des Sprites formen. Nach getaner Arbeit wird einfach Enter eingegeben. Wo der Cursor während des Enters steht, ist egal. Ebenso haben Buchstaben außerhalb der Matrix keinen Einfluß auf das Sprite. Andere Zeichen als "0" und "1" sollte man nicht verwenden, denn dadurch wird das Ergebnis ziemlich sicher verfälscht.

Nun wird man gefragt, ob man die Figur vergrößert oder verkleinert sehen will. Hier wird einfach "J" oder "N" eingegeben. Nachdem man die Form seiner Gestalt genug begutachtet hat, wird noch gefragt, ob Änderungen nötig sind. Sind keine nötig, druckt der Computer die Zahlen aus, die in das Programm eingegeben werden müssen, um das Sprite zu verwenden. Bei eventuellen Änderungen wird das verbesserungswürdige Sprite noch einmal angezeigt, und man kann wieder mit den oben genannten Tasten ans Werk gehen.

Mit diesem Programm ist es möglich, ein Sprite in einem anderen Programm nur mehr durch folgende Zeilen zu definieren (gilt für 8*8, 16*16 muß entsprechend geändert werden) :

```
XXXX FORI=1TO8:READA:S$=S$+CHR$(A):NEXT
XXXY DATA   8 ZAHLEN
```

Das Menü und die Anzeige dürften klar sein, bleibt nur noch das große Sprite. Hier ist zu sagen, daß es fast ganz genau wie das kleine generiert wird, nur mit den entsprechenden Umwandlungen, die für die 16*16 Matrix notwendig sind.

Interessant ist vielleicht noch, daß der Computer anhand der Variablen Q unterscheidet, ob es sich um ein 16*16- oder 8*8-Sprite handelt. Je nachdem, ob ein kleines oder ein großes Sprite bearbeitet wird, hat Q den 7 oder den Wert 31.

Wie funktioniert nun das Programm?

Bevor wir uns das Programm im Einzelnen ansehen, wollen wir das Prinzip klären! Der Computer legt sich eine Anzahl von dimensionierten Variablen zurecht, für 8*8 8 Zahlen und für 16*16 32 Variablen. Jede dieser Variablen beherbergt den ASCII-Code eines Zeichens vom Sprite.

Am Anfang wird nun der Binärcode jeder Zahl auf den Bildschirm ausgegeben, in der Form, wie auch der Computer das Sprite am Bildschirm ausgiebt. Nun kann der Bediener seine Gestalt formen. Der Computer hilft ihm dabei, indem er die eingegeben Zeichen (Cursor-Tasten, "1" und "0") auf dem Bildschirm an der richtigen Stelle anzeigt. Nach Enter holt sich der Computer die Werte des neu geformten Sprites aus dem Video-RAM. Auf dem Bildschirm steht ja das Binärmuster, dieses wird mittels "VPEEK" und einigen kleinen Umwandlungen in die dezimale Zahl rückverwandelt, das Sprite definiert und angezeigt. Wenn noch Änderungen notwendig sind, beginnt das Ganze von vorne, ist man mit dem Sprite zufrieden wird das Programm automatisch mit "RUN" neu gestartet, nachdem die Zahlen angezeigt wurden.

Sehen wir uns nun die einzelnen Teile des Programms an. Da haben wir zuerst das Menü bis Zeile 50. Danach wird das 8*8-Sprite definiert. Es folgt die Anzeige von Zeile 130 bis 190, welche auch die Abfragen auf Vergrößerung und Änderung beinhaltet. Die Spritezahlen werden von 200-210 ausgegeben. Zuletzt folgt das 16*16-Sprite.

In Zeile 70 erkennen wir zuerst die Ausgabe des kleinen Sprites auf dem Bildschirm. Es wird immer der Binärcode einer Zahl aus dem dimensionierten Feld erzeugt, auf 8-Bit umgeformt ("BIN$" liefert nicht immer einen acht Zeichen langen String!) und angezeigt. Zeile 90 erledigt die Formung des Sprites und die Zeile 100, 110 und 120 werden für das Hereinholen der Gestalt gebraucht. Wir wollen hier nicht näher auf die Syntax eingehen.

```
10 CLS:DIMA(40)
20 REM *** MENUE
30 PRINT"SPRITEGENERIERPROGRAMM
40 PRINT:PRINT" 1) 8*8":PRINT" 2) 16*16
50 PRINT" GEWUENSCHTE TASTE DRUECKEN":ONASC(
INPUT$(1))-48GOTO60,240:RUN
60 REM *** 8*8
70 Q=7:CLS:LOCATE,5:FORI=1TO8:B$=BIN$(A(I)):
PRINTTAB(14);STRING$(8-LEN(B$),"0")+B$:NEXT
80 PRINT:PRINT"FIGUR MIT EINSEN KENNZEICHNEN
":PRINT:PRINT"ABSCHLUSS MIT ENTER":LOCATE14,
5
90 S$=INPUT$(1):IFASC(S$)<>13THENPRINTS$;:GO
TO90
100 PRINTCHR$(13):FORI=215TO495STEP40:S$=""
110 FORY=0TO7:S$=S$+CHR$(VPEEK(I+Y)+32):NEXT
120 A(W)=VAL("&B"+S$):T$=T$+CHR$(A(W)):W=W+1
:NEXT
130 REM *** ANZEIGE
140 CLS:PRINT:PRINT"VERGROESSERT? (J/N)":S$=
INPUT$(1)
150 IFS$="J"ORS$="j"THENW=3ELSEIFS$="N"ORS$=
"n"THENW=2ELSEGOTO130
160 SCREEN1,W:SPRITE$(1)=T$:PUTSPRITE1,(128,
96),,1
170 LOCATE60,180:PRINT"TASTE DRUECKEN":S$=IN
PUT$(1)
180 SCREEN0:PRINT:PRINT" AENDERUNGEN NOETIG?
 (J/N)":S$=INPUT$(1)
190 IFS$="J"ORS$="j"THENONQ-66GOTO60:GOTO240E
LSEIFS$="N"ORS$="n"THEN200ELSEGOTO180
200 REM *** SPRITEZAHLEN
210 SCREEN0:PRINT"DIE ZAHLEN DES SPRITES:":P
RINT:FORI=0TOQ:PRINTA(I):FORT=1TO200:NEXT:NE
XT
220 PRINT:PRINT"TASTE DRUECKEN, UM NAECHSTES
 SPRITE ZU ZEICHNEN
230 S$=INPUT$(1):RUN
240 REM *** 16*16
250 Q=31:CLS:LOCATE,2:FORI=1TO16:B$=BIN$(A(I
-1)):C$=BIN$(A(I+15)):PRINTTAB(9);STRING$(8-
LEN(B$),"0")+B$+STRING$(8-LEN(C$),"0")+C$:NE
XT
260 PRINT:PRINT"FIGUR MIT EINSEN KENNZEICHNE
N":PRINT:PRINT"ABSCHLUSS MIT ENTER":LOCATE9,
2
270 S$=INPUT$(1):IFASC(S$)<>13THENPRINTS$;:G
OTO270
280 PRINT:FORI=90TO690STEP40:S$=""
290 FORY=0TO15:S$=S$+CHR$(VPEEK(I+Y)+32):NEX
T
300 A(W)=VAL("&B"+LEFT$(S$,8)):A(W+16)=VAL("
&B"+RIGHT$(S$,8)):T$=T$+CHR$(A(W)):E$=E$+CHR
$(A(W+16)):W=W+1:NEXT
310 T$=T$+E$:GOTO130
```

Noch zwei Programme zum Thema Sprites:

```
1 COLOR15,0,0
10 SCREEN1,2:M=1
20 FORA=1TO32
30 IFA=17THENRESTORE:M=9
40 READA$:B$=MID$(A$,M,8)
50 S$=S$+CHR$(VAL("&b"+B$))
60 NEXTA
70 SPRITE$(1)=S$
80 GOTO 250
90 DATA 0000000000000000
100 DATA0000000000000000
110 DATA0000000000000000
120 DATA0000000110000000
130 DATA0000011001100000
140 DATA0001100000011000
150 DATA0110000000000110
160 DATA0001100000011000
170 DATA0000011001100000
180 DATA0000000110000000
190 DATA0000011001100000
200 DATA0001100000011000
210 DATA0110000000000110
220 DATA0001100000011000
230 DATA0000011001100000
240 DATA0000000110000000
250 X=50:Z=200:C=+2:COLOR0
255 T=T+.2
260 X=X+C:Z=Z-C:IFX>200 THEN C=-2
270 IFX<50 THENC=+2
280 Y=96+40*SIN(T)
290 PUTSPRITE1,(X-8,Y-10),15,1
300 PUTSPRITE2,(Z-8,Y-10),15,1
305 LINE(X,Y)-(X1,Y1)
306 LINE(Z,Y)-(Z1,Y1):Z1=Z:X1=X:Y1=Y
307 COLOR 10
310 GOTO 255'by R. B.
```

```
1 X=200
2 COLOR15,0,0
3 SCREEN1,2:M=1:X=128:D=X:E=X:F=X:G=X
4 PSET(0,50),5
5 DRAW"e10r10e5r15f10r10f5r15r20e10r10e5r15f
10r10f5e5r10e5r5e10r20f5r5f10r15"
6 LINE(0,55)-(256,55),5
7 PAINT(1,51),5
8 FORB=1 TO 2
9 X=X-8
10 RESTORE
11 FORA=1TO32
12 IFA=17THENRESTORE:M=M+8
13 READA$:B$=MID$(A$,M,8)
14 S$=S$+CHR$(VAL("&b"+B$))
15 NEXTA
16 SPRITE$(B)=S$:M=M+8:S$=""
17 NEXTB
18 GOTO 35
19 DATA00000000000000000000000000000000
20 DATA00000000000000000000000000000000
21 DATA00000000000000000000000000000000
22 DATA00000000000000000000000000000000
23 DATA00000000000000000000000000000000
24 DATA00000000000000000000000000000000
25 DATA00000000000000000000000000000000
26 DATA11111111000000000000000111111110
27 DATA00000001000000010000000100000001
28 DATA00001111110000011000011111000011
29 DATA00010001111111111000100011111111
30 DATA11100001111000001110000111100000
31 DATA11111111110000001111111111000000
32 DATA01111111100000000111111110000000
33 DATA00000000000000000000000000000000
34 DATA00000000000000000000000000000000
35 PUTSPRITE1,(X,96),15,1
36 PUTSPRITE2,(D,96-10),15,2
37 PUTSPRITE3,(E,96-20),15,1
38 PUTSPRITE4,(F,96-30),15,2
39 PUTSPRITE5,(G,96-40),15,1
40 PUTSPRITE1,(X,96),15,2
41 PUTSPRITE2,(D,96-10),15,1
42 PUTSPRITE3,(E,96-20),15,2
43 PUTSPRITE4,(F,96-30),15,1
44 PUTSPRITE5,(G,96-40),15,2
45 X=X-3:D=D-2.5:E=E-2:F=F-1.5:G=G-1
46 GOTO 35'by R. B.
```

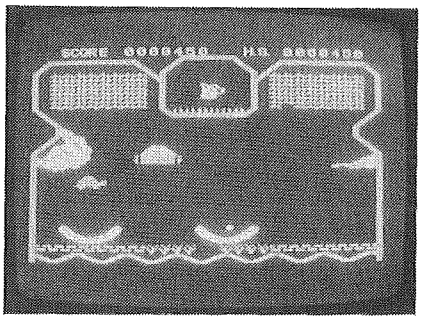
Zum bevorstehenden Jahreswechsel lieferte uns Herr Gaganas ein ungemein nützliches Programm für solche Leute, die noch keinen Kalender haben, es generiert nämlich welche!

```
1000 ''''''''''''''''''''''''''''''''''''''
1010 '     Kalender erstellen              '
1020 '                                     '
1030 'Dem S V I - C L U B gewidmet         '
1040 '                                     '
1050 '  von C .  G A G A N A S             '
1060 '                                     '
1070 ''''''''''''''''''''''''''''''''''''''
1080 CLS
1090 DIM TAGEIMM%(12),KAL%(42),MNAM$(12)
1091 H1$=CHR$(27)+"p":H2$=CHR$(27)+"q"
1100 TGE$=" SO  MO  DI  MI  DO  FR  SA"
1110 'Namen der Monate einlesen
1120 RESTORE 1210
1130 FOR I%=1 TO 12
1140   READ MNAM$(I%)
1150 NEXT I%
1160 'Anzahl der Tage im Monat einlesen
1170 RESTORE 1230
1180 FOR I%=0 TO 12
1190   READ TAGEIMM%(I%)
1200 NEXT I%
1210 DATA JANUAR,FEBRUAR,MAERZ
1215 DATA APRIL,MAI,JUNI
1220 DATA JULI,AUGUST,SEPTEMBER
1225 DATA OKTOBER,NOVEMBER,DEZEMBER
1230 DATA 0,31,28,31,30,31,30
1235 DATA 31,31,30,31,30,31
1240 'Einlesen des Jahres
1250 INPUT "Welches Kalenderjahr";YEAR%
1260 IF YEAR% < 0 THEN GOTO 1250
1270 JAHR%=YEAR%
1280 CLS
1290 'finde den 1. Januar des Jahres
1300 GOSUB 1580
1310 IF LEAPYR% THEN TAGEIMM%(2)=29
1320 WEEKDAY%=F%
1330 FOR II%=1 TO 12
1340     AUXM%=TAGEIMM%(II%-1)
1350   DAUX%=WEEKDAY%+AUXM%
1360   WEEKDAY%= DAUX% MOD 7
1370   IF WEEKDAY%=0 THEN WEEKDAY%=7
1380   FOR KK%=1 TO 42
1385     KAL%(KK%)=0
1387   NEXT KK%
1390   J%=0
1395   HLFS%=WEEKDAY%+TAGEIMM%(II%)-1
1400   FOR I%=WEEKDAY% TO HLFS%
1410     J%=J%+1
1420     KAL%(I%)=J% ' Kalender fuellen
1430   NEXT I%
1440   PRINT H1$;MNAM$(II%);JAHR%
1445   PRINT H2$;:PRINT
1450   PRINT H1$;TGE$
1460   PRINT H2$;:PRINT
1470   FOR L%=0 TO 5
1480     FOR M%=1 TO 7
1490       K%=KAL%(7*L%+M%)
1500       IF K%>0 THEN
              PRINT USING " ## ";K%;
1510       IF K% <=0 THEN PRINT "     ";
1520     NEXT M%
1530     PRINT
1540   NEXT L%
1550   E$=INKEY$
1555   IF E$="" THEN 1550 ELSE CLS
1560 NEXT II%
1570 END
1580 REM Errechne den 1. Januar
1590 IF (YEAR% MOD 4=0) AND
        (YEAR% MOD 100<>0)
        THEN LEAPYR%=1 ELSE LEAPYR%=0
1600 MNTH%=1:DAY%=1
1610 IF MNTH% > 2
       THEN MNTH%=MNTH%-2
       ELSE MNTH%=MNTH%+10:YEAR%=YEAR%-1
1620 C%=INT(YEAR%/100)
1630 A%=YEAR% MOD 100
1640 IF A%=0 THEN A%=99:C%=C%-1
1650 B%=INT((13*MNTH%-1)/5)+
        INT(A%/4)+INT(C%/4)
1660 F%=(A%+B%+DAY%-2*C%) MOD 7
1670 F%=F%+1
1680 RETURN
```

```
*****************************************
*                                       *
*            SVi-Programmecke           *
*                                       *
*****************************************
```

FLIPPER-SLIPPER

"Es müssen nicht immer Gefechte sein!" Unter diesem Motto könnte das Video-Spiel "Flipper-Slipper" erzeugt worden sein. Und in der Tat, ohne irgendwelche Feuerorgien gegen feindliche außerirdische Eindringlinge oder Asteroiden veranstalten zu müssen, kann man hier Reaktionsgeschwindigkeit und Berechnungsgabe üben.



Sinn dieses Spieles ist es nämlich, einen Ball in einem abgegrenzten Feld mit zwei Schalen immer am Bildschirm zu halten. Von den Grenzen des Feldes wird der Ball reflektiert. Den unteren Rand des Bildschirms muß man selber im Auge behalten und seine beiden Schalen so stellen, daß der Ball auf einer der beiden auftrifft und wieder weggeschleudert wird. Während dieser Balance-Akte muß man natürlich auch auf irgendeine Weise Punkte sammeln.

Dazu hat man mehrere Möglichkeiten. Im linken und rechten oberen Viertel des Feldes sind viele kleine zweigartige Objekte abgebildet. Wird der Ball in diese Erscheinungen hineingeschleudert, so wird jede vernichtet, an welche der Ball ankommt. Dafür gibt es allerdings Gutpunkte. Daß es immer schwieriger wird, ein Objekt zu treffen, je weniger es sind, ist wohl klar. Nachdem alle "Zweige" abgeschossen sind, werden sie nach einer gewissen Zeit wieder regeneriert. Ebenso kann man seinen Score durch Treffen eines Tierchens erhöhen, daß ab und zu über die Bildschirmfläche kriecht. Zur Wahl stehen hier Schildkröten, Fische und Skorpione. Sie können von hundert aufwärts bis zu achthundert Punkten "Wert" sein.

Zu guter Letzt kann man noch einen Hund befreien, der in einem Gefängnis sitzt. Dieses unfreiwillige "Hundehüttl" befindet sich zwischen den beiden Zweigansammlungen. Man muß dabei sechs Segmente eines Zauns je zweimal treffen. Nach dem ersten Treffer erscheint der Teil weiß, nach dem zweiten schwarz-weiß schraffiert. Wenn alle Segmente schraffiert sind, sinkt der Hundekopf aus seinem Gefängnis, und der Bediener bekommt 3600 Punkte Bonus und einen zusätzlichen Ball als Vorrat. Nach dieser Vorstellung wird der Hund wieder in sein Gefängnis eingesperrt.

Nach 10000 Gutpunkten tritt ein undefinierbares Objekt in der Bildschirmmitte in Kraft. Laut Bedienerhandbuch ist es ein Haus an der Küste, für den unvoreingenommenen Spieler kommt dieses Objekt eher einem Eisberg näher. Sei es wie es sei, jedenfalls bewegt sich dieses Objekt langsam auf einer Linie auf dem Schirm herum und reflektiert den Ball, wenn der Ball es berührt (meistens so unglücklich, daß man keine Chance mehr hat, seine Schale auf den neuen Aufschlagpunkt zu bewegen).

Nach 20000 Punkten erhöht sich die Geschwindigkeit des Balles.

Diese Beschreibung gilt jedoch nur für Level "1". In anderen Leveln kann man die Geschwindigkeit konstant auf "Schnell" stellen. Ebenso ist es möglich, einen weiteren Schwierigkeitsgrad einzustellen. In dieser ändert sich die Farbe des Balles zufällig zwischen Rot und Weiß. Nun muß man die Schale, in welcher vermutlich der Ball landet, der jeweiligen Ballfarbe angleichen. Dies tut man indem man die Space-Taste drückt, wenn man die Farben der Schalen wechseln will. Eine Schale ist dabei immer Weiß, eine Rot. Den gleichen Effekt kann man übrigens auch im Level "1" ab 30000 Punkten genießen.

Die letzte Feinheit, die dieses Programm zu bieten hat, ist das Kippen der Schalen. Durch einen Druck auf die Cursor-Taste "Aufwärts" kippt die Schale nach rechts. Durch einen Druck auf die Cursor-Taste "Abwärts" wandern die beiden Behälter wieder in ihre Normalposition. Dabei wurde es dem Autor nicht ersichtlich, ob dieser Effekt eine Erleichterung oder eine Erschwernis erzeugen soll. Er tippt zwar eher auf eine kleine Hilfe, doch ist dies eine vage Annahme.

Erzeugt wurde dieses Programm von Spectravideo, vertrieben wird es unter anderem von der Firma "Wehsner" im Computer-Studio (Adresse sollte jedem "ordentlichen" Clubmitglied bekannt sein).

Gratis wird es natürlich auch nicht abgegeben. Da das Spiel nur in Modulform erhältlich ist, kostet es 790 österreichische Schillinge.

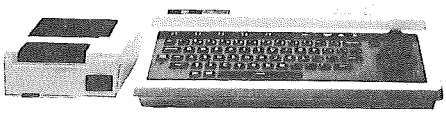
```
*****************************************
```

Und noch einmal Sprites:

```
1 COLOR15,9,9
10 SCREEN1,2:M=1
20 FORA=1TO32
30 IFA=17THENRESTORE:M=9
40 READA$:B$=MID$(A$,M,8)
50 S$=S$+CHR$(VAL("&b"+B$))
60 NEXTA
70 SPRITE$(1)=S$
80 GOTO 250
90 DATA0000000000000000
100 DATA0000000000000000
110 DATA0000000000000000
120 DATA0000000000000000
130 DATA0000000000000000
140 DATA0000000000000000
150 DATA0000000000000000
160 DATA0000000000000000
170 DATA0000000000000000
180 DATA0000000000000000
190 DATA0000000000000000
200 DATA0001111100000000
210 DATA0010001010000000
220 DATA1111111111111110
230 DATA1111111111111111
240 DATA0011000000001100
250 A=&H7D47
260 X=30:Y=50:C=0
270 C=C+1:IFC<6THEN290ELSEC=0:GOTO280
280 LOCATEX-4,Y+8:PRINTCHR$(PEEK(A));:A=A+1:
IFA>&H7D8ETHEN320
290 IFX>215THENY=Y+9:X=30:IFY>67THENA=A+1
300 PUTSPRITE1,(X,Y),1,1:X=X+1
305 PSET(X,Y+16),1
310 GOTO 270
320 GOTO 320'by R. B.
```

Die Super-Computer.

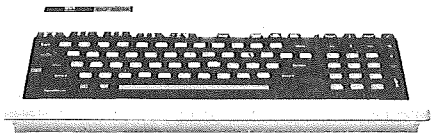
SVI



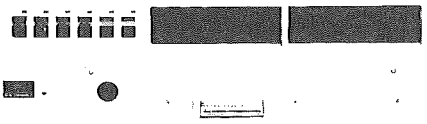
SVI-318 32 K RAM, erweiterbar bis **144 K RAM.** Erweitertes MICROSOFT-BASIC, integrierte Cursorsteuerung **öS 4.990,—**

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,—**



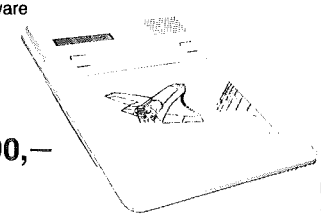
Super-Expander SVI-605, ein eingebautes Diskettenlaufwerk (160 K), Centroniks-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 12.990,—**

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,—**

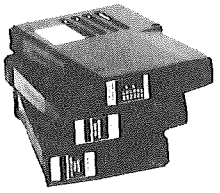
Grafik-Tablett SVI-105, 186 x 158 mm Zeichenfläche, Kassette mit Anwender-Software inkl. **öS 2.090,—**



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,—**



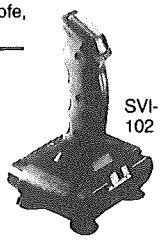
SVI-806 80-Zeichenkarte **öS 2.390,—**

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. (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 7 34 2 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 7 3	**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 7 8 08

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

```
*****************************************
*                                       *
*              RÄTSEL-ECKE              *
*                                       *
*****************************************
```

Nachdem wir in der vorigen Ausgabe über den "DIM"-Befehl "hergezogen" sind und seine Fehlermeldungen kennengelernt haben, so werden wir uns diesmal einem etwas kniffligeren Fehler widmen.

Gegeben ist ein Programm mit einem Unterprogramm, welches nur ungleiche Zahlen "Oder"- verknüpfen darf. Doch es sind einige Fehler in diesem Programm, wie man bald merken wird. Im Endeffekt soll die Entwicklung beliebig oft Zahlen verarbeiten, die Verknüpfung anzeigen und erkennen lassen, ob die Rechenoperation gemacht werden durfte.

```
5 GOSUB 10
6  PRINT "Die Verknüpfung zweier  ungleicher
Zahlen war erfolgreich
7 PRINT "Die Verknüpfung zweier  ungkleicher
Zahlen war nicht erfolgreich
10 REM UNTERPROGRAMM
20 PRINT "Dieses Programm manipuliert Zahlen
30 INPUT SE
40 INPUT T
50 IF T=SETHENPRINT "Diese Zahlen
sind gleich, daher nicht verwendbar!:RETURN
60 D=TORSE:PRINTT"OR-verknüpft mit"SE"ergibt
"D
70 RETURN
```

Viel Spaß beim Fehlersuchen!

```
*****************************************
*                                       *
*     Auflösung des Rätsels aus Heft 5  *
*                                       *
*****************************************
```

Es gibt viele Wege, nach Rom zu kommen, und es gibt ebenso viele Wege, ein Programm zu verbessern. Aus diesem Grund können wir nur eine der vielen möglichen Lösungen bieten. Unsere Auflösung soll lediglich als Denkanstoß für noch nicht fündige Leser gelten, wir sind uns im Klaren, daß es noch einige andere Wege gibt, unser Programm richtigzustellen.

Das verbesserte Programm hat folgende Form:

```
10 DIM S(12)
20 FOR T=0 TO 12
30 INPUT S(T):NEXT
40 FOR T=0 TO 12
50 PRINT S(T):NEXT
60 ERASE S:GOTO 10
```

Wie kommt man nun zu den einzelnen Lösungen? Nach dem ersten "RUN" und der Eingabe von 13 Zahlen werden 14 Nullen und eine Fehlermeldung "Redimensioned array in 10" ausgegeben.

Nun müssen wir uns überlegen, was diese Fehlermeldung bedeutet. "Redimensioned array" wird immer dann ausgegeben, wenn ein Variablenfeld zweimal hintereinander ohne "ERASE" dimensioniert wird. Direkt nach "RUN" wird das Feld S zum ersten Mal reserviert. Danach folgen Ein- und Ausgabe des Feldes und ein Rücksprung zu Zeile 10. Aber Zeile 10 hat ja das "DIM S (12)" inne. Ohne "ERASE" wird nun noch einmal S dimensioniert.

Wir wissen nun, wie dieser Fehler entstanden ist, doch wie wird er behoben? Ganz einfach, vor das "GOTO 10" schreiben wir ein "ERASE S" (man kann natürlich Zeile 60 nur durch "RUN" auch ersetzen, das bewirkt in diesem Fall das Gleiche: 60 RUN).

Sehen wir uns nun Fehler Nummer Zwei an! Der Computer druckt nur Nullen aus. Unser logischer Schluß: Der Ausgabebefehl in Zeile 50 kann nicht stimmen, da wir nicht lauter Nullen eingegeben haben. Und wirklich, wenn wir uns Zeile 50 näher betrachten, können wir "A(F)" erkennen. Das Feld A wurde jedoch weder dimensioniert noch eingegeben. Ebenso bleibt die Variable F immer auf Null, weil kein "FOR...NEXT-Befehl F erhöht. Unsere Schleife wirkt auf die Variable T. So ändern wir Zeile 50 auf "PRINT S(T)".

Nach einen neuerlichen "RUN" werden wir aber wieder enttäuscht. Die Zahlen werden zwar alle ausgegeben, aber am Ende steht "Suscript out of range in 50". Offensichtlich beherbergt der "PRINT"-Befehl noch einige andere Fehler. Wir blättern neuerdings in unserem Handbuch nach und erkennen, daß ein Index verwendet wurde, welcher größer als erlaubt ist. Wir überprüfen, ob dies stimmt, und drücken im Direktmodus "PRINT T". T hat den Wert 13, wir haben aber nur bis 12 dimensioniert. Nun erkennen wir auch in Zeile 40, daß im "FOR...NEXT"-Befehl nicht bis 12 sondern bis 13 "hochgeschraubt" wurde. Der Fehler lag also nicht in 50, sondern in Zeile 40. Wir ändern 13 auf 12 um.

Wenn wir nun "RUN" drücken, funktioniert alles tadellos, unser Programm ist entwanzt. Eine der endgültigen Versionen ist oben abgebildet.

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

Turbo Pascal 2.0

angepaßt auf SVI-328 prompt lieferbar, mit Texteditor und ausführlichem Handbuch in d e u t s c h e r Sprache nur S 1.890,-

TOOL BOX mit deutschem Handbuch S 1.890,-

SVI-728

S 7.990,- prompt lieferbar

SVI-328 S 6.990,-

Wir liefern weiterhin Markendrucker zu besonders günstigen Preisen:

EPSON BROTHER SILVER-REED STAR

Wir wünschen allen unseren Kunden
und Geschäftsfreunden
ein frohes Weihnachtsfest und
ein erfolgreiches Jahr 1985!
Wehsner GmbH.

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