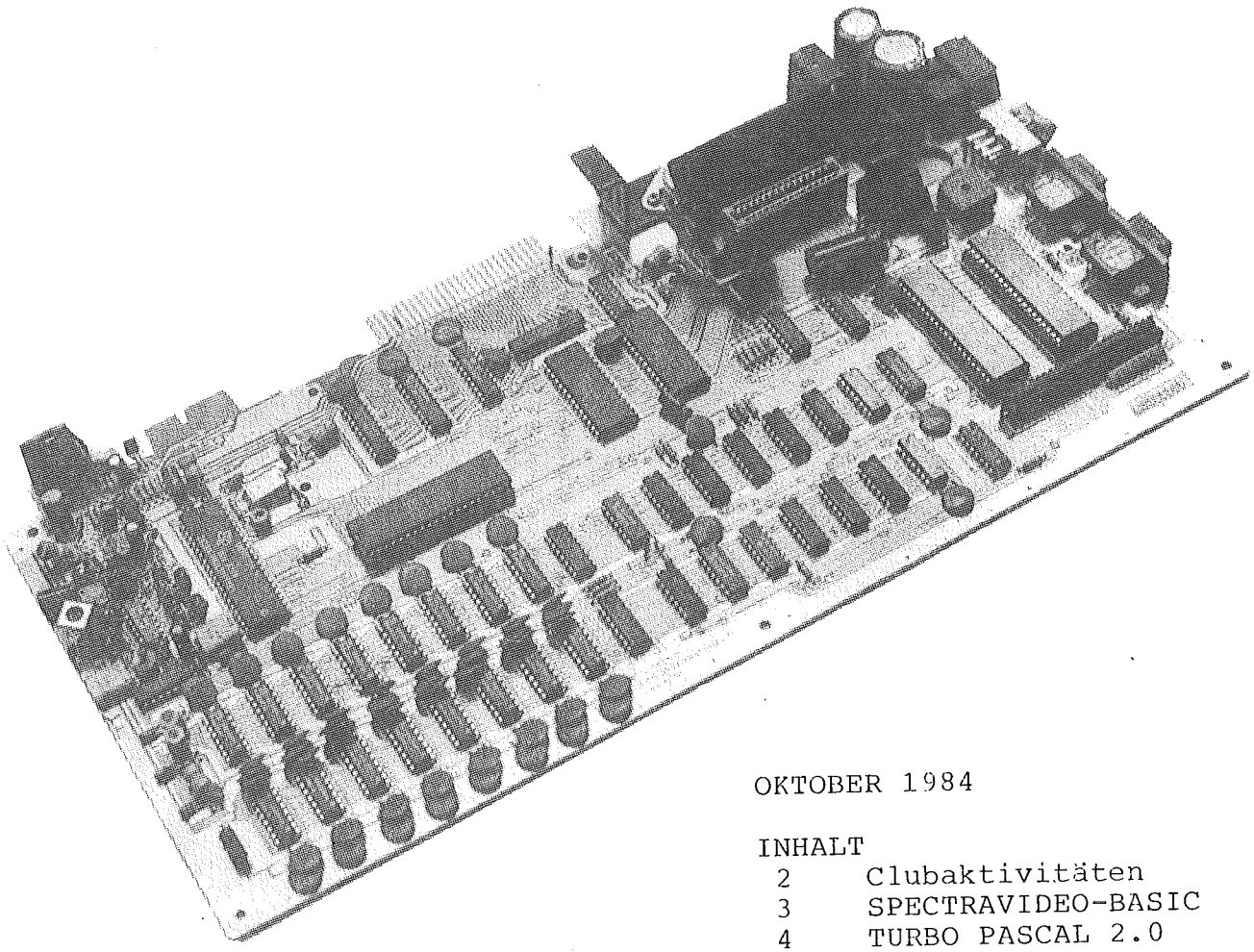
SVI JOURNAL

Die Zeitschrift des Spectra Video Club Austria



OKTOBER 1984

INHALT

Heft 4/84 **S 15.-**

Liebes Clubmitglied!

Kein SVi-Journal ohne Neuigkeiten! Nachdem wir den verkleinerten Druck eingeführt haben, wagen wir uns nun daran, Kursivdruck zu verwenden. Durch diese Schrägschrift können wir vor jeden Artikel ein Motto schreiben. Dieses Motto gibt einen kurzen Überblick über den jeweils nachfolgenden Bericht.

Nach wie vor gibt es übrigens die Möglichkeit, Vorschläge für unsere Teilnahme an der "Hobby-Elektronik"-Messe einzubringen, die vom 15.November bis zum 18.November im Wiener Messepalast stattfindet. Wie im vorigen Heft angekündigt, stellt die österreichische Zeitschrift "itm-praktiker" dem Club einen Platz auf der Messe zur Verfügung. Wir werden mit drei Computern vertreten sein. Nun brauchen wir aber die oben erwähnten Vorschläge, was wir mit den SVI's zeigen sollen. Ebenfalls brauchen wir Aufpasser für unsere Mikros und Vertreter für den Club. Vor allem am Samstag und am Sonntag erwarten wir einen großen Ansturm an Besuchern.

Einige Vorschläge bekamen wir übrigens schon! So werden wir den Messebesuchern vorführen, wie angenehm es ist, auf einem Computer mit 208 BASIC-Befehlen zu programmieren. Aus diesem Grund veranstalten wir einen Kurzprogrammierwettbewerb. Jedem Teilnehmer werden 20 Minuten Zeit gegeben. In dieser Zeit muß er ein Kurzprogramm schreiben. Das beste Ergebnis wird belohnt. Clubmitglieder dürfen nicht am Wettbewerb teilnehmen. Einerseits brauchen wir nämlich ein paar Leute, die den Teilnehmern die Befehle erklären, und andererseits soll jeder Programmierer die gleichen Chancen haben. Nicht zuletzt soll dieser Wettbewerb vor allem auch Personen, die keinen SVI-Computer besitzen zeigen, daß man in kurzer Zeit viel aus dem Computer herausholen kann.

Doch nicht nur der umfangreiche Befehlssatz wird demonstriert. Die Fähigkeiten des SVI-328 als "Musikinstrument" werden ebenfalls hergezeigt. Mit dem SPECTRAVIDEO-Spiel "MUSIC-MENTOR" (Testbericht in diesem Heft) hat man die Möglichkeit, Lieder zu komponieren und abzuspeichern. Einige schon vorbereitete Stücke werden ebenfalls abgespielt (Für Elise, Invention von Bach, Prelude von Bach usw.).

Einen Hardwarezusatz haben wir von einem Linzer Kollegen bekommen. Damit wird unser SVI sprachkundig. Ja, richtig gelesen! Man braucht lediglich einen Satz eingeben und der SVI-328 spricht ihn aus.

Nicht zuletzt werden auch Grafikeffekte gezeigt, kurzum, die ganze Palette der Vorteile des Spectravideo-Systems.

Im Übrigen danken wir für die überwältigende Teilnahme an der neuen Rubrik "Programmecke". Wir haben bis jetzt sage und schreibe 1 (in Worten: "Ein") Programm aus der Reihe unserer Leser bekommen. Sollte hier vielleicht ein Mißverständnis aufgekommen sein? Diese Ecke hilft den Clubmitgliedern, ihre Programme publik zu machen, also durchaus ein Vorteil. In dieser Rubrik werden keine Programmiergeheimnisse verraten, außer wie dieses Programm bedient wird und wer es gemacht hat (und das soll ja im Interesse des Entwicklers kein Geheimnis sein!).

Niemand braucht sich wegen irgendwelcher fehlender literarischer Eigenschaften Sorgen zu machen. Die Programmecke soll nicht mit kulturell hochstehenden Novellen oder spannenden Kurzgeschichten ausgefüllt sein, sondern dient, wie das gesamte SVi-Journal auch, ausschließlich dem Informationsaustausch der einzelnen Clubmitglieder. Und sollte jemand wirklich ganz und gar nicht schreiben wollen, dann braucht er sich nur an einen Mitarbeiter des SVi-Journals zu wenden. Wir packen das Ganze in eine nette Form und fertig ist der Artikel. Dank gebührt hier unseren Linzer Kollegen, die uns das oben erwähnte einzige Programm geliefert haben.

Etwas erfolgreicher als unsere Rubrik war der erste "außerordentliche" Clubabend, der über das Thema "Floppies" stattgefunden hat. Es kamen ungefähr zehn Leute. Behandelt wurde die breite Palette von Diskettenanwendungen. In den zweieinhalb Stunden wurde genauso über das richtige Formatieren wie über das Anlegen von Sicherheitskopien gesprochen. Wie ja viele wissen werden, kann man am Spectravideo sowohl Disk-BASIC als auch das Betriebssystem CP/M auf Disketten laufen lassen. Deshalb kam auch die Anwendung dieser beiden Entwicklungen zur Sprache. Bei dieser Gelegenheit wurde auch gleich ein neue Beilage für Diskettenlaufwerke vorgestellt, in der alle wichtigen Daten und Fakten über Disketten stehen.

Ebenfalls vorgestellt wurde das Programm "DSKED", mit dem man die Inhalte von Disketten untersuchen kann. Dieses Programm läuft unter CP/M und gewährt entweder sektoren- oder spurenweise Einblick in den Inhalt von Disketten. Der nächste außerordentliche Clubabend wird übrigens wie schon im vorigen Heft angekündigt am 7. November durchgeführt.

An diesem Abend werden wir versuchen, entgültig die altbekannte Frage zu klären, wann man sich warum welche Computerperipherie kaufen soll! Es ist nämlich gar nicht so einfach, auf diese an sich banal klingende Frage eine richtige vollständige Antwort zu finden. Trotzdem hoffen wir auf genauso großen Andrang wie auf den ersten Clubabend. Wissen Sie denn ganz genau, was Sie sich noch zu Ihrem Spectravideo kaufen wollen?

Ihr SVi-Journal Chefredakteur Gerhard Fally!

```
*******************************************
*                                         *
*  Die nächsten Clubabende:               *
*                                         *
*   Mi, dem 24. Oktober   1984, ab 19 Uhr *
*   Sa, dem 10. November  1984, ab 17 Uhr *
*   Mi, dem 21. November  1984, ab 19 Uhr *
*                                         *
*   Für Dezember gibt es eine Sonderrege- *
*   lung:                                 *
*                                         *
*   Sa, dem  1. Dezember  1984, ab 17 Uhr *
*   Mi, dem 12. Dezember  1984, ab 19 Uhr *
*                                         *
*  wie  immer  im Computer-Studio,        *
*  1040  Wien,  Paniglgasse  18-20.       *
*  Nichtmitglieder sind willkommen.       *
*  Ende jeweils  ca. 22 Uhr!              *
*                                         *
*  Aktivitäten  an  den  Clubabenden:     *
*  Arbeiten an Spectravideo-Systemen,     *
*  Informationsaustausch zwischen Club-   *
*  mitgliedern.                           *
*                                         *
*   Im November gibt  es zusätzlich  zu   *
*  den Clubabenden einen Vortrag  über    *
*  Spectravideo-Computer,  deren Anwen-   *
*  dung und deren Peripherie!             *
*                                         *
*  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                      *
*                                         *
*******************************************
```

Spectravideo-BASIC

In der vierten Folge unserer Serie wechseln wir wieder von den Funktionen zu den Befehlen über. Als erstes besprechen wir nun sämtliche Kommandos. Dabei werden wir uns allerdings, wie schon bei den Funktionen, an die Reihenfolge des User-Manuals halten. Der einzige Unterschied besteht darin, daß wir nun keinen Befehl auslassen!

Angefangen wird mit dem "AUTO"-Befehl. Wenn man einen gleichmäßigen Zeilenabstand haben will und zu faul ist, jede Zeilennummer einzeln einzugeben, dann wird man dieses Kommando schätzen. Mit "AUTO" berechnet der Computer automatisch nach jedem <ENTER> die nächste Zeilennummer. Ist diese Nummer schon verwendet worden, dann malt der Interpreter einen Stern hinter die Zeilennummer. Mit "CTRL-STOP" wird der Zyklus unterbrochen.

Direkt hinter "AUTO" schreibt man die als erstes gewünschte Zahl, danach kommt der Zeilenabstand. Läßt man den ersten Parameter weg, dann fängt der Computer ab 10 an, läßt man den zweiten weg, dann ist der Abstand jeweils 10. Steht nur "AUTO" ohne Anhängsel da, so werden beide Effekte kombiniert. "AUTO" generiert die Zeilen 10,20,30,40,... und "AUTO 50,3" erzeugt 50,53,56,59,.... Mit "AUTO ,17" spuckt der Computer 10,27,44,61,... aus.

Erscheint, wie oben erwähnt, ein Stern hinter der Nummer, dann wurde in diese Zeile schon etwas eingegeben. Drückt man nun <ENTER>, dann bleibt der vorher eingegebene Inhalt erhalten, ansonsten wird der neu eingegebene übernommen.

Eine weitere nützliche Anweisung ist "CONT". Sollte durch "CTRL-STOP", "END" oder "STOP" ein Programm unterbrochen worden sein, dann setzt der Computer mit "CONT" seine Programmabarbeitung fort. Ist eine Programmzeile inzwischen geändert, dann schnauzt der Interpreter den Bediener mit "Can't continue" an, was übersetzt soviel wie "Kann nicht fortsetzen" lautet.

Möchte man einige Zeilen löschen, dann ist man mit "DELETE" gut bedient. Schreibt man nur eine Zahl hinter die Anweisung, dann wird nur diese Zeile gelöscht (entspricht dem: "Zeilennummer eingeben und <ENTER> drücken"). Eine Spanne zwischen zwei Zeilennummern (zum Beispiel 40-80) löscht alle Zeilen einschließlich der angegebenen. Möchte man vom Programmanfang weg Zeilen löschen, dann schreibt man "-Zeilennummer". "DELETE.-<Zeilennummer>" setzt für den Punkt die zuletzt bearbeitete Zeile ein. Leider kann man kein "<Zeilennummer>-" schreiben. (also zum Beispiel "DELETE 40-" ist nicht erlaubt). Ebenso meldet sich der Computer unwirsch, wenn ihm eine Zeilennummer eingegeben wurde, die nicht vorhanden ist.

