SVI JOURNAL

Die Zeitschrift des Spectra-Video-Club Austria

ifabo 85 — wir sind dabei

MSX

Heft 4/85

S 15.-

Lieber SVI-Journal-Leser!

Nun haben wir auch formal bestätigt, was ohnehin schon jeder Eingeweihte gewußt hat. Das SVI-Journal bietet natürlich auch allen MSX-Usern viele Informationen. Da der SVI-728 voll und der SVI-318/328 beinahe MSX-kompatibel ist, haben auch die Bediener von Philips- oder Yashica-Computersystemen (oder anderen MSX-Computern) die Möglichkeit, Interessantes über ihre Computer aus unserem Journal zu erfahren. Damit dabei der Name unserer Zeitschrift nicht irreführt, haben wir unserem Titelblatt eine zusätzliche Zeile "mit MSX-Teil" spendiert. Eingefleischte Spectravideo-Fans brauchen jetzt übrigens nicht auf die Barrikaden zu springen, wir werden weiterhin MSX nur am Beispiel Spectravideo demonstrieren.

Für alle Clubmitglieder gibt es wieder einige Neuigkeiten. Im Mai werden wir unsere erste Vereinssitzung einberufen. Nun werden Obmann, Kassier und andere wichtige Repräsentanten unseres Clubs gewählt. Bitte beachten Sie auch die Informationen darüber auf dieser Seite.

Besonders Aktive dürfen sich wieder in die Arbeit stürzen, denn wir werden auf der "Ifabo"-Messe vertreten sein. Das Computer-Studio der Wiener Firma Wehsner hat uns auf ihrem Messestand Platz spendiert. Wer die hektischen Vorbereitungen für die Hobby-Elektronik im November mitbekommen hat, wird wissen, was wir auch nun wieder brauchen. Da wären einerseits Programme oder Hardware, die die Fähigkeiten der Spectravideo-Computer schön demonstrieren. Andererseits brauchen wir wieder Clubmitglieder, die während der Messezeit unseren Stand betreuen und Informationen an die Messebesucher weitergeben. Vorallem am Vormittag werden wir wenig Personen zur Verfügung haben. Sollten Sie daher interessiert sein, unseren Club an einem Tag (kann auch länger als ein Tag sein, wenn Sie wollen) zu vertreten, dann melden Sie sich bitte im Clublokal.

Besonders großes Echo haben wir wegen des 3D-Graphik-Programms aus Heft 2/85 bekommen. Wir hatten nämlich vergessen, daß ein Mathematik-Student mit Ebenen im Raum sehr gut umgehen kann. Doch im Alltag vergißt man solche theoretischen Dinge leicht. Daher wußten viele nicht, wie man schöne Ebenen auf den Schirm zaubert, beziehungsweise wie die geheimnisvollen Fragen am Anfang des Programms zu beantworten wären. Wir haben uns nun hingesetzt und werden von Anfang an alles erklären. Natürlich können und wollen wir keinen "Darstellende Geometrie"-Kurs kreieren. Doch es ist im Interesse aller, daß auf Computern nicht nur stur programmiert wird, sondern auch etwas über die Zusammenhänge der Probleme, die in den Computer getrichtert werden, gelehrt wird.

Schon voriges Jahr rumorten immer wieder Gerüchte über eine Subroutinenbibliothek im SVI-Journal herum. Nun nehmen diese flüchtigen Gedanken Form an. Deshalb haben wir die neuesten Fakten auf diesem Gebiet zusammengefaßt und abgedruckt.

Ihr SVI-Journal-Chefredakteur Gerhard Fally!

INHALT

Die nächsten Clubtermine:

Sa, dem 4. Mai 1985, ab 17 Uhr
--- 7. Mai - 11. Mai ifabo 85 ---
Mi, dem 15. Mai 1985, ab 19 Uhr

Mo, dem 20. Mai 1985, 18.30 Uhr
Generalversammlung
siehe Einladung auf dieser Seite

Sa, dem 1. Juni 1985, ab 17 Uhr

Clubabende wie immer im Clublokal im Computer-Studio,
1040 Wien, Paniglgasse 18-20
Nichtmitglieder sind willkommen
Ende jeweils ca. 22 Uhr!

Aktivitäten an den Clubabenden:
Arbeiten an Spectravideo-Systemen, Informationsaustausch zwischen Clubmitgliedern, Gelegenheit zum Ausdrucken von Programmlistings.

Telefonische Auskünfte über den Club und seine Aktivitäten erhalten Sie unter der Telefonnummer 65 88 93.

Einladung zur Generalversammlung des Spectra Video Club Austria

Achtung, Clubmitglieder! WICHTIG!

Am Montag, dem 20. Mai 1985 wird die erste Generalversammlung abgehalten. Es werden unter anderem auch der Obmann, sein Stellvertreter, der Kassier und der Schriftführer gewählt. Clubmitglieder, die als Kandidaten fungieren möchten, mögen dies bitte im Clublokal bekanntgeben.

Ort: Cafe-Restaurant Landtmann
Dr. Karl Lueger-Ring 4
1010 WIEN
(beim Burgtheater)

Zeit: Montag, dem 20. Mai 1985
um 18.30 Uhr

Tagesordnung:
1) Wahl der Mitglieder des Vorstandes (Obmann, Obmannstellvertreter, Schriftführer, Kassier)
2) Wahl der Funktionäre
3) Wahl der Rechnungsprüfer
4) Festsetzung der Höhe der Mitgliedsbeiträge
5) Allfälliges

Erinnern Sie sich noch? In der vorigen Ausgabe lernten wir den Befehl RSET, beziehungsweise LSET, kennen. Damals konnten wir aber noch nicht alles klären. Dies holen wir nun nach und sehen uns außerdem noch weitere "I/O"-("Input/Output" = Ein-/Ausgabe)-Befehle an.

Der Befehl LSET wird zum "Stutzen" von Strings eingesetzt. Seine Arbeitsweise ist ähnlich der des PRINT USING-Befehls bei der Stringausgabe auf dem Bildschirm, nur daß bei LSET eben nicht der Bildschirm sondern eine Variable angesprochen wird.

Konkret: Wir geben eine Stringvariable an, die wir in ein bestimmtes Format bringen wollen. Vorher müssen wir jedoch schon eine weitere Puffervariable festgelegt haben, die festlegt, wie lange der formatierte String werden darf. In die Puffervariable wird dann der formatierte String eingeschrieben. Wenn die unformatierte Zeichenkette länger als die Puffervariable ist, wird der überflüssige Teil des Strings abgeschnitten. Bei LSET eliminiert der Computer den unnützen rechten Teil, bei RSET den unnützen linken Teil.

Anders verhält sich die Sache, wenn der String kürzer als gefordert ist. Dann wird die Zeichenkette bei LSET linksbündig, beziehungsweise bei RSET rechtsbündig abgelegt und der Rest des Puffers mit Blanks vollgefüllt.

Wie definiert man nun die Puffervariable? Auch das ist leicht, man muß lediglich einen String mit der gewünschten Zahl von Zeichen erzeugen. Die kürzeste und eleganteste Möglichkeit ist zweifellos das Verwenden der Funktion SPACE$. Hier muß man lediglich den Namen des Puffers und seine Länge angeben.

Zur Verdeutlichung des Gesagten wurde wieder ein kleines Programm geschrieben:

```
10 INPUT "Länge des gewünschten Puffers";A
15 PU$=SPACE$(A)
20 INPUT "Eingabe unformatierter String";B$
30 PRINT "Links- oder rechtsbündig (L/R)";
40 S$=INPUT$(1):D=(ASC(S$)AND31)+64
50 IF D=76 THEN PRINT "L":LSETPU$=B$:GOTO70
60 IF D=82 THEN PRINT "R":RSETPU$=B$ ELSE G
OTO30
70 PRINT PU$:PRINT "Taste drücken"
80 IF INPUT$(1)=CHR$(13) THEN10 ELSE GOTO15
```

Dieses Programm beinhaltet einen kleinen Programmiertrick. Normalerweise muß ein Programmierer bei einer Abfrage "L/R" mit zwei IF...THEN-Anweisungen hantieren, die mit OR auf Klein- oder Großbuchstaben testen (zum Beispiel: IF S$="L" OR S$="l"THEN...). Dies kann bei längeren Abfragen lästig werden. Mit der Formel in Zeile 40 wandelt man sämtliche Buchstaben automatisch in Großbuchstaben um. Doch es werden nicht nur Groß- und Kleinbuchstaben verwandelt, auch die CTRL-Sequenzen formt der Interpreter in die analogen Großbuchstaben um.

Leider umfaßt die Formel auch sämtliche andere ASCII-Codes. Daher darf man in der obigen Fasung zum Beispiel kein Plus oder kein Minus drücken, ohne Angst vor einer Fehlfunktion haben zu müssen. Der Vollständigkeit halber muß man in einem anderem Programm durch IF...THEN-Anweisungen die Sonderzeichen vor dem Umwandeln "retten". Mit ENTER läßt sich übrigens die Länge des Puffers neu bestimmen.

Wenden wir uns nun einem neuen Befehl zu, der die Bildschirmausgabe wesentlich vereinfacht. Bei vielen Computern kann man Texte nur mit PRINT und TAB auf dem Schirm positionieren. Dies ist natürlich mühsam. Das MSX-BASIC schafft hier Abhilfe, und zwar mit dem Befehl LOCATE. Das Prinzip ist einfach, man fügt zwei Zahlen an das Befehlswort, die als Koordinaten verstanden werden. Anhand dieser Koordinaten wird dann die Stelle errechnet, an der das erste Zeichen aufscheinen soll. Es gibt allerdings Unterschiede zwischen dem "Graphik-LOCATE" und dem LOCATE im Textmodus.

