SV JOURNAL

Die Zeitschrift des Spectra Video Club Austria

INHALT

Heft 3/84

S 15.-

Liebes Clubmitglied!

Erstmalig probieren wir in unserer Zeitschrift einen verkleinerten Druck. Die Vorteile liegen auf der Hand! Wir können nun zwei Spalten zu je 44 Buchstaben verwenden. Eine Spalte mit 44 Zeichen ist leichter lesbar, als eine mit 65.

Da wir eine Fachzeitschrift sind, möchten wir natürlich auch möglichst viel Information in unser SVi-Journal zu packen. Mit dem neuen System wird es uns möglich gemacht, theoretische 80% mehr Inhalt in die Zeitschrift zu schreiben. "Theoretische" schreibe ich deshalb, weil in Wirklichkeit das Titelblatt noch immer das gleiche Format hat, genauso, wie einige andere Teile der Zeitschrift. Praktisch ergeben sich aber immer noch ungefähr 50% mehr Information.

Eine neue Rubrik haben wir auch eröffnet. Hier können Clubmitglieder mitteilen, wenn sie gute Programme kreiert haben. Zum Unterschied zu den Programmen von früher, die gelistet wurden und eine Programmbeschreibung hatten, wird in der neuen Rubrik lediglich eine Bedienungsanleitung geliefert. Das Programm selber ist dann beim Entwickler erhältlich. Aus diesem Grund bitten wir wieder alle Clubmitglieder, uns Programme beliebiger Länge zu geben. Ist die Entwicklung kurz, kann man sie als fertig dokumentiertes Lernobjekt verwenden. Ist das Programm lang, so wird es für die neue Rubrik verwendet.

Leichte Schwierigkeiten gibt es leider noch bei den Dokumentationen unserer gelisteten Programme. Diese sollen ja weniger fortgeschrittenen Clubmitgliedern als Hilfe dienen, um schneller und besser die Programmiersprachen BASIC und Assembler zu lernen. Deshalb sollte nicht nur beschrieben werden, was das Spiel alles kann, sondern auch, wie es aufgebaut ist, welche Variablen warum verwendet werden und so weiter.

```
*****************************************
*                                       *
*  Die nächsten Clubabende:             *
*                                       *
*   Mi, dem 26. September 1984, ab 19 Uhr *
*   Sa, dem 13. Oktober   1984, ab 17 Uhr *
*   Mi, dem 24. Oktober   1984, ab 19 Uhr *
*   Sa, dem 10. November  1984, ab 17 Uhr *
*                                       *
*  wie  immer  im Computer-Studio,      *
*  1040  Wien,  Paniglgasse  18-20.     *
*  Nichtmitglieder sind willkommen.     *
*  Ende jeweils  ca. 22 Uhr!            *
*                                       *
*  Aktivitäten  an  den  Clubabenden:   *
*  Arbeiten  an  Spectravideo-Systemen, *
*  Informationsaustausch zwischen Club- *
*  mitgliedern.                         *
*                                       *
*  Ab Oktober zusätzlich  Vorträge über *
*  Spectravideo-Computer,  deren Anwen- *
*  dung und deren Peripherie!           *
*                                       *
*  Mittwoch, dem 10. Oktober 1984, 19 Uhr *
*    Themenkreis: Floppy-Disks          *
*    - Schutz der Floppy-Disks          *
*    - Formatieren der Disketten        *
*    - Disk-BASIC                       *
*    - Diskettenbetrieb unter CP/M      *
*    - und anderes                      *
*                                       *
*  Mittwoch, dem 7. November 1984, 19 Uhr *
*    Themenkreis: Anwendung der         *
*                 SVi-Computer          *
*     - Warum ein SVi-Computer?         *
*     - Warum 80 Zeichen-Karte?         *
*     - Wann welche Peripherie?         *
*     - Anwendungsmöglichkeiten         *
*     - und anderes                     *
*                                       *
*****************************************
```

Neu in unserem Club sind Vortragsabende. Nähere Informationen über die beiden bisher festgelegten Abende sind unter den Clubnachrichten zu finden. Während am ersten Abend über den Sinn von Floppy-Disks, deren Handhabung und ihren Anwendungsbereich gesprochen wird, versuchen wir am zweiten Abend den Sinn von Computern allgemein zu eruieren. Dazu geben wir auch Tips, wann man sich welche Peripherie kaufen sollte. Denn es hat keinen Sinn, sich eine SVi-Anlage mit Drucker, Floppy und 80-Zeichenkarte anzuschaffen, wenn man nur Video spielen will (da genügt die Zentraleinheit SV-318 oder SV-328). Andererseits sei dann natürlich vor falscher Sparsamkeit gewarnt, wenn es um den gewerblichen Einsatz geht. Aber näheres erfahren sie an den Vortragsabenden. Die sind übrigens für Mitglieder und Nichtmitglieder gleichermaßen gratis.

Weniger kostenlos ist der BASIC-Kurs, welchen die Firma Wehsner veranstaltet. Dieser Kurs ist speziell auf die Anwender der Spectra-Video-Computer zugeschnitten. Dadurch lernt der Anfänger mit dem Gerät umzugehen und kann sein Wissen gleich in die Praxis umsetzen, da auch Hardware zur Verfügung steht.

Anwender anderer Computer können übrigens sicher sein, daß auch für sie der Lehrgang interessant wird, denn der SVi-328 hat so gut wie alle vernünftigen BASIC-Befehle. Der Kurs wird 11 Tage mit je 2 Stunden belegen. Der Anfang ist am 16.Oktober festgelegt. Ab dann wird jeden Montag und jeden Donnerstag ein Abend abgehalten. Eine Ausnahme bildet lediglich der 1.November, weil er ein Feiertag ist. Damit auch Berufstätige den Kurs besuchen können, fängt er an jedem Abend erst um 19.00 Uhr an.

Kommen wir zu den Kosten! Der Kurs kostet einen Teilnehmer pro Stunde 40 Schilling. Da man aber den ganzen Kurs besuchen und den gesamten Beitrag bezahlen muß, kann man mit S 880.- rechnen. Der Preis ist inklusive Mehrwertsteuer.

Eine weitere Neuigkeit für Clubmitglieder bietet der "itm-praktiker". Dieses österreichische Technikmagazin bietet unserem Club die Möglichkeit, an der Hobby-Elektronik-Messe teilzunehmen. Der "praktiker" stellt uns einen Teil des sogenanntes Action-Centers zur Verfügung, wo die Fähigkeiten der einzelnen Computer vorgeführt werden können.

Wir lassen uns dort mit 3 SV-Mikros nieder. Wir brauchen nun findige Clubmitglieder, welche unseren Spectravideos gute Demonstrationsprogramme schreiben. Außerdem benötigen wir Ideen, was man am Spectravideo herzeigen kann, wie man den Club vertreten sollte und nicht zuletzt brauchen wir auch noch einige Aufpasser für die SV's. Die Hobby-Elektronik findet vom 15.November bis zum 18.November statt. Anregungen können jederzeit beim Computer-Studio abgegeben werden.

Da es immer wieder Leute gibt, die aus rationellen Überlegungen nicht dem Club beitreten möchten, wohl aber das SVi-Journal lesen wollen, haben wir uns entschlossen, es diesen Leuten mit einem Abonnement zu ermöglichen, unsere Zeitung regelmäßig zu beziehen. Das SVi-Journal kommt ja jedes Monat heraus und kostet 15 Alpendollar. Beim Abo bekommt man zwei Hefte gratis und zahlt so für 12 Hefte pro Jahr nur 150 Schilling. Die Zeitschriften werden ohne Mehrkosten zugesandt. Ebenfalls verbilligt ist das Halbjahresabonnement. Für sechs Hefte zahlt man 80 Schilling. Die Preise sind selbstverständlich alle mit Mehrwertsteuer.

Ihr SVi-Journal Chefredakteur Gerhard Fally!

In dieser Folge wird die Beschreibung der Funktionen fortgesetzt. Die Reihenfolge des User-Manuals wird eingehalten. Allerdings werden Funktionen, denen nichts hinzuzufügen ist, weggelassen!

Leider hat die ASCII-Code-Tabelle am Ende des SV-Manuals einige Schwächen. Deshalb werden in untenstehender Tabelle alle 256 Zeichen und deren Funktionen wiederholt. Die CTRL-Sequenzen sind in der Tabelle auf der folgenden Seite angegeben.

Das Beispiel bei "RIGHT$" stimmt nicht. Selbstverständlich muß bei "PRINT RIGHT$("#" +"1234567",3+2) ⟨CR⟩ das Ergebnis "34567" lauten.

Der große Vorteil von "SPACE$" kommt im Buch nicht voll zum Ausdruck. Gegenüber der Anweisung "SPC" kann man nämlich "SPACE$" auch nach eine alphanumerische Variable schreiben. Folgendes Beispiel zeigt die Möglichkeiten von "SPACE$" besser!

```
5 INPUT B
10 A$=SPACE$(B+5)
20 PRINT "HALLO"A$"EMIL"SPACE$(3)"HALLO"
```

RUN ⟨CR⟩ ergibt für B=2:

```
HALLO       EMIL   HALLO
```

Mit "STR$" kann man einen numerischen Wert in Stringdarstellung ummodellieren. Wird zum Beispiel eine Zahl von der Variable A ausgegeben, so schreibt der Computer je ein Blank vor und hinter die Zahl. In der Stringdarstellung (zum Beispiel S$=STR$(A)) wird nur vor der Zahl ein Space ausgegeben (dieses wird als Platzreservierung für das Minus einer negativen Zahl gebraucht).

```
A$=STR$(584.39+27):?  "Das ist die"A$"ste Folge!" ⟨CR⟩
```

Das ist die 611.39ste Folge!

0	0	No Function	64	40	@	128	80		192	C0	[RIGHT]+G
1	1	Control+ A	65	41	A	129	81		193	C1	[RIGHT]+H
2	2	Control+ B	66	42	B	130	82		194	C2	[RIGHT]+I
3	3	Control+ C	67	43	C	131	83		195	C3	[RIGHT]+J
4	4	Control+ D	68	44	D	132	84		196	C4	[RIGHT]+K
5	5	Control+ E	69	45	E	133	85		197	C5	[RIGHT]+L
6	6	Control+ F	70	46	F	134	86		198	C6	[RIGHT]+M
7	7	Control+ G	71	47	G	135	87		199	C7	[RIGHT]+N
8	8	Control+ H	72	48	H	136	88		200	C8	[RIGHT]+O
9	9	Control+ I	73	49	I	137	89		201	C9	[RIGHT]+P
10	A	Control+ J	74	4A	J	138	8A		202	CA	[RIGHT]+Q
11	B	Control+ K	75	4B	K	139	8B		203	CB	[RIGHT]+R
12	C	Control+ L	76	4C	L	140	8C		204	CC	[RIGHT]+S
13	D	Control+ M	77	4D	M	141	8D		205	CD	[RIGHT]+T
14	E	Control+ N	78	4E	N	142	8E		206	CE	[RIGHT]+U
15	F	Control+ O	79	4F	O	143	8F		207	CF	[RIGHT]+V
16	10	Control+ P	80	50	P	144	90		208	D0	[RIGHT]+W
17	11	Control+ Q	81	51	Q	145	91		209	D1	[RIGHT]+X
18	12	Control+ R	82	52	R	146	92		210	D2	[RIGHT]+Y
19	13	Control+ S	83	53	S	147	93		211	D3	[RIGHT]+Z
20	14	Control+ T	84	54	T	148	94		212	D4	
21	15	Control+ U	85	55	U	149	95		213	D5	
22	16	Control+ V	86	56	V	150	96		214	D6	
23	17	Control+ W	87	57	W	151	97		215	D7	
24	18	Control+ X	88	58	X	152	98		216	D8	
25	19	Control+ Y	89	59	Y	153	99		217	D9	
26	1A	Control+ Z	90	5A	Z	154	9A		218	DA	
27	1B	ESC-Taste	91	5B	[	155	9B		219	DB	
28	1C	Cursor right	92	5C	\	156	9C		220	DC	
29	1D	Cursor left	93	5D	]	157	9D		221	DD	
30	1E	Cursor up	94	5E	^	158	9E		222	DE	
31	1F	Cursor down	95	5F	_	159	9F		223	DF	
32	20	Space-Taste	96	60	`	160	A0	[LEFT]+A	224	E0	
33	21	!	97	61	a	161	A1	[LEFT]+B	225	E1	
34	22	"	98	62	b	162	A2	[LEFT]+C	226	E2	
35	23	#	99	63	c	163	A3	[LEFT]+D	227	E3	
36	24	$	100	64	d	164	A4	[LEFT]+E	228	E4	
37	25	%	101	65	e	165	A5	[LEFT]+F	229	E5	
38	26	&	102	66	f	166	A6	[LEFT]+G	230	E6	
39	27	'	103	67	g	167	A7	[LEFT]+H	231	E7	
40	28	(	104	68	h	168	A8	[LEFT]+I	232	E8	
41	29	)	105	69	i	169	A9	[LEFT]+J	233	E9	
42	2A	*	106	6A	j	170	AA	[LEFT]+K	234	EA	
43	2B	+	107	6B	k	171	AB	[LEFT]+L	235	EB	
44	2C	,	108	6C	l	172	AC	[LEFT]+M	236	EC	
45	2D	-	109	6D	m	173	AD	[LEFT]+N	237	ED	
46	2E	.	110	6E	n	174	AE	[LEFT]+O	238	EE	
47	2F	/	111	6F	o	175	AF	[LEFT]+P	239	EF	
48	30	0	112	70	p	176	B0	[LEFT]+Q	240	F0	
49	31	1	113	71	q	177	B1	[LEFT]+R	241	F1	
50	32	2	114	72	r	178	B2	[LEFT]+S	242	F2	
51	33	3	115	73	s	179	B3	[LEFT]+T	243	F3	
52	34	4	116	74	t	180	B4	[LEFT]+U	244	F4	
53	35	5	117	75	u	181	B5	[LEFT]+V	245	F5	
54	36	6	118	76	v	182	B6	[LEFT]+W	246	F6	
55	37	7	119	77	w	183	B7	[LEFT]+X	247	F7	
56	38	8	120	78	x	184	B8	[LEFT]+Y	248	F8	
57	39	9	121	79	y	185	B9	[LEFT]+Z	249	F9	
58	3A	:	122	7A	z	186	BA	[RIGHT]+A	250	FA	
59	3B	;	123	7B	{	187	BB	[RIGHT]+B	251	FB	
60	3C	<	124	7C	|	188	BC	[RIGHT]+C	252	FC	
61	3D	=	125	7D	}	189	BD	[RIGHT]+D	253	FD	
62	3E	>	126	7E	~	190	BE	[RIGHT]+E	254	FE	
63	3F	?	127	7F	DEL-Taste	191	BF	[RIGHT]+F	255	FF	

Nur wenn Zahlen mit "MKI$", "MKS$" oder "MKD$" in Srings umgewandelt wurden, kann man sie mit "CVI", "CVS" oder "CVD" wieder in Zahlen umwandeln. Im Übrigen steht "I" für Integer (ganzzahlig), "S" für Single (einfachgenau) und "D" für Double (doppeltgenau). Die Abkürzungen beziehen sich auf die verschiedenen Arten der Variablen. Ebenso sind auch die Abkürzungen "INT" (Integer), "SNG" (Single) und "DBL" (Double) zu verstehen.