Eine der einfachsten Anweisungen ist "END". Hier gibt es keine Zahlen anzuschließen. Ja es ist nicht einmal möglich, durch falsche Handhabung irgendwelche Fehlermeldungen zu bekommen. Dieser Befehl schließt alle Dateien und kommt in den Kommando-Modus des BASIC zurück. Wie oben erwähnt, kann man dann noch mit "CONT", sofern keine Programmänderungen vorgenommen wurden, weitermachen. Zu beachten ist aber, daß alle Dateien geschlossen sind, wenn mit "CONT" fortgesetzt wird (weil "END" alle zugemacht hat). Am Ende eines Programms braucht man kein "END", der Computer weiß auch ohne diesen Befehl, daß es nicht mehr weitergeht.

Ein etwas umfangreicherer Befehl ist "LIST". Wie der Name schon sagt, kann man mit "LIST" Programme "listen", das heißt auflisten. Ohne irgendwelche Anhängsel am "LIST" sieht der Betrachter das ganze Programm von Anfang bis zum Ende durchlaufen. Wenn man einen Teil näher betrachten will, kann man mit der "STOP" Taste das Programm am Bildschirm verharren lassen. Ein zweites "STOP" läßt das Listing weiter ablaufen. "CTRL-STOP" unterbricht den Ablauf vollständig. Nur nach einem "CTRL-STOP" kann man übrigens mit dem Cursor auf den Bildschirm herumfahren und Ausbesserungen vornehmen, nach einem einfachen "STOP" während des Listings funktioniert das nicht.

Sollte ein sehr langes Programm gelistet werden und man möchte Zeit sparen, dann kann man, während das Listing läuft, den nächsten Befehl eingeben. Gleich nach dem Ende des Programmes wird dann dieser Befehl ausgeführt. Natürlich kann es auch sein, daß man einen bestimmten Teil des Programmes sehen will. Hier ergeben sich folgende Möglichkeiten:

Nur eine Zahl listet die Zeile mit der Zahl als Nummer auf (zum Beispiel "LIST50" listet Zeile 50). Setzt man hinter die Zahl einen Beistrich, so wird ab der Zeile alles bis zum Ende hergezeigt (zum Beispiel "LIST50-" zeigt ab Zeile 50 bis zur letzten alles). Kommt der Strich vor die Zahl, was auch möglich ist, dann zeigt der Interpreter den Anfang des Programms bis zur definierten Zeile (zum Beispiel LIST -50). Die kombinierte Form (zum Beispiel "LIST 50-100") läßt den Ausschnitt Zeile 50-100 am Bildschirm erscheinen.

Die Zeilennummern können durch einen Punkt ersetzt werden. Dann setzt der Computer die zuletzt vom Bediener bearbeitete Zeile ein (die Zeile, wo das letzte <ENTER> gedrückt wurde). Ist der Computer während des Programmablaufes auf einen Fehler gestossen, so nimmt er diese Zeile, in welcher der Fehler auftrat, in sein Gedächtnis. Nach einer Fehlermeldung wird also immer mit "LIST." die Zeile aufgelistet, in welcher sich der Fehler befindet.

Die gleiche Syntax (=Grammatik) wie "LIST" hat auch "LLIST", mit dem einzigen Unterschied, daß "LLIST" den Drucker anspricht. Der Computer nimmt bei dieser Anweisung an, daß mit einem Drucker mit höchstens 132 Zeichen pro Zeile gearbeitet wird.

Wer kennt nicht das folgende Problem? Man hat ein wunderbares Programm geschrieben, und danach wurde ein kleiner Zusatz außerhalb dieses Programmes entwickelt. Doch wie verbindet man jetzt, die "Mutter-" mit der "Tochterentwicklung"? Die einfachste Methode ist, den kleinen Zusatz, einfach händisch zu übertragen. Bei 2KBytes Länge wird das aber ein mühseliges Unterfangen. Die zweite Methode wäre, mit einigen "PEEKS" und "POKES" im Programmspeicher und im Video-RAM so herumzuwüten, daß man den zweiten Teil der Entwicklung als Bild in den Bildschirmspeicher lädt. Danach holt man sich die Bytes in den Speicher und stellt obendrein noch einige BASIC-Zeiger richtig ein. Wie das genau funktioniert, sei hier nicht verraten, weil so eine Beschreibung den Rahmen dieser Serie sprengen würde.

Zum Glück gibt es "MERGE". Dieser Befehl nimmt dem Bediener alle Schwierigkeiten mit dem Einstellen von BASIC-Zeigern ab. Man braucht lediglich mit "SAVE<Dateiname>,A" die Entwicklung auf Kassette abzuspeichern (oder auf Diskette). Danach holt man sich das Mutterprogramm in den Speicher und tippt "MERGE<Dateiname>" ein. Nun braucht man nichts weiter zu machen. Automatisch mischt

Will man von seinem Programm nichts mehr wissen, so vernichtet BASIC auf Wunsch mit "NEW" das ganze Programm samt Variablen. Alle Files werden geschlossen. Nebenbei sei bemerkt, falls ein Programm unabsichtlich gelöscht wurde, daß man mit einigen gezielten "POKES" das Programm wieder herzaubern kann. Denn "NEW" vernichtet nämlich nur den Programmzeiger auf die erste Zeile. Ist der erst wieder gerichtet, dann ist das Programm wieder gerettet. Doch wie das genau vor sich geht, wird in einem gesonderten Artikel besprochen werden.

Eine weitere wichtige Anweisung ist "RENUM". Mit diesem Befehl werden Programmzeilen umnummeriert. Wenn "RENUM" alleine eingegeben wird, dann wird das ganze Programm umnummeriert, und zwar mit der ersten Zeile als 10 und jeweils in Zehnerabständen. Will man nicht von Anfang an umnummerieren, dann muß "RENUM,<Zahl>" eingegeben werden. Ab der angegebenen Nummer wird dann das Programm in 10er Abständen nummeriert. Der Abstand ändert sich durch eine Zahl hinter der angefügten Nummer (zum Beispiel "RENUM,40,30).

Selbstverständlich kann man auch angeben, welche Zeilennummer man als erste haben will. Mit "RENUM<Zahl>" wird diese Möglichkeit erfüllt. Fehlermeldungen kann es auch geben. Werden Programmzeilen erzeugt, die größer als 65529 sind, oder wenn sich durch ein geschicktes "RENUM" die Zeilenabfolge ändern würde, dann streikt der Computer mit "Illegal Function Call". Ein "Undefined Line" spukt der Interpreter aus, wenn eine Zeilennummer hinter "GOTO", "GOSUB" und Ähnlichem steht, die nicht existiert. Der genaue Wortlaut dieser Meldung: "Undefined Line xxxxx in yyyyy".

Einer der wichtigsten Befehle ist "RUN". Dieses Kommando startet das im Speicher befindliche Programm. Will man erst ab einer bestimmten Zeile mit den Laufenlassen beginnen, dann stellt man hinter das Wort "RUN" eine Zahl (zum Beispiel "RUN 40" startet das Programm ab Zeile 40). Bei "RUN" werden alle Variablen auf Null gesetzt und alle Files geschlossen. Will man einen Wert vom Direktcode für den Programmablauf erhalten, dann kann man auch mit "GOTO" oder "GOSUB" in ein Programm einsteigen (Mit "CONT" geht es nur dann, wenn keine Zeile verändert wurde!). Diese beiden Befehle zerstören keine Variablen.

Im Gegensatz zu "END" zerstört "STOP" keine Variablen oder Files. Alles bleibt so, wie es der Computer gerade verwendet hat. Außerdem wird eine Meldung "BREAK in line xxxxx" ausgegeben. Der Interpreter wartet nach "STOP" im Direktmodus auf weitere Anweisungen.

In der nächsten Folge setzen wir dann mit den "normalen" BASIC-Anweisungen fort, die im User-Manual mit 2.2 nummeriert sind.

Turbo Pascal 2.0

Der TURBO-Pascal-Compiler hat seinen Namen zu recht: In unglaublich kurzer Zeit ist ein Programm kompiliert. TURBO-Pascal 2.0 bietet auch noch andere Features zu einem besonders günstigen Preis. Jetzt ist es auch für den SVI-328 lieferbar.

Als ich die Diskette mit der Aufschrift TURBO-Pascal erhielt, begleitet von einem ca. 280 Seiten starken Reference Manual (in englischer Sprache), ging ich mit gemischten Gefühlen ans Werk. Ich hegte schon immer eine Aversion gegen Compiler, die mühselige Prozedur des Editierens vor Augen (CP/M ED oder ein anderer Editor) und die zeitraubende Fehlersuche.

Ich war angenehm überrascht, bereits auf den ersten Seiten des Handbuchs die ausführliche Beschreibung eines implementierten Editors vorzufinden. Der Editor ist für alle, die WordStar kennen, ganz leicht zu bedienen. Die Steuerzeichen sind dieselben wie bei WordStar. Darüber hinaus gibt es aber noch einige zusätzliche Annehmlichkeiten. So springt der Cursor beispielsweise unter das erste Zeichen der vorhergehenden Zeile, was die zeitraubende Eingabe von Leerzeichen entfallen läßt. Ebenso ist es auch möglich, den Cursor über leere Textstellen zu bewegen, eine Möglichkeit, die einem beim WordStar manchmal abgeht. In Verbindung mit dem Programm TLIST.COM, welches sich ebenfalls auf der Pascal-Diskette befindet, verfügt man praktisch auch über eine Textverarbeitung (das Programm druckt die Source-Files und auch beliebige Texte aus).

Sie können die Steuerzeichen aber auch ändern und mittels des mitgelieferten Install-Programms auf Ihre gewohnte Textverarbeitung anpassen.

Mit der Vorfreude auf den implementierten Texteditor ausgestattet, nahm ich nun die "TURBO-Pascal 2.0"-Diskette in Betrieb. Und ich wurde nicht enttäuscht. Nachdem ich meinem Source-File einen Namen gegeben hatte, meldete sich der Texteditor. Turbo Pascal wird, wie bereits erwähnt, mit einem Install-Programm geliefert. Eine Installierung war jedoch nicht mehr erforderlich, da die mir gelieferte Version bereits auf den SVI-Computer angepaßt war.

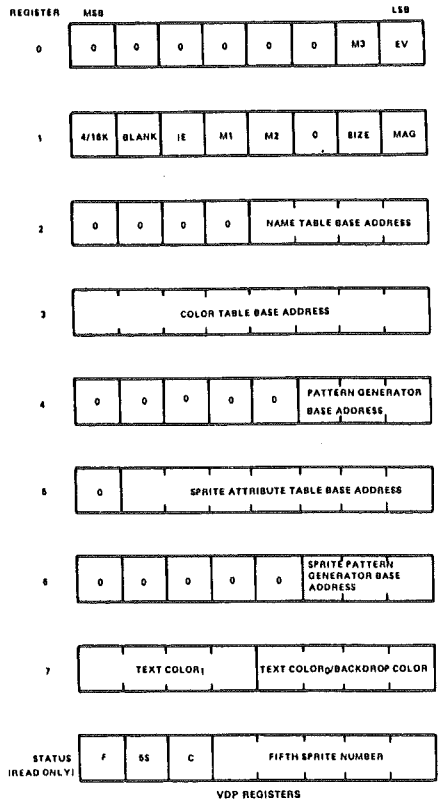
Gleich am Beginn meiner Arbeit mit TURBO-Pascal entdeckte ich eine weitere Annehmlichkeit. Im Deklarationsteil muß man sich an keine bestimmte Reihenfolge wie bei anderen Pascal-Versionen zu halten, weiters kann man jeden Teil beliebig oft aufrufen. Wenn nun das Programm fertig eingegeben ist, kehrt man mit CTRL-K-D zum Compiler zurück. Man kann das Programm sowohl im Speicher als auch auf die Diskette compilieren.

Zur allfälligen Fehlersuche wird man zuerst nur im Speicher compilieren. Compiliert wird in einem Durchgang in verblüffend kurzer Zeit. Ein 245 Zeilen langer Source-File wurde in weniger als 7 Sekunden fertig compiliert, wobei der Maschinencode immerhin rund 2500 Byte umfaßte. TURBO-Pascal erzeugt übrigens optimierten Z80-Maschinencode und Assembler-Routinen können direkt im Source-File eingefügt werden.

Wird während des Compilierens ein Fehler gefunden, wird die Fehlernummer angezeigt und (auf Wunsch) auch die Fehlermeldung ausgegeben, beispielsweise " ';' expected ". Nach Drücken der ESCAPE-Taste wird der Editor aufgerufen und der Cursor steht genau an

Mit dem Wissen über die Anordnung der Sprite-Parameter ausgestattet, ist es möglich, die Geschwindigkeit der Sprites zu beschleunigen. Man kann dies durch "POKEn" der X- und Y-Koordinaten in die Parameterliste im Video-RAM erreichen (also mit VPOKE). Wenn dies noch immer zu langsam sein sollte, kann man die Koordinaten im Maschinencode eintragen.

Wir wenden uns nun den Registern des Video-Chips zu. Er verfügt über 9 Register. Die folgende Darstellung gibt einen Überblick über die Register und die Bedeutung der Bits der Register.



Und nun zu den Registern im einzelnen:

Register 0

Dieses Register enthält nur zwei Kontroll-Bits:

Bit 6 M3 (Mode Bit 3): Erläuterung bei Bit 3 und 4 von Register 1
Bit 7 Zuständig für externes Video-Signal

Alle anderen Bits sind für zukünftige Verwendung reserviert.

Register 1 enthält 8 Kontroll-Bits:

Bit 0 beim SVI-Computer immer 1
Bit 1 wird dieses Bit auf 0 gesetzt, so wird der aktive Bildschirm gelöscht
Bit 2 IE (Interrupt Enable)
0 = Disable VDP-Interrupt
1 = Enable VDP-Interrupt
Bit 3 M1 (Mode Bit 1)
Bit 4 M2 (Mode Bit 2)

Die Mode Bits bestimmen die Darstellungsart und kommen wie folgt zum Einsatz:

M1	M2	M3	
0	0	0	Grafikmode I (vom BASIC nicht realisiert)
0	0	1	Grafikmode II (das ist SCREEN 1 beim SVI-Computer
0	1	0	Multicolormode (im BASIC SCREEN 2)
1	0	0	Textmode (SCREEN0)

Bit 5 ist für zukünftige Verwendung reserviert

Bit 6 zuständig für die Sprite-Größe:
0 bedeutet Größe 1 (8 x 8 Bit)
1 bedeutet Größe 2 (16 x 16 Bit)

Bit 7 bestimmt die Vergrößerungsoption für die Sprites:
0 bedeutet Darstellung ohne Vergrößerung
1 bedeutet Sprite-Darstellung in doppelter Größe

Turbo Pascal 2.0
Fortsetzung von Seite 4

der Stelle des gefundenen Fehlers (beziehungsweise bei dem dem Fehler folgenden Zeichen). Mehr Komfort kann sich ein Programmierer wohl kaum wünschen.