Graphik: Sowohl in der hoch- als auch in der niederauflösenden Graphik gelten die Koordinatenregeln, die auch für sämtliche andere Graphikbefehle gültig sind. Man kann also jeden einzelnen Punkt adressieren. Da es in der Graphik keinen sichtbaren Cursor gibt, hat der Befehl auch nur zwei Parameter für die Koordinaten zur Verfügung. Der dritte Wert zum Einschalten des Cursors ist nicht vorhanden.

Im Textmodus werden nicht die einzelnen Punkte adressiert, sondern die Zeichenblöcke. Daher hat man in der Waagrechte 40 (beziehungsweise 39 oder 80) Einheiten und in der Senkrechte 24 zur Verfügung (bei SCREEN,1 sind es nur 23, da die Funktionstastenleiste eine Zeile "schluckt"). Der dritte Parameter steuert den Cursor. Setzt man eine Null ein, wird der Cursor ausgeschalten, bei positiven Zahlen von 1 bis 255 prangt der Cursor wieder in seiner vollen Größe am Schirm. Alle anderen Zahlenwerte werden mit einem "Illegal function call" beantwortet.

Die Adressierung fängt in beiden Fällen (sowohl Graphik als auch Text) bei Null an.

Ein "sehr komplizierter" Befehl kommt nun an die Reihe. MOTOR ON/OFF steuert den Kassettenrekordermotor. Wenn eine Taste am Rekorder gedrückt ist (REW, FF, PLAY oder RECORD), läßt sich das Gerät per Computer stoppen (OFF) oder starten (ON). Was hier lächerlich klingt, da man mit einem Handgriff STOP oder PLAY schneller erreichen kann, als durch den Befehl MOTOR ON/OFF, ist in der Praxis gar nicht so unsinnig. Es kann vorkommen, daß man einem Lade- oder Speichervorgang nicht vollständig beiwohnt. Wenn nun während der Kommunikation mit dem Rekorder ein Vorlauf (oder ein Stop) erwünscht ist, kann man dies mit dem oben erwähnten Befehl bewerkstelligen.

Der SVI-328 ist ja bekanntlich nicht nur als Personal-Computer konzipiert, sondern hat auch vorzügliche Home-Computer-Eigenschaften. Darunter fallen auch seine drei Tongeneratoren. Doch im Gegensatz zu anderen Computern, die ihre Töne noch mit POKEs aktivieren müssen, hält das MSX-BASIC komfortablere Befehle zur Verfügung. Der wichtigste davon ist PLAY.

Der Befehl PLAY ist so reichhaltig, daß man schon von einem Makrobefehl sprechen kann,

also ein Befehl, der praktisch schon kleine Programme beinhaltet. So hat PLAY auch seine eigene Zeichensprache. Bevor wir uns nun mit den einzelnen Symbolen auseinandersetzen, werden wir Grundsätzliches klären.

Hinter dem Befehlswort PLAY können bis zu drei Zeichenketten angehängt werden, die jeweils durch Beistriche getrennt sind. Jeder String spricht dann einen der drei Tongeneratoren an. Die Zeichenketten bestehen aus den Symbolen des PLAY-Befehls und bilden so die Informationen für eine Melodie. Diese Daten werden dem Tongenerator übergeben, wobei hier ein kleiner Puffer verwendet wird. Daher kann sich der Computer schon anderen Befehlen widmen, wenn der Tongenerator noch läuft. Dies ist besonders bei Spielen vorteilhaft, die ihre Action mit Musik untermalen. Der Nachteil dieser Methode ist jedoch, daß ein PLAY-Befehl nur mit einem CTRL-STOP früher als vorgesehen gestoppt werden kann. Findige Programmierer haben jedoch schon eine Befehlserweiterung geschrieben, die PLAY OFF heißt. Sie schneidet alle nicht gewünschten Töne ab. Im SVI-Journal 1/85 wurde diese Erweiterung veröffentlicht.

Sehen wir uns nun die einzelnen Symbole an, die verwendet werden dürfen:

Die einzelnen Töne werden durch ihre Notennamen aktiviert. Will man nun ein "C" spielen, dann erweitert man seinen String um ein "C". Alle Noten werden dadurch mit den Zeichen "A-G" benannt. Der Buchstabe "B" steht für unser "H". Das "B" in unserer Notensprache wird als erniedrigtes "B" angesehen.

Man möchte natürlich auch einen Notenwert um einen Halbton erhöhen oder erniedrigen, um alle zwölf Töne pro Oktave zu erreichen. Für das schräge Doppelkreuz in der Musik darf man ein "#" oder ein "+" (Erhöhung des Tones) verwenden. Ein "-" ersetzt dementsprechend das Erniedrigungszeichen (schräges "b"). Will man die verwendete Oktave verlassen, kann man mit "O" (wie "Oktave") und einer Zahl zwischen 1 und 8 beliebig in der Notenskala wandern. Voreingestellt ist der Wert 4. Er ist der Oktave mit dem eingestrichenen "C" (C') Musik äquivalent.

Doch man kann nicht nur die Tonhöhe sondern auch die Tonlänge fast beliebig manipulieren. Durch "L" wird die Länge des Tones bestimmt. Der nachfolgende Wert darf sich im Bereich von 1 bis 64 befinden. Die Notenlänge wird dann mit 1/'Zahl' errechnet. So ist L4 eine Viertelnote, L16 eine Sechzehntel und so weiter. Daraus folgt natürlich, daß L64 die kürzeste Note ist. Der mit L eingestellte Wert gilt solange, bis das nächste "L" definiert wird. Voreingestellt ist 4.

Will man nur für eine Note die Länge ändern, so kann man die Längenzahl auch ohne L direkt hinter die Note schreiben (etwa E8). Ebenso läßt sich eine Note um die Hälfte ihrer Dauer verlängern, wenn man einen Punkt hinter sie setzt ("C." oder "C8.", jedoch nicht "C.8"). Selbstverständlich ist es auch möglich, Pausen zu generieren, indem man einfach ein R einsetzt. Hier gelten dann die gleichen Regeln wie für alle anderen Noten, mit dem einzigen Unterschied, daß "R." nicht funktioniert (wohl aber zum Beispiel "R8." oder "R").

Mit T stellt man das Tempo der Melodie ein. Hier darf man eine Zahl zwischen 32 und 255 wählen. Voreingestellt ist der Wert 120. 255 aktiviert das schnellste Tempo. Die Geschwindigkeit der Melodie muß man nach seinem eigenen Fingerspitzengefühl auswählen. Es gibt zwar eine Angabe des Tempos in Sekundenwerten im User-Manual, doch scheint diese Information nicht zu stimmen.

Natürlich sind die Möglichkeiten der Klangmanipulationen damit noch nicht erschöpft. Die Lautstärke braucht man nicht nur am Lautsprecher des Fernsehers verstellen, sondern kann sie auch softwaremäßig ändern. Dies geht mit "V" und einer Zahl zwischen Null und Fünfzehn.

Den Klangcharakter darf man mit zwei Zeichen manipulieren. Es ist zwar nicht möglich, die Form der Schwingungen direkt zu ändern, aber dafür ist die Hüllkurve regelbar. Man hat acht verschiedene Formen verfügbar, die durch S (wie "Shape", engl.: Form, Gestalt) aufgerufen werden. Die einzelnen Formen der Hüllkurve sind im Bedienermanual anschaulich dargestellt.

Um den Klang besser manipulieren zu können, darf man die Hüllkurve in ihrer Frequenz ändern (nicht den Ton!). Die im Buch angegebene Form wird dabei entweder schneller oder langsamer ausgeführt. Selbstverständlich kann das Manual nur einen Ausschnitt der Form geben. Wenn man nun diese Hüllkurve schneller macht, heißt das daher noch lange nicht, daß deshalb der Ton kürzer wird. Die Frequenz wird mit M angegeben. Hier gibt es wieder eine Umrechnungsformel, da man den Frequenzwert nicht direkt einsetzen kann.

```
n = 3579545*Frequenz/768
```

"n" ist der Wert hinter M, "Frequenz" die gewünschte Hüllkurvenfrequenz. Mit S und M kann man beschränkt den Klangcharakter formen. Es erfordert jedoch etwas Geduld und Experimentierfreudigkeit, bis man mit diesen zwei Parametern einen realistischen Klang von einem bestimmten Instrument findet. Im übrigen ist PLAY auch eher für Melodien als für Klangstudien geeignet.

Mit dem Symbol "N" und einer Zahl zwischen 0 und 84 läßt sich eine Note aktivieren. Der Clue dabei ist, daß die einzelnen Noten, zwölf pro Oktave aufsteigend sortiert sind, und durch eine Zahl aufgerufen werden können. 0 entspricht einer Pause.

Wenn man einzelne Teile im String durch Variablen ersetzen will, so kann man dies, indem man ein X und den Namen der Stringvariable in die Zeichenkette hinter dem Befehlswort PLAY einsetzt.

Es ist natürlich auch möglich, den ganzen String eines Tonkanals als Variable anzugeben. Dann braucht man keinen Zusatz, sondern setzt einfach statt der direkten Zeichenkette die Variable ein.