Die Funktionen "DSKF", "EOF", "FPOS", "LOC", "LOF", "ATTR$" und "DSKI$" werden im Kapitel Datei & Dateiverarbeitung besprochen!

Vor einem "STRIG" muß unbedingt ein "STRIG (<Steuerknüppelnummer>) ON" ausgeführt worden sein. "STRIG (<Steuerknüppelnummer>)" liefert nicht 1, sondern -1, wenn die Leer- (beim Computer), oder die Feuertaste (bei Joysticks) gedrückt wurde.

Man darf "A$=SPRITE$(12)" auf keinen Fall mit "SPRITE$(12)=A$" verwechseln! Während bei letzterem das Sprite mit der Nummer 12 den String A$ zugewiesen bekommt, weist die erste Funktion der Variable A$ den String des Sprites 12 zu. Es kann ohne weiteres vorkommen, daß man in einem Programm folgendes sieht: SPRITE$(11)=SPRITE$(12) (das Sprite Nummer 11 bekommt das Aussehen des Sprites Nummer 12).

'CTRL'-B	CHR$(2)	bewegt den Cursor zum Anfang des vorigen Wortes
'CTRL'-E	CHR$(5)	löscht den Text von der Cursorposition bis zum Zeilenende
'CTRL'-F	CHR$(6)	bewegt den Cursor zum Anfang des nächsten Wortes
'CTRL'-H	CHR$(8)	(auch Taste über ENTER) löscht das Zeichen links vom Cursor und zieht den Rest der Zeile ein Zeichen nach links
'CTRL'-I	CHR$(9)	(auch Taste mit dem Doppelpfeil nach rechts über CTRL) Tabulation. Erzeugt acht Blanks
'CTRL'-K	CHR$(11)	(auch SHIFT-CLS) bewegt den Cursor zur Home-Position.
'CTRL'-L	CHR$(12)	(auch CLS) löscht den Bildschirm und bewegt den Cursor zur Home-Position
'CTRL'-M	CHR$(13)	(auch ENTER) übergibt die Eingabezeile an das BASIC-System
'CTRL'-N	CHR$(14)	bewegt den Cursor an das Ende der Zeile (nicht unbedingt 40. Bildschirmspalte)
'CTRL'-R	CHR$(18)	(auch INS) schaltet zwischen Einfüge- und Normalmodus hin und her
'CTRL'-U	CHR$(21)	löscht die logische Zeile (eventuell mehrere Bildschirmzeilen) und positioniert den Cursor an den Anfang der Zeile
'CTRL'-\	CHR$(28)	Cursor nach rechts
'CTRL'-]	CHR$(29)	Cursor nach links
'CTRL'-'SHIFT'-6	CHR$(30)	Cursor nach oben
'CTRL'-_	CHR$(31)	Cursor nach unten
'DEL'	CHR$(127)	löscht das Zeichen auf dem der Cursor steht und schiebt den Rest der Zeile ein Zeichen nach links

Für die Funktion "INP" gelten die gleichen IO-Port Adressen, die im Heft 2/84 auf Seite 8 angegeben sind.

Normalerweise ruft man ein Asembler-Unterprogramm auf, indem man für den Ausdruck eine Dummyvariable einsetzt (zum Beispiel "A=USR(0))".0 ist beim Beispiel in der Klammer die Dummyvariable. Dummyvariablen sind Werte, die vom Interpreter formal gebraucht werden, die aber den Programmablauf in keiner Weise beeinflußen.

Wenn jemand aber einen Wert wirklich mit "USR" für das Assembler-Unterprogramm übergeben möchte, der hat im Anhang F des deutschen BASIC-Manuals wertvolle Hinweise, wie und wo dieser Wert im Speicher abgelegt wird. Es kann aber immer nur eine Variable für ein Unterprogramm übergeben werden. Sollten mehr Werte gebraucht werden, muß man die Variablen selber mittels "POKE" in den Programmspeicher schreiben. Im Maschincodeprogramm kann man dann mit einem "LD XX, (nn)" auf die Variablen zugreifen. "XX" bedeuted hier ein Register oder ein Registerpaar des Z80-Prozessors, "nn" bezeichnet die Adresse, bei der Variable abgelegt ist.

Das Video-RAM (=Bildschirmspeicher) hat einen Speicher von 16 KBytes. Nur die ersten 4 K werden vom Textbildschirm gebraucht. Die restlichen 12 K sind für Datenspeicherung frei! Wohlgemerkt, nur für Daten! Programme darf man keine ins Videoram schreiben. Man kann dann mit "VPOKE" und "VPEEK" im Video-RAM genauso hantieren, wie mit "POKE" und "PEEK" im "normalen" Speicher.

Der einzige Unterschied besteht darin, daß man mit "POKE" und "PEEK" den Bereich von &H8000 bis &HFFFF ansprechen kann. Der Bildschirmspeicher hat den Bereich von &H0000 bis &H4000 inne. Achtung! Mit "PEEK" alleine kann man nur das ROM des BASIC-Interpreters lesen. der den Bereich von &H0000 bis &H7FFF besitzt. Die &H0000-&H4000 sind aber unabhängig von den 32 KByte des ROM-Interpreters.

Mit "A=PEEK(&H100)" liest der Computer das Byte mit der Adresse &H100 aus dem BASIC-ROM, mit "A=VPEEK(&H100)" das Byte mit der Adresse &H100 aus dem Videoram.

Bekanntlich braucht unser Computer einen Platz im Speicher, wo er seine Variablen hinlegt. Dies macht er bei numerischen Variablen hinter dem BASIC-Programm. Diese werden dort fein alphabetisch sortiert abgelegt. Natürlich gibt es auch hier ein festes Schema, wie so eine Ablage auschaut. Da

steht einmal als erstes Byte entweder eine Zwei, eine Drei, eine Vier oder eine Acht. Die Zwei steht für ganzzahlige Werte, die Vier für einfachgenaue, die Acht für doppeltgenaue Werte und die Drei für Stringvariablen.

Lassen wir jedoch die Strings weg und kümmern uns momentan nur um die numerischen Variablen. Bei ihnen kommen nach der Kennnummer (2,4,8) zwei Bytes mit dem ASCII-Code des Variablennamens. Hier bemerken wir übrigens, warum der Computer nur zwei Zeichen als Namen registriert, er kann nämlich gar nicht mehr Buchstaben abspeichern.

Der eigentliche Wert wird je nach Variablentyp anders abgelegt. Bei den Integerwerten begnügt sich der Computer mit 2 Bytes. Der Zahlenbereich von 32768 bis -32768 kann damit abgedeckt werden. Vielleicht ist Ihnen aufgefallen, daß das hexadezimale &HA000 dezimal als -xxxx vom Computer interpretiert wird. Das Hexadezimale hat nämlich kein Vorzeichen. Daher verwendet unser Mikro die dezimalen Minuszahlen für den Bereich &H8000 bis &HFFFF.

Die anderen numerischen Variablen haben folgende mathematische Grundlage: Eine Zahl, zum Beispiel 3456.456 kann als Ziffernfolge (3456456) interpretiert werden, vorher wird jedoch angegeben, wo das Komma steht. Demnach sieht 3456.456 so aus: Komma an 4.Stelle;3456456. Und nach genau diesem Schema legt der Computer die Zahl auch ab. Er nimmt sich ein Byte und markiert Komma und das Vorzeichen und reserviert dann 3 (für einfachgenaue Variable) oder 7 Bytes (für doppeltgenaue) für die Zahlenfolge. Die Markierung für das Komma sieht so aus:

0-63 Zahlenbereich von 10^-64 bis 10^-1
64-127 Zahlenbereich von 10^0 bis 10^63
128-191 Zahlenbereich von -10^-64 bis -10^-1
192-255 Zahlenbereich von -10^0 bis -10^63

Die Stringvariablen bieten zwei Möglichkeiten. Beiden ist gemein, daß sie eine Drei als Markierung besitzen und zwei Bytes als Namen reservieren. Danach kommt bei beiden ein Softwarezeiger, der an eine andere Speicherstelle zeigt, wo der String abgelegt ist. Wurde ein String zwischen zwei Anführungszeichen einer Variablen zugewiesen, zum Beispiel "A$="HALLO", dann weist der Zeiger auf diese Stelle im Programm. Hat man allerdings einen String rechnerisch erzeugt oder mit "INPUT" eingegeben, dann wird er von &HF2D0 (bei Disk-BASIC-Betrieb und nach "CLEAR"-Anweisungen kann sich dieser Wert allerdings ändern) abwärts abgelegt.

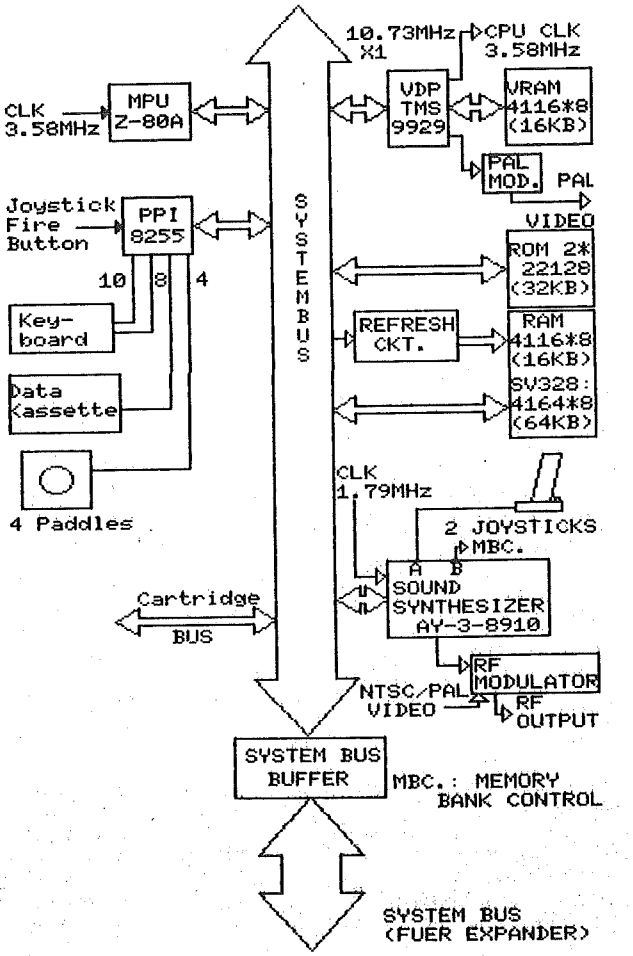
Die Anweisung "VARPTR" erzeugt nun die Zeilennummer, ab welcher der Wert einer Variablen abgelgt ist. Bei numerischen wird die Nummer des Bytes angegeben, in dem das Komma steht, bzw das erste Byte der ganzzahligen Werte. Die Stringvariablen erzeugen bei "VARPTR" den Zeiger, der auf die jeweilige Zeichenkette zeigt. "VARPTR" kommt übrigens von dem Wort VARiablen-PoinTeR.

S V - H a r d w a r e

Die Hardware der Spectra Video Computer in Fortsetzungen

3. F o l g e

Wie die Z80-CPU über den Systembus mit den Peripheriebausteinen in Verbindung steht, zeigt die untenstehende Graphik in übersichtlicher Darstellung. Links oben erkennt man den Mikroprozessor. Die Taktfrequenz wird, wie bereits erwähnt, vom Video-Prozessor generiert (rechts oben in der Abbildung zu sehen. Setzen wir hier fort: Das Video-RAM (ganz rechts oben dargestellt) besteht aus acht RAM-Bausteinen 4116. Weiters sieht man, daß das Videosignal über den PAL-Modulator zum Videoausgang gebracht wird.