Sind alle Fehler beseitigt, wird man den compilierten File auf der Diskette anlegen. Das Compilieren dauert nur geringfügig länger als beim Compilieren im Speicher. Der solcherart erstellte COM-File ist natürlich selbständig lauffähig (auch auf einer Diskette ohne Pascal). Auch rechtlich ist es nunmehr möglich, unter TURBO-Pascal erstellte Programme ohne zusätzliche Gebühren zu verkaufen.

Dieses Wunderding von Compiler verbraucht nur wenig Speicherplatz: inklusive Editor nur 28 KByte. Sollte der Programmierer dennoch zu wenig Speicherplatz vorfinden, kann er zu große Programme in Overlays aufsplitten, die jeweils bei Bedarf von der Diskette aufgerufen werden. Dies ist eine Eigenschaft, die erst bei der Version 2.0 zu finden ist, ebenso die DISPOSE-Procedure, die eine gezielte Freigabe des Speicherplatzes von nicht mehr benötigten dynamischen Variablen ermöglicht.

Gegenüber anderen Pascal-Arten hat Turbo-Pascal die Einschränkung, daß ein Sprung aus Blöcken heraus nicht möglich ist. Dies stellt aber nicht unbedingt einen Nachteil dar. Dafür können aber Labels richtige Namen sein (nicht nur Ziffern).

Auf der Turbo-Pascal-Diskette befindet sich auch noch ein Programmbeispiel mc.pas. Es ist ein Tabellenkalkulationsprogramm (ein einfaches, kein CalcStar oder Multiplan). Es gibt aber jedenfalls einen lehrreichen Einblick in die Pascal-Programmiertechnik. Der Anfänger kann sich ein erstes Erfolgserlebnis durch Compilieren dieses Source-Files verschaffen.

Für alle, die gerne strukturiert programmieren möchten oder denen BASIC-Programme zu langsam sind, stellt Turbo Pascal 2.0 eine preiswerte Alternative zum BASIC-Interpreter dar. Die SVI-angepaßte Version ist in Österreich schon ab S 2.240,- zu erhalten (samt Handbuch).

Register 2
enthält die Basisadresse für die Namenstabelle. Die 4 Bits enthalten eine Zahl zwischen 0 und 15 und stellen die oberen 4 Bits der Basisadresse dar. Im SVI-Computer beginnt diese Tabelle im Video-RAM bei Adresse 0000H.

Register 3
definiert die Basisadresse für die Farbtabelle. Das Register enthält die oberen 8 Bit der 14 Bit-Basisadresse und kann also einen Wert zwischen 0 und 255 annehmen. Beim SVI-Computer beginnt die Farbtabelle bei 2000H, d.h. dieses Register enthält den Wert 80H.

Register 4
definiert die Basisadresse des Patterngenerators (Tabelle beginnt in unserem Fall bei 1800H = Wert des Registers 4 mal 800H).

Register 5
bestimmt die Basis der Sprite-Attributliste (Parameter). Das Register kann einen Wert zwischen 0 und 127 enthalten. Der Inhalt dieses Registers stellt die 7 höherwertigen Bits der Basisadresse dar. Die Adresse selbst ist dann dieser Wert mal 80H. Beim SVI liegt die Basisadresse bei 1B00H.

Register 6
zeigt auf die Basisadresse der Sprite-Muster-Tabelle. Diese liegt bei 3800H im Video-RAM und ergibt sich aus dem Inhalt dieses Registers mal 800H (3 höherwertige Bits).

Register 7
enthält Zeichenfarbe im Textmodus bzw. die Hintergrundfarbe in allen Darstellungsarten.

Bitte beachten Sie, daß es sich bei diesen acht Registern um Write only-Register handelt, das heißt, man kann nur in diese Register schreiben, den Inhalt jedoch nicht auslesen.

Umgekehrt ist es beim 9. und letzten Register des Video-Chips. Dieses ist das Statusregister, ein Read only-Register. Man kann also dieses Register auslesen, jedoch nicht hineinschreiben.

Wie aus der Darstellung ersichtlich enthält dieses Register drei Flaggen: das Interrupt Flag (F) wird auf 0 zurückgesetzt, wenn das Statusregister ausgelesen wurde. Das Coincidence Flag (C) wird gesetzt, wenn zwei oder mehrere Sprites zusammenstoßen (vom BASIC her mit ON SPRITE abgefragt). Nach jeder Abfrage wird die Flagge wieder zurückgesetzt.

Das sogenannte Fifth Sprite Flag (5S) wird gesetzt, wenn sich fünf Sprites auf einer horizontalen Linie befinden. Wie bereits erwähnt, wird in diesem Fall ein Sprite transparent. Diese Tatsache wird in dieser Flagge festgehalten und die Nummer dieses unsichtbaren Sprites bis zur Wiederverwendung in der Nummer des fünften Sprites im selben Register festgehalten.

Der TMS 9918(9929) verfügt über sechzehn Farben, die im SCREEN 1 gleichzeitig darstellbar sind. Es ist jedoch eine Einschränkung zu beachten, die wir später beschreiben werden. Vorerst jedoch eine Zusammenfassung der verfügbaren Farben und der zugehörigen Farbcodes, die auch im BASIC für die Darstellung der Farben verwendet werden.

0	transparent	8	mittelrot
1	schwarz	9	hellrot
2	mittelgrün	10	dunkelgelb
3	hellgrün	11	hellgelb
4	dunkelblau	12	dunkelgrün
5	hellblau	13	magenta
6	dunkelrot	14	grau
7	cyan	15	weiß

Der Aufbau des Bildschirms beginnt in der linken oberen Ecke. Es werden acht Bytes (acht Muster von je acht Bildpunkten) untereinander dargestellt. Dann wird rechts davon fortgesetzt und so fort insgesamt 32 mal. Der Vorgang wiederholt sich nun für weitere 23 Zeilen (insgesamt 24 Zeilen) bis zur rechten unteren Ecke des Bildes, wie die folgende Darstellung zeigt. Jede Zahl steht dabei für ein Byte (acht Bildpunkte). Die Nummer entspricht der fortlaufenden Nummerierung der Bytes im Video-RAM. Die Position entspricht der Position am Bildschirm.

Reihe 0	0	8	16	24	-	-	-	248
	1	9	17	25	-	-	-	249
	2	10	18	26	-	-	-	250
	3	11	19	27	-	-	-	251
	4	12	20	28	-	-	-	252
	5	13	21	29	-	-	-	253
	6	14	22	30	-	-	-	254
	7	15	23	31	-	-	-	255
Reihe 1	256	264	-	-	-	-	-	504
	257	-	-	-	-	-	-	505
	258	-	-	-	-	-	-	506
	259	-	-	-	-	-	-	507
	260	-	-	-	-	-	-	508
	261	-	-	-	-	-	-	509
	262	-	-	-	-	-	-	510
	263	-	-	-	-	-	-	511
Reihe 2	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
-------	-	-	-	-	-	-	-	-
-------	-	-	-	-	-	-	-	-
Reihe 23	5888	-	-	-	-	-	-	6136
	5889	-	-	-	-	-	-	6137
	5890	-	-	-	-	-	-	6138
	5891	-	-	-	-	-	-	6139
	5892	-	-	-	-	-	-	6140
	5893	-	-	-	-	-	-	6141
	5894	-	-	-	-	-	-	6142
	5895	-	-	-	-	-	-	6143

Man sieht aus dieser Übersicht, daß ein Bildschirminhalt im Grafikmodus aus 6144 Byte, das sind 6 KByte besteht. In Bildpunkten ausgedrückt sind das 6144 mal 8 (49152) darstellbare Bildpunkte. Die Auflösung unseres Bildschirms beträgt 256 Bildpunkte horizontal und 192 Bildpunkte vertikal, was wieder eine Gesamtzahl von 49152 Punkten entspricht. Diese Punkte werden übrigens auch Pixels genannt.

In jedem Byte können die acht Punkte zwei Zustände annehmen: gesetzt oder nicht gesetzt, 1 oder 0. Gesetzte Punkte nehmen die Farbe des Vordergrundes (die Zeichenfarbe) an, nicht gesetzte Punkte sind durchsichtig, man sieht dort also die Hintergrundfarbe. Jedem der oben dargestellten Bytes entspricht auch ein Byte in der Farbtabelle. Diese ist ebenfalls 6144 Byte lang und beginnt, wie schon erwähnt, bei der Adresse 2000H im Video-RAM.

In jedem dieser Farbbytes enthalten die vier höherwertigen Bits den Farbcode für die Zeichenfarbe, die niederwertigen vier Bits den Farbcode für die Hintergrundfarbe. Die Farbcodes entsprechen der vorstehenden Farbtabelle. Daraus ergibt sich auch die erwähnte Einschränkung für die Farbdarstellung im SCREEN 1. Da ein Byte nur zwei Farben darstellen kann, kann also auch ein Byte am Bildschirm (= eine Reihe von acht Bildpunkten entsprechend obiger Bildschirmorganisation) nur zwei Farben beinhalten. Versucht man eine dritte Farbe einzutragen, wird die vorhergehende Farbe überschrieben. Bei einiger Sorgfalt des Programmierers, kann man aber trotzdem schöne Grafiken auf den Bildschirm zaubern.

Bitte beachten Sie auch den Beitrag über den Hardcopy-Ausdruck eines Bildschirms auf Seite in diesem Heft. Bei der Erstellung dieses Programms war die Organisation des Bildaufbaus entsprechend zu berücksichtigen.

Z 80 - Programmieren in Assembler

Assemblerfortsetzungskurs Nr.: 4

Nachdem in der letzten Ausgabe alle wichtigen Ladebefehle besprochen wurden, wollen wir uns diesmal den arithmetischen Befehlen zuwenden. Leider besitzt der Z-80 Mikroprozessor wie alle anderen 8-Bit- Prozessoren nur Additions- und Subtraktionsbefehle, und keine Multiplikations- beziehungsweise Divisionsbefehle.

Zuerst sollen die Additionsmöglichkeiten am Z-80 Mikroprozessor besprochen werden. Der Additionsbefehl wird in der mnemotechnischen Darstellung mit "ADD" abgekürzt. Mit dem "ADD"-Befehl kann nur zum Akkumulator ein anderes 8-Bit-Register oder Byte addiert werden (ausgenommen sind die 16-Bit- Register). Genauso wie bei den Ladebefehlen muß zuerst das Zielregister und dann das Quellregister angegeben werden. Folgendes Programm addiert die Bytes aus den Speicherzellen mit der Adresse C000 und C001.

```
E000    LD   A,(C000)  ; Lade 1.Byte
E003    LD   B,A       ; nach B
E004    LD   A,(C001)  ; Lade 2.Byte
E007    ADD  A,B       ; Addiere 1.
                         Byte zum 2.
                         Byte
E008    LD   (C002),A  ; Speichere
                         Ergebnis
                         nach C002
```

Der oben angeführte "ADD"-Befehl wird wie folgt gesprochen: "Addiere zum Akkumulator B". Dabei werden die Werte, die im Akkumulator und in B stehen, addiert. Das Produkt wird im Akkumulator abgelegt. Dabei bleibt der Inhalt von B unverändert. Um zwei 16-Bit-Zahlen zu addieren, muß man logischerweise zwei 8-Bit- Additionen durchführen.

Es sind also zuerst die beiden Low- und dann die beiden Highbytes zu addieren. Wenn nun aber bei der Addition der beiden Lowbytes der 16-Bit-Zahlen ein Übertrag auftritt, das heißt, wenn die Summe größer als 255 ist, so muß zu der Summe der beiden Highbytes 1 dazuaddiert werden, um das Ergebnis richtigzustellen. Wenn nun so ein Übertrag auftritt, wird eines der Flags vom Flagregister beansprucht.

Wie schon früher erwähnt wurde, zeigen die einzelnen Flags (=Bit, das einen Zustand signalisiert und den Wert 0 oder 1 annehmen kann), welche im Flagregister abgelegt sind, Zustände oder Ergebnisse an, die bei der letzten Rechenoperation aufgetreten sind. Eines dieser Flags ist das "Carryflag (C)", welches anzeigt, ob bei der vorhergehenden arithmetischen Operation ein Übertrag aufgetreten ist. Da man dieses Carryflag oft braucht, gibt es bei der Addition einen eigenen Befehl. Das ist der "ADC"-Befehl.

Er ist die Abkürzung für "Addiere mit Carry". Dieser Befehl ist genauso zu handhaben wie der "ADD"-Befehl. Mit ihm wird zum Akkumulator das angegebene Register addiert. Doch es wird zusätzlich noch der Wert des Carryflags zum Ergebnis dazugezählt. Das Carryflag kann natürlich nur zwei Zustände haben. Es wird unter anderem auch durch den "ADD"-Befehl beeinflußt. Ist beim letzten "ADD"-Kommando ein Übertrag aufgetreten, das heißt, ist die Summe der beiden addierten Zahlen größer als 255 (Dezimal) gewesen, so wird das Carryflag automatisch auf Eins gesetzt. Ist kein Übertrag aufgetreten, so bekommt es den Wert Null.

Zum besseren Verständnis wird gleich ein Programm zur Addition zweier 16-Bit-Zahlen beschrieben. Die beiden Zahlen sollen in den Speicherzellen mit den Adressen C000, C001 und C002, C003 abgelegt sein. Da es 16-Bit-Zahlen sind, wird natürlich zuerst das Lowbyte und dann das Highbyte abgelegt. Das Lowbyte des ersten zu addierenden Zahl liegt bei C000 und das Highbyte bei C001 und das Lowbyte der zweiten Zahl liegt bei C002 und das Highbyte bei C003. Das Lowbyte des 16-Bit-Ergebnisses wird bei C004 und das Highbyte bei C005 abgelegt.

```
LD    A,(C000)  ; Lade das Lowbyte
LD    B,A       ; der ersten Zahl
                  nach B
LD    A,(C001)  ; Lade das Lowbyte
                  der zweiten Zahl
                  in den Akkumulator
ADD   A,B       ; Addiere die beiden
                  Lowbytes
LD    (C004),A  ; Speichere Lowbyte
                  nach C004
LD    A,(C001)  ; Lade das Highbyte
LD    B,A       ; der ersten  Zahl
                  nach B
LD    A,(C003)  ; Lade das Highbyte
                  der zweiten Zahl
                  in den Akkumulator
ADC   A,B       ; Addiere die beiden
                  Highbytes und zu-
                  sätzlich den Wert
                  des Carryflags.
LD    (C005),A  ; Speichere Highbyte
                  des Ergebnisses nach
                  C005
RET
```

Doch mit der Addition alleine kann man noch nicht allzuviel anfangen. Daher gibt es auch einen Subtraktionsbefehl. Der Subtraktionsbefehl wird in der mnemotechnischen Darstellung wie folgt geschrieben: "SUB" (SUBtrahiere). Beim "ADD" - Befehl gibt es, wie schon kurz gesagt wurde, die Möglichkeit auch 16-Bit Register zu addieren. Da es diesen Komfort beim "SUB"-Befehl nicht gibt, und daher nur zum Akkumulator addiert werden kann, ist der Akkumulator in der mnemotechnischen Darstellung nicht mehr anzuschreiben. Die Syntax, um das Register D vom Akkumulator zu subtrahieren, lautet "SUB D".