PLAY wird vor allem verwendet, um Melodien und Lieder einzuspeichern oder zu komponieren. Diese Anweisung sticht durch die komfortablen Symbole hervor, die es ermöglichen, innerhalb kürzester Zeit mit wenig Speicherverbrauch lange Lieder zu erzeugen. Es gibt jedoch noch einen zweiten Befehl, der auf den Tongenerator einwirkt. SOUND ist nicht so komfortabel, doch dieses Kommando spricht direkt die Register des PSG an.

Im SOUND-Befehl gibt man ähnlich dem POKE eine Adresse und ein Datenbyte an. Die Adresse muß sich im Rahmen von 0 bis 13 bewegen und spricht eines der Register des PSG an. Das Datenbyte, das in das Register geschrieben wird, darf zwischen 0 und 255 liegen.

In der nächsten Folge werden wir klären, warum der SOUND-Befehl besser zum Experimentieren von Musik geeignet ist. Außerdem lernen wir alle Musikregister und ihre Bedeutung kennen. Danach widmen wir uns dem Rest der I/O-Anweisungen (WIDTH und SWITCH).

Assemblerfortsetzungskurs Nr.: 010

Ein Prozessor muß mit seinen Bausteinen (zum Beispiel: Tastatur, Bildschirm, Drucker und so weiter) kommunizieren können. Ein Programm muß fähig sein, die Tastatur abzufragen und die gedrückte Taste wieder am Bildschirm darzustellen. Externe Bausteine kann man mit Hilfe von zwei verschiedenen Befehlen ansteuern und abfragen. Der Befehl, mit dem es möglich ist, Daten hinauszusenden, ist der OUT (engl.: hinaus)-Befehl. Zum Einlesen wird der IN (engl.: hinein)-Befehl verwendet. Die Befehle sind identisch mit den BASIC-Kommandos OUT und INP.

Jedem Baustein wird mindestens ein Port zugewiesen, über den man mit dem Baustein kommunizieren kann. Ein Port ist mit einer Adresse gleichzusetzen, oder genauer gesagt, jedem Port wird eine Adresse zugewiesen. Daher gibt es 65536 (2^16) verschiedene Ports, an die man einen Baustein anschließen könnte. Normalerweise werden jedoch nur die acht niederwertigen Bits einer Adresse verwendet und die acht höherwertigen Bits unbeachtet gelassen, da ein Computer nie sehr viele Ports benötigt.

Der Z-80 besitzt einen Befehl, mit dem ein Byte an einen Port geschickt werden kann. Mit diesem Befehl sind auch nur die acht niederwertigen Bits für einen Port ansteuerbar. Der Befehl hat folgendes Format: OUT (n),A. Damit läßt sich an das Port n der Wert des Akkumulators schicken. Um ein Byte von einem Port einzulesen, gibt es auch einen Befehl: IN (n),A. Will man nun etwa nicht nur den Akkumulator an ein Port schicken oder verwendet ein System auch die acht höherwertigen Bits des Adreßbusses, so ist ein Befehl vorhanden, der es ermöglicht, daß ins Registerpaar BC die 16-Bit Adresse des Ports geladen werden kann. Damit ist es möglich, jedes beliebige Register an das indirekt adressierte Port BC zu schicken. Der Befehl sieht folgendermaßen aus: OUT (C),r (r=Register). Sind nur die acht niederwertigen Bits eines Ports verwendet, so kann im Register B ein beliebiger Wert stehen.

Jetzt kommen wird zu den Befehlen, die den Z-80 als besonderen Prozessor auszeichnen. Das sind die sogenannten "Blocktransfer-Befehle". Mit ihnen ist es möglich, mit Hilfe eines Befehles einen Block von Daten im Speicher zu bewegen.

Der am häufigsten verwendete Befehl ist der LDIR-Befehl (wiederholtes LaDen mit InkRementieren). Der Befehl verschiebt einen definierten Speicherbereich auf einen anderen Platz im Speicher. Dabei werden alle drei Registerpaare verwendet. In HL steht die Adresse, ab der der zu verschiebende Block beginnt, in DE steht die Adresse, wo er hinbewegt werden soll, und in BC die Länge des Datenblocks. Stößt die CPU auf einen LDIR Befehl, so wird das Byte, das in (HL) steht, nach (DE) übertragen. Daraufhin werden DE und HL inkrementiert. BC dagegen dekrementiert der Prozessor und vergleicht es mit dem Wert Null. Ist BC noch nicht Null, so wird der Befehl wiederholt und dieselbe Prozedur so lange durchlaufen, bis BC gleich Null ist. Will man einen Block nur um einige Bytes nach hinten schieben, deren Differenz zwischen Anfangsadresse und Endadresse kleiner als die Länge ist, so wird der Speicherbereich mit einer Sequenz gleicher Bytes mit der Länge der Differenz vollgeschrieben und der Block zerstört. Um dies zu vermeiden, gibt es den LDDR-Befehl (wiederholtes LaDen mit DekRementieren). Er ist ident mit dem LDIR-befehl, mit dem einzigen Unterschied, daß HL das Ende des zu verschiebenden Blockes angibt und DE das Ende des verschobenen Blockes. Das heißt, daß HL und DE nicht inkrementiert sondern dekrementiert werden.

Doch nicht immer werden wiederholende Blocktransferbefehle benötigt. Daher gibt es noch die Befehle LDI (LaDen mit Inkrementieren) und LDD (LaDen mit Dekrementieren). Zum Unterschied zu LDIR und LDDR wiederholen sich die Befehle nicht, wenn BC ungleich Null ist. Es wird jedoch das Parity Flag gesetzt, wenn BC nach dem Erhöhen oder Erniedrigen ungleich Null war. Ist BC nach Durchlaufen des Befehles gleich Null, so wird das Parity Flag zurückgesetzt.

Doch es gibt noch andere Blocktransferbefehle, die sich nicht auf den Speicher beziehen, sondern zum Beispiel auf die Ports. Dabei gibt es wie LDIR den Befehl OTIR (wiederholendes Ausgeben mit inkrementieren) und OTDR (wiederholendes Ausgeben mit dekrementieren), das so ähnlich wie LDDR arbeitet. Bei beiden Befehlen muß im Register C das Port stehen, auf das ausgegeben werden soll (dabei sind nur die niederwertigen 8-Bits möglich) und C dient als Schleifenzähler. In HL steht die Adresse, bei der die Daten beginnen, die auszugeben sind. Bei OTIR wird HL inkrementiert und bei OTDR dekrementiert. Bei beiden wird B erniedrigt und auf Null überprüft. Wenn B noch nicht Null ist, wird der Befehl noch einmal wiederholt. OUTD und OUTI unterscheiden sich nur dadurch, daß der Befehl nicht wiederholt wird, wenn B ungleich Null ist. Dafür wird aber das Zero-Flag entsprechend dem Wert für B gesetzt.

Es gibt jedoch weiters noch die Blockvergleich-Befehle, mit denen es möglich ist, ein Byte in einem Block zu suchen. Hierbei existieren wiederum die Befehle CPIR (Blockvergleich mit Inkrementieren) und CPDR (Blockvergleich mit Dekrementieren). Bei CPIR steht in HL die Startadresse, bei der zum Vergleichen begonnen wird. Im Akkumulator steht das Byte, das mit dem Inhalt von (HL) verglichen wird, und BC beinhaltet die Länge des zu testenden Speicherbereichs. Nachdem der Akku mit dem Inhalt von (HL) verglichen wurde und der Vergleich negativ ausfiel, wird HL inkrementiert und BC dekrementiert, auf Null abgefragt und bei ungleich Null der Befehl nocheinmal durchlaufen. Wurde das gesuchte Byte gefunden, so ist das Zero-Flag gesetzt. Ist jedoch nichts im gesuchten Bereich gefunden worden, setzt der Prozessor das Parity-Flag zurück. Bei CPDR funktioniert es genauso, es wird jedoch bei jedem Durchlauf HL dekrementiert. Die weiters noch existierenden Befehle CPD (Vergleiche und dekrementiere) und CPI (Vergleiche und inkrementiere) haben als einzigen Unterschied zu den Blockvergleichbefehlen, daß sie nicht nocheinmal durchlaufen werden, wenn BC ungleich Null ist.

Das diesmal angeführte Programm behandelt noch nicht die Blocktransfer-Befehle, jedoch werden die OUT und IN Befehle verwendet. Das

Programm scrollt den Textbildschirminhalt in vier Richtungen, wobei dies mit den Cursortasten zu steuern ist. Es hat folgende Funktionsweise: Der gesamte Bildschirminhalt wird in den Speicher eingelesen und verschoben wieder ausgegeben. Dabei ist für den einzulesenden Bildschirminhalt ein Feld vorgesehen, das 42 Zeichen zu 26 Zeilen umfaßt (daher auch "DS 1092" am Ende des Programmes, das den Speicher für den Bildschirminhaltes reserviert). Der Bildschirminhalt wird genau in die Mitte dieses Feldes hineingeladen. Bei SCRL wird der Bildschirm wieder aufgebaut. Hierbei ist in HL die Startadresse des abgespeicherten Bildschirminhaltes anzugeben. Würde man nun etwa SPEICHER+42+1 nach HL laden, so wird der Bildschirm unverändert aufgebaut. Bei RAUF wird nach HL die Adresse des Speichers geladen, die um eine Zeile höher liegt als der eigentlich Bildschirminhalt. Daher wird oben eine Zeile zusätzlich dargestellt und so der Bildschirminhalt nach unten verschoben. Bei RUNTER wird nach HL die Adresse der zweiten Zeile geladen und daher am Bildschirm noch eine Zeile zusätzlich dargestellt. Um nach links zu scrollen, wird HL mit der Adresse der ersten Zeile plus einem Byte zusätzlich geladen. Daher wird jede Zeile um eins nach links verschoben. Das letzte Byte ist ein Leerzeichen. Beim Scrollen nach rechts, ist das genau umgekehrt.