An der rechten Seite ist auch die Verbindung zu 2 x 16 KB ROM (BASIC-Interpreter) zu sehen, weiters der Anschluß der dynamischen RAMs an den Systembus und die Steuerung zum Auffrischen der RAMs.

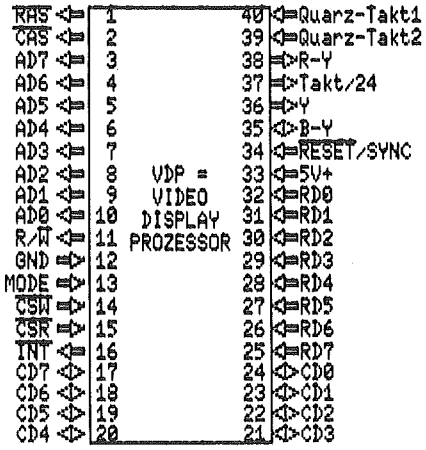
Der "Programmierbare Tongenerator" AY-3-8910 erfüllt neben der Musik- und Geräuschproduktion noch zwei weitere Aufgaben, nämlich die Abfrage der Joysticks und das "Bank-Switching". Beim "Bank-Switching" werden eigentlich nicht Banken sondern halbe Banken ("Pages" = Seiten) umgeschaltet, also jeweils die oberen oder unteren 32 KByte.

Im linken Teil der Zeichnung findet man weiters den programmierbaren Ein-/Ausgabebaustein 8255, der die Tastaturabfrage erledigt, weiters den Feuerknopf an den Joysticks sowie das Graphik-Tablett und die Paddles abfrägt und den Kassettenrecorder steuert.

Der Steckschacht für Module liegt ebenfalls am Systembus. Gepuffert wird der Systembus zum Expanderstecker geführt.

Der Video-Prozessor TMS-9918A (TMS-9929)

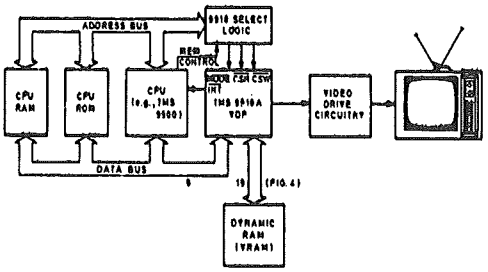
Der TMS-9929 ist die PAL-Version des TMS-9918 von Texas Instruments und wird daher in der europäischen Version der SVI-Computer eingesetzt (Bildschirm im 625 Zeilen-Format). Er ist pinkompatibel mit dem 9918 mit Ausnahme der Pins 35, 36 und 38 (siehe Graphik mit Anschlußbelegung).



Drei Kontrollsignale kommen von der CPU: CSW, CSR und MODE. Sie werden zur Einstellung der jeweiligen Betriebsart des Videoprozessors verwendet. CSW bedeutet CPU schreibt zum Videoprozessor, CSR bedeutet CPU liest vom Videoprozessor und MODE ist das Signal, welches die Quelle oder das Ziel der Operation vom bzw. zum Video-RAM bestimmt (Pin 13 bis 15 im Anschlußbild).

RD0 bis RD7 stellt den Video-RAM-Lesebus dar (nur Input), AD0 bis AD7 ist der Video-RAM-Adreß-/Datenbus (gemultiplext Highbyte, Lowbyte und Datenbyte - nur Output). CD0 bis CD7 ist der Datenbus der CPU und wird für I/O-Operationen verwendet. Zu beachten ist, daß D0 bei den drei Bussen jeweils das "most significant Bit" darstellt.

Der Video-Prozessor (VDP) verfügt über vier Darstellungsarten: Graphikmode I, Graphikmode II, Multicolormode und Textmode. Graphikmode II entspricht der hochauflösenden Graphik bei den SVI-Computern (SCREEN 1), Multicolor entspricht SCREEN 2, Textmode ist auch der SVI-Textmodus SCREEN 0. Der Graphikmode I (von der Farbauflösung her schlechter als Graphikmode II) kann vom SVI-BASIC aus nicht angesprochen werden.



Typische Systemkonfiguration

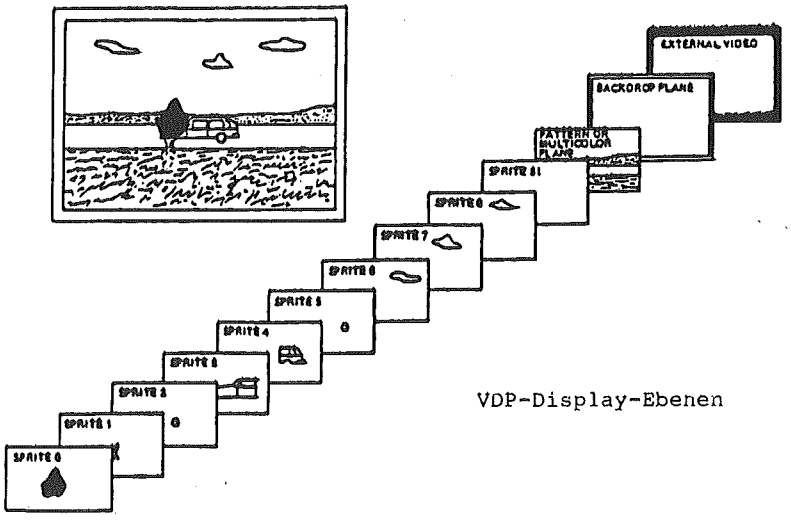
Der VDP bringt ein Bild auf den Schirm, das man sich aus mehreren Lagen (Ebenen) zusammengesetzt denken kann (wie ein Sandwich). Darstellungen in Ebenen näher zum Betrachter haben die höhere Priorität und verdecken dahinterliegende Darstellungen. Die Sprites 0 bis 31 befinden sich in den vorderen 32 Ebenen (siehe Darstellung), dahinter liegt die sogenannte "Pattern"-Ebene (also die eigentliche Zeichenebene, dahinter der Hintergrund (nur eine Farbe möglich), als nächstes die Ebene, in die man ein externes Video-Bild einspeisen kann (bei den SVI-Computern derzeit hardwaremäßig nicht realisiert) und dann als Abschluß eine "schwarze" Fläche.

Noch ein Wort zu den Sprites. Sprites sind bewegliche Objekte am Bildschirm, deren Position durch horizontale und vertikale Koordinaten im Video-RAM bestimmt wird. Jede der ersten 32 Ebenen beinhaltet einen Sprite. Außerhalb des Sprites ist die jeweilige Ebene transparent. Da die Koordinaten der Sprites in Pixel (kleinste Einheit in der Bildschirmdarstellung) bestimmt werden, können Sprites sehr genau positioniert und bewegt werden.

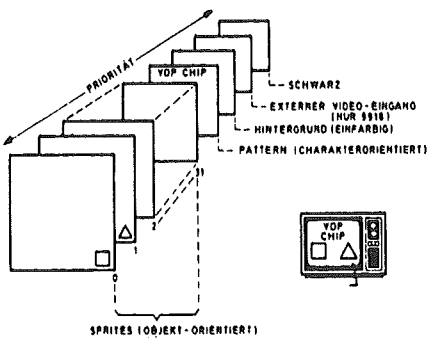
Sprites können in den Größen 8 x 8, 16 x 16 und 32 x 32 Pixels dargestellt werden (definieren kann man sie in Feldern 8 x 8 oder 16 x 16). Im Textmodus sind die Sprite-Ebenen transparent.

Es gibt übrigens eine Restriktion hinsichtlich der Darstellung von Sprites. Nur vier Sprites können auf einer horizontalen Linie dargestellt werden. Weitere Sprites auf dieser Linie werden transparent.

Fortsetzung auf Seite 10



VDP-Display-Ebenen



Z 80 - PROGRAMMIEREN IN ASSEMBLER

Assemblerfortsetzungskurs Nr.:003

Diesmal will ich schon mit den einfachsten Befehlen beginnen. Doch zuerst muß ich erklären, wie der Z-80 mit den Befehlen umgeht.

Jeder seiner knapp 1500 Befehle hat einen eigenen Operationskode, kurz Opkode genannt, welcher der CPU (Central Processing Unit= Mikroprozessor, Zentrale Recheneinheit) angibt, was sie zu tun hat. Ein Opkode kann maximal zwei Bytes lang sein. Diesem Opkode können maximal zwei Bytes an Daten oder Adressen folgen. Daher kann ein Befehl bis zu vier Bytes im Speicher belegen. Der Mikroprozessor benötigt für jedes Byte eines Befehles vier Taktzyklen zum Einlesen aus dem Speicher.

Was ist ein Taktzyklus? Sie werden bestimmt schon von der Taktfrequenz gehört haben, mit der ein Mikroprozessor arbeitet. Die Taktfrequenz gibt an, wieviele Operationen der Mikroprozessor in einer Sekunde durchführen kann. Dies wird meistens in MHz (=Megahertz 1MHz = 1 Million Schwingungen in der Sekunde) gemessen. Der ursprüngliche Z-80 arbeitete mit einer Taktfrequenz bis zu 2 MHz. Da das aber nicht überwältigend schnell war, entwickelte man einige Z-80 Mikroprozessoren, die sich nur durch ihre Taktfrequenz unterscheiden. Heute gibt es bereits einen Z-80A (bis 4 MHz), Z-80B (bis 6 MHz) und einen Z-80H (bis 8 MHz). Der Spectravideo besitzt zum Beispiel einen Z-80A, der mit 3.56 MHz arbeitet.

Doch nun zurück zum Opkode. Programmiert man in Maschinensprache, so gibt man nur die Zahlenfolgen der einzelnen Befehle ein. Da dies sehr aufwendig ist, gibt es die Programmierung in Assembler. Statt der Opkodes wird nur eine symbolische Abkürzung des Befehls eingegeben. Befehle, die man symbolisch darstellt, werden "MNEMONICS" (Merkwörter) genannt. Ein ganzes Programm, das in Mnemonics eingegeben worden ist, nennt man Source-File. Diese Mnemonics können mit Hilfe eines Assemblers in die dementsprechenden Opkodes übersetzt werden.

Doch nun zu den einfachsten Befehlen, die der Z-80 besitzt. Das sind die Ladebefehle. Mit ihnen kann man Informationen, Daten oder den Wert eines Registers in ein angegebenes Register laden. Die mnemotechnische Darstellung für diesen Befehl lautet "LD", was von LoaD (engl.) kommt. Ladebefehle zwischen den Universalregistern (auch dem Akkumulator) belegen immer nur ein Byte an Speicher.

Ein Ladebefehl zwischen zwei Registern sieht zum Beispiel folgendermaßen aus: LD C,E (Lade Register "C" mit dem Wert des Registers "E"). Dieser Befehl bewirkt, daß der Wert, der im Register "E" steht, in das Register "C" übertragen wird. Dabei wird der Inhalt des Registers "E" nicht geändert. Bitte merken Sie sich auch, daß immer von rechts nach links geladen wird. Das heißt, daß immer der Wert, der im rechten Register (Quellregister) steht, in das linke Register (Zielregister) geladen wird. Natürlich ist das Laden zwischen den anderen Universalregistern (inkl. Akkumulator) ebenfalls möglich. Ausserdem kann man beliebige 8-Bit-Daten in eines der Register speichern. Zum Beispiel : LD H,3E. Wenn in dieser Serie eine Zahl ohne eine Kennzeichnung (H oder D oder B) geschrieben wird, so handelt es sich immer um eine Hex-Zahl. Bei manchen Assembler-Versionen muß man hinter jede Hexzahl ein "H" schreiben um diese als solche zu kennzeichnen. Hinter eine dezimale Zahl kommt ein "D" und hinter eine binäre Zahl ein "B". Einige Beispiele: LD A,EAH, LD E,00101110B, LD L,215D.

Doch nach einiger Zeit werden die Register als Speichermöglichkeit zu wenig sein, und daher kann der Z-80 Daten und Informationen auch in Speicherzellen ablegen. Diese Daten können aber nur über den Akkumulator in den Speicher geladen werden. Die Syntax (Schreibweise) sieht folgendermaßen aus: LD (nn),A. "nn" bezeichnet hier eine beliebige 16-Bit Adresse. Dieser Befehl veranlaßt die CPU, den Wert, der im Akkumulator steht, in die angegebene Speicherzelle zu laden. Um nun den Akku in die Speicherzelle mit der Adresse A4E1 zu laden muß man folgendes schreiben: LD (A4E1),A.

Natürlich ist auch der Befehl wie folgt möglich: LD A,(A4E1). Das bewirkt, daß der Akkumulator mit dem Wert aus der Speicherzelle mit der Adresse A4E1 geladen wird. Diese Möglichkeit, ein Byte aus einer Speicherzelle zu laden, ist aber nur dem Akkumulator vorbehalten. Der Befehl LD A,(A4E2) wird "Lade den Akkumulator aus A4E2" gesprochen. Im umgekehrten Fall LD (A4E2),A heißt es "Lade den Akkumulator nach A4E2". Das "...aus A4E2" oder "nach A4E2" wird in der Mnemonic durch die Klammern, zwischen denen die Adresse steht, symbolisiert. Diese Klammern bedeuten, daß der Akkumulator den Wert erhält, der in der Speicherzelle mit der angegebenen Adresse steht, oder daß der Wert, der im Akkumulator steht, in diese Speicherzelle geladen wird.