Beim "SUB"-Befehl gibt es natürlich auch die Möglichkeit, das Carryflag mit einem speziellen Befehl abzufragen. Dieser Befehl ist der "SBC"-Befehl und die Abkürzung von "SuBtrahiere mit Carry". Tritt beim Subtrahieren zweier 8-Bit Zahlen ein Übertrag auf (bei der Subtraktion ist ein Übertrag, wenn die Summe kleiner Null ist), so bekommt das Carryflag den Wert Eins. Ist das Ergebnis größer oder gleich Null, so wird das Carryflag auf Null gesetzt. Beim "SBC"-Befehl wird, nachdem der Operand vom angegebenen Register subtrahiert wurde, dann auch noch der Wert des Carryflags subtrahiert.

Da es aber beim "SBC"-Befehl schon möglich ist, im Gegensatz zum "SUB"-Befehl, 16-Bit-Register zu subtrahieren, muß hier der Akkumulator angeschrieben werden. Wir wollen gleich einmal eine 16-Bit Subtraktion erstellen. Bei diesem Beispiel liegen die beiden Operanden bei C000 und C002. Das Ergebnis soll bei C004 abgelegt werden.

```
LD    A,(C000)  ; Lade Lowbyte des
                  ersten Operanden
LD    B,A       ; nach B
LD    A,(C002)  ; Lade Lowbyte des
                  zweiten Operanden
                  in den Akkumulator
SUB   B         ; Subtrahiere B vom
                  Akkumulator
LD    (C004),A  ; Speichere Ergebnis
```

```
LD   A,(C001)  ; Lade Highbyte des
                 ersten Operanden
LD   B,A       ; nach B
LD   A,(C003)  ; Lade Highbyte des
                 zweiten Operanden
                 in den Akkumulator
SBC  A,B       ; Subtrahiere B vom
                 Akkumulator und
                 dann das Carryflag
LD   (C005),A  ; Speichere Ergebnis
RET            ; Ende
```

Da nun alle Möglichkeiten von 8-Bit-Additionen und Subtraktionen erklärt wurden, wollen wir uns jetzt der Addition und Subtraktion unter Verwendung von 16-Bit-Registern zuwenden. Schon bei den Ladebefehlen wurde aufgezeigt, daß man die Registerpaare HL, BC und DE als 16-Bit-Register verwenden kann. Bei vielen Befehlen, welche auch das Arbeiten mit den Registerpaaren erlauben, wird HL bevorzugt. Auch beim Addieren ist das so. Hierbei ist es möglich, das Registerpaar HL wie einen 16-Bit-Akkumulator zu verwenden. Zu HL kann wieder eines der drei anderen Registerpaare addiert werden. Zusätzlich gibt es, ebenso wie bei der 8-Bit-Addition, den Befehl "ADC". Wendet man ihn an, so wird zum 16-Bit-Ergebnis noch der Wert des Carryflags addiert.

Um die Vereinfachung der Programme durch Verwendung von 16-Bit-Registerpaaren zu veranschaulichen, werde ich jetzt ein Programm für eine 32-Bit-Addition erstellen. Die beiden zu addierenden 32-Bit-Zahlen sollen in C000, C001, C002, C003 und in C010, C011, C012 und C013 abgelegt sein. Natürlich sind wieder zuerst die Lowbytes und dann die Highbytes abgelegt.

```
LD HL,(C000)    ; Lade die beiden
                ; Lowbytes des ersten
                ; Operanden nach HL
LD DE,(C010)    ; Lade die beiden
                ; Lowbytes des zweiten
                ; Operanden nach DE
ADD HL,DE       ; Addiere die beiden
                ; Operanden
LD (D000),HL    ; Speichere Lowbytes
                ; des Ergebnisses nach
                ; D000
LD HL,(C002)    ; Lade die beiden High-
                ; bytes des ersten Ope-
                ; randen nach HL
LD DE,(C012)    ; Lade die beiden High-
                ; bytes des zweiten Ope-
                ; randen nach DE
ADC HL,DE       ; Addiere beide Operan-
                ; den und zusätzlich den
                ; Wert des Carryflags
LD (D002),HL    ; Speichere Highbytes
                ; des Ergebnisses nach
                ; D002
RET             ; Beende Programm
```

Obwohl es nicht gesagt wurde, wird natürlich nur der Inhalt von HL verändert. Das angegebene Registerpaar wird in keiner Weise geändert, das Ergebnis der Addition wird nur in HL abgelegt.

Doch nun zur 16-Bit-Subtraktion. Hierbei wird es jedoch etwas komplizierter. Wie schon vorher erwähnt worden ist, können mit dem "SUB"-Befehl keine 16-Bit-Subtraktionen durchgeführt werden. Es gibt daher leider nur die Möglichkeit, 16-Bit-Register voneinander zu subtrahieren, indem man den "Subtrahiere mit Carry (SBC)"-Befehl anwendet.

Da beim "SBC"-Befehl immer der Wert des Carryflags wegsubtrahiert wird, scheint es unmöglich, eine normale Subtraktion durchzuführen. Eine normale Subtraktion wäre nur dann möglich, wenn das Carryflag nicht wegsubtrahiert werden würde, oder wenn es den Wert Null hätte. Denn ist der Wert Null, so wird vom Ergebnis Null subtrahiert und das Ergebnis keineswegs beeinflußt. Das einzige Problem ist also, das Carryflag zu löschen. Dafür gibt es leider keinen eigenen Befehl, aber "OR A" erreicht das Gewünschte. Was dieser Befehl bewirkt, wird erst später erklärt. Wichtig ist nur, daß durch die Ausführung dieses Befehls das Carryflag auf Null gesetzt wird.

Will man nun eine "normale" Subtraktion durchführen, so schreibt man unmittelbar davor den Befehl "OR A". Es gibt, wie schon kurz gesagt wurde, keinen eigenen Befehl zum Löschen des Carryflags. Es sind aber zwei Befehle vorhanden, die nur das Cary beeinflussen. Der eine Befehl lautet "SCF" (Set CarryFlag). Dieser Befehl gibt dem Carryflag den Wert Eins, es ist genau der verkehrte Befehl, den wir gesucht haben. Der zweite Befehl ist "CCF" (Complement CarryFlag). Dieses Kommando darf nie mit dem hier nicht vorhandenen Befehl "Clear CarryFlag)" verwechselt werden, denn "CCF" invertiert das Flag. Hatte es zuvor den Wert Null, dann hat es nachher den Wert Eins und umgekehrt.

Es wäre aber möglich, durch Ausführen des "SCF" und des "CCF" das Carryflag zu löschen, nur diese beiden Anweisungen würden zwei Bytes belegen, und "OR A" belegt nur ein Byte. Nachdem ein 32-Bit-Additionsprogramm geschrieben wurde, wird jetzt natürlich auch ein Programm für eine 32-Bit-Subtraktion erstellt. Dabei sind wiederum die beiden Operanden in C000-C003 und in C010-C013 abgelegt. Das Ergebnis wird nach D000 abgespeichert.

```
LD HL,(C000)    ; Lade Lowbytes des
                ; ersten Operanden
                ; nach HL
LD BC,(C010)    ; Lade Lowbytes des
                ; zweiten Operanden
                ; nach BC
OR A            ; Lösche Carryflag
SBC HL,BC       ; Subtrahiere BC von HL
LD (D000),HL    ; Speichere Lowbytes
                ; des Ergebnisses
                ; nach D000
LD HL,(C002)    ; Lade Highbytes des
                ; ersten Operanden
                ; nach HL
LD BC,(C012)    ; Lade Highbytes des
                ; zweiten Operanden
                ; nach BC
SBC HL,BC       ; Subtrahiere BC und
                ; Carry von HL
LD (D000),HL    ; Speichere Highbytes
                ; des Ergebnisses
                ; nach D002
```

Nachdem Addition und Subtraktion gründlich besprochen wurden, wollen wir in der nächsten Ausgabe mit den "BCD"-Zahlen fortsetzen.

Tips & Tricks

Die interne Struktur des Basic-Programmspeichers 4. Teil

Zu Beginn dieses vierten Teils muß ich leider mitteilen, daß es mir aus Zeitgründen nicht mehr möglich sein wird, die Rubrik TIPS & TRICKS weiterzuführen. Ebenso ist es mir aus dem oben genannten Grund nicht möglich, das in der ersten Folge versprochene Programm, im Rahmen dieser Serie zu veröffentlichen. Ich würde mich allerdings freuen, wenn sich jemand findet, der für diese Rubrik weiterschreibt.

In diesem letzten Teil der ersten Serie möchte ich nun noch die Punkte aufzählen, welche beim Generieren (=Erzeugen) von Programmzeilen aus einem vorhandenen Programm heraus, zu beachten sind.

Als erstes wäre hier zu erwähnen, daß man sich für die noch notwendigen Programmzeilen bereits im vorhinein genügend Platz schaffen muß. Dies hat folgende Gründe: Der BASIC-Interpreter braucht den Platz hinter dem Programm für Variablen und deren Zeiger. Vom Programm erzeugte Zeilen werden aber vom Computer nicht erkannt. Die Folge ist nun, daß einerseits die Variablen beim Erzeugen der Zeile gelöscht werden. Aber andererseits existiert diese Zeile nicht lange, denn der Computer lädt seine Variablen an den alten Platz, wo die neuerzeugte Zeile steht. Der Interpreter betrachtet diese Zeile nämlich als undefinierbare zufällige Speicherwerte und beachtet sie nicht.

Der einfachste Weg, um den benötigten Platz zu schaffen, besteht darin, daß man am Programmbeginn einige "leere" Zeilen einfügt. Bei diesem Vorgang hat man außerdem noch einen gewaltigen Vorteil; es ist nämlich von Haus aus genau definiert, wo dieser "leere" Raum im Speicher liegt. Dies allerdings nur solange, als diese Zeilen wirklich am Anfang des Programms stehen (auf den Programmablauf hat dies keinen Einfluß). Diese leeren Programmzeilen schafft man dadurch, daß man eine Zeilennummer eingibt und danach mittels der Leertaste oder dem Doppelpunkt beliebig viele Bytes reservieren kann (es wird kein "REM" benötigt).

Es ist jedoch folgendes zu beachten: Wenn man diesen Vorgang mit der Leertaste durchführt, so muß die Zeile mit einem Doppelpunkt abgeschlossen werden, da ansonsten der BASIC-Interpreter nur die Zeilennummer erkennen kann und eine Fehlermeldung bringt (Undefined Line Number). Und um sich dies alles besser vorstellen zu können, ist unten ein Speicherauszug und das dazugehörige Programmlisting (="Speicherauszug 1") abgedruckt.

Wenn Sie nun diesen Speicherauszug und das Listing genauer betrachten, werden Sie mit ziemlicher Sicherheit die Zeiger auf den Beginn der nächsten Programmzeile erkennen können (siehe Heft 1). Diese Zeiger müssen bei jeder Veränderung der Zeilenlänge auf den aktuellen Stand geändert werden. Um diesen Aufwand so gering als möglich zu halten (d.h. so wenig Zeigerwerte als möglich ändern) glaubte ich folgende Lösung gefunden zu haben. Ich habe nämlich versucht, den Zeiger der ersten Zeile so zu ändern, daß er nicht auf die nächste, sondern auf die letzte der vorbereiteten "Leerzeilen" zeigt. Dadurch braucht man schließlich beim Programmzeilenerzeugen nur noch einen Wert - nämlich den ersten Zeiger - zu ändern (siehe "Speicherauszug 2").

Ich habe allerdings feststellen müssen, daß dieser Vorgang, so effektvoll er auch sein mag (man könnte ganze Programmteile "verstecken" und nach der Wiederherstellung der Zeiger aufrufen), aus zwei Gründen ungeeignet ist. Der erste Grund ist der, daß bei einem einfachen Überspringen der Zeilen nach einem Speichern und erneuten Laden die Zeiger wieder die alten Werte enthielten.

Ein weiterer Nachteil ergibt sich daraus, daß der übersprungene Bereich beim Laden mit "MERGE" verloren geht. Bei intakten Zeigerwerten (d.h. die Zeigerwerte stehen direkt nach dem Zeilenende), auch wenn diese Werte bereits geändert wurden (z.B. es wurden andere Zeilenlängen benötigt), traten diese Probleme jedoch nicht auf. Zum besseren Verständnis wieder ein Speicherauszug mit zugehörigem Listing, in welchem ein kurzes Unterprogramm dargestellt wird (siehe "Speicherauszug 3").

Als Abfallprodukt meiner Versuche habe ich einen vielleicht interessanten Punkt entdeckt. Es ist nämlich möglich, durch Veränderung der entsprechenden Speicherwerte beliebig viele Zeilen mit der gleichen Zeilennummer zu versehen. Diese Nummerierung bleibt auch nach einem Abspeichern und erneutem Laden vorhanden. Bei einem Ladevorgang mit "MERGE" bleibt allerdings nur die letzte Zeile erhalten, da alle vorhergehenden Zeilen mit gleichlautender Nummer sofort überschrieben werden. Weiters ist zu beachten, daß bei Sprungbefehlen immer die erste Zeile im Programm mit der betreffenden Zeilennummer angesprungen wird.