Man kann nun vor jedem Scrollen, etwas in den "Rand" schreiben, der normalerweise leer ist. Dies wird dann in den Bildschirm gescrollt.

Doch zuletzt noch zu den OUT und IN Befehlen. Sie dienen bei meinem Programm dazu, den Bildschirmspeicher (VRAM) anzusteuern. Bei dem Port 81h kann man die Adressen an den Videochip hinausschreiben, auf die man zugreifen will. Man muß jedoch unterscheiden, ob aus dem VRAM gelesen oder ins VRAM geschrieben werden soll. Man schreibt zuerst das Lowbyte der Adresse bei 81h hinaus und dann das Highbyte. Da sowieso nur 16k VRAM möglich sind, sind die höherwertigen zwei Bits des Highbytes normalerweise auf Null. Wie gesagt muß man zwischen lesen und schreiben differenzieren. Will man lesen, so müssen die beiden Bits auf Null gesetzt sein. Beim Schreiben hingegen muß das sechste Bit auf Eins gesetzt sein. Am Beginn der Routine LE_BUFF wird die Leseadresse Null in den VDP geschickt, das heißt, es wird zweimal eine Null ans Port 81h geschickt. Am Ende wird die Schreibadresse Null hinausgeschrieben. Zuerst wird das Lowbyte ganz normal geschrieben und dann das Highbyte mit OR 40h verknüpft, was bewirkt, daß das sechste Bit gesetzt wird. Mit IN A,(84h) wird ein Byte vom VRAM eingelesen. Danach wird vom VDP automatisch die Leseadresse um Eins erhöht. So ist es nicht notwendig, dauernd eine neue Adresse hinauszuschreiben. Bei SCRL wird mit OUT (80h),A ein Byte an das VRAM geschrieben, wobei wiederum die Schreibadresse automatisch um eins erhöt wird.

In der nächsten Ausgabe wird ein Bytesuchprogramm vorgestellt, das mit den Blocksuchbefehlen arbeitet.

****

Achtung Clubmitglieder!

Endlich sind wieder handelsübliche Kassettenrekorder an den SVI anschließbar! Das Interface dazu kostet öS 325,-
Vertrieben wird es von Rafael Razim

****

```
; Dieses Programm scrollt nach allen
; vier Richtungen nach druecken einer
; Cursortaste.
; Achtung: nur im TEXT-Modus moeglich


          ORG  A000h

LOOP      CALL 403Dh      ;lese Taste ein
          CP   29         ;rechts
          JR   Z,RECHTS
          CP   28         ;links
          JR   Z,LINKS
          CP   30         ;rauf
          JR   Z,RAUF
          CP   31         ;runter .
          JR   Z,RUNTER
          CP   27         ;ESC
          RET  Z
          JR   LOOP       ;falsche Taste
                          ;gedruecked


; lade nach HL die Pointer fuer den
; Bildschirmaufbauf (=Startadresse)

RAUF      LD   HL,SPEICHER+1
          JR   SCRL

RUNTER    LD   HL,SPEICHER+84+1
          JR   SCRL

RECHTS    LD   HL,SPEICHER+42
          JR   SCRL

LINKS     LD   HL,SPEICHER+42+2


; baue den Bildschirm neu auf
; HL muss die Startadresse fuer den
; abgespeicherten Bildschirminhalt
; beinhalten.

SCRL      CALL LE_BUFF
          LD   C,24       ;Zeilen
SCRL.0    LD   B,40       ;Zeichen

SCRL.1    LD   A,(HL)     ;lese Zeichen
          INC  HL         ;aus Buffer
          OUT  (80h),A    ;und schreibe
                          ;es in den
                          ;Videospeicher
          DJNZ SCRL.1
          INC  HL
          INC  HL
          DEC  C
          JR   NZ,SCRL.0
          JR   LOOP


; lese Bildschirminhalt in den Speicher

LE_BUFF   PUSH HL
          XOR  A          ;loesche Akku
          OUT  (81h),A    ;und schreibe
                          ;Schreibeadr.
          OUT  (81h),A    ;0 an den VDP
          LD   HL,SPEICHER+42+1
          LD   C,24       ;Zeilen
LE_BUF0   LD   B,40       ;Zeichen
LE_BUF1   IN   A,(84h)    ;lese Zeichen
          LD   (HL),A     ;ein und
          INC  HL         ;speichere nach
                          ;(HL)
          DJNZ LE_BUF1
          INC  HL
          INC  HL
          DEC  C
          JR   NZ,LE_BUF0
          XOR  A          ;loesche Akku
          OUT  (81h),A    ;und schreibe
          OR   40h
          OUT  (81h),A    ;Leseadresse
                          ;0 an den VDP
          POP  HL
          RET

SPEICHER  DS   1092       ;42*26
```

3.4 Der Warmstart

Mit dem Durchlauf der in den letzten Beiträgen besprochenen zwei Programmabschnitte, dem ersten Booten und dem Kaltstart, befindet sich nun der komplette hardwareabhängige Teil des CP/M-Betriebssystems, das BIOS im Speicher. Auch die Peripherie, 80Zeichen-Karte u. a., die einen ordentlichen Betrieb im CP/M-Modus des SVI-328 gewährleisten soll, ist initialisiert.

Für das Laden des noch fehlenden hardwareunabhängigen Teils des CP/M, dem eigentlichen Systemkern, ist die Warmstart-Routine zuständig. Wie wir sehen werden, tritt die Warmstart-Routine, ganz im Gegenteil zur Kaltstart-Routine, nicht nur bei der Initialisierung, sondern auch während des Normalbetriebs, zum Beispiel durch Drücken von ^C oder nach dem Ausstieg aus einem Arbeitsprogramm, in Aktion.

Sie muß vor einer ersten bzw. einer neuen Bereitschaftsmeldung die restlichen oder fehlenden Teile des CP/M-Betriebssystems in den Speicher bringen. Darunter wird zum einen der CCP und zum zweiten das BDOS verstanden. Der CCP, als Consol-Command-Prozessor übersetzt, fungiert als CP/M-Monitor, der die über die Tastatur eingegebenen Befehle dekodiert und die entsprechenden Schritte einleitet. Das Disk-Operating-System, abgekürzt BDOS, ist in erster Linie für die Verwaltung des Massenspeichers, aber auch für das Weiterreichen von Daten an die vorhandenen Ein- und Ausgabegeräte durch das BIOS verantwortlich. Da es sich beim CP/M um ein Plattenspeicher-orientiertes Betriebssystem handelt, beinhaltet das BDOS vorwiegend Disk-Operations-Routinen.

Diese Teile, CCP und BDOS, des CP/M-Betriebssystems sind, wie schon früher erwähnt, außer den absoluten Adressen bei allen CP/M-Rechnern nahezu gleich. Nahezu deshalb, weil beim SVI-328, ausgenommen bei CP/M-Version 2.20, bei der Ausgabe des Inhaltsverzeichnis einer Diskette durch den Befehl "DIR" aus dem CCP herausgesprungen wird, um die Bildschirmausgabe dem vorhandenen Sichtgerät anzupassen (40 oder 80 Zeichen). Es ist zu vermuten, daß auch andere Computerhersteller zur Verwirklichung solcher durchaus praktischen Anpassungen, aus ihrem CCP oder BDOS ausbrechen, um die erforderlichen Unterprogramme irgendwo in einem freibleibenden Speicherbereich im BIOS unterzubringen. Beim SVI-328 sind diese Routinen bei CP/M 2.22 und bei CP/M 2.23 beginnend bei Adresse E715H untergebracht.

Während der Initialisierung, also nach dem Einschalten der Anlage, kann der Warmstartvorgang eben als Initialisierung, bei einem regulären Warmstart als Erneuerung des Betriebssystems betrachtet werden. Mehrere Anwenderprogramme überschreiben nämlich in erster Linie den CCP, der ja nur beim Arbeiten im "Direkt-Modus", solange kein CP/M-Programm aufgerufen ist, benötigt wird.

Ein Hauptgrund, den CCP zu überschreiben, ist zum einen der dadurch größere zur Verfügung stehende Speicherplatz, zum anderen das "Verstecken" von Programmen gleich unter dem BDOS, um dem Programmierer, es handelt sich hier meistens um Programm-Entwicklungshilfen, einen fast uneingeschränkten RAM-Bereich ab 100H zur Verfügung zu stellen, was gerade bei Entwicklung und Simulation von CP/M-Programmen, die ja immer auf 100H geladen werden und dort starten, sehr wichtig ist. Eine solche Programm-Entwicklungshilfe ist zum Beispiel der DDT, der mit dem Befehlssatz des 8080 Mikroprozessors arbeitet, oder der ZSID, der die Mnemonics des Z80 beherrscht.