Doch jetzt wollen wir uns ansehen, wie der Befehl LD (A4E2),A im Speicher abgelegt ist. Bei diesem Beispiel steht der Befehl ab der Adresse F000 im Speicher.

F000	32	ist der Opcode für den Befehl LD (nn),A.
F001	E2	= Adresse
F002	A4	= Adresse

Nun fällt auf, daß die Adresse nicht richtig im Speicher steht. Die beiden Bytes der Adresse können in verkehrter Reihenfolge vorkommen. Es steht nicht "A4" und "E2", sondern "E2" und "A4" in den Speicherzellen. Diese beiden Bytes der Adresse haben nun ihre eigenen Namen. Das erste Byte einer Adresse wird Highbyte (Höherwertiges Byte genannt. Bei unserem Beispiel (A4E2) ist es "A4". Das zweite Byte heißt daher Lowbyte (Niederwertiges Byte) und ist bei der Adresse A4E2 das Byte E2. Weil die Adresse verkehrt im Speicher steht, heißt das somit,

daß immer das Lowbyte zuerst und dann das Highbyte steht. Es ist eine Besonderheit vieler Mikroprozessoren, daß bei jeder Adresse zuerst das Lowbyte und dann das Highbyte kommt.

Zur Verständlichkeit dieser Artikel-Serie möchte ich noch folgende Abkürzungen klarstellen. Mit "nn" wird wie schon oben erwähnt eine Adresse oder 16-Bit-Zahl bezeichnet. "n" bezeichnet eine 8-Bit-Zahl und "r" ein beliebiges 8-Bit-Register, ausgenommen das Interruptvektorregister und das Memory Refresh Register.

Doch nun weiter, wie läßt sich jetzt ein Universalregister aus einer Speicherzelle laden? Während man nur den Akkumulator direkt aus einer Speicherzelle laden kann, ist es mit den Universalregistern nur über einen kleinen Umweg möglich. Man kann den Akku aus dem Speicherteil laden, den man im Universalregister haben möchte. Danach wird in das entsprechende Register der Inhalt des Akkus geladen.

Doch es gibt eine viel elegantere Möglichkeit. Wie schon früher erwähnt wurde, kann man die sechs 8-Bit Universalregister zu drei 16-Bit Registern zusammenfassen: BC, DE und HL. Davon hat das Registerpaar "HL" eine gewisse Vormachtstellung. HL kann mit einer 16-Bit-Adresse geladen werden. Daraufhin ist es möglich, eines der Universalregister aus der Speicherzelle mit der Adresse, die in HL steht zu laden. Wenn man hingegen BC oder DE als Adreßregister verwendet, kann man den Akku aus der Speicherzelle mit der Adresse von BC oder DE laden. Um nun das Register "D" aus der Speicherzelle mit der Adresse 345A zu laden, benötigt man folgende Befehle:

LD	HL,345A	Registerpaar HL wird mit der Adresse &H345A geladen.
LD	D,(HL)	Register D wird mit dem Wert geladen, der in der Speicherzelle mit der Adresse 345A liegt.

Überall wo es möglich ist eines der Universalregister zu verwenden, kann man auch "(HL)" verwenden. Dabei wird der Wert, der in HL steht, nicht verändert, sondern nur der Wert, der in der Speicherzelle steht, die durch HL adressiert wird.

Das nächste Mal handelt übrigens von den arithmetischen Befehlen.

Die interne Struktur des BASIC-Programmspeichers 3. Teil

In diesem Teil folgt nun die komplette Aufstellung der BASIC-Tokens mit einigen Beispielen. Es ist übrigens interessant, daß die Funktionen alle zwei Byte-Tokens haben. Die arithmetischen Rechenzeichen (+,-,*,/, und so weiter) werden nicht im ASCII-Code, sondern mit eigenen Tokens verwendet. Es gibt auch Befehle aus mehreren Einzeltokens! "PRINT USING" bekommt man zum Beispiel, wenn man die Tokens von "PRINT" und "USING" ermittelt und sie einfach hintereinander stellt. "91 (PRINT) und DD (USING)= 91 DD"

Befehl	Token	Befehl	Token	Befehl	Token
ABS	FF 86	FILES	B7	OPEN	B0
AND	F8	FIX	FF A1	OR	F9
ASC	FF 95	FN	DE	OUT	9C
ATN	FF 8E	FOR	82	PAD	FF A5
ATTR$	E9	FPOS	FF A7	PAINT	BF
AUTO	A9	FRE	FF 8F	PDL	FF A4
BEEP	C0	GET	B2	PEEK	FF 97
BIN$	FF 9D	GO TO	89	PLAY	C1
BLOAD	CD	GOSUB	8D	POINT	ED
BSAVE	CE	GOTO	89	POKE	98
CDBL	FF A0	HEX$	FF 9B	POS	FF 91
CHR$	FF 96	IF	8B	PRESET	C3
CINT	FF 9E	IMP	FC	PRINT	91
CIRCLE	BC	INKEY	EC	PSET	C2
CLEAR	92	INP	FF 90	PUT	B3
CLICK	C8	INPUT	85	READ	87
CLOAD	9B	INSTR	E5	REM	8F
CLOSE	B4	INT	FF 85	RENUM	AA
CLS	9F	IPL	D5	RESTORE	8C
CMD	D7	KEY	C7	RESUME	A7
COLOR	BD	KILL	D4	RETURN	8E
CONT	99	LEFT$	FF 81	RIGHT$	FF 82
COPY	D6	LEN	FF 92	RND	FF 88
COS	FF 8C	LET	88	RSET	B9
CSAVE	9A	LFILES	BB	RUN	8A
CSNG	FF 9F	LINE	AF	SAVE	BA
CSRLIN	E8	LIST	93	SCREEN	C5
CVD	FF AA	LLIST	9E	SET	D2
CVI	FF A8	LOAD	B5	SGN	FF 84
CVS	FF A9	LOC	FF AC	SIN	FF 89
DATA	84	LOCATE	D8	SOUND	C4
DEF	97	LOF	FF AD	SPACE$	FF 99
DEFDBL	AE	LOG	FF 8A	SPC(	DF
DEFINT	AC	LPOS	FF 9C	SPRITE	EE
DEFSNG	AD	LPRINT	9D	SQR	FF 87
DEFSTR	AB	LSET	B8	STEP	DC
DELETE	A8	MAX	CA	STICK	FF A2
DIAL	D0	MDM	CF	STOP	90
DIM	86	MERGE	B6	STR$	FF 93
DRAW	BE	MID$	FF 83	STRIG	FF A3
DSKF	FF A6	MKD$	FF B0	STRING$	E3
DSKI$	EA	MKI$	FF AE	SWAP	A4
DSKO$	D1	MKS$	FF AF	SWITCH	C9
ELSE	A1	MOD	FD	TAB(	DB
END	81	MON	CB	TAN	FF 8D
EOF	FF AB	MOTOR	CC	THEN	DA
EQV	FB	NAME	D3	TIME	EF
ERASE	A5	NEW	94	TO	D9
ERL	E1	NEXT	83	TROFF	A3
ERR	E2	NOT	E0	TRON	A2
ERROR	A6	OCT$	FF 9A	USING	E4
EXP	FF 8B	OFF	EB	USR	DD
FIELD	B1	ON	95	VAL	FF 94
				VARPTR	E7
				VPEEK	FF 98
				VPOKE	C6
				WAIT	96
				WIDTH	A0
				XOR	FA
				>	F0
				=	F1
				<	F2
				+	F3
				-	F4
				*	F5
				/	F6
				^	F7
				\	FE

Bei zwei BASIC-Befehlen gibt es Besonderheiten:

```
INTERVAL FF 85 45 52 FF 94
         3A 8F E6
```

Beim BASIC-Befehl "INTERVAL" erklärt sich die Länge dadurch, daß er aus zwei Befehlen und zwei Buchstaben zusammengestzt ist. Und zwar:

```
INT      FF 85
E        45
R        52
VAL      FF 94
```

Bei der Kurzform von "REM" sieht es ähnlich aus:

```
:        3A
REM      8F
Token '  E6
```

Nun folgen noch einige Beispiele von kompletten BASIC-Anweisungen, wobei ich zuerst die BASIC-Zeile im Klartext, danach als Speicherauszug darstelle und zum Schluß die einzelnen Speicherwerte erkläre.

```
B$=STRING$(12,"y")

42 24 F1 E3 28 0F 0C 2C 22 79 22 29

42 24          Variablenname im ASCII-Code
F1             Token von "="
E3             Token von "STRING$"
28             "(" im ASCII-Code
0F 0C          Kodierter Zahlenwert 12
2C             "," im ASCII-Code
22 79 22       String
29             ")" im ASCII-Code
```

```
F=VAL(D$(3))

46 F1 FF 94 28 44 24 28 14 29 29

46              Variablenname im ASCII-Code
F1              Token von "="
FF 94           Token von "VAL"
28              "(" im ASCII-Code
44 24           Variablenname im ASCII-Code
28              "(" im ASCII-Code
14              Kodierter Zahlenwert 3
29 29           "))" im ASCII-Code

PRINT 32+C AND 255

91 20 0F 20 F3 43 20 F8 20 0F FF

91              Token von "PRINT"
20              Space im ASCII-Code
0F 20           Kodierter Zahlenwert 32
F3              Token von "+"
43              Variablenname im ASCII-Code
20              Space im ASCII-Code
F8              Token von "AND"
20              Space im ASCII-Code
0F FF           Kodierter Zahlenwert 255
```

Bei "DATA"-Zeilen muß darauf geachtet werden, daß sämtliche "DATA"-Werte im ASCII-Code abgelegt werden.

```
DATA 15,Test

84 20 31 35 2C 54 65 73 74

84              Token von "DATA"
20              Space im ASCII-Code
31 35           Zahlenwert 15 als String
2C              "," im ASCII-Code
54 65 73 74     String
```

Bei Sprunganweisungen gibt es ebenfalls eine Besonderheit, nämlich daß alle Sprungadressen als ganzahlige Werte in je zwei Bytes abgelegt werden, wobei zuerst das niederwertige Byte und danach das höherwertige gespeichert wird. Und außerdem werden die Sprungadressen jeweils vom Wert &H0E eingeleitet.

```
GOSUB 35

8D 20 0E 23 00

8D              Token von "GOSUB"
20              Space im ASCII-Code
0E              Zahlenwerteinleitung
23 00           Kodierter Zahlenwert 35
```

Fortsetzung folgt. Wolfgang Rotschek

SV-Hardware, Fortsetzung von Seite 6

Die Sprite-Parameter werden im Video-RAM wie folgt abgelegt:

von 1B00H bis 1B7FH (6912 bis 7039):

```
1B00H  Y-Koordinate von Sprite der Ebene 0
1B01H  X-Koordinate von Sprite der Ebene 0
1B02H  Spritemusterzeiger (Spritenummer)
       von Sprite der Ebene 0
1B03H  Farbe des Sprites in der Ebene 0

1B04H  Y-Koordinate von Sprite der Ebene 1
1B05H  X-Koordinate von Sprite der Ebene 1

usw. bis zum Sprite der Ebene 31
```

Die Sprite-Muster werden im Video-RAM von Adresse 3800H bis 3BFFH (14336 bis 15359) abgelegt (32 Spritemuster x 32 Bytes).

In der kommenden Folge wird die Beschreibung des Videoprozessors fortgesetzt.

GOSUB "LABEL" MIT SVI-3x8

Durch diese kurze Routine in Maschinensprache werden die im SV-Basic fehlenden Befehle GOSUB "label" und GOTO "label" ersetzt.

Ein Beispiel:

```
10 A$=USR1("label"):GOSUB xxx
20 END
30 REM*label*
40 PRINT"Unterprogramm"
50 RETURN
```

In Zeile 10 dieses Programms wird ein Programmteil mit dem Namen "label" aufgerufen. Die Routine sucht nun nach einem REM*label*. Wenn sie dieses gefunden hat, erfolgt ein GOSUB (oder ein GOTO) dorthin. Die Zeilennummer wird dabei automatisch ermittelt. Danach kann ein normales RETURN (Zeile 50) ausgeführt werden.

Als Namen können alle Zeichen verwendet werden, die Länge darf jedoch höchstens 255 Zeichen betragen. Jedes Label kann beliebig oft aufgerufen, jedoch nur einmal mit REM*label* definiert werden. Die Sterne vor und hinter dem Namen sind unbedingt erforderlich.

Der Aufruf erfolgt durch

```
A$=USR1("label"):GOTO (GOSUB) xxx.
```

Dabei ist xxx eine beliebige Zeilennummer (muß nicht existieren).

Vorher muß das Maschinenprogramm durch ein DEFUSR1=&HD000 (bei Disk-Basic) bzw. durch ein DEFUSR1=&HF000 (bei Cassette) definiert werden.

Das nachstehende Programm POKEd das Maschinenprogramm ein. Dabei erkennt es selbständig, ob mit Cassette oder Diskette gearbeitet wird.

Ist das gesuchte Label nicht vorhanden, dann zeigt dies der Computer mit der Fehlermeldung 'ILLEGAL FUNCTION CALL' an.

Auf der folgenden Seite finden Sie ein kommentiertes Listing des Maschinenprogrammes.

RAFAEL RAZIM

```
10 REM--------------------------------
20 REM-   GOTO,GOSUB LABEL           -
30 REM- VERS 2.1 VOM 29.8.1984       -
40 REM-                              -
50 REM-   WRITTEN BY RAFAEL RAZIM    -
60 REM--------------------------------
70 DEFINTF,Y:FORF=&HE000TO&HE0FF:Y=YXOR(PEEK
(F)):NEXT
80 IFY=145THENCLEAR200,&HD000:I=0ELSECLEAR20
0,&HF000:I=1
90 Q=&HD000+I*8192
100 FORF=QTOQ+116
110 READA$
120 IFA$="d0"THENA$=HEX$(VAL("&h"+A$)+I*32)
130 PRINTHEX$(F)" "A$
140 POKEF,VAL("&h"+A$)
150 NEXT
200 PRINT"fertig!!!"
64999 END
65000 DATAc3,0d,d0,00,00,00,00,00,00,00,00,0
0,00,23,23,4e,23,46,c5,e1,0a,32,03,d0,23,4e,
23,46,ed,43,04,d0,e1,c1,ed,43,06,d0,c5,e5
65010 DATA21,00,80,01,00,50,3e,8f,ed,b1,c2,7
2,d0,3e,2a,be,c2,2e,d0,c5,e5,3a,03,d0,47,23,
ed,5b,04,d0,1a,be,c2,6d,d0,23,13,10,f7,3e,2a
65020 DATAbe,c2,6d,d0,e1,c1,2b,2b,2b,2b,2b,2
b,ed,5b,06,d0,13,13,3e,0d,12,13,7d,12,13,7c,
12,c9,e1,c1,c3,2e,d0,c3,9e,0f
```

```
D000  C3,0D,D0,     JP D00D
D003  00,           NOP
D004  00,           NOP
D005  00,           NOP                 10 Bytes Daten
D006  00,           NOP
D007  00,           NOP
D008  00,           NOP
D009  00,           NOP
D00A  00,           NOP
D00B  00,           NOP
D00C  00,           NOP
D00D  23,           INC HL              HL=BASIC Akkumulator
D00E  23,           INC HL
D00F  4E,           LD C,(HL)           Laden  BC  mit  Adresse  von  Länge  des
D010  23,           INC HL              übergebenen Labels
D011  46,           LD B,(HL)
D012  C5,           PUSH BC             Laden von HL mit BC
D013  E1,           POP HL
D014  0A,           LD A,(BC)           Laden von A mit Länge
D015  32,03,D0,     LD (D003),A         Saven von A in (D003)
D018  23,           INC HL
D019  4E,           LD C,(HL)           Laden  von  BC mit Adresse des  übergebenen
D01A  23,           INC HL              Labels
D01B  46,           LD B,(HL)
D01C  ED,43,04,D0,  LD BC,(D004)        Saven von BC in (D004)
D020  E1,           POP HL              HL mit Stapel laden
D021  C1,           POP BC              BC mit Stapel laden=Rücksprung ins BASIC
D022  ED,43,06,D0,  LD BC,(D006)        Saven von BC in (D006)
D026  C5,           PUSH BC             Stapel wiederherstellen
D027  E5,           PUSH HL
D028  21,00,80,     LD HL,8000          Laden von HL mit Anfangsadresse von BASIC
D02B  01,00,50,     LD BC,5000          Laden von BC mit Länge von BASIC-Speicher
D02E  3E,8F,        LD A,8F             Laden von A mit Token von REM (8F)
D030  ED,B1,        CPIR                Suchen nach REM
D032  C2,72,D0,     JP NZ,D072          Sprung wenn BC=0 (Illegal Function Call)
D035  3E,2A,        LD A,2A             Laden von A mit Code von "*"
D037  BE,           CP (HL)             Vergleich, ob (HL)="*"
D038  C2,2E,D0,     JP NZ,D02E          Wenn nicht, Suche nach REM fortsetzen
D03B  C5,           PUSH BC             BC und HL auf Stapel retten
D03C  E5,           PUSH HL
D03D  3A,03,D0,     LD A,(D003)         Laden von A mit Länge des Labels
D040  47,           LD B,A              Laden von B mit A
D041  23,           INC HL
D042  ED,5B,04,D0,  LD DE,(D004)        Laden von DE mit Adresse des Labels
D046  1A,           LD A,(DE)           Lade A mit (DE)
D047  BE,           CP (HL)             Vergleich ob (HL)=A
D048  C2,6D,D0,     JP NZ,D06D          Wenn nicht, hole BC und HL vom Stapel
D04B  23,           INC HL
D04C  13,           INC DE
D04D  10,F7,        DJNZ,F7     7D046   Vergleich, ob B=0, wenn nicht Sprung
D04F  3E,2A,        LD A,2A             Laden von A mit Code "*"
D051  BE,           CP (HL)             Vergleich, ob (HL)="*"
D052  C2,6D,D0,     JP NZ,D06D          Wenn nicht, hole BC und HL vom Stapel
D055  E1,           POP HL              HL vom Stapel holen
D056  C1,           POP BC              BC vom Stapel holen
D057  2B,           DEC HL              Subtrahiere 6 von HL
D058  2B,           DEC HL
D059  2B,           DEC HL
D05A  2B,           DEC HL
D05B  2B,           DEC HL
D05C  2B,           DEC HL
D05D  ED,5B,06,D0,  LD DE,(D006)        Lade DE mit Rücksprungsadresse ins BASIC
D061  13,           INC DE
D062  13,           INC DE
D063  3E,0D,        LD A,0D
D065  12,           LD (DE),A
D066  13,           INC DE
D067  7D,           LD A,L              Lade A mit Lowbyte von Adr.REM-Zeile
D068  12,           LD (DE),A           Sprungadresse von GOSUB verändern
D069  13,           INC DE
D06A  7C,           LD A,H              Lade A mit Highbyte von Adr.REM-Zeile
D06B  12,           LD (DE),A           Sprungadresse von GOSUB verändern
D06C  C9,           RET                 RETURN ins BASIC
D06D  E1,           POP HL
D06E  C1,           POP BC
D06F  C3,2E,D0,     JP D02E
D072  C3,9E,0F,     JP 0F9E
```

SOFTWARE UHR

Das Programm ist sowohl auf dem SV-328 als auch auf dem SV-318 lauffähig. Da Sie das Assemblerlisting des Programmes vor sich haben, können Sie das Programm leicht im Speicher verschieben (z.B. nach F300H). Das System der Routine erlaubt es Ihnen auch, einen eigenen vom Interrupt gesteuerten Zählvorgang zu programmieren. Sie können sowohl die Stellenanzahl frei festlegen als auch die maximale Größe jeder einzelnen Ziffer bestimmen. Sie erweitern oder verkürzen die Tabelle, die bei MS beginnt mit der Variablen LN, die die Tabellenlänge enthält. Bedenken Sie aber, daß die eigentliche Tabelle doppelt so lange ist, da sie ja zusätzlich die Eintragungen für die maximale Zifferngröße enthält. Sie müssen dann weiters das erste Byte auf 0 und das zweite Byte auf die maximale Größe setzen.

Um das Programm generell in Gang zu setzen, müssen Sie in die Variable ON eine 0 schreiben (bzw. POKEn) und das Programm 'STRT/STOP' aufrufen, welches das Programm 'CLOCK' an die Interruptroutine anhängt. Wollen sie die Uhr wieder abkoppeln, so schreiben Sie lediglich in die Variable ON eine 1 und rufen wieder 'STRT/STOP' auf.

Wenn Sie die Uhr stoppen möchten, ohne sie von der Interruptroutine abzuhängen, dann schreiben Sie eine 0 in Variable PM. PM enthält die Erlaubnis zum Weiterzählen. Wenn diese gleich 0 ist, wird die Routine zwar aufgerufen, sie zählt aber nicht weiter. Um die Uhr wieder in Gang zu bringen, schreiben Sie einfach einen Wert ungleich 0 in PM.

Um die Uhrzeit zu setzen, ohne vom Weiterzählen behindert zu werden, sollten Sie PM auf 0 setzen und dann die enstprechenden Werte in die Variablen MS (sie gibt die Zeit in 20 Millisekunden an), SE (gibt die Zeit in Sekunden an), MI (in Minuten) und ST (in Stunden) schreiben. Dann starten Sie die Uhr durch Setzen der Variablen PM ungleich 0. Wenn Sie die Zeit dann wissen wollen, PEEKen Sie sie aus den Variablen heraus.

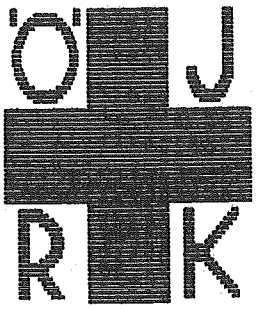
```
        SV-328 Z80 ASSEMBLER
        VERSION 3.7
        11.09.1984
        PHILIPP OTT

 ASSEMBLERTEXT GEFUNDEN
DURCHLAUF NR: 1


 KEINE ERRORS
 77 ZEILEN

DURCHLAUF NR: 2
ADDR  OPCODES  SYMBOLE  BEF.  OPERAND        KOMMENTAR

                                            ;Programm STRT/STOP

D200                    ORG   0D200H        ;Das Programm startet oder

D200  F3       STRT/ST: DI                  ;stoppt die Uhr.
D201  1179FE            LD    DE,0FE79H     ;Adresse, zu der der Computer
                                            ;bei jedem Interr. hinspringt.
D204  211FD2            LD    HL,CLON       ;Tabelle CLock ON.
D207  3A1ED2            LD    A,(ON)        ;Ist (ON) gleich 0, dann
D20A  A7                AND   A             ;wird die Uhr gestartet,
D20B  2805              JR    Z,EIN         ;ansonsten wird sie
D20D  2123D2            LD    HL,CLOFF      ;gestoppt.
D210  1801              JR    CONT          ;Springe zu CONT.

D212  3C       EIN:     INC   A             ;Starte die Uhr.

D213  321ED2   CONT:    LD    (ON),A        ;Setzte oder loesche (ON).
D216  010400            LD    BC,04         ;Lade [BC] mit 4.
D219  EDB0              LDIR                ;Verschiebe 4 Bytes nach FE79H.
D21B  FB                EI                  ;Gib Interrupts frei.
D21C  76                HALT                ;Warte auf Interrupt.
D21D  C9                RET                 ;Kehre zurueck.

D21E  00       ON:      DEFB  00            ;Enthaelt Starterlaubnis fuer Uhr.

D21F  CD30D2   CLON:    CALL  CLOCK         ;Befehlesfolge fuer CLock ON.
                                            ;Rufe Uhrprogramm auf und
D222  C9                RET                 ;springe zurueck.

D223  C9       CLOFF:   RET                 ;Befehlesfolge fuer CLock OFF.
                                            ;Springe direkt zur Interrupt-

D224  0000              DEFW  00            ;routine zurueck.
D226  C9                RET

                                            ;Ende der Start und Stop-
                                            ;routine fuer die Uhr.


                                            ;Programm CLOCK
                                            ;Das Programm stellt eine
                                            ;Softwareuhr dar, die Inter-
                                            ;rupt gesteuert die Zeit
                                            ;in 20 Millisekunden, Sekunden,
                                            ;Minuten und Stunden misst.

D230                    ORG   0D230H        ;Lege das Programm
                                            ;ab 0D230H ab.
```

```
D230  F3        CLOCK:  DI              ;Sperre Interrupts.
D231  D9                EXX             ;Vertausche Universalregister.
D232  DD2162D2          LD   IX,MS      ;Lade [IX] mit Tabellen-
                                        ;anfang.
D236  3A60D2            LD   A,(LN)     ;Lade [B] mit der Tabellen-
D239  47                LD   B,A        ;laenge.
D23A  3A61D2            LD   A,(PM)     ;Lade [C] mit Erlaubnis
D23D  4F                LD   C,A        ;zum Zeiterhoehen.

D23E  79        LOOP:   LD   A,C        ;Siehe nach, ob die Erlaubnis
D23F  A7                AND  A          ;zum erhoehen da ist ([C] <> 0).
D240  2812              JR   Z,NXT      ;Wenn nicht, dann springe
                                        ;zu NXT.
D242  DD3400            INC  (IX+00)    ;Erhoehe Zeitzaehler und
D245  DD7E00            LD   A,(IX+00)  ;sieh nach, ob er groesser ist
                                        ;als das Vergleichsbyte.
D248  DDBE01            CP   (IX+01)    ;Wenn dem nicht so ist,
D24B  0E00              LD   C,0        ;dann loesche die Erlaubnis
D24D  3805              JR   C,NXT      ;zum Erhoehen
                                        ;und springe zu NXT.
D24F  0C                INC  C          ;Setze Erlaubnisflag.
D250  DD360000          LD   (IX+00),00 ;Loesche Zeitzaehler.

D254  DD23      NXT:    INC  IX         ;Erhoehe Pointer auf naechsten
D256  DD23              INC  IX         ;Zeitzaehler.
D258  10E4              DJNZ LOOP       ;Das ganze [B] mal.
D25A  D9                EXX             ;Vertausche wieder die
                                        ;Universalregister und
D25B  C9                RET             ;kehre zurueck.

D260                    ORG  0D260H     ;Tabelle fuer Zeitzaehler be-
                                        ;ginnt bei D260H.

D260  04        LN:     DEFB 04         ;Laenge der Tabelle.

D261  01        PM:     DEFB 01         ;Enthaelt die Erlaubnis zum
                                        ;Zeiterhoehen.

D262  00        MS:     DEFB 00         ;Millisekunden.

D263  32                DEFB 50         ;Vergleichsbyte.

D264  00        SE:     DEFB 00         ;Sekunden.

D265  3C                DEFB 60         ;Vergl.

D266  00        MI:     DEFB 00         ;Minuten.

D267  3C                DEFB 60         ;Vergl.

D268  00        ST:     DEFB 00         ;Stunden.

D269  18                DEFB 24         ;Vergl.
                                        ;Ende der Softwareuhr.
```

```
KEINE ERRORS
PROGRAMMBEGINN : D200
PROGRAMMENDE   : D26A
PROGRAMMLAENGE:  106 BYTES

SYMBOLE:
STRT/STOP : D200 ,  53760      ; EIN      : D212 ,  53778
CONT      : D213 ,  53779      ; ON       : D21E ,  53790
CLON      : D21F ,  53791      ; CLOFF    : D223 ,  53795
CLOCK     : D230 ,  53808      ; LOOP     : D23E ,  53822
NXT       : D254 ,  53844      ; LN       : D260 ,  53856
PM        : D261 ,  53857      ; MS       : D262 ,  53858
SE        : D264 ,  53860      ; MI       : D266 ,  53862
ST        : D268 ,  53864
```

****

JUGENDROTKREUZ-KURSE



Es genügt nicht nur, daß man helfen will, man muß auch helfen können! Deshalb veranstalten das Jugendrotkreuz und das Rote Kreuz Kurse! Im Schuljahr 1983/84 hat zum Beispiel jeder 7.Schüler an einem Jugendrotkreuzkurs teilgenommen.

76.600	Teilnehmer bei Schwimmprüfungen
60.000	Teilnehmer bei Radfahrerprüfungen
55.000	Teilnehmer bei Erste-Hilfe-Kursen
3.200	Teilnehmer bei Mutter-Kind-Kursen
2.000	Teilnehmer bei Hauskrankenpflegekursen
1.500	Teilnehmer bei Mopedfahrerprüfungen
198.300	Teilnehmer insgesamt

Besuchen auch Sie einen Kurs des Jugendrotkreuzes oder des Roten Kreuzes! Sie können Leben retten!

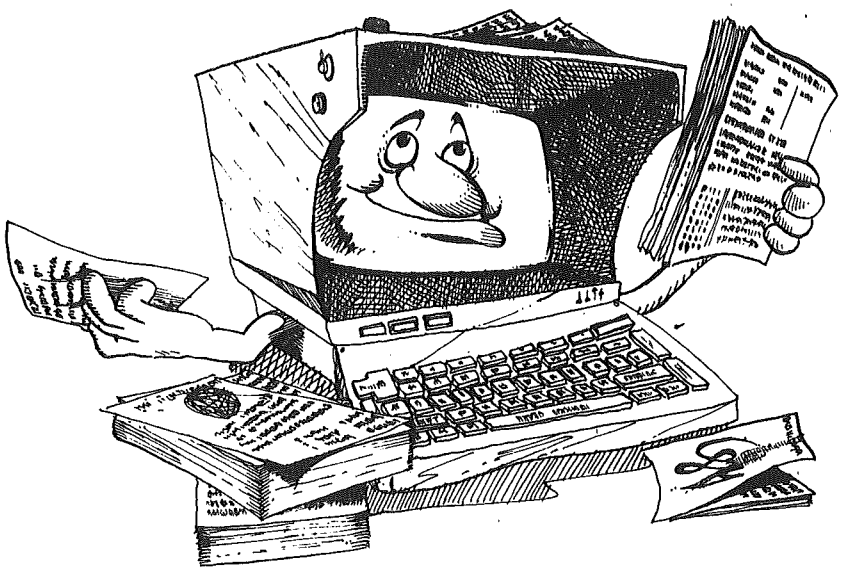
```
*****************************************
*                                       *
*    MUSIK AUF DEM SPECTRAVIDEO SV-328  *
*                                       *
*****************************************
```

Unsere SVi-Computer haben nicht nur eine exzellente Grafik (siehe Heft 1/84), sondern auch Nachwuchskomponisten kommen auf ihre Rechnung.

Sowohl der SV-318 als auch der SV-328 haben drei Tongeneratoren mit einem Frequenzbereich von je 8000 Hz. Dieser Bereich kann mit dem BASIC-Befehl "PLAY" ausgenutzt werden. Mit der anderen Tonerzeugungsanweisung "SOUND" kann man theoretisch gar von 6Hz bis 100000 Hz gehen. Allerdings liegt der Bereich der hörbaren Töne zwischen 16 und 20000 Hz.

Aber nicht nur im Frequenzbereich kann der SV-328 auftrumpfen. Man kann Hüllkurven bestimmen und die Oberwellen verändern. Mit Hüllkurven werden die Ein- und Ausschwingphasen von natürlichen Instrumenten simuliert, und das Oberwellenspektrum bestimmt die Klangfarbe eines Tones.

Die Programmierung erfolgt, im Gegensatz zu einem anderen bekannten Home-Computer, sehr komfortabel. Einerseits gibt es, wie oben erwähnt, den BASIC-Befehl "PLAY". Er ist primär für das Komponieren von Liedern zuständig. Deshalb wird der Komfort groß geschrieben, und man kann zum Beispiel die einzelnen Noten in mit ihren Namen eingeben ("c" oder "d" oder andere). Mehr fürs Experimentieren gedacht ist der "SOUND"-Befehl. Er ist nicht so komfortabel wie die "PLAY"-Anweisung, weil man die Frequenzen, und nicht die Notennamen eingeben muß, aber andererseits bietet das mehr Möglichkeit, unharmonische Töne und Geräusche zu entwickeln.



```
10 REM*********************************
20 REM*   HELMUT  KVASNICKA           *
30 REM*  "PRELUDE VON BACH"           *
40 REM*  WIEN, AUGUST 1984"           *
50 REM*  VERSION 4.3T 1984            *
60 REM*********************************
70 PRINT"KLEINES PRAELUDIUM"
80 PRINT:PRINTTAB(9);"von Johann Sebastian B
ach"
90 FOR I=1TO2
100 PLAY"v8t90o418c4r8fg4r8o5co4ffo5do4ffer4
","t90o3g4r818o4dc4r8eddfddcr4","v 9t90116o2
r16cego3c4r16o2cego3c4r16cdco2bo3do2gbo3co2c
ego3cdef"
110 PLAY"d4r8ef4r8feg116gfgae8d8r16v9gbo5d",
"o3b4r8o4cd4r8dceccco3br4","go2gbo3dg4r16116
o2gbo3dg4r16o2go3ceo2e8f8go3gdo2bg8o3v8g8"
120 PLAY"edco4bagf+ef+df+af+df+ao5dco4bagf+e
dcegecego5co4bagf+gaf+def+gabo5co4a o5d8o4g
8bagf+gdo3bgr4","18cgcadacao2bo3ao2bo3gcgo2b
o3go2ao3go2ao3gf+af+do2bo3ecdgr8o2v9g4"
130 NEXT
140 FOR I=1TO2
150 PLAY"v8o414gr8o5c8d4r818efffedcr4c4r8o4b
-","o4d4r818ag4r8o5cdddco4bar4g4r8g","v9o211
6r16gbo3dg4r16o2gbo3dg4r16abag+beg+ao2ao3cea
bo4cdeo3ego4ce4"
160 PLAY"a4r8o518abo6co5effdr16116o4v9gbo5d"
,"f4r818o5cfedcco4br4","r16o3fao4cfefedcdco3
bgo4cegf+gdo3bo4do3gb"
170 PLAY"gfedco4b-agafao5co4afao5cfedco4bagf
gfgo5do4gfgo5dedco4bagfedfao5co4bgbo5dg8c8ed
co4bo5co4gecr4","18eo4co3eo4co3fo4co3eo4co3d
bcao2bo3fedcefafdgfeafgc4o2v9c4"
180 NEXT
```

Keine Geräusche, sondern sehr wohlklingende Lieder haben uns zwei Clubmitglieder gegeben. Das eine ist von dem bekannten Komponisten Ludwig van Beethoven und heißt "Für Elise". Nicht weniger bekannt ist Johann Sebastian Bach, der die Vorlage zum zweiten Computerständchen geliefert hat.

Beide Lieder sind mit "PLAY" in den Speicher gebannt und man hört dezenten Computersound, weil die beiden Programmierer sich nicht einmal die Mühe gemacht haben, mit den vorhandenen Möglichkeiten der Klangmanipulation einen Klavier- oder Geigenklang nachzuahmen.

Sehen wir uns kurz das zweite Programm an Das "PRELUDE" von Johann S. Bach ist für eine Orgel komponiert worden. Da die üblichen Akzentuierungen, die für ein Klavier-Stück notwendig sind, entfallen, eignet sich dieses Werk bestens dafür, in Computersound umgewandelt zu werden.

Ludwig van Beethoven machte es dem zweiten Programmierer schon etwas schwerer. "Für Elise" wird sehr oft auf Klavieren gespielt, daher mußte hier etwas vom ausgefeilten Originalklang abgewichen werden.

Nicht nur wegen der fehlenden Akzentuierung ist "Für Elise" etwas schlechter im Klang. Durch den Versuch, mit "READ" und "DATA" zu arbeiten, gab es zwei Probleme. Durch noch ungeklärte Ursachen verschluckte der Compu-

ter einige Töne, wenn keine "FOR...NEXT"-Schleifen die einzelnen Passagen trennten. Sie können das ausprobieren, wenn Sie sämtliche Schleifen auf 2000 verkürzen. Als allerdings die Schleifen angebracht waren, da erkannte man, daß zwischen den Passagen zu große Pausen waren. Leider blieb auch das ein ungeklärtes Rätsel. Eine Verkürzung der Schleifen brachte keinen Erfolg.

Ungeachtet solcher Schwierigkeiten solten Sie sich das Musikerlebnis nicht trüben lassen und wenn Sie halbwegs Noten schreiben können, dann versuchen Sie doch einmal ein Lied zu "computeriesieren".

Bei dem oben erwähnten Konkurrenzprodukt vom Spectravideo muß man übrigens jeden Wert einzeln berechnen und in den Soundgenerator poken. Die beiden kurzen Programme auf dem Spectravideo wären dort seitenlange Anhäufungen von "DATA"-Zeilen.

```
10 REM ********************************
20 REM *                              *
30 REM *  FUER ELISE VON BEETHOVEN    *
40 REM *     VON GERHARD FALLY        *
50 REM *    1020 WIEN, JUNI 1984      *
60 REM *         VERSION 2.7          *
70 REM *                              *
80 REM ********************************
90 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
100 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAA","R16XB$;R16":FORI=1TO2500:NEXT
110 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
120 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAA","R16XB$;"
130 RESTORE300:READA$,B$:PLAY"XA$;","XB$;":FORI=1TO2500:NEXT
140 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
150 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAA","R16XB$;"
160 READA$,B$:PLAY"XA$;","XB$;":FORI=1TO2500:NEXT
170 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
180 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAA","R16XB$;"
190 PLAY"T70L16V1105CR64CR64CR64CCCCFL32FEL8EDL32B-B-B-AL16AGFEDCO4B-B-AAL32AGAB-","T70L1604V10CR64CR64CR6403FAO4CO3AO4CO3AFB-O4DO3B-O4DO3B-FO4EO3B-O4EO3B-O4EO3FAO4CO3AO4CO3A","T70L16V1003B-AGR2R4R803FR16FR16"
200 PLAY"O5L16CCCCDD+EEEEFO4AO5CCCCDL32DO4BO5CGO4GO5GO4AO5GO4BO5GCGDGEG","FAO4CO3AO4CO3AEAO4CO3AO4DO3FGO4EO3GO4EO3GO4FEER16L32FR32GR32FR32L16G","R2DR16R4R8O4CCR8L32GR32ER32DR32L16E"
210 PLAY"O6CO5BAGFEDGFDCGO4GO5GO4AO5GO4BO5GCGDGEGO6CO5BAGFEDGFDEFEE-EO4BO5EE-EO4BO5EE-L16EEEO4BO5EE-EEEO4BO5EE-EE-EE-","GO3AABBO4CCR8L32GR32GR32GR32L16GGO3AABBG+G+","EO3FFGGR8R1604L32FR32ER32FR32L16EEO3FFGGBB"
220 FORI=1TO2500:NEXT
230 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
240 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAA","R16XB$;":FORI=1TO2500:NEXT
250 RESTORE300:READA$,B$:PLAY"XA$;","XB$;":FORI=1TO2500:NEXT
260 RESTORE:READA$,B$:PLAY"XA$;G+B05CCR8","XB$;":FORI=1TO2500:NEXT
270 RESTORE:READA$,B$:PLAY"O4EXA$;O5CO4BAAAAV10AV9AV7AV4AA","R16XB$;"
280 DATAV11L16T9005EE-EE-EO4BO5DCO4AAR8CEABBR8E
290 DATAV10L16T90R202AO3EAR402EO3EG+R402AO3EA
300 DATABO5CDEEEO4GO5FEDDDO4FO5EDCCCO4EO5DCO4BBR16O4EO5ER8EO6ER8O5E-ER8E-
310 DATAR16R8O3CGO4CR8R1602GO3GBR8R1602AO3EAR8R1602EO3EO4ER8EO5ER8E-ER8E-ER16
```

BYTES-SUCHPROGRAMM

In dieser Ausgabe bringen wir ein dokumentiertes Assemblerlisting zu dem Bytesuchprogramm aus dem vorigen Heft. Das Programm hatte leider ein paar Schönheitsfehler. Durch unglückliche Umstände konnte es passieren, daß es abstürzt. Weiters ist mir ein Fehler in der Umsetzung des Programmes in die Form von DATA-Zeilen unterlaufen. In der Zeile 370 ist das letzte '++' durch ein 'D0' zu ersetzen. Das voranstehende '10' entspricht dem 'DJNZ'-Befehl des Z80 und das 'D0' ist die relative Versetzungszahl zum PC.

Ich habe das Programm geringfügig verändert und hoffe, daß nun alle Fehler beseitigt sind.

Das untenstehende BASIC-Programm enthält das neue Maschinenprogramm. Das BASIC-Programm erkennt, ob mit Cassette oder mit Diskette gearbeitet wird und legt den Maschincode dementsprechend an (F000 oder D000).

```
10 ' BYTES SUCHEN
20 ' Version 2.7 vom 14.09.1984
30 ' Philipp Ott
40 GOSUB 340
50 IFB=145THENCLEAR200,&HD000:D=0ELSECLEAR200,&HF000:D=1
60 RESTORE150:A=&HD000+8192*D
70 READI$:IFI$="*"THEN120
80 IFI$="++"THENI$=HEX$(&HD0+D*32)
90 IFI$="+-"THENI$=HEX$(&HD1+D*32)
100 PRINT HEX$(A);"    ";I$
110 POKEA,VAL("&H"+I$):A=A+1:GOTO70
120 PRINT"Ok. Das Programm ist im Speicher.
130 DEFUSR0=&HD000+D*8192
140 END
150 DATA C3,8,++,0,0,0,0,0,FE,3,C2,5
160 DATA 9,23,23,4E,23,46,C5,E1,7E,A7,CA,9E
170 DATA F,1F,1E,18,DA,7,9,FE,40,1E,F,D2
180 DATA 7,9,32,7,++,23,4E,23,46,ED,43,3
190 DATA ++,F3,2A,3,++,11,0,+-,3A,7,++,47
200 DATA CD,C6,++,87,87,87,87,4F,23,CD,C6,++
210 DATA 81,12,23,13,10,EE,21,0,0,22,5,++
220 DATA FB,2A,5,++,23,22,5,++,11,0,+-,3A
230 DATA 7,++,47,7C,B5,CA,AE,++,1A,BE,13,23
240 DATA C2,59,++,10,F2,2A,5,++,11,0,+-,E7
250 DATA CA,55,++,3A,6,++,CD,9B,++,3A,6,++
260 DATA CD,A1,++,3A,5,++,CD,9B,++,3A,5,++
270 DATA CD,A1,++,3E,20,CD,AC,++,C3,55,++,E6
280 DATA F0,1F,1F,1F,1F,E6,F,C6,30,FE,3A,FA
290 DATA AC,++,C6,7,DF,C9,3E,D,CD,AC,++,3E
300 DATA A,C3,AC,++,FE,30,D8,FE,3A,3F,D0,FE
310 DATA 41,D8,FE,47,3F,C9,CD,B,17,CD,B8,++
320 DATA DA,9E,F,D6,30,FE,A,D8,D6,7,C9,E7
330 DATA *
340 DEFINTA-Z:FORA=&HE000TO&HE0FF
350 B=B XOR(PEEK(A)):NEXT:RETURN
```

Das dokumentierte Assemblerlisting beginnt auf der folgenden Seite.

Bytes-Suchprogramm
Assemblerlisting

```
     SV-328 Z80 ASSEMBLER
     VERSION 3.7
     10.09.1984
     PHILIPP OTT

 ASSEMBLERTEXT GEFUNDEN
DURCHLAUF NR: 1


 KEINE ERRORS
 121 ZEILEN

DURCHLAUF NR: 2
ADDR  OPCODES  SYMBOLE  BEF.  OPERAND        KOMMENTAR

                                            ;BYTES-SUCH-PROGRAMM
                                            ;Das Programm sucht N (max. 64)
                                            ;Bytes in der Bank 0 des SV-328.
                                            ;Die Adressen, bei denen die
                                            ;Bytes gefunden werden, werden
                                            ;direkt als vierstellige Hexa-
                                            ;dezimalzahlen ausgegeben
                                            ;(CRT oder LPT).

D000                    ORG   0D000H        ;Programm ab 0D000 ablegen.
D000  C308D0            JP    START         ;Sprung zum Hauptprogramm.

D003  0000      USR:    DEFW  0             ;(USR) enthaelt Adresse des
                                            ;ersten Zeichens des Strings.

D005  0000      ADR:    DEFW  0             ;Aktuelle Suchadresse.

D007  00        LEN:    DEFB  0             ;Laenge der Tabelle.

D008            TAB:    EQU   0D100H        ;Tabellenbegin, Tab. enthaelt
                                            ;zu suchende Bytes.

D008  FE03      START:  CP    03            ;Vergleich, ob Argument ein
                                            ;String ist.
D00A  C20509            JP    NZ,0905H      ;Wenn kein String, dann
D00D  23                INC   HL            ;TYPE MISMATCH.
D00E  23                INC   HL
D00F  4E                LD    C,(HL)
D010  23                INC   HL
D011  46                LD    B,(HL)
D012  C5                PUSH  BC            ;[BC] enhaelt Pointer auf
                                            ;Stringlaenge.
D013  E1                POP   HL            ;[HL] zeigt nun auf Laenge.
D014  7E                LD    A,(HL)        ;hole Laenge des Stringes und
D015  A7                AND   A             ;wenn Laenge = 0, dann
D016  CA9E0F            JP    Z,0F9EH       ;ILLEGAL FN CALL.
D019  1F                RRA                 ;Wenn die Laenge un-
D01A  1E18              LD    E,24          ;gerade ist, dann
D01C  DA0709            JP    C,907H        ;MISSING OPERAND
D01F  FE40              CP    64            ;Wenn die Laenge > 64,
D021  1E0F              LD    E,15          ;dann springe zu
D023  D20709            JP    NC,907H       ;STRING TOO LONG.
D026  3207D0            LD    (LEN),A       ;Ok, die Laenge ist korrekt.
D029  23                INC   HL            ;Lies Adresse des ersten
D02A  4E                LD    C,(HL)        ;Zeichens des Strings ein
D02B  23                INC   HL            ;und speichere Adresse in
D02C  46                LD    B,(HL)        ;[BC], lege [BC] in
D02D  ED4303D0          LD    (USR),BC      ;(USR) ab.
D031  F3                DI
D032  2A03D0            LD    HL,(USR)      ;[HL] zeigt auf erstes Zeichen,
D035  1100D1            LD    DE,TAB        ;[DE] enthaelt Tabellenanfang
D038  3A07D0            LD    A,(LEN)       ;und [B] wird mit der Tabellen-
D03B  47                LD    B,A           ;laenge geladen.

D03C  CDC6D0    STRING: CALL  CHECK         ;Ueberpruefen, ob (HL) in
D03F  87                ADD   A,A           ;der Menge (0..9,A..F) und wenn
D040  87                ADD   A,A           ;(HL) in der Menge ist, dann
D041  87                ADD   A,A           ;multipliziere die Elementsnum-
D042  87                ADD   A,A           ;mer-1 mit 16 und lade
D043  4F                LD    C,A           ;[C] mit dem Ergebnis.
D044  23                INC   HL            ;Erhoehe Pointer und hole Ele-
D045  CDC6D0            CALL  CHECK         ;mentsnummer-1,addiere [C]
D048  81                ADD   A,C           ;und speichere zu suchendes Byte
D049  12                LD    (DE),A        ;in der Tabelle.
D04A  23                INC   HL            ;Erhoehe beide Pointer.
D04B  13                INC   DE            ;Setze die Umwandlung so lange
D04C  10EE              DJNZ  STRING        ;fort, bis das Tabellenende er-
                                            ;reicht ist.
D04E  210000            LD    HL,00         ;Loesche [HL] und beginne
D051  2205D0            LD    (ADR),HL      ;Suche bei 00.
D054  FB                EI

D055  2A05D0    MAINLOOP:LD   HL,(ADR)      ;[HL] enthaelt Suchadresse.
D058  23                INC   HL

D059  2205D0    MAINL1.: LD   (ADR),HL      ;Aktualisiere Adresse.
D05C  1100D1             LD   DE,TAB        ;[DE] zeigt auf Tabellenanfang.
D05F  3A07D0             LD   A,(LEN)       ;[B] wird mit Tabellenlaenge
D062  47                 LD   B,A           ;geladen.
```

```
D063  7C       LOOP:   LD   A,H          ;Sieh nach, ob [HL] = 00
D064  B5               OR   L            ;und wenn, dann beende die
D065  CAAED0           JP   Z,EXIT       ;Sucherei.
D068  1A               LD   A,(DE)       ;Vergleiche, ob (HL) gleich (DE)
D069  BE               CP   (HL)         ;und erhoehe die Pointer.
D06A  13               INC  DE
D06B  23               INC  HL           ;Wenn (DE) ungleich (HL),dann
D06C  C259D0           JP   NZ,MAINL1.   ;springe nach MAINL1.
D06F  10F2             DJNZ LOOP         ;Ueberpruefe [B] Zeichen.
D071  2A05D0           LD   HL,(ADR)     ;[HL] enthaelt gefundene Adresse.
D074  1100D1           LD   DE,TAB       ;Vergleiche ueber ROM-Routine,
                                         ;ob [HL] gleich Tabellenanfang.
D077  E7               RST  20H          ;Wenn [HL] = [DE], dann
D078  CA55D0           JP   Z,MAINLOOP   ;springe zur Hauptschleife.
D07B  3A06D0           LD   A,(ADR+1)    ;Ausgabe der gefundenen
D07E  CD9BD0           CALL CONVERT      ;Adresse in Form einer
D081  3A06D0           LD   A,(ADR+1)    ;vierstelligen Hexadezimal-
D084  CDA1D0           CALL ASCII        ;zahl.
D087  3A05D0           LD   A,(ADR)
D08A  CD9BD0           CALL CONVERT
D08D  3A05D0           LD   A,(ADR)
D090  CDA1D0           CALL ASCII
D093  3E20             LD   A,32         ;Ausgabe eines
D095  CDACD0           CALL PRINT        ;Leerzeichens und
D098  C355D0           JP   MAINLOOP     ;naechste Adresse suchen.

D09B  E6F0     CONVERT: AND  0F0H        ;Loesche unteren Nibble
D09D  1F               RRA               ;und verschiebe [A]
D09E  1F               RRA               ;um 4 Bits nach rechts.
D09F  1F               RRA
D0A0  1F               RRA

D0A1  E60F     ASCII:  AND  15           ;Loesche oberen Nibble,
D0A3  C630             ADD  A,30H        ;addiere 30H und wenn
D0A5  FE3A             CP   3AH          ;[A] < 3AH, dann springe
D0A7  FAACD0           JP   M,PRINT      ;zu PRINT, ansonsten addiere
D0AA  C607             ADD  A,07         ;7 zu  Akku.

D0AC  DF       PRINT:  RST  18H          ;Schicke Zeichen an CRT
D0AD  C9               RET               ;oder LPT und kehre zurueck.

D0AE  3E0D     EXIT:   LD   A,13         ;Schicke CR-LF
D0B0  CDACD0           CALL PRINT        ;und beende das Programm.
D0B3  3E0A             LD   A,0AH
D0B5  C3ACD0           JP   PRINT

D0B8  FE30     LOOK:   CP   30H          ;Sieh nach, ob das Zeichen in
D0BA  D8               RET  C            ;[A] in der Menge '0' bis '9' oder
D0BB  FE3A             CP   3AH          ;in der Menge 'A' bis 'F' vorkommt.
D0BD  3F               CCF               ;Ist das Carryflag gesetzt, liegt
D0BE  D0               RET  NC           ;das Zeichen nicht in der Menge.
D0BF  FE41             CP   41H
D0C1  D8               RET  C
D0C2  FE47             CP   47H
D0C4  3F               CCF
D0C5  C9               RET

D0C6  CD0B17   CHECK:  CALL 170BH        ;Hole Zeichen aus (HL)
D0C9  CDB8D0           CALL LOOK         ;in [A], ueberpruefe, ob [A]
D0CC  DA9E0F           JP   C,0F9EH      ;in erlaubter Menge und wenn
D0CF  D630             SUB  30H          ;nicht, dann ILLEGAL FN CALL.
D0D1  FE0A             CP   10           ;Sonst stelle Elementsnummer-1
D0D3  D8               RET  C            ;her und kehre zurueck.
D0D4  D607             SUB  07
D0D6  C9               RET
```

```
KEINE ERRORS
PRGRAMMBEGINN : D000
PROGRAMMENDE  : D0D7
PROGRAMMLAENGE:  215 BYTES

SYMBOLE:
USR       : D003 ,  53251
ADR       : D005 ,  53253
LEN       : D007 ,  53255
TAB       : D100 ,  53504
START     : D008 ,  53256
STRING    : D03C ,  53308
MAINLOOP  : D055 ,  53333
MAINL1.   : D059 ,  53337
LOOP      : D063 ,  53347
CONVERT   : D09B ,  53403
ASCII     : D0A1 ,  53409
PRINT     : D0AC ,  53420
EXIT      : D0AE ,  53422
LOOK      : D0B8 ,  53432
CHECK     : D0C6 ,  53446

ENDE DER ASSEMBLIERUNG
```

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

SINGLE STEP

Leider ist es eine hartgesottene Tatsache, daß neuentwickelte Maschincodeprogramme auf dem Papier besser funktionieren, als im Computer. Besonders bei SV-Computer-Besitzern macht sich große Freude breit, wenn man vor einem leeren Bildschirm sitzt, auf dem eigentlich schon hunderte Daten vorbeiflimmern müßten. Passiert so etwas im BASIC, so kann man mit "TRON/TROFF" und einem reaktionsschnellen Griff zur STOP-Taste manche Fehler schneller als erwartet finden. Es gibt auch die Möglichkeit, sich nach einer Errormeldung noch sämtliche Variablen aufzulisten und deren tatsächliche Werte mit den Sollwerten zu vergleichen.

Nichts davon hat man im Maschincode. Oft stürzt der Computer ab, oder läßt sich durch nichts mehr bewegen, vom MC-Programm zum BASIC-Modus zurückzukehren. Auch sind die Registerinhalte unwiederbringlich verloren, wenn man sich wieder im BASIC befindet. Selbst das Einfügen von Testmechanismen, im BASIC zum Beispiel in der Form "10 IF A=0 THEN END" realisiert, verursacht meist mehr Fehler, als dadurch ausgemerzt werden.

In einer so aussichtslosen Situation gibt es eigentlich nur zwei Möglichkeiten. Entweder man wirft frustriert die Flinte ins Korn und wendet sich vom Maschincode ab zum einfacheren BASIC, oder man holt sich einen Debugger oder Single Stepper. Dieser hat die Aufgabe, ein Maschincodeprogramm Befehl für Befehl abzuarbeiten und auf Wunsch alle Register und sonstigen Informationen sichtbar zu machen, die eigentlich nur für den Prozessor bestimmt sind, die aber auch den Programmierer brennend interessieren. Und genau so einen Debugger gibt es jetzt bei einem "Spectra Video Club Austria"-Mitglied.

Gerhard Fally hat im BASIC einen Single Stepper entwickelt, der in seiner Grundfunktion wartet, bis die Leertaste gedrückt worden ist. Dann arbeitet er einen Befehl ab, zeigt sämtliche Register an und geht wieder in Wartestellung, bis die Leertaste nocheinmal gedrückt wird. Selbstverständlich muß man am Anfang die Startadresse angeben, ab der das Programm abgelagert wurde. Der Debugger erkennt automatisch die Länge des Opcodes und simuliert auch die Sprungadressen.

Damit man aber nicht nur untätig zusehen muß, wenn sich ein falscher Registerwert einschleicht, kann man mit einer Funktionstaste Inhalte verändern. Damit lassen sich nun "Was wäre, wenn " Überlegungen weiterverfolgen.

Mit weiteren Funktionstasten kann man noch einen beliebigen Speicherbereich und seine Inhalte anzeigen lassen. Ebenso ist es möglich, das Interrupt-Flip Flop zu bewundern. Um am Anfang auch Maschincodeprogramme abzulegen, braucht man lediglich im geeigneten Modus die Hex-Werte über die Tastatur eingeben. Gibt es noch zwei Dinge zu sagen:

Interrupts können nicht getestet werden, weil der Simulator nicht unter Echtzeitbedingungen arbeitet, sondern wie oben erwähnt nur im BASIC programmiert ist.

Außerdem verbraucht das Programm Platz. Obwohl mit zum Teil ausgefallenen Methoden programmiert wurde, braucht der Debbugger über 7 kBytes im Speicher. Deshalb ist übrigens die Erklärung im Programm so spärlich, weil jedes Byte gespart wurde. Trotzdem kann es vorkommen, daß man just in den Speicherbereich von &H8000 bis &H9A00 etwas "reinpoken" möchte. Deshalb ist ein Zusatz in Entwicklung, der es ermöglicht, in besagten Speicherbereich zu poken.

Der Debugger fängt nämlich die Daten ab und lädt sie in das Video-RAM. Wenn aus dem Speicherbereich rausgelesen werden soll, dann fängt das Programm diesen Befehl wieder ab und läßt das MC-Programm in Wirklichkeit die Daten aus dem Video-RAM wieder rauslesen. So sieht es für das Maschincodeprogramm so aus, als ob zum Beispiel nach &H8300 gespeichert wurde, in Wirklichkeit steht das Ganze aber im Video-RAM.
Das Programm ist bei Gerhard Fally erhältlich. Auskünfte über Programm und Programmierer erteilen außerdem noch gerne die Redaktion des SVi-Journals oder das Computer-Studio in der Paniglgasse.

SPECTRON

Dieses Spiel besticht durch gute Farb- und Toneffekte. Es ist auf Kassette erhältlich. Man muß zuerst ein kurzes Hilfsprogramm in den Computer laden (mit "CLOAD" ⟨CR⟩), danach holt sich der Computer selber das Spiel rein (Nach RUN ⟨CR⟩).

Das Videospiel selber handelt, wie kann es anders sein, auf einer Raumbasis. Man hat vier bodengestützte Raumkreuzer, die in der Basis hin und her fahren. Drei davon sind in Reserve, mit jeweils einem muß man angreifende Raumschiffe (Hobbits) abschießen. Zum Schutz der Basis gibt es Schilde, die aber durch die Bomben der feindlichen Raumschiffe zerstört werden. Sind die Schilde wegbombardiert, muß der Raumkreuzer durch geschicktes Manövrieren den Bombenteppichen ausweichen.

Mit den Cursortasten des Computers oder dem Steuerknüppel der Joysticks kann man nach links oder nach rechts lenken. Die Raketen zur Vernichtung der Angreifer werden mit der Leertaste oder der Feuertaste abgefeuert. Nach jeweils drei Durchgängen bekommt man einen Zusatzraumkreuzer Verstärkung und neue Schilde.

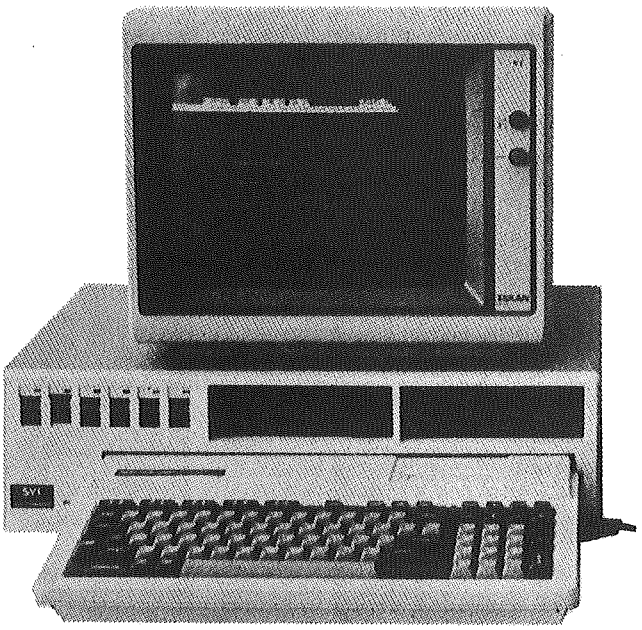
Nach dem dritten Durchgang können die Hobbits auch in die Raumbasis eindringen. Die Angreifer (dann SINKER genannt) durchqueren sie in zwei Ebenen. In der Oberen kann der Raumkreuzer die Angreifer noch abschießen, in der Unteren legen sie eine Mine in die Basis und können beim Durchqueren nicht abgeschossen werden. Sind 10 Minen gelegt, explodiert die Raumbasis und das Spiel ist aus.

Der Spieler kann den Sinkern ausweichen, indem er den Raumkreuzer mit den Cursortasten hinauf oder hinunter manövriert. Der Raumkreuzer hat nämlich auch die Möglichkeit, in beiden Ebenen herumzufahren.

Für jeden abgeschossenen Hobbit bekommt man 30 Gutpunkte, für jeden Sinker 50. Ab und zu kreuzt am oberen Rand des Bildschirms ein sogenanntes Mutterschiff auf. Wenn es abgeschossen wird, bekommt man 70 Points gut geschrieben.

Das Spiel hat 4 Schwierigkeitsgrade, die sich nur durch die Geschwindigkeit und Wendigkeit der Angreifer und deren Bomben unterscheiden. Außerdem kann man zu zweit oder alleine spielen. Spielt man zu zweit, dann wechseln sich die Spieler nach jedem Durchgang ab.

Erzeugt wurde dieses Spiel von Spectravideo und ist im Computer-Studio der Firma Wehsner erhältlich. Preis: S 170.- inkl. MWSt.



Wir sind Spezialisten für die SVI-Computer!

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

Wir haben auch immer die aktuellsten Preise:

Die Sonderaktion konnte verlängert werden, daher gilt noch kurze Zeit unser

SONDERANGEBOT

SV-328 + Datenrecorder SVI-904

+ 2 Joysticks QuickShot I

+ 4 Programmkassetten

Paketpreis nur S 7.990,-

Weiters eingetroffen:

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*, ClacStar*, DataStar*, ReportStar*).

Alle Erweiterungen und Peripheriegeräte sind prompt lieferbar.

Matrixdrucker, Typenraddrucker, Monitore.

Auch Postversand per Nachnahme.

Alle Preise inkl. MWSt.

* eingetragene Warenzeichen von MicroPro International Corporation

Jetzt auch T U R B O P A S C A L 2 . 0

angepaßt auf SVI-328 prompt lieferbar (mit Texteditor und ausführlichem Handbuch)

S 2.365,-

Computer-Studio

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

Hardcopy zu Programm SINGLE STEP

```
M O N I T O R  F U E R  Z 8 0 A
BITTE STARTADRESSE ANGEBEN:&H9000

PC= 9357   SP = F2D0

A= 23      A'= 54
F= DE      F'= D8
B= AD      B'= FE
C= 65      C'= D4
D= 25      D'= B4
E= 85      E'= F7
H= 7F      H'= 5E
L= 41      L'= 3C

IX= A125   IY = A547

          SZ H PNC   SZ H PNC
FLAGS     11011110   11011000   FLAGS'
```

```
10 A$ = "SPECTRAVIDEO"
20 FOR L=1 TO LEN (A$)
30 PRINT LEFT$ (A$,L),
40 PRINT MID$(A$,L,1);
50 PRINT SPACE$ (2+LEN(A$)-L);
60 PRINT MID$(A$,LEN(A$)+1-L,L)
70 NEXT L
```

Was bietet Ihnen der
Spectra Video Club Austria?

Regelmäßige Clubabende mit Gelegenheit zum Informationsaustausch! Kostenloses Arbeiten an SV-318 und SV-328-Systemen und Möglichkeiten zum Verwenden von Druckern!

Freier Bezug des SVi-Journals, unserer Clubzeitschrift! Verbilligte Angebote von Spectravideo-Produkten!

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

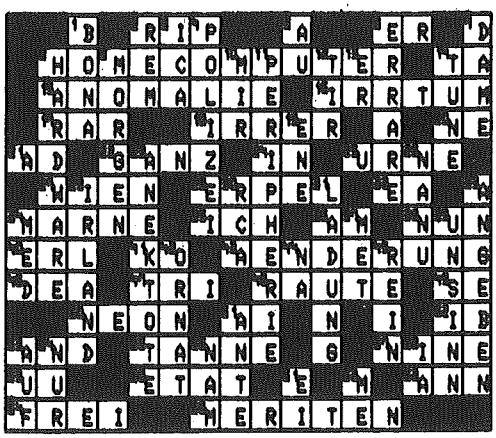
Ich interessiere mich für die Mitgliedschaft beim "Spectra Video Club Austria" und wünsche die Zusendung näherer Informationen und der Anmeldungsunterlagen.

Name .....

Adresse .....

Bitte per Post an "Spectra Video Club Austria", p.A. Computer-Studio, 1040 Wien, Paniglgasse 18-20 senden oder persönlich abgeben. Die Abgabe dieses Coupons ist unverbindlich!

Lösung des Rätsels aus Heft 2:



IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Philipp Ott, Wolfgang Rotschek, Heinz Schmid, Christian Thomas (Grafiker), Georg Wolfbauer

Medieninhaber (Verleger): Spetra Video Club Austria, p.A. Computerstudio, A-1040 Wien, Paniglg.18-20, Tel (0222) 65 88 93

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

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

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

Silbenrätsel:

amerik. Tal mit viel Computerindustrie

Zentraleinheit eines Computers

Fachgebiet der Schwachstromtechnik

Übersätzer von hohen Programmiersprachen

elektronisches Bauelement (aktiv!)

unpräziser Begriff für ALU

Programmiersprache

Bildschirmspeicher

ein Register des Z 80A

externes Speicherelement

Hochzahl in einer Potenz

elektronisches Bauelement (aktiv!)

Mit den untenstehenden Silben müssen obrigen die Wörter gebildet werden. Die Anfangsbuchstaben von oben nach unten gelesen, ergeben den Namen einer amerikanischen Computerfirma.

AS/BLER/CHEN/COM/CON/DEO/DEX/DIS/EIN/ELEK
EX/GI/HEIT/IN/KER/KET/LER/LEY/LI/NENT/NIK
ONS/OPE/PI/PO/PRO/RA/RAM/RE/RE/SEM/SI/SI
SOR/STAER/STER/STOR

P. b. b. Erscheinungsort Wien, Verlagspostamt A-1040 Wien