Wolfgang Rotschek

Speicherauszug 1

```
10          :        8000 00 10 80 0A 00 20 20 20 20 20 20 20 20 20 3A 00 .....          :.
15           :       8010 20 80 0F 00 20 20 20 20 20 20 20 20 20 3A 00  ...          :.
20          :        8020 30 80 14 00 20 20 20 20 20 20 20 20 20 3A 00 0...          :.
25          :        8030 40 80 19 00 20 20 20 20 20 20 20 20 20 3A 00 @...          :.
30          :        8040 50 80 1E 00 20 20 20 20 20 20 20 20 20 3A 00 P...          :.
                     8050 5E 80 50 00 95 20 90 20 8D 20 0E B2 61 00 72 80 ^.P.. . . ..a.r.
```

Speicherauszug 2

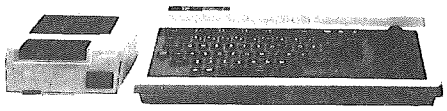
```
10          :        8000 00 50 80 0A 00 20 20 20 20 20 20 20 20 20 3A 00 .P...          :.
                     8010 20 80 0F 00 20 20 20 20 20 20 20 20 20 3A 00  ...          :.
                     8020 30 80 14 00 20 20 20 20 20 20 20 20 20 3A 00 0...          :.
                     8030 40 80 19 00 20 20 20 20 20 20 20 20 20 3A 00 @...          :.
                     8040 50 80 1E 00 20 20 20 20 20 20 20 20 20 3A 00 P...          :.
                     8050 5E 80 50 00 95 20 90 20 8D 20 0E B2 61 00 72 80 ^.P.. . . ..a.r.
```

Speicherauszug 3

```
10 DEF FNT(X)=SIN(X)     8000 00 13 80 0A 00 97 20 DE 54 28 58 29 F1 FF 89 28 ....... .T(X)....
15 RETURN                8010 58 29 00 19 80 0F 00 8E 00 20 80 14 00 20 3A 00 X)..... ... :.
20  :                    8020 30 80 19 00 20 20 20 20 20 20 20 20 20 20 3A 00 0...          :.
25          :            8030 40 80 1E 00 20 20 20 20 20 20 20 20 20 20 3A 00 @...          :.
30          :            8040 50 80 23 00 20 20 20 20 20 20 20 20 20 20 3A 00 P.#.          :.
35          :            8050 5E 80 50 00 95 20 90 20 8D 20 0E B2 61 00 72 80 ^.P.. . . ..a.r.
```

Die Super-Computer.

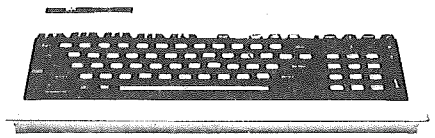
SVI



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

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

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



SVI-328 32 K ROM, 80 K RAM, Erweitertes MICROSOFT-BASIC, Schreibmaschinentastatur, 10 Funktionstasten, 10er-Block **öS 7.990,—**

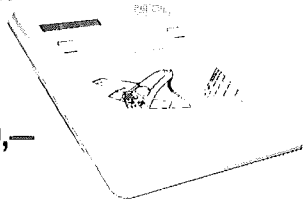
Super-Expander SVI-605, ein eingebautes Diskettenlaufwerk (160 K), Centroniks-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 14.990,—**

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

Super-Expander SVI-605 B, mit Supersoftware-Paket, zwei eingebaute Diskettenlaufwerke (je 320 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 WordStar, Mailmerge, CalcStar ReportStar, DataStar **öS 29.990,—**

SVI-328 Pro Profisystem bestehend aus: Computer SVI-328, Super-Expander SVI-605 A (inkl. WordStar, Mailmerge, CalcStar, DataStar, ReportStar) Betriebssystem CP/M 2.2, 80-Zeichenkarte SVI-806, **öS 35.690,—**

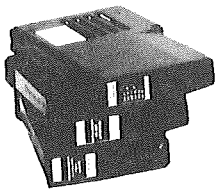
Grafik-Tablett SVI-105, 186 x 158 mm Zeichenfläche, Kassette mit Anwender-Software inkl. **öS 2.590,—**



Erweiterungskarten für SVI-605, A, B

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

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



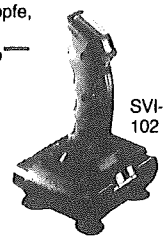
SVI-806 80-Zeichenkarte **öS 3.890,—**

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

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

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

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



SVI-102

Die Software

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

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

Diskettenlaufwerk SVI-905, 160 K, zur Erweiterung des Super-Expanders SVI-605 **öS 7.690,—**

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

Preise incl. MWSt.

SVI-FACHBERATUNG UND VERKAUF BEI:

Großhandel, Facheinzelhandel **BASTL-COMPUTER-SYSTEME** **2700 Wr. Neustadt,** Hauptplatz 5 Tel. (0 26 22) 22 7 20 - 59 80 Telex 16 522	**DAHMS-PRAKTIKER-ELEKTRONIK** Pilgramgasse 11 **1050 Wien** Tel. (02 22) 54 34 21	**ESV ELEKTROTECHNISCHER SERVICE MBH.** Bayerhamerstraße 19—21 **5020 Salzburg** Tel. (06 62) 74 7 51	**SCHILLER MICRO-COM-BOUTIQUE** Fasangasse 21 **1030 Wien** Tel. (02 22) 78 35 99, 78 56 61	**TARGET ELECTRONIC** Bergstraße 6 **6900 Bregenz** Tel. (0 55 74) 23 7 18
BYTE COMPUTER Favoritenstraße 20 **1040 Wien** Tel. (02 22) 65 73 42 Telex 132 827	**EDV-STUDIO PORSCH** Kinderspittalgasse 13 **1090 Wien** Tel. (02 22) 42 63 44	**FEDCON ELEKTRONIK** Ing. Franz Krenn Kanalplatz 86 **9400 Wolfsberg** Tel. (0 43 52) 42 73	**TARGET ELECTRONIC** Reichsstraße 123a **6800 Feldkirch** Tel. (0 55 22) 21 5 29 Telex 52 300	**WEHSNER GMBH. COMPUTER-STUDIO** Paniglgasse 18—20 **1040 Wien** Tel. (02 22) 65 88 93, 65 78 08

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

SVI-Hardcopy mit EPSON-Drucker

Mir ist es nun endlich gelungen, ein Hardcopy-Programm für den SV-318/328 zu schreiben. Dieses Programm besteht hundertprozentig aus Maschinencode und ist dementsprechend schnell. Ein weiterer Vorteil gegenüber dem Hardcopy-Programm in Heft 1 hat das neue ebenfalls zu bieten: Das Bild erscheint am Drucker nicht mehr um 90 Grad gedreht.

Bei der Verwendung dieses Programmes, muß aber einiges beachtet werden. Erstens braucht der Computer einen transparenten Hintergrund, außerdem werden daher transparente Linien und Punkte nicht gedruckt. Wenn das vergessen wird, und der Hintergrund zum Beispiel die Farbe grün hat, so wird am Drucker alles schwarz ausgedruckt. Ich habe transparent gewählt, weil es nicht nur wesentlich einfacher zu programmieren ist, sondern weil viele bereits generierten Bilder die Hintergrundfarbe transparent haben. Natürlich wird auch eine schwarze Linie als Bildschirmbegrenzung ausgedruckt, obwohl diese am Bildschirm nicht sichtbar sind.

Versuchen Sie bitte einige komplizierte Graphiken zu erstellen, um damit mein Hardcopy-Programm zu testen. Wenn Fehler auftreten, so teilen Sie mir das bitte an einem der Clubabende mit.

Doch nun zum Programm. Als erstes werden die Interrupts gesperrt, damit das Programm nicht alle 20 Millisekunden unterbrochen wird. Danach werden die Labels "DRUCK" und "TABEL" definiert. "DRUCK" bezeichnet die Einsprungadresse für die Ausgabe eines Bytes an den Drucker. Dabei muß im Akkumulator das zu druckende Zeichen stehen. "TABEL" wird nur zum Zwischenspeichern von acht Bytes aus dem Video-RAM gebraucht.

Als Nächstes werden drei Leerzeilen gedruckt, der Zeilenabstand auf Graphik eingestellt und dann wird angegeben, daß 512 Graphikzeichen (getreu den 256 waagrechten Bildschirmpunkten, die in doppelter Ausführung auf das Papier kommen) gedruckt werden sollen. Doch bevor noch die Graphik eingeschalten wird, lädt der Computer das Register 'C' mit den zu druckenden Zeilen.

Da der Drucker immer das Bit-Muster eines Graphikbytes vertikal druckt, sind es 24 Zeilen (24*8=192). Vor jeder Zeile wird dann auf 512 Graphikzeichen umgeschaltet. Bei dem Label "HC7" wird die Anzahl der acht Byte-Blöcke pro Zeile geladen =32.

Da der Videoprozessor seine Speicher etwas pervers angeordnet hat, brauchte ich so lange, dieses Programm zu erstellen. Damit Sie verstehen können, was mit "pervers" gemeint ist, möchte ich kurz den Aufbau des Videospeichers erklären. Ein Byte im VDP (Video-Prozessor)-RAM, beinhaltet acht Punkte, die am Bildschirm angezeigt werden. Das heißt, daß jedes Bit in einem Byte am Bildschirm einen Punkt charakterisiert. So einem Byte ist wiederum ein zweites Byte zugeordnet, in welchem der Farbcode für die ungesetzten Punkte (=Hintergrundfarbe) und für die gesetzten Punkte (=Vordergrundfarbe) steht. Die Adresse für den Farbcode von einem Byte (=8 Punkte) bekommt man, indem zur Adresse des Bytes hexadezimal 2000 dazuaddiert wird.

Das erste Byte liegt links oben, horizontal (=die 8 Punkte sind waagrecht dargestellt) am Bildschirm bei der Adresse 0, das zweite Byte liegt unter dem ersten Byte. Das geht so lang bis zum achten Byte. Denn das neunte Byte befindet sich nicht unter dem achten sondern neben dem ersten Byte. Das sechzehnte Byte liegt daher neben dem achten und das siebzehnte liegt neben dem neunten Byte. Acht untereinander liegende Bytes werde "Blöcke" oder "Pattern" genannt. Da eine Zeile 256 Punkte hat, sind das logischerweise 32 Bytes (256/8). Doch normalerweise sind mit einer Zeile 32 Pattern gemeint, welche aus 256 (32*8) Bytes bestehen, wovon es 24 Stück gibt. Daher ist zu ersehen, daß ich 24 Zeilen zu 32 Zeichen (Pattern) nahm.

Doch das eigentlich Problem bestand darin, daß die Bytes nicht genauso auszugeben sind, wie sie im VDP-RAM stehen. Am Drucker werden von einem Byte ebenfalls 8 Punkte ausgegeben. Aber diese Punkte werden nicht so wie am Bildschirm waagrecht, sondern vertikal ausgegeben. Ich mache es daher so, daß der Computer zuerst ein Pattern (8*8) einliest. Dann nehme ich von diesen Bytes der Reihe nach das erste Bit jeder Zeile, und stopfe diese in den Akkumulator.

Habe ich von allen acht Bytes die ersten Bits im Akkumulator, so enthält dieser die ersten acht vertikalen Punkte für den Drukker. Dann nehme ich die zweiten Bits von den acht Bytes und so weiter. Nachdem alle Bits von einem Byte "herausgekletzelt" worden sind (=8 mal), so lasse ich das nächste Pattern einlesen und es geht wieder von vorne los. Nachdem dies 32mal geschehen ist, sende ich ein Carriage Return, welches mir den Druckkopf an den Zeilenanfang bringt und ein Line Feed, wodurch das Blatt um eine Zeile nach vor geschoben wird.

Es wird bestimmt schon die Frage aufgetaucht sein warum die einzelnen Points, wie oben erwähnt, doppelt ausgedruckt werden (insgesamt 512 Zeichen bei 256 Points pro Zeile). Dies hat etwas mit dem Seitenverhältnis der Punkte am Drucker zu tun. Ein ausgegebener Punkt am Drucker ist nicht quadratisch, sonder cirka um das Doppelte höher. Daher gebe ich jedes Punktmuster zweimal auf den Drucker aus (256*2=512).

Doch das Programm wird hauptsächlich durch die Feststellung der sichtbaren Punkte am Bildschirm in die Länge gezogen. Wenn zum Beispiel kein Punkt gesetzt ist, so muß der Hintergrund aber nicht transparent sein. Wenn die Hintergrundfarbe <> Null ist, so scheinen alle acht Punkte am Bildschirm gesetzt. Dies passiert hauptsächlich beim Paint-Befehl, wo nicht die Punkte gesetzt werden, sonder die Hintergrundfarbe den Wert der Farbe bekommt, mit der "gePaintet" wird (ausgenommen Randgebiete).

Das gleiche ist der Fall, wenn der Hintergrund <> Null ist, da dann angenommen wird, daß der Befehl "PAINT" eingesetzt wurde. Weiters kann es vorkommen, daß die Vordergrundfarbe transparent und die Hintergrundfarbe nicht transparent ist, so muß das Punktmuster invertiert werden. Doch das sind noch nicht alle Möglichkeiten. Es kann ja noch vorkommen, daß die Vordergrund- sowie die Hintergrundfarbe ungleich transparent sind. Somit haben alle Punkte gesetzt zu sein.

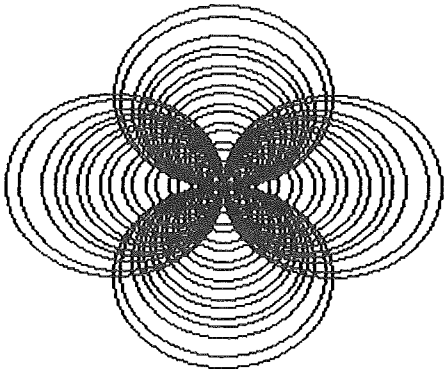
Soweit die programmtechnischen Aspekte! Unten steht nun das fertige Programm in einem Assemblerlisting. Wie man dieses MC-Programm in den Computer lädt, steht in dem Artikel "Ladeprogramm" in diesem Heft (Achtung! Im Ladeprogramm muß man die Hex-Codes eintippen!).

Weiters sind noch ein paar Vorführbeispiele angeführt, um zu zeigen, was die Hardcopy kann.

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

 ASSEMBLERTEXT GEFUNDEN
DURCHLAUF NR: 1


 KEINE ERRORS
 146 ZEILEN


 DURCHLAUF NR: 2
 ADDR  OPCODES  SYMBOLE  BEF.  OPERAND          KOMMENTAR

 D000  F3                DI                    ; Hardcopy fuer transparenten Hin-
                                                ; tergrund
                 DRUCK:  EQU   3915H           ; Unterroutine fuer Zeichen auf
                                                ; Drucker
                 TABEL:  EQU   0D200H          ; 8-Bytes Tabelle
                                                ;
 D001  3E0D              LD    A,13            ; Carriage Return + drei Leerzeilen
 D003  CD1539            CALL  DRUCK
 D006  3E0A              LD    A,10
 D008  CD1539            CALL  DRUCK
 D00B  CD1539            CALL  DRUCK
 D00E  CD1539            CALL  DRUCK
                                                ;
 D011  3E1B              LD    A,27            ; Zeilenabstand
 D013  CD1539            CALL  DRUCK           ; auf Graphik
 D016  3E33              LD    A,"3"           ; einstellen
 D018  CD1539            CALL  DRUCK
 D01B  3E18              LD    A,24
 D01D  CD1539            CALL  DRUCK
                                                ;
 D020  210000            LD    HL,0000
 D023  0E18              LD    C,24            ; = Anzahl der Zeilen

 D025  0620      HC7:    LD    B,32            ; = Zeichen pro Zeile (32 8*8 Bit
                                                ; Felder)
 D027  3E1B              LD    A,27            ; Graphik einschalten
 D029  CD1539            CALL  DRUCK           ; 512 Graphikzeichen
 D02C  3E4C              LD    A,"L"           ;
 D02E  CD1539            CALL  DRUCK
 D031  3E00              LD    A,0             ; Lowbyte von 512
 D033  CD1539            CALL  DRUCK
 D036  3E02              LD    A,02            ; Highbyte von 512
 D038  CD1539            CALL  DRUCK

 D03B  C5        HC6:    PUSH  BC
 D03C  0608              LD    B,08            ; 8 Bytes
 D03E  1100D2            LD    DE,TABEL
 D041  CDDBD0            CALL  VDP             ; Werden vom Video-

 D044  DB84      HC1:    IN    A,(132)         ; speicher eingelesen.
 D046  FE00              CP    00              ; Wenn [A]=0 -> [A]=0 oder [A]=255
 D048  CAC0D0            JP    Z,GROUND        ; Wenn Hintergrund <> 0 -> [A]=255
 D04B  C5                PUSH  BC
 D04C  D5                PUSH  DE
 D04D  E5                PUSH  HL
 D04E  110020            LD    DE,2000H
 D051  19                ADD   HL,DE           ; [HL] = Farbcodeadresse
 D052  CDDBD0            CALL  VDP
 D055  DB84              IN    A,(132)         ; [A] = Farbcode einlesen
 D057  B7                OR    A
 D058  2823              JR    Z,TREND         ; Wenn [A] = 0 dann Punktmuster
                                                ; bleibt unveraendert
 D05A  5F                LD    E,A
 D05B  E6F0              AND   11110000B       ; [A] = Vodergrundfarbe
 D05D  4F                LD    C,A             ; [C] = ----- "" ------
 D05E  7B                LD    A,E
 D05F  E60F              AND   00001111B       ; [A] = Hintergrundfarbe
 D061  B7                OR    A
 D062  2819              JR    Z,TREND         ; Wenn Hintergr. = 0 -> Punkt-
                                                ; muster bleibt unveraendert
 D064  79                LD    A,C
 D065  B7                OR    A               ; [A] = Vordergrundfarbe
 D066  200B              JR    NZ,FULL         ; wenn [A]=0 -> [A]=255
 D068  E1                POP   HL
 D069  CDDBD0            CALL  VDP             ; Wenn Vordergr.=0 und Hintergr.
                                                ; <> 0
 D06C  DB84              IN    A,(132)         ; -> invertiere Punktmuster
 D06E  2F                CPL
 D06F  D1                POP   DE
 D070  C1                POP   BC
 D071  1812              JR    GRRET
```

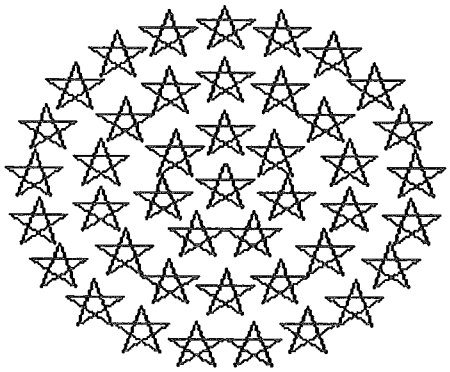
```
D073  E1        FULL:   POP  HL              ; Wenn Hintergrund und
D074  D1                POP  DE              ; Vordergrund <>0
D075  C1                POP  BC              ; -> [A]=255
D076  CDDBD0            CALL VDP
D079  3EFF              LD   A,255
D07B  1808              JR   GRRET

D07D  E1        TREND:  POP  HL
D07E  D1                POP  DE
D07F  C1                POP  BC
D080  CDDBD0            CALL VDP
D083  DB84              IN   A,(132)         ; lese Byte aus VDP und lasse un-
                                             ; veraendert
                                             ;

D085  12        GRRET:  LD   (DE),A          ; Punktmuster in Tabelle ablegen
D086  23                INC  HL
D087  13                INC  DE
D088  10BA              DJNZ HC1             ; 8 mal wiederholen bis [B] = 0
D08A  E5                PUSH HL
                                             ;
D08B  0608              LD   B,08            ; In dieser Routine werden die acht
                                             ; eingelesenen Bytes in das fuer
                                             ; den Drucker richtige Punktmuster
                                             ; ungeaendert

D08D  2100D2    HC2:    LD   HL,TABEL
D090  0E08              LD   C,08

D092  CB06      HC3:    RLC  (HL)            ; Rotiere Punkt ins Carry
D094  17                RLA                  ; Schiebe Punkt im Akku nach
D095  23                INC  HL
D096  0D                DEC  C
D097  20F9              JR   NZ,HC3
D099  CD1539            CALL DRUCK           ; Sende acht vertikale Punkte (=1
                                             ; Byte) zum
D09C  CD1539            CALL DRUCK           ; Drucker. Seitenverhaeltnis der
                                             ; Punkte: Eins zu Zwei
D09F  10EC              DJNZ HC2
D0A1  E1                POP  HL
D0A2  C1                POP  BC
D0A3  1096              DJNZ HC6             ; wiederhole 32 mal (32*8=255) 255
                                             ; Punktmuster fuer Drucker
D0A5  3E0D              LD   A,13            ; naechste Zeile
D0A7  CD1539            CALL DRUCK
D0AA  3E0A              LD   A,10            ; Line Feed
D0AC  CD1539            CALL DRUCK
D0AF  0D                DEC  C
D0B0  C225D0            JP   NZ,HC7
D0B3  3E0A              LD   A,10
D0B5  CD1539            CALL DRUCK           ; drei Leerzeilen
D0B8  CD1539            CALL DRUCK
D0BB  CD1539            CALL DRUCK
D0BE  FB                EI
D0BF  C9                RET                  ; zurueck ins Basic
                                             ;

D0C0  E5        GROUND: PUSH HL
D0C1  D5                PUSH DE
D0C2  110020            LD   DE,2000H        ; Beginn Farbspeicher
D0C5  19                ADD  HL,DE
D0C6  CDDBD0            CALL VDP
D0C9  DB84              IN   A,(132)         ; [A] = Farbcode (1.Nibble Hinter-
                                             ; grund, 2.Nibble Vordergrund)
D0CB  E60F              AND  00001111B       ; [A] = Hintergrund
D0CD  2802              JR   Z,GR1
D0CF  3EFF              LD   A,255           ; Wenn Hintergrund <> 0 ->

D0D1  D1        GR1:    POP  DE              ; -> [A] = 255
D0D2  E1                POP  HL
D0D3  23                INC  HL
D0D4  CDDBD0            CALL VDP             ; Schreibe Adresse nach VDP
D0D7  2B                DEC  HL
D0D8  C385D0            JP   GRRET

D0DB  F5        VDP:    PUSH AF
D0DC  7D                LD   A,L
D0DD  D381              OUT  (129),A         ; Adresse in [HL]
D0DF  7C                LD   A,H             ; an den
D0E0  D381              OUT  (129),A         ; VDP
D0E2  F1                POP  AF              ; hinausschreiben
D0E3  E3                EX   (SP),HL
D0E4  E3                EX   (SP),HL         ; Zeitverzoegerung
D0E5  C9                RET

 KEINE ERRORS
 PROGRAMMBEGINN : D000
 PROGRAMMENDE   : D0E6
 PROGRAMMLAENGE:  230 BYTES
```

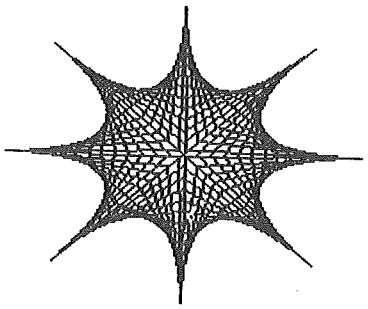
```
SYMBOLE:
DRUCK     : 3915 ,  14613      ; TABEL    : D200 ,  53760
HC7       : D025 ,  53285      ; HC6      : D03B ,  53307
HC1       : D044 ,  53316      ; FULL     : D073 ,  53363
TREND     : D07D ,  53373      ; GRRET    : D085 ,  53381
HC2       : D08D ,  53389      ; HC3      : D092 ,  53394
GROUND    : D0C0 ,  53440      ; GR1      : D0D1 ,  53457
VDP       : D0DB ,  53467

ENDE DER ASSEMBLIERUNG
```



```
100   COLOR8,8,1
110 SCREEN1:R=37
120 CIRCLE(128+R, 96),R,1
130 CIRCLE(128-R, 96),R,1
140 CIRCLE(128, 96-R),R,1
150 CIRCLE(128, 96+R),R,1
160 IFR>5THENR=R-5:GOTO120
170 R=48
180 CIRCLE(128+R, 96),R,1
190 CIRCLE(128-R, 96),R,1
200 CIRCLE(128, 96-R),R,1
210 CIRCLE(128, 96+R),R,1
220 PAINT(1,1),1
230 PAINT(250,3),1
240 PAINT(250,190),1
250 PAINT(1,190),1
260 IFR>5THENR=R-3:GOTO120
270 GOTO270
```

```
100 CLS
110 INPUT"ZAHL DER ECKEN(UNGERADE)";A
120 INPUT"ZAHL DER STERNE(1.KRANZ)";B
130 SCREEN1:RR=13:DIMX(3*B),Y(3*B)
140 FORI=0TO3:N=B*I:R=28*I:IFN=0THENN=1
150 FORJ=1TON:W=J*2*3.1415/N
160 X(J)=128+R*SIN(W):Y(J)= 96-R*COS(W)
170 FOR K=1TOA:W1=(A-1)*3.1415*K/A:W2=(A-1)*
3.1415*(K-1)/A
180 X1=X(J)+RR*SIN(W1):Y1=Y(J)-RR*COS(W1)
190 X2=X(J)+RR*SIN(W2):Y2=Y(J)-RR*COS(W2)
200 LINE(X1,Y1)-(X2,Y2)
210 NEXT:NEXT:NEXT
220 GOTO 220
```





```
10 CLS
20 INPUT"Dichte ";D
30 SCREEN1:P=.7071*D
40 X=57+P:FORI=0TO80STEPD:X=X-P
50 LINE(I+127,96)-(X+127,X+96)
60 LINE(X+127,X+96)-(127,96+I)
70 LINE(X+127,96-X)-(127,96-I)
80 LINE(127-X,96-X)-(127,96-I)
90 LINE(127-X,96-X)-(127-I,96)
100 LINE(127-X,96+X)-(127-I,96)
105 LINE(127-X,96+X)-(127,96+I)
110 LINE(I+127,96)-(X+127,96-X)
120 NEXT
130 A$=INPUT$(1):GOTO10
```

****

ÖJ
RK

Sommertherapielager für:

50 asthmatische Kinder im Juli in St. Hemma/Pack
30 rheumatische Kinder im Juli in Warmbad Villach
40 körperbehinderte Kinder im Juli in Litzlberg
30 hämophile Kinder im Juli in Pinkafeld
40 diabetische Kinder im Juli in Eben/Pongau

Das JUGENDROTKREUZ konnte helfen.
Allen Unterstützern herzlichen Dank dafür.

Sollte Ihr Kind Hilfe benötigen, wenden Sie sich bitte an:

ÖSTERREICHISCHES JUGENDROTKREUZ
Generalsekretariat
Gußhausstraße 3, 1041 Wien
Tel. 65 36 61 / 45

****

```
*****************************************
*                                       *
*       MASCHINCODE-LADEPROGRAMM        *
*                                       *
*****************************************
```

Der Spectravideo mag ein sehr guter Computer sein, eines geht bei ihm hundsmiserabel schlecht. Er hat nicht die geringste Andeutung eines Monitors für Maschincodeprogramme. Nicht einmal das Hineinladen eines MC-Programms wird erleichtert. Wie schafft man es nun, seine mühsam erarbeitete Entwicklung in den Computer zu bringen?

Dazu muß man ein Ladeprogramm erstellen! Folgende Bedingungen muß dieses Programm erfüllen: Es muß Schutz gegen Zerstörung des MC-Programms durch den Interpreter bieten, eine Startadresse muß angegeben werden können und man möchte ein automatisches Ende haben, das heißt, wenn der Computer eine vorgegebene Marke (zum Beispiel ++) erreicht, dann soll er aufhören, in den Speicher zu schreiben.

Warum braucht nun ein MC-Programm Schutz gegen Zerstörungen! Der Interpreter legt sehr viel für ihn wichtiges im Speicher ab. Wenn nun zufälligerweise gerade dort das MC-Programm liegt, dann wird es überschrieben, ist doch logisch, oder? Deshalb müssen wir dem Computer mitteilen, daß in dem und dem Speicherbereich "etwas Unantastbares" liegt. Diesen Schutz bekommen wir mit "CLEAR". Diese Anweisung gibt an, ab welcher Speicherstelle der Interpreter seine Strings ablegen darf. Das Ende (Strings werden von hohen Adressen ab zu niedrigen abgelegt) wird nach dem Beistrich angegeben. Üblicherweise wird "CLEAR 200, Stringspeicherende " verwendet, die 200 ist unwichtig für unsere Betrachtungen. In Zeile 10 im untenstehenden Programm ist diese Anweisung zu sehen.

Danach setzen wir eine Schleife, die folgendes macht: Sie holt mit "READ" einen "DATA"-Wert in die Variable A$. Diese Werte sind nichts anderes als die hexadezimalen Opcodes des MC-Programms. Als erstes vergleicht der Interpreter nun, ob dieser Wert eine Marke ist (zum Beispiel ++). Wenn es keine ist, dann kann es sich nur um einen HEX-Code handeln. Wir dürfen aber keine Strings (A$ ist ja einer) in den Speicher "poken", denn "POKE" verträgt keine Strings. Außerdem müssen wir dem Computer erst einmal zeigen, daß es hexadezimale Werte sind, die hier vorliegen.

Die Umwandlung in hexadezimale Zahlenwerte besorgt die Funktion B=VAL("&H"+A$). Wie wir ja alle wissen, verbindet "+" zwei Strings miteinander und verkettet sie. Nun ist der Wert der Variablen A$ zum Beispiel 4E, dann entsteht durch "&H"+A$ "&H4E". Und das, wissen wir, interpretiert der Computer nun als hexadezimale Zahl, wenn wir das ganze noch mit "VAL" in eine Zahl umwandeln. "VAL ("&H"+A$)" wird allerdings nicht wie oben mit B angeschrieben, sondern gleich in den "POKE"-Befehl gepackt. Die Speicheradresse I wird anschließend um eins erhöht, und der Kreislauf fängt von vorne an.

Kommt nun die Marke (zum Beispiel ++), dann führt der Computer den "THEN"-Zweig aus, in unserem Fall "GOTO 50". Sollte man mehrere Marken verwenden, dann muß man mehrere "IF...THEN"-Anweisungen einsetzen. Hinter diese Schleife kommen die Hex-Codes in den "DATA"-Zeilen. Zur Veranschaulichung des Gesagten ist unten das Ladeprogramm mit einem MC-Programm (Die Softwareuhr vom vorigen Heft) abgedruckt. Manche Assembler-Spezialisten arbeiten in ihren Ladeprogrammen manchmal mit einigen Tricks, die aber so vielfältig und problemspezifisch sind, daß wir sie nicht beachten können.

```
10 CLEAR 200,&HD1FF:I=&HD200
20 READA$
30 IF A$="++" THEN 50
40 POKEI,VAL("&H"+A$):I=I+1:GOTO20
50 END
60 DATAF3,11,79,FE,21,1F,D2,3A,1E,D2,A7,28,0
5,21,23,D2,18,01,3C,32,1E,D2,01,04
70 DATA00,ED,B0,FB,76,C9,00,CD,30,D2,C9,C9,0
0,00,C9,00,00,00,00,00,00,00,00,00,F3,D9,DD,
21,62,D2,3A,60,D2
80 DATA47,3A,61,D2,4F,79,A7,28,12,DD,34,00,D
D,7E,00,DD,BE,01,0E,00,38,05,0C,DD
90 DATA36,00,00,DD,23,DD,23,10,E4,D9,C9,00,0
0,00,00,04,01,00,32,00,3C,00,3C,00,18,++
```

```
*****************************************
*                                       *
*       ZUSATZ ZU SOFTWAREUHR           *
*                                       *
*****************************************
```

Das Softwareuhrenprogramm weist einige Besonderheiten in seiner Bedienung auf. Das Programm sorgt nicht dafür, daß die Zeit angezeigt wird, sondern speichert nur die Werte der Stunden, Minuten und Sekunden in Speicherzellen ab. Ebenfalls das Starten ist etwas Schwieriger. Deshalb haben wir jetzt noch zusätzlich zum Artikel des Ladeprogramms einen kleinen Nachtrag für die Bedienung des Programms "Softwareuhr" speziell entwickelt. Dieses kleine BASIC-Programm nimmt dem Bediener lästige Kleinigkeiten ab, die für den Computer wichtig sind. Man braucht lediglich beim ersten Starten des Programms die Zeit eingeben und eine Taste drücken. Danach läuft das Programm vollautomatisch ab.

```
10 REM ZUSATZ ZU SOFTWAREUHR
20 DEFUSR=&HD200:A=USR(0):POKE&HD21E,0
30 INPUT"UHRZEIT STUNDEN:";A
40 INPUT"MINUTEN:";B
50 INPUT"SEKUNDEN:";C
60 POKE&HD261,0
70 POKE&HD262,0
80 POKE&HD264,C
90 POKE&HD266,B
100 POKE&HD268,A
110 PRINT"ZUM STARTEN DER UHR TASTE DRUECKEN
!"
120 S$=INPUT$(1):POKE&HD261,1
130 CLS:REM ANZEIGE DER UHRZEIT
140 LOCATE10,10,0:PRINTPEEK(&HD268)":"PEEK(&
HD266)":"PEEK(&HD264)
150 GOTO140
```

Wenn man das BASIC-Programm stoppt, dann läuft die Uhr aber trotzdem weiter. Um sie dann noch einmal aufzurufen, muß man RUN130 drücken. Man hat dann automatisch die richtige Uhrzeit. Das BASIC-Programm selber wird hier nicht erklärt, denn es ist nur ein Hilfsprogramm und seine Funktionsweise ist für den Bediener nicht sehr interessant.

Zum Laufenlassen der Uhr muß man nun folgende Punkte durchführen:

1) Das Ladeprogramm mit der Uhr rennen lassen! Wenn das MC-Programm im Speicher ist, meldet sich der Computer mit einem Ok. Andernfalls ist ein Fehler passiert!

2) Das BASIC-Hilfsprogramm laufen lassen. In diesem Programm werden alle Informationen gegeben, was wann eingegeben werden muß. Das Programm kann man mit "CTRL-STOP" stoppen. Die Uhr läuft dann weiter! Mit RUN 130 läßt sich das Programm mit der richtigen Uhrzeit wieder aktivieren.

```
*****************************************
*                                       *
*                CMDPOKE                *
*                                       *
*****************************************
```

CMDPOKE. Einer der im SVI-BASIC verwendeten Befehle hat interessanterweise keine Bedeutung. Wenn Sie "CMD" eingeben, antwortet Ihnen der Computer mit einem Biepser und der Fehlermeldung "Illegal function call". Ein zweites Befehlswort, "MON" führt zu dem gleichem Resultat. Nun könnte man meinen, die beiden Wörter hätten zumindestens unter DISK-BASIC eine Bedeutung, aber weit gefehlt. Auch hier kommt der Computer mit einer Fehlermeldung zurück. Nach einigem Suchen im ROM des Computers konnte ich die Behandlungsroutine für "CMD" finden. Sie besteht aus 2 Befehlen, nämlich einem Aufruf nach 0FFC9H und einem Sprung zu "Illegal function call". Genauso sieht es bei "MON" aus. Somit kann man die beiden Wörter für eigene Befehle verwenden. Das habe ich getan und eine Erweiterung des "POKE"-Befehls eingebaut.

Der neue "POKE"-Befehl hat folgende Syntax: CMD POKE XX TO YY DATA A1,A2,A3,.. wobei die Wörter "POKE","TO" und "DATA" nur eine bessere Lesbarkeit des gewünschten herbeiführen soll. Die erste Adresse XX gibt an, wo das Programm beginnen soll, die nach "DATA" angegebenen Daten hineinzuschreiben. Für die Daten gilt dasselbe wie bei "POKE", es müssen Integerzahlen sein, doch dürfen sie auch größer 255 sein. Wenn ein Wert kleiner als 256 ist, schreibt das Programm ihn als ein Byte in den Speicher, wenn er größer als 255 ist, so schreibt das Programm zuerst das Lowbyte und dann das Highbyte des Wertes. Dadurch wird nun auch ein Doppelpoke möglich. Aber Sie können auch einen bestimmten Speicherbereich mit einer konstanten Bytefolge auffüllen. So bewirkt z.B.

'CMDPOKE &HCF00 TO &HCFFF DATA 00,01,02'

das Auffüllen das Bereiches von 0CF00 bis 0CFFF mit 00,01 und 02 bevor es wieder mit 00 beginnt.

Leider müssen Sie ähnlich dem Uhrenprogramm ein paar Bytes in der Tabelle ab 0FFC9 ändern. Dort springt die "CMD"-Routine hin um gleich darauf unverrichteter Dinge wieder zurückzukehren. Sie leiten die Routine mit 3 "POKE"-Befehlen um.

POKE &HFFC9,&HC3 : POKE &HFFCA,0 : POKE &HFFCB,&HD0

Danach ist der Befehl "CMDPOKE" in der oben erwähnten Form verfügbar.

Philipp Ott

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

 ASSEMBLERTEXT GEFUNDEN
DURCHLAUF NR: 1

 KEINE ERRORS
 76 ZEILEN

DURCHLAUF NR: 2
ADDR  OPCODES  SYMBOLE  BEF. OPERAND        KOMMENTAR

D000  F3               DI                   ;Sperre Interrupts
D001  1809             JR   START           ;Sprung zum Hauptprogramm

D003  0000     BEG:    DEFW 00              ;Erste Adresse

D005  0000     END:    DEFW 00              ;Zweite Adresse

D007  80D0     DTA:    DEFW START+74H       ;Tabelle fuer DATA

D009  0000     PTR:    DEFW 00              ;Programmpointer

D00B  00       LEN:    DEFB 00              ;Laenge der Tabelle

D00C  AF       START:  XOR  A               ;Loesche Tabellen-
D00D  320BD0           LD   (LEN),A         ;laenge, sieh nach,
D010  CF               RST  8               ;ob das Token von 'POKE'

D011  98               DEFB 98H             ;vorhanden ist
D012  CD991A           CALL 1A99H           ;Lies erste Adresse ein
D015  ED5303D0         LD   (BEG),DE        ;und speichere sie
D019  2B               DEC  HL              ;Sieh nach, ob das Token
D01A  D7               RST  16              ;von 'TO' vorhanden ist
D01B  CF               RST  08              ;

D01C  D9               DEFB 0D9H            ;
D01D  CD991A           CALL 1A99H           ;Lies zweite Adresse ein
D020  ED5305D0         LD   (END),DE        ;und speichere sie
D024  2B               DEC  HL              ;Sieh nach, ob das
D025  D7               RST  16              ;Token von 'DATA'
D026  CF               RST  08              ;vorhanden ist

D027  84               DEFB 84H             ;
D028  ED5B07D0         LD   DE,(DTA)        ;Lade [DE] mit dem Tabellen-
                                            ;anfang

D02C  2B       LOOP:   DEC  HL              ;Sieh nach, ob der Befehl
D02D  D7               RST  16              ;abgeschlossen ist
D02E  2828             JR   Z,EXIT          ;Sprung, wenn (HL) = ':'
```

```
D030  D5       L1:     PUSH DE          ;Rette Tabellenpointer
D031  CD991A           CALL 1A99H       ;Lies einen Wert ein
D034  CD50D0           CALL LENUP       ;erhoehe Laenge
D037  7B               LD   A,E         ;[A] ist Lowbyte des Wertes,
D038  4A               LD   C,D         ;[C] ist Highbyte ....
D039  D1               POP  DE          ;Hole Tabellenpointer vom
D03A  12               LD   (DE),A      ;Stapel, trage Lowbyte in
D03B  79               LD   A,C         ;Tabelle ein, sieh nach,
D03C  A7               AND  A           ;ob Highbyte ungleich 0
D03D  2805             JR   Z,B1        ;Sprung, wenn [A] <> 0
D03F  13               INC  DE          ;erhoehe Tabellenpointer
D040  12               LD   (DE),A      ;trage Highbyte in Tabelle
D041  CD50D0           CALL LENUP       ;ein und erhoehe LEN

D044  13       B1:     INC  DE          ;erhoehe Tabellenpointer
D045  2B               DEC  HL          ;Hole naechstes Zeichen
D046  D7               RST  16          ;wenn (HL) = ':', dann
D047  280F             JR   Z,EXIT      ;Ausstieg, ebenso wenn
D049  FE2C             CP   2CH         ;(HL) ungleich ','
D04B  200B             JR   NZ,EXIT     ;
D04D  D7               RST  16          ;Erhoehe Programmpointer
D04E  18E0             JR   L1          ;und hole naechsten Wert.

D050  3A0BD0   LENUP:  LD   A,(LEN)     ;Lade [A] aus (LEN)
D053  3C               INC  A           ;erhoehe [A] um 1
D054  320BD0           LD   (LEN),A     ;und speichere [A]
D057  C9               RET              ;neu in (LEN)

D058  2209D0   EXIT:   LD   (PTR),HL    ;Rette  Programmpointer
D05B  2A03D0           LD   HL,(BEG)    ;Lade [HL] mit der ersten

D05E  ED5B07D0 WR1:    LD   DE,(DTA)    ;Adresse, lade [DE] mit
D062  3A0BD0           LD   A,(LEN)     ;Tabellenanfang, lade [B] mit
D065  47               LD   B,A         ;Tabellenlaenge

D066  1A       WRITE:  LD   A,(DE)      ;Transferiere (DE)
D067  77               LD   (HL),A      ;nach (HL) und
D068  D5               PUSH DE          ;rette [DE] auf den Stapel
D069  ED5B05D0         LD   DE,(END)    ;Sieh nach, ob [HL] die
D06D  E7               RST  32          ;zweite Adresse schon erreicht
D06E  D1               POP  DE          ;hat, hole [DE] vom Stapel
D06F  3006             JR   NC,AUS      ;Sprung, wenn [HL] >= [DE]
D071  13               INC  DE          ;erhoehe beide Pointer und
D072  23               INC  HL          ;fuehre den Transfer [B] mal
D073  10F1             DJNZ WRITE       ;durch
D075  18E7             JR   WR1         ;Tabellenende erreicht, also
                                        ;Tabellenpointer auf Anfang
                                        ;und weiter transferieren

D077  E1       AUS:    POP  HL          ;Hole Einsprung von 0FF9CH vom
D078  2A09D0           LD   HL,(PTR)    ;Stapel, lade [HL] mit Pro-
D07B  FB               EI               ;grammpointer, gib Inter-
D07C  C9               RET              ;rupts frei und springe
                                        ;zurueck
```

```
KEINE ERRORS
PRGRAMMBEGINN : D000
PROGRAMMENDE  : D07D
PROGRAMMLAENGE:  125 BYTES

SYMBOLE:
BEG       : D003 ,  53251        ; END       : D005 ,  53253
DTA       : D007 ,  53255        ; PTR       : D009 ,  53257
LEN       : D00B ,  53259        ; START     : D00C ,  53260
LOOP      : D02C ,  53292        ; L1        : D030 ,  53296
B1        : D044 ,  53316        ; LENUP     : D050 ,  53328
EXIT      : D058 ,  53336        ; WR1       : D05E ,  53342
WRITE     : D066 ,  53350        ; AUS       : D077 ,  53367

ENDE DER ASSEMBLIERUNG
```

```
*****************************************
*                                       *
*        Stringeingabe ohne INPUT       *
*                                       *
*****************************************
```

Ein interessantes Unterprogramm hat uns unser Klubkollege Rotschek geliefert. Mit dieser Entwicklung wird es möglich, ohne den Befehl "INPUT" Texte einzugeben. Das Programm selber weist zusätzlich eine Besonderheit auf: Der Cursor blinkt am Fernsehbildschirm!

Wie das funktioniert? Ganz einfach: Es gibt im BASIC eine Anweisung, die sich "ON INTERVAL=<Zeit> GOSUB <Ziel>" nennt. Diese Anweisung bewirkt regelmäßige Interrupts, abhängig von der verstrichenen Zeit. Die "Zeit" gibt nämlich die Zeitspanne an, die zwischen zwei Interrupts vergehen soll, und zwar in 1/50 Sekunden pro Einheit. Jeder Interrupt bewirkt einen Sprung in ein Unterprogramm beginnend mit der Zeilennumer "<Ziel>".

Und genau diese Anweisung verwendet dieses Programm, um den Cursor blinken zu lassen. Es wird nämlich jede halbe Sekunde ein Unterprogramm aufgerufen, dessen erste Zeile 5000 ist. In diesem Unterprogramm wird nun bei jedem Aufruf der Cursor umgeschaltet. Ob der Cursor ein- oder ausgeschaltet ist, entscheidet der Inhalt der Variable IV (in IV steht entweder 1 oder 0). Nun kann man aber leicht mit der Funktion "XOR" Eins zu Null und Null zu Eins ändern. Wir verknüpfen logisch die Variable IV XOR 1 (0 XOR 1 = 1, 1 XOR 1 = 0). Nachdem nun IV invertiert wurde, wird mit "LOCATE,,IV" der Cursor umgeschaltet, ohne seine Position zu verändern.

```
  10 ' Stringeingabe ohne INPUT
  20 ' Aufgebaut als direkt zu
  30 ' verwendendes Unterprogramm
  40 ' mit blinkenden CURSOR auf
  50 ' Fernsehbildschirm
  60 '
  70 ' 10.10.1984
  80 ' Rotschek Wolfgang
  90 '
 100 ' Verwendete Variablen:
 110 ' IK$    Zwischenvariable bei INKEY$
 120 ' IN$    bisher eingegebner String
 130 ' IV     Zählvariable für Intervalroutine
 140 '


1000 ON INTERVAL=25 GOSUB 5000
1010 INTERVAL ON
1020 IN$=""



2000 IK$=INKEY$
2010 IF IK$="" THEN 2000
2020 IF IK$=CHR$( 13) THEN 4000
2030 IF IK$=CHR$(  8) THEN 3000
2040 IF LEN(IN$)>50   THEN 2000

2050 IF IK$<CHR$( 32) THEN 2000
2060 IF IK$>CHR$(126) THEN 2000
2070 IN$=IN$+IK$
2080 PRINT IK$;
2090 GOTO 2000



3000 IF LEN(IN$)=0 THEN 2000
3010 IN$=LEFT$(IN$,LEN(IN$)-1)
3020 PRINT CHR$(127);
3030 GOTO 2000



4000 INTERVAL OFF
4010 LOCATE ,,0
4020 RETURN



5000 IV=IV XOR 1
5010 LOCATE ,,IV
5020 RETURN
```

Die eigentliche Eingabe erfolgt mit der Eingabeschleife mit "INKEY$" (in Zeile 2000-2010). Der Computer arbeitet solange diese Schleife ab, bis eine Taste gedrückt worden ist. Danach prüft er die Eingabe auf den ASCII-Code 13 (=<ENTER>) und auf den Code 8 (=ein Zeichen löschen mit Doppelpfeil nach links) ab (Zeile 2020-2030). Wurde <ENTER> gedrückt, so unterbricht der Computer das Blinken des Cursors, indem er die Interrupts sperrt (mit "INTERVAL OFF") und springt zum Hauptprogramm zurück. Wurde der Doppelpfeil nach links gedrückt, dann löscht der Computer in einer Programmschleife das letzte Zeichen.

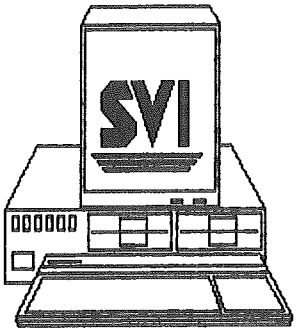
Mit der Funktion "LEFT$ (IN$,LEN (IN$)-1)" wird der Eingabestring um ein Zeichen verkürzt (IN$ enthält alle bisher eingegeben Zeichen). Danach schickt der Interpreter noch ein "DELETE"-Zeichen auf den Bildschirm, um das Zeichen, daß schon im Eingabestring gelöscht wurde, auch am Bildschirm verschwinden zu lassen.

Weiters prüft der Computer, ob die maximale Eingabelänge erreicht wurde (Zeile 2040). Wenn ja, dann wartet er solange, bis entweder ein Zeichen gelöscht wird, oder bis <ENTER> gedrückt wird.

Zuletzt werden noch alle anderen Graphik- und Steuerzeichen abgefangen (Zeile 2050-2060). Erst wenn sicher ist, daß nur ein alphanumerisches Zeichen eingetippt wurde, wird das Zeichen in den Eingabestring IN$ übernommen. Danach zeigt der Computer das Zeichen am Bildschirm an und springt wieder zur Eingabeschleife zurück, um das nächste Zeichen in Empfang zu nehmen.

Noch ein kurzes Wort zur Verwendung dieses Unterprogramms. Man kann es natürlich "solo" einspeichern, und dann mit "GOSUB1000" in das Programm vom Direktmodus aus einsteigen. Nach dem <ENTER> würde man dann automatisch wieder in den Direktmodus gelangen und der eingegebene String in IN$ liegen. Im Normalfall kann dieses Programm aber gute Dienste als Teil eines größeren Programms leisten. Man springt lediglich vom Hauptprogramm aus in die Subroutine.

Es gibt übrigens die Möglichkeit, die maximale Länge des Eingabestrings beliebig zu verändern. In der Zeile 2040 ist die maximale Stringlänge mit 50 Zeichen vordefiniert. Diesen Wert kann man nun beliebig verändern (Achtung! Bei Werten, die größer als 200 sind, braucht man ein "CLEAR"!).



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

MUSIC MENTOR

Es gibt Leute, die lernen Flöte spielen, es gibt Leute, welche die Bratsche spielen lernen und welche, die sich am Klavier einüben. Es gibt aber auch einige, die keines der drei Instrumente gelernt haben, und die trotzdem alle drei spielen können. Wieso? Sie besitzen den "MUSIC MENTOR"!

Spaß beiseite! Tatsache ist, daß Spectravideo vor kurzem dieses Spiel auf den Markt gebracht hat. Der "MUSIC MENTOR" hat drei Modes. Der erste ist der Spiel- oder Pianomode. Während man eine Klaviatur am Bildschirm sieht, die anzeigt, welche Tasten am Computer den Tasten am Klavier entsprechen, kann man musizieren. Drückt man eine Taste, so färbt sich diese am Bildschirm rot. Auf diese Weise kann man Melodien spielen, leider nur einstimmig.

Drückt man jedoch zwei Tasten auf einmal, so ertönt ein Triller (Schneller Wechsel zwischen zwei Tönen). Drückt man mehr als zwei Tasten, dann ist das Programm übrigens irritiert und spielt Töne, deren Tasten nie gedrückt wurden.

Wahlweise Klavier-, Flöten-, Bratschen- oder Gongklänge erfüllen den Raum. Für Computerenthusiasten spielt der Mentor obendrein noch im Sondermode "Regular". Dieser Klang entspricht dem normalen Sound des Spectravideo-Synthesizers ohne Klangverzerrung. Umstellen kann man die Klangarten mit den Funktionstasten, genauso wie die Rhythmen.

Der "MUSIC MENTOR" wartet mit vier verschiedenen Rythmusarten auf. Walzer, Tango, Marsch und Disko stehen zur Verfügung. Ebenso kann man den Rhythmus abschalten ("No Rhythm"). Weiters gibt es die Möglichkeit, die Oktavlage auf "High" oder "Low" zu schalten. Hier ändert sich das Manual auf dem Bildschirm auf die obere beziehungsweise untere Hälfte einer Klaviatur.

Das Tempo kann man in fünf Stufen regulieren. Die gleichen Umstellmöglichkeiten gibt es im zweiten Modus, dem "Replay"-Mode. Hier kann man zwei schon vorprogrammierten Songs ("Turkey in Straw", "Donauwalzer") oder seine eigenen Kompositionen abspielen. Die eigenen Liedchen können sich sowohl im Speicher des Computers als auch auf dem Band einer Kassette befinden.

Mit der Taste "INS/PASTE" wird gestartet, und der Computer spielt solange Wiederholungen, bis "STOP" gedrückt wird. Der dritte Mode, das Aufnehmen eines Liedchens, gestaltet sich etwas anders. Es sind zwar wieder alle vier Umstellmöglichkeiten vorhanden, doch kann man leider in diesen vier Arten, bevor man "INS" gedrückt hat, die Funktionstaste 5 nicht benutzen. Denn dann schaltet der Computer automatisch auf "Aufnehmen auf das Band".

Erst nach dem Starten der Aufnahme ist zum Beispiel der Rhythmus abschaltbar (F5= Rhythmus abschalten). Beim Aufnehmen selber spielt man einfach wie im "PIANO"-Mode. Hat man fertig gespielt, drückt man wieder "INS", um die Aufnahme zu beenden. Der Computer schaltet dann wieder auf das Hauptmenü zurück.

Das Programm ist sehr nett ausgeführt, und der Anfang ist ganz untypisch für ein Computerspiel. Es wurde nämlich versucht, gleich nach der Kennsignation für SPECTRAVIDEO, ein Landschaftsbild so realistisch wie möglich darzustellen, in dem obendrein noch Schmetterlinge fliegen. Eine Aufschrift "Welcome in Music-Land" und eine Begleitmusik (nämlich das "Turkey in Straw") vervollständigen das "Idyll" noch.

Zu bekommen ist das Programm in Modulform im Computer-Studio Wehsner. Das Modul braucht man lediglich in den Schacht rechts über dem Zehnertastaturfeld hineinstecken, bevor der Computer eingeschaltet wird. Nach dem Einschalten schaltet sich das Programm automatisch ein. Zum Herausnehmen des Moduls muß übrigens unbedingt der Computer ausgeschalten sein, ebenso beim Hineinstecken!

Das Programm kostet S 990,- inklusive Mehrwertsteuer!

```
10000 REM ****************************
10010 REM *                          *
10020 REM *  DEUTSCHER ZEICHENSATZ   *
10030 REM *                          *
10040 REM *    FÜR SPECTRAVIDEO      *
10050 REM *                          *
10060 REM *   SVI-318 UND SVI-328    *
10070 REM *                          *
10080 REM ****************************
10090 REM UNTERPROGRAMM
10100 REM DEUTSCHE ZEICHEN AUF DEN TASTEN
10110 REM RECHTS VON "P" ANGEORDNET
10120 REM PÄÜÖ ... päüß
10130 RESTORE10140
10140 DATA80,0,112,8,120,136,120,0
10150 DATA144,0,144,144,144,144,104,0
10160 DATA96,144,144,160,144,144,160,0
10170 DATA136,32,80,136,248,136,136,0
10180 DATA80,0,112,136,136,136,112,0
10190 DATA136,0,136,136,136,136,112,0
10200 FORI=0TO7:READAE:VPOKE2776+I,AE:NEXT
10210 FORI=0TO15:READAE:VPOKE2792+I,AE:NEXT
10220 FORI=0TO23:READAE:VPOKE2520+I,AE:NEXT
10230 RETURN
```

Lösung des Rätsels aus Heft 3:

Silicon-Valley
amerik. Tal mit viel Computerindustrie

Prozessor
Zentraleinheit eines Computers

Elektronik
Fachgebiet der Schwachstromtechnik

Compiler
Übersetzer von hohen Programmiersprachen

Transistor
elektronisches Bauelement (aktiv!)

Recheneinheit
unpräziser Begriff für ALU

Assembler
Programmiersprache

Video-RAM
Bildschirmspeicher

Indexregister
ein Register des Z 80A

Diskette
externes Speicherelement

Exponent
Hochzahl in einer Potenz

Operationsverstaerker
elektronisches Bauelement (aktiv!)

Die amerikanische Computerfirma ist, wie kann es anders sein,

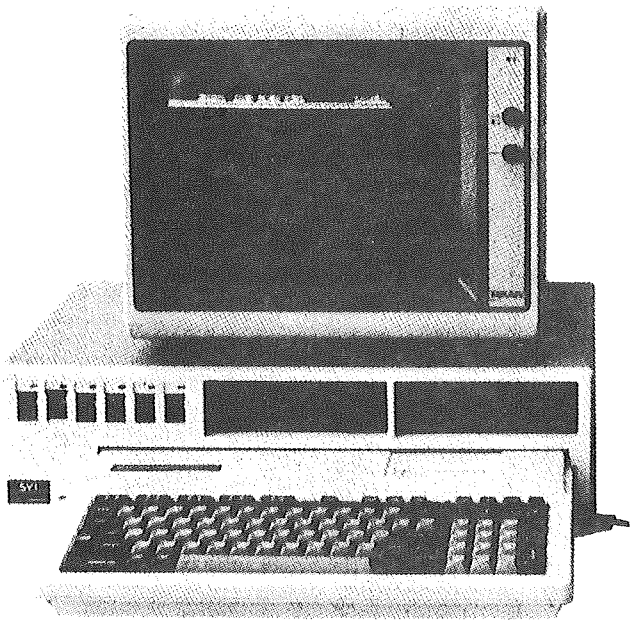
SPECTRAVIDEO !

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.



Endlich 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*, CalcStar*, DataStar*, ReportStar*).

* eingetragene Warenzeichen von Micropro International Corporation

Alle Erweiterungen und Peripheriegeräte sind prompt lieferbar.

Druckerwochen

Wir können Drucker folgender Marken derzeit besonders preisgünstig liefern:

EPSON BROTHER SILVER-REED STAR

Bitte verlangen Sie unsere Druckerpreisliste.

Auch Postversand per Nachnahme.

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)

inkl. MWSt. S 2.240,-

Weitere Programmiersprachen:

COBOL, FORTRAN, LISP, C

Computer-Studio

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

IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Philipp Ott, Wolfgang Rotschek, Heinz Schmid, Georg Wolfbauer

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

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

Erscheinungsort Wien, Verlagspostamt 1040 Wien