Nach Verlassen solcher Anwenderprogramme wird mit einem Sprung auf die Adresse 0 die Warmstart-Routine angesprungen, die eine Restaurierung eventuell veränderter Betriebssystemteile durchführt. Das BIOS ist dabei natürlich ausgenommen. Im Falle einer unkontrollierten Zerstörung dieses Abschnitts, aber auch der anderen Teile, hilft nur mehr das Betätigen des Netzschalters. Wird das BIOS "kontrolliert" verändert oder soll es, was eigentlich in beiden Fällen nicht der Praxis entspricht, neu geladen werden, dann kann aus dem Inhalt der Warmstartadresse die Lage der BIOS-Sprungleiste errechnet und über diese auch ein Kaltstart ohne Ausschalten eingeleitet werden.

Obwohl bei beiden Warmstart-Vorgängen, dem Initialisierungs- und dem regulären Warmstart die gleiche Sequenz durchlaufen wird, sei auf einen kleinen, aber wichtigen Unterschied, die Ansprungspunkte betreffend, hingewiesen.

Wie im anschließenden Listing der Warmstart-Routine zu sehen, gibt es zwei mögliche Eintrittspunkte. Der eine ist "WBOOT" der andere "WSTART". Auf den letzteren bezieht sich nur die Kaltstart-Routine. Hier erübrigt sich nämlich ein eventuelles Sichern von noch nicht auf die Diskette geschriebenen Daten aus dem HOST-Buffer, da der Computer ja vorher ausgeschaltet gewesen sein wird, und deshalb keine gültigen Daten für einen Disketten-Sektor vorbereitet werden konnten. Natürlich gibt es in der am Beginn des BIOS befindlichen Sprungleiste auch einen fixen Aufruf der Kaltstart--Routine, aber hier muß der Programmierer wissen, das sich "niemand" mehr um den Zustand des HOST-Buffers kümmert.

Als HOST-Buffer wird der vom BIOS verwaltete Disketten-Sektor-Buffer bezeichnet, der im Sinne der Zugriffsoptimierung (u. a. Zahl der Zugriffe auf Diskette) zeitweise Daten, die für einen Disketten-Sektor bestimmt sind, zwischenspeichert, um diese bei einer wahrscheinlich folgenden Disk-Operation, spätestens aber vor einem Warmstart, auf die Diskette zu retten.

Außerdem, was eigentlich noch viel wichtiger ist, wird der Buffer auch für die logische Umsetzung zwischen physikalischem und logischem Sektorformat benötigt. Denn beim CP/M-Betriebssystem wird, wie wir später sehen werden, nur mit logischen Sektorgrößen von 128 Byte gearbeitet. Der physikalische Sektor beim SVI-328 hat jedoch 256 Bytes. Andere Computer haben abhängig vom gewählten Sektorformat bis zu 1024 Bytes. Die Wahl eines größeren physikalischen Sektorumfangs bedingt durch die Notwendigkeit des Zwischenspeicherns eines ganzen Sektors jedoch einen entsprechend großen HOST-Buffer. Schließlich muß ja aus diesem dann der ge-

Fortsetzung auf Seite 11

```
                        TITLE   '(INITIALISIERUNGS)-WARMSTART-SEQUENZ FÜR DAS CP/M-BETRIEBSSYSTEM'

                                .Z80
0000'                           ASEG
                                ORG     0EFEDH

ED1A                    LOKWRH  EQU     0ED1AH          ;überprüfungsroutine für HOST-Buffer
E603                    TWBOOT  EQU     0E603H          ;Warmstarteinsprung in der BIOS-Tabelle
0000                    WBOOTP  EQU     0               ;Warmstartzeiger
0001                    WBOOTA  EQU     1               ;Adressenteil des Zeigers (Warmstart)
D806                    BDOS    EQU     0D806H          ;BDOS-Einsprung
0005                    BDOSP   EQU     5               ;BDOS-zeiger
0006                    BDOSAD  EQU     6               ;Adressenteil des Zeigers (BDOS)
EC9D                    SETDMA  EQU     0EC9DH          ;Routine für das Setzen der DMA-Adresse
                                                        ;(BDOS-Disk-Buffer)
F319                    HSTACT  EQU     0F319H          ;HOST-Aktiv-Flag (BIOS-Disk-Buffer)
F318                    HSTWRT  EQU     0F318H          ;HOST-Write-Flag (HOST schon auf Diskette)
0004                    CDISK   EQU     4               ;Aktuelles Laufwerk und Benutzernummer
D000                    CCP     EQU     0D000H          ;Einsprung und Beginn des CCP
F312                    DESTXS  EQU     0F312H          ;Aktuelle RAM-Anfangsadresse für das Schreiben
                                                        ;oder Lesen eines Sektors
F314                    FDCSEC  EQU     0F314H          ;Aktuelle Sektornummer für BIOS-Zugriff
ED81                    RDORWR  EQU     0ED81H          ;Schreib bzw. Leseroutine (abhängig vom C-Flag)
F315                    FDCTRK  EQU     0F315H          ;Aktuelle Spurnummer für BIOS-Zugriff

EFED                    WBOOT:                          ;Ansprung für REGULÄREN Warmstart
EFED    CD ED1A                 CALL    LOKWRH          ;überprüfe, ob Rettung des HOST-Buffers
                                                        ;auf Diskette notwendig

EFF0                    WSTART:                         ;Ansprung für INITIALISIERUNGS-Warmstart
EFF0    3E C3                   LD      A,0C3H          ;Lade Akku mit "JP"-Instruktion
EFF2    21 E603                 LD      HL,TWBOOT       ;Aktualisiere WARMSTART-Zeiger in Adresse 0 mit der
EFF5    32 0000                 LD      (WBOOTP),A      ;entsprechenden Adresse in der BIOS-Sprungtabelle
EFF8    22 0001                 LD      (WBOOTA),HL

EFFB    21 D806                 LD      HL,BDOS         ;Aktualisiere BDOS-Zeiger in Adresse 5 mit dem zur
EFFE    32 0005                 LD      (BDOSP),A       ;Auswertung von Funktionsaufrufen entsprechenden
F001    22 0006                 LD      (BDOSAD),HL     ;Eintrittspunkt ins BDOS

F004    31 0080                 LD      SP,80H          ;Setzte Stapelbeginn auf 80H (wird hinuntergezählt!)

F007    CD F024                 CALL    READXS          ;Lese über READXS mehrere Sektoren wie folgt:
F00A    D000                    DW      0D000H                  ;Erste Einleseadresse
F00C    0D                      DB      0DH                     ;Erster zu lesender Sektor
F00D    01                      DB      01H                     ;Erste zu lesende Spur
F00E    00                      DB      00H                     ;Laufwerksnummer
F00F    16                      DB      16H                     ;Anzahl der zu lesenden Sektoren

F010    01 0080                 LD      BC,80H          ;Initialisiere über die Routine SETDMA die DMA-Adresse
F013    CD EC9D                 CALL    SETDMA          ;auf 80H (wird hinaufgezählt!)

F016    AF                      XOR     A
F017    32 F319                 LD      (HSTACT),A      ;Zeige dem Betriebssystem an, daß noch kein Zugriff
                                                        ;auf den "HOST"-Buffer erfolgt ist; HOST = nicht aktiv
F01A    32 F318                 LD      (HSTWRT),A      ;Zeige dem System an, daß der HOST-Buffer nicht auf
                                                        ;Diskette gerettet werden muß, also leer ist.

F01D    3A 0004                 LD      A,(CDISK)       ;übergib, mit aktiver Laufwerks- und Benutzernummer
F020    4F                      LD      C,A             ;in Register C, die weitere Kontrolle an
F021    C3 D000                 JP      CCP             ;den CCP = Consol-Command-Prozessor

F024                    READXS:                         ;Einsprung fürs LESEN mehrerer Sektoren
F024    E6                      DB      0E6H ;="AND"    ;Verhindere durch Ausführung des "2-Byte!"-Befehls
                                                        ;"AND 37H"(37H="SCF") das Setzen des Carry-Flags
                                                        ;und leite damit das LESEN von Sektoren ein

F025                    WRITXS:                         ;Einsprung fürs BESCHREIBEN mehrerer Sektoren
F025    37                      SCF                     ;Ermögliche das SCHREIBEN durch Setzen des Carry-Flags


F026    E3                      EX      (SP),HL         ;Lade HL mit der am Stapel liegenden "Rücksprungadresse"
F027    5E                      LD      E,(HL)          ;Lese, beginnend mit der als "Rücksprungadresse" auf den
F028    23                      INC     HL              ;Stapel gelegten Speicherzelle, zwei Bytes und lege sie
F029    56                      LD      D,(HL)          ;als erste RAM-Schreib- oder Leseadresse in DESTXS ab
F02A    ED 53 F312              LD      (DESTXS),DE
```

```
F02E  23                    INC     HL

F02F  11 F314               LD      DE,FDCSEC       ;Transferiere die drei folgenden Bytes als Sektor, Spur und
F032  01 0003               LD      BC,3            ;Laufwerknummer in die für die BIOS-Lese- bzw. Schreib-
F035  ED B0                 LDIR                    ;Routine vorgesehenen Speicherzellen

F037  46                    LD      B,(HL)          ;Lege die unmittelbar nach den Parametern folgende Adresse
F038  23                    INC     HL              ;auf den Stapel, und somit als neue Rücksprungadresse fest
F039  E3                    EX      (SP),HL

F03A                  NXTSEC:
F03A  C5                    PUSH    BC              ;Rette Sektorzähler
F03B  F5                    PUSH    AF              ;Rette Akku, insbesondere das Carry-Flag
F03C  2A F312               LD      HL,(DESTXS)     ;HL = Aktuelle RAM-adresse für RDDRWR

F03F  CD ED81               CALL    RDDRWR          ;Führe über die Routine RDDRWR in Abhängigkeit vom Zustand
                                                    ;Carry's das Lesen oder Beschreiben eines Sektors durch

F042  CD F04A               CALL    SETXSP          ;Aktualisiere Parameter für weitere Zugriffe

F045  F1                    POP     AF              ;Restauriere Akku
F046  C1                    POP     BC              ;Restauriere Sektorzähler

F047  10 F1                 DJNZ    NXTSEC          ;Führe solange weitere Sektoroperationen durch
                                                    ;bis Sektorzähler gleich Null ist

F049  C9                    RET                     ;Springe ins Hauptprogramm zurück

F04A                  SETXSP:

F04A  3A F315               LD      A,(FDCTRK)      ;Lese Aktuelle Spurnummer und setze für späteres Testen
F04D  A7                    AND     A               ;bei gewählter Spur 0 die Z-Flagge

F04E  11 0100               LD      DE,256D         ;Wähle vorerst Sektorgröße von 256 Bytes (Double Density)
F051  0E 12                 LD      C,18D           ;Lade zur Überprüfung eines eventuellen Spurwechsels
                                                    ;C mit der bei Double-Density maximalen Sektoranzahl + 1

F053  20 04                 JR      NZ,SC256B       ;überspringe bei Spur > 0 Einstellung für Single-Density

F055  11 0080               LD      DE,80H          ;Setze für Single-Density Sektorgröße auf 128 Bytes
F058  0C                    INC     C               ;Erhöhe für Single-Density die maximale Sektorzahl um 1

F059                  SC256B:
F059  2A F312               LD      HL,(DESTXS)     ;Lese letzte Sektor-Einleseadresse
F05C  19                    ADD     HL,DE           ;und erhöhe sie entsprechend der folgenden Sektorgröße
F05D  22 F312               LD      (DESTXS),HL     ;Sichere diese für weitere Operation

F060  21 F314               LD      HL,FDCSEC       ;Erhöhe aktuelle Sektornummer für folgenden Zugriff
F063  34                    INC     (HL)

F064  7E                    LD      A,(HL)          ;überprüfe ob Sektornummer die maximale Sektoranzahl
F065  B9                    CP      C               ;überschreitet
F066  D8                    RET     C               ;wenn nicht ==> Springe sofort nach NXTSEC zurück

F067  36 01                 LD      (HL),1          ;wenn ja ==> Setze Sektornummer auf 1 (Spurbeginn)

F069  23                    INC     HL              ;Erhöhe die der Sektornummer folgende Tracknummer
F06A  34                    INC     (HL)

F06B  C9                    RET                     ;Springe nach NSTSEC zurück

                      ;Abkürzungen: CCP             Console-Command-Prozessor (CP/M-Monitor)
                      ;             DMA             Direct-Memory-Adress (BDOS-Disk-Buffer)
                      ;             HOST(-Buffer)   BIOS-Disk-Buffer

                            END                     ;Ende der Quelldatei
```

No Fatal error(s)

In dieser Folge soll zuerst der IF-THEN-ELSE-Befehl fertig besprochen werden. Danach stellen wir eine weitere Möglichkeit vor, ein Programm zu gliedern, den FUNCTION-Block.

In der letzten Folge habe ich schon die Syntax von IF-THEN erklärt. Dazu kommt noch der ELSE-Zweig. Der vollständige Befehl sieht dann so aus:

```
IF bedingung=TRUE THEN
                  ...
               ELSE
                ...
```

Das Programm "Logarithmus" zeigt die Anwendung von IF-THEN-ELSE. In PASCAL gibt es die Standardfunktion "LN", um den natürlichen Logarithmus zu berechnen. Diese mathematische Funktion liefert den Wert, mit dem die Euler'sche Zahl (= 2.718281828...) potenziert werden muß, um den Funktionswert zu erhalten. Bei der Beziehung : x=e^y ist e die Euler'sche Zahl, und y der natürliche Logarithmus der Zahl x. Dabei muß man darauf achten, daß x immer größer als Null ist, da e positiv ist. Es kann nämlich nie durch Potenzieren einer postitiven Zahl ein negatives Ergebnis herauskommen. So muß man die negativen Fälle ausschließen. Das geschieht mit einer IF-Anweisung.

Aber nun zum Aufbau des Programms. Zuerst werden wie üblich die Variablen deklariert. Dabei dient die Variable "eingabe" für die eingetippte Zahl, die BOOLEAN-Variablen "ende" und "falsch" für die logischen Entscheidungen. Der logische Aufbau des Programms ist einfach. Der Hauptblock besteht im wesentlichen aus einer REPEAT-UNTIL-Schleife, die von der Bedingung "ende=TRUE" abhängt. "ende" wird durch eine IF-Anweisung dann TRUE, wenn eine "0" eingegeben wird. Dadurch wird bei Eingabe von "0" das Programm beendet. Innerhalb der REPEAT-UNTIL-Schleife sind zwei IF-Anweisungen ineinander geschachtelt. Es ist zu beachten, daß der THEN-Zweig nicht mit einem ";" abgeschlossen werden darf, wenn noch ein ELSE-Zweig folgt. Sonst nimmt der Compiler nämlich an, die IF-Anweisung sei beendet, und das nachfolgende ELSE steht dann plötzlich ganz alleine da. Ein Fehler wird erzeugt.

Die IF-Anweisung fängt Fehler auf, und setzt die Variable "fehler" auf TRUE. In einer anderen IF-Anweisung wird dann, wenn "fehler" TRUE ist, die Prozedur "meldefehler" aufgerufen. Das Programm zeigt, wie man in Verbindung von Schleifen und einem IF-Befehl ohne GOTO-Befehl Verzweigungen aufbauen kann. In BASIC wäre das nicht möglich.

Mit dem IF-Befehl sind alle wichtigen Verzweigungs- und Entscheidungsbefehle in PASCAL erklärt. Bei den Befehlen zur Blockstruktur bin ich noch eine Möglichkeit schuldig, den "FUNCTION"-Block. Dieser Befehl ist entfernt vergleichbar mit dem Befehl "DEF FN" in BASIC. Mit ihm können Funktionen definiert werden, die dann ganz normal wie eine eingebaute Funktion (SIN, COS, TAN,....) aufgerufen werden können. Im Gegensatz zu BASIC jedoch können ganze Befehlsgruppen die Funktion darstellen (in BASIC nur eine Zeile mit mathematischen Operationen). Die allgemeine Syntax lautet:

```
FUNCTION name (parameterliste):typ;
VAR lokale Variable

  BEGIN
    .
    .
  END;
```

Der "FUNCTION"-Block ähnelt also syntaktisch dem "PROCEDURE"-Block. Aber anders als eine Prozedur, die wie eine Befehlserweiterung wirkt und aufgerufen wird, ist "FUNCTION" eine neue Funktion. Deswegen muß auch ein Variablentyp definiert werden. Es muß vereinbart werden, ob das Ergebnis BOOLEAN oder REAL oder vielleicht STRING sein soll. Das Programm "kubus" verdeutlicht die Wirkung von "FUNCTION". In PASCAL gibt es nur die Quadratfunktion als eingebaute Funktion. Der Kubus (x^3) oder höhere Potenzen sind nicht vorhanden. Dieses Programm definiert eine Funktion "kbs" die den Kubus eines Wertes berechnet.

Der Aufbau des Programmes ist sehr einfach. Es werden zuerst eine REAL- und eine INTEGER- Variable vereinbart, um zu zeigen, daß eine REAL-Funktion auch INTEGER-Zahlen verarbeiten kann. Dann wird die REAL-Funktion "kbs" definiert, die dann im Hauptprogramm zweimal aufgerufen wird. Die Funktion "kbs" wird genauso wie die Funktion "LN" im ersten Program verwendet. So können mit dem FUNCTION-Block leistungsfähige Strukturen programmiert werden.

Nachdem ich auch den Befehl "FUNCTION" definiert habe, will ich nun die übrigen eingebauten numerischen Funktionen erklären und aufzählen.

ABS:	liefert den Absolutwert, eine INTEGER-oder REAL-Variable
ARCTAN:	liefert den Arkustangens im Bogenmaß
COS:	liefert den Kosinus
EXP:	liefert die Zahl e^x (Exponentialfunktion)
FRAC:	liefert den Kommateil einer REAL-Zahl
INT:	liefert den ganzzahligen Teil einer Zahl
LN:	natürlicher Logarithmus
SIN:	liefert den Sinus
SQR:	liefert das Quadrat einer Zahl
SQRT:	liefert die Wurzel einer Zahl

Wie man sieht, gibt es in PASCAL nicht allzuviele Funktionen. PASCAL wurde nicht für rein mathematische Verwendungen entwikkelt (wie FORTRAN), dafür können aber neue Funktionen sehr leicht definiert werden.

Nachdem ich die mathematischen Fähigkeiten von PASCAL beleuchtet habe, soll nun die STRING-Verarbeitung näher betrachtet werden. In PASCAL muß die maximale Länge eines Strings definiert werden. Dabei darf ein String auf keinen Fall länger als 255 Zeichen lang sein. Die allgemeine Syntax für die Vereinbarung einer Zeichenkette lautet so:

```
VAR name:STRING(.länge.);
```

Statt (. und .) können wie immer auch eckige Klammern verwendet werden. Für die Verarbeitung von Strings stellt uns PASCAL eine

Reihe von Funktionen und Prozeduren zur Verfügung:

DELETE (name,pos,anz)

Diese Prozedur löscht den String "name" von der Position "pos" bis "pos+anz". Ein Beispiel:

```
str:='1234567890';
DELETE(str,2,3);
```

Dieser Befehl weist dem Namen "str" den String "1567890" zu.

INSERT (ins,name,anz)

ist das Gegenteil zu DELETE. Ab dem Zeichen mit der Nummer "anz" wird im String "name" der String "ins" eingefügt. Die Befehlsfolge:

```
str:='1567890';
ins:='234';
INSERT(ins,str,2);
```

verwandelt "str" in "1234567890".

```
PROGRAM kubus;

VAR eingabe1:REAL;
    eingabe2:INTEGER;

FUNCTION kbs(wert:REAL):REAL;

  BEGIN
    kbs:=wert*wert*wert;
  END;

BEGIN
  CLRSCR;
  WRITE('Bitte eine REAL-Zahl eingeben ');
  READLN(eingabe1);
  WRITELN('');
  WRITE(eingabe1:5:3,'^3= ');
  WRITELN(kbs(eingabe1):5:3);
  WRITELN('');
  WRITELN('');
  WRITE('INTEGER-Zahl eingeben ');
  READLN(eingabe2);
  WRITELN('');
  WRITE(eingabe2,'^3= ');
  WRITELN(kbs(eingabe2):10:0);
END.
```

In der nächsten Folge werde ich dann weiter die Stringfunktionen beschreiben, und sie durch Beispiele verdeutlichen. Außerdem will ich beginnen, einen bildschirmorientierten Editor zu schreiben, der uns die nächsten Folgen beschäftigen wird.

RAFAEL RAZIM

```
PROGRAM logarithmus;

VAR eingabe:REAL;
    fehler,ende:BOOLEAN;

PROCEDURE meldefehler;

  BEGIN
    WRITELN('');
    WRITELN('FEHLER !!!!');
    WRITELN('Eingabe   0 unzulässig');
    WRITELN('Bitte Eingabe wiederholen');
    WRITELN('');
    fehler:=FALSE;
  END;

BEGIN
  CLRSCR;
  REPEAT
    WRITE('Zahl eingeben (0=Ende) : ');
    READLN(eingabe);
    IF eingabe = 0
      THEN ende:=TRUE
      ELSE
        BEGIN
          IF eingabe   0
            THEN fehler:=TRUE
            ELSE
              BEGIN
                WRITELN('');
                WRITE('LN(',eingabe:5:5);
                WRITE(')=',LN(eingabe):5:5);
                WRITELN('');
                WRITELN('');
                fehler:=FALSE;
                ende:=FALSE;
              END;
        END;
    IF fehler THEN meldefehler;
  UNTIL ende;
  WRITELN('');
  WRITELN('Ende des Programmes');
END.
```

Das CP/M-Betriebssystem am SVI-328
Fortsetzung von Seite 7

wünschte logische Sektor (128 Bytes) dem BDOS übergeben werden oder umgekehrt vom BDOS in den Buffer übergeben werden.

Diese Fakten erklären beinahe von selbst, warum es den ersteren Eintrittspunkt in die Warmstart-Sequenz gibt. Hier wird vor dem eigentlichen Warmstart, der meistens ein Ausstieg aus einem Programm mit Diskettenzugriffen war, über die Routine LOKWRH überprüft, ob nicht sowieso schon ein vorangegangener Diskettenzugriff den HOST-Buffer geleert, also auf die Diskette geschrieben hat. Ist das nicht der Fall, ein Flag HSTWRT gibt Auskunft darüber, wird das Sichern des Buffers auf Diskette nachgeholt und anschließend der Warmstart durchgeführt.

Der Unterprogrammaufruf READXS bzw. WRITXS leitet das Lesen von mehreren Sektoren in einen zusammenhängenden Speicherbereich, oder eben umgekehrt das Schreiben aus einem solchem auf Diskette ein. Nicht ohne Grund wird dieser Programmkomplex, neben der Warmstart-Routine, hier allerdings nur READXS, auch vom Sysgen-Programm aufgerufen. Um Kompatibilität innerhalb unserer bis jetzt drei Versionen zu haben, ist jeweils eine Ansprungadresse, beginnend bei Adresse E654H für READXS bzw. WRITXS, also nach der offiziellen BIOS-Sprungleiste angeordnet.

Im voraus sei schon jetzt darauf hingewiesen, daß diese Unterprogrammaufrufe nicht zur Standard-BIOS-Sprungleiste gehören, sondern eine echte gerätespezifische Erweiterung darstellen, und bei einem anderen CP/M-Rechner wahrscheinlich nicht zu finden sind.

Nach dem wir jetzt alle am Laden des CP/M-Betriebssystem beteiligten Programmabschnitte besprochen haben, werden wir uns in den folgenden Beiträgen mit den uns zur Verfügung stehenden Sprungtabellen, Unterprogrammaufrufen und Betriebssystem-Routinen beschäftigen.

Die Reihenfolge, das Laden des Betriebssystems vor allem anderen zu behandeln, liegt in der Absicht den wirklichen Ablauf unmittelbar nach dem Einschalten des Computers nachzuvollziehen. Schließlich ist es ja auch interessant, wie denn eigentlich das CP/M, mit dem man ja dann die ganze Zeit arbeitet, in den Speicher kommt.

```
*****************************************
*                                       *
*       MC-HEXEDITOR für SVI-3x8        *
*                                       *
*****************************************
```

In der vorigen Ausgabe des SVI-Journals (Heft 3/85) hat der Chefredakteur Ihnen die Anwendungsmöglichkeiten eines Hexeditors vorgestellt. Da sämtliche MC-Programme als Hexdump abgedruckt werden, ist so ein Programm zur Eingabe von Hexdumps erstklassig geeignet. Für die Programme, die in den Zeitschriften abgedruckt werden, reicht die BASIC-Version wahrscheinlich voll aus, aber falls es unter den werten Lesern einige gibt, die einen Hexeditor in einen größeren, fähigeren Maschinencodemonitor einbauen wollen, könnten sie den von mir entwickelten Hexeditor verwenden. Der Hexeditor ist so geschrieben, daß er leicht angepaßt werden kann (z.B. 80 oder 40 Zeichen). Er ist mit seinen knapp 780 Bytes auch relativ groß ausgefallen, aber in diesen 780 Bytes sind einige äußerst praktische Routinen enthalten.

Der Hexeditor verwendet die PGE 21 (zweite Bank, untere Hälfte) und die PGE 02 (normales BASIC-RAM), weshalb er es auch gestattet, Programme in der PGE 21 zu entwickeln. Weiters ermöglicht er ein äußerst simples Eingeben von Texten direkt in das RAM des Computers (ein kleiner Texteditor ist vorhanden). Weiters gestattet er ein Seitenscrolling (auf und ab) sowie das Suchen nach einem Byte. Alle Funktionen sind sowohl im Hexeditor als auch im Texteditor verfügbar.

Das Darstellungsformat auf dem Schirm ist gleich dem der bisher erschienen Hexdumps. Ganz links haben Sie eine vierstellige Hexzahl als Adresse, danach folgen je nach Bildschirmbreite 8 oder 16 zweistellige Hexzahlen und rechts haben Sie die ASCII-Zeichen, die den Hexzahlen entsprechen. Bytes mit Inhalt zwischen 32 und 126 bzw. 160 und 215 werden durch die ASCII-Zeichen dargestellt, alle anderen werden durch einen Punkt (".") dargestellt, da es für sie keine eigenen Zeichen gibt.

Nachdem Sie den Hexeditor geladen haben, müssen Sie ihn mit DEFUSR=&HC800 definieren. Das Argument, das Sie beim Aufruf im USR mitübergeben, ist gleichzeitig die Startadresse, bei der der Hexeditor zu "dumpen" beginnt. Nachdem Sie ENTER gedrückt haben, erscheint die erste Zeile. Links sehen Sie die Adresse (nächstkleineres Vielfaches von 8 oder 16), in der Mitte die Hexzahlen und rechts die ASCII-Zeichen. Dann werden Sie irgendwo (bei der Adresse, die Sie eingegeben haben) den Cursor finden. Er wird auf dem Highnibble des Bytes stehen und auf Ihre Eingabe warten. Mit CTRL-STOP können Sie jederzeit aus dem Editor aussteigen (auch wenn er gerade nicht auf eine Eingabe wartet). Sie haben nun folgende Möglichkeiten:

1.) Sie geben Hexzahlen ein. Das geschieht durch Drücken der Zifferntasten und der Tasten 'A' bis 'F'. Sie können somit die Hexdumps eingeben.

2.) Sie können Texte eingeben. Dazu müssen Sie erst aus dem Hexeditor aussteigen und in den Texteditor umsteigen. Dies und auch der Rückweg geschieht über CTRL-T. Jedesmal, wenn Sie CTRL-T eingeben, schaltet der Computer auf den anderen Modus um. Der Cursor springt dabei immer auf die entsprechende Position im anderen Editor. Wenn Sie also im Hexeditor auf der Zahl 43H stehen, so wechseln Sie mit CTRL-T auf das 'C' im Texteditor und mit einem weiteren CTRL-T wieder zurück. Im Texteditor sind alle Ziffern und Buchstaben zugelassen. Spezialcodes und CTRL-Sequenzen sind nur beschränkt erlaubt. Jene Spezialcodes, die der Editor benötigt, sind im Folgendem aufgezählt.

CTRL-A Sucht das Byte, auf dem Sie sich mit dem Cursor befinden in fallenden Adressen im RAM. Wenn der Computer das Byte findet, bleibt er darauf stehen.

CTRL-M oder ENTER, springt an den Anfang der nächsten Zeile.

CTRL-S wie CTRL-A, nur wird in aufsteigenden Adressen gesucht.

CTRL-W bringt den Cursor um eine Seite (genauer 20 Zeilen) nach oben.

CTRL-Z wie CTRL-W, nur nach unten.

Die Cursortasten und deren Codes sind ebenfalls nicht verfügbar, weil sie vom Editor verwendet werden.

Zu den CTRL-Sequenzen und Cursortasten ist noch hinzuzufügen, daß sie sowohl im Hexeditor als auch im Texteditor gelten.

Das Hexdump des Hexeditors ist 780 Bytes lang und hier abgedruckt. Da man aus dem Hexdump nicht die Struktur des Programmes erkennen kann, möchte ich einige Systemvariablen des Programm aufführen und deren Bedeutung zeigen.

Der Einsprung bei &HC800 erfolgt bei einem JP-Befehl, der 3 Bytes belegt. Danach, ab &HC803, folgen 18 Bytes, die für den Editor von großer Bedeutung sind. Das IX-Register des Z-80 zeigt, während der Hexeditor läuft, immer auf &HC803, die Basis jener Tabelle. Im Programm selbst greift der Editor mit '(IX+OF)' auf diese Tabelle zurück. 'OF' ist das Offset und adressiert die Systemvariablen. Beginnend bei &HC803 haben die Bytes folgende Bedeutung:

00: Position, enthält die aktuelle Position des Cursors. Sie variiert von 0 bis 7 oder von 0 bis 15 (je nach Dumpbreite).

01: enthält die Anzahl der Speicherzellen, die angezeigt werden: 8 für 40 Zeichen und 16 für 80 Zeichen Bildschirmbreite. Voreingestellt ist 8, mit POKE &HC804,16 haben Sie 16 Hexzahlen und ASCII-Codes angezeigt.

02,03,04: zeigen an, ab welcher Cursorposition in der Zeile die Hexadresse, die Hexzahlen und die ASCII-Codes angezeigt werden sollen. Für 40 Zeichen voreingestellt sind die Werte 1,7 und 31. Für 80 Zeichen sollten Sie POKE &HC807,62 eingeben, damit sich Hexdump und ASCII-Codes nicht überlagern.

05: Dieses Byte sagt dem Editor, ob er Hexzahlen oder Text editieren soll.

06: Dieses Byte sagt dem Hexeditor, ob er sich im Highnibble oder im Lownibble eines Bytes befindet.

07,08: Diese Systemvariablen braucht der Hexeditor zum Rechnen.

09: Diese Variable enthält den letzten Tastendruck.

10 bis 18: Sie enthalten in folgender Reihenfolge die Codes für Cursor HINAUF, RECHTS, HINUNTER und LINKS, danach folgen Suche HINAUF, HINUNTER, dann ENTER und Seite HINAUF, HINUNTER. Durch "umpoken" der Werte können Sie die Tastenbelegungen ändern.

Leider ist das Assemblerlisting des Hexeditors mit 380 Zeilen zu lang um es abzudrucken. Wenn Sie es haben wollen, können Sie es sich an einem Clubabend holen (auf Diskette, Kassette, Druckpapier oder eventuell photokopiert). Ich werde mich bemühen, es verständlich zu dokumentieren.

```
C800: C3 18 C8 00 08 01 07 1F ........
C808: 00 00 00 00 00 1E 1C 1F ........
C810: 1D 01 13 0D 17 1A 00 00 ........
C818: 2A 25 F9 E5 3A 04 C8 EE *%..:...
C820: FF 3C A5 6F 22 16 C8 E1 .<.o"...
C828: 3A 04 C8 3D A5 DD 21 03 :..=..!.
C830: C8 DD 77 00 DD 36 05 00 ..w..6..
C838: DD 36 06 00 DD 36 07 00 .6...6..
C840: DD 36 08 00 DD 36 09 00 .6...6..
C848: CD 4C C8 C9 DD 21 03 C8 .L...!..
C850: CD 9A CA CD B7 C8 CD E2 ........
C858: C8 18 F5 DD 7E 02 CD 00 ....~...
C860: CB 3A 17 C8 CD EC CA 3A .:.....:
C868: 16 C8 CD EC CA 3E 3A C3 .....>:.
C870: 18 00 DD 7E 03 CD 00 CB ...~....
C878: DD 46 01 2A 16 C8 CD C0 .F.*....
C880: CA CD EC CA 23 3E 20 DF ....#> .
C888: 10 F4 C9 DD 7E 04 CD 00 ....~...
C890: CB DD 46 01 2A 16 C8 CD ..F.*...
C898: C0 CA FE 20 30 04 3E 2E ... 0.>.
C8A0: 18 10 FE 7F 28 F8 FE D8 ....(...
C8A8: 30 F4 FE 80 38 04 FE A0 0...8...
C8B0: 38 EC DF 23 10 E1 C9 DD 8..#....
C8B8: CB 05 46 0E 03 DD 66 03 ..F...f.
C8C0: 28 05 0D 0D DD 66 04 DD (....f..
C8C8: 46 00 78 A7 28 06 7C 81 F.x.(.|.
C8D0: 67 05 18 F6 DD CB 06 46 g......F
C8D8: 28 01 24 3A 03 FA 6F C3 (.$:..o.
C8E0: 3E 39 CD 57 CA CD 3D 40 >9.W..=@
C8E8: DD 77 09 CD 95 64 CD 4C .w...d.L
C8F0: CA DD 7E 09 FE 14 20 0D ..~... .
C8F8: DD 7E 05 EE 01 DD 77 05 .~....w.
C900: DD 36 06 00 C9 DD CB 05 .6......
C908: 46 C2 53 C9 CD 62 C9 DD F.S..b..
C910: 7E 09 FE 30 38 CC FE 3A ~..08..:
C918: 38 0D CD 0C 17 FE 41 38 8.....A8
C920: C1 FE 47 30 BD D6 07 D6 ..G0....
C928: 30 DD 77 07 DD 36 08 F0 0.w..6..
C930: DD CB 06 46 20 0B DD 36 ...F ..6
C938: 08 0F 17 17 17 17 DD 77 .......w
C940: 07 CD A7 CA DD A6 08 DD ........
C948: B6 07 DD 77 07 CD AE CA ...w....
C950: C3 C8 C9 CD 62 C9 DD 7E ....b..~
C958: 09 DD 77 07 DD 36 08 00 ..w..6..
C960: 18 DF E1 DD 7E 09 DD BE ....~...
C968: 0A CA 9E C9 DD BE 0B CA ........
C970: C8 C9 DD BE 0C CA 0C CA ........
C978: DD BE 0D CA 2A CA DD BE ....*...
C980: 0E CA 62 CA DD BE 0F CA ..b.....
C988: 7E CA DD BE 10 CA 0E CA ~.......
C990: DD BE 11 CA 16 CA DD BE ........
C998: 12 CA 20 CA E5 C9 CD 9A .. .....
C9A0: CA 2A 16 C8 DD 5E 01 16 .*...^..
C9A8: 00 AF ED 52 22 16 C8 3A ...R"..:
C9B0: 03 FA 3D 20 0B 3E 0B DF ..= .>..
C9B8: 3E 1B DF 3E 4C DF 3E 01 >..>L.>.
C9C0: 6F 3A 04 FA 67 C3 3E 39 o:..g.>9
C9C8: DD CB 05 46 20 32 DD 34 ...F 2.4
C9D0: 06 DD CB 06 4E 28 07 DD ....N(..
C9D8: 36 06 00 DD 34 00 DD 7E 6...4..~
C9E0: 01 DD BE 00 C0 DD 36 00 ......6.
C9E8: 00 CD 9A CA 2A 16 C8 DD ....*...
C9F0: 5E 01 16 00 19 22 16 C8 ^....".."
C9F8: 3E 0A DF 3E 0D C3 18 00 >..>....
CA00: DD 34 00 DD 7E 01 DD BE .4..~...
CA08: 00 C0 18 D9 18 DB CD 0C ........
CA10: CA DD 36 00 00 C9 06 14 ..6.....
CA18: C5 CD 9E C9 C1 10 F9 C9 ........
CA20: 06 14 C5 CD 0C CA C1 10 ........
CA28: F9 C9 DD CB 05 46 20 16 .....F .
CA30: DD 35 06 F0 DD 35 00 DD .5...5..
CA38: 36 06 01 F0 DD 7E 01 3D 6....~.=
CA40: DD 77 00 C3 9E C9 DD 35 .w.....5
CA48: 00 F0 18 F0 3E 1B DF 3E ....>..>
CA50: 78 DF 3E 35 C3 18 00 3E x.>5...>
CA58: 1B DF 3E 79 DF 3E 35 C3 ..>y.>5.
CA60: 18 00 CD A7 CA DD 77 07 ......w.
CA68: 18 07 CD A7 CA DD BE 07 ........
CA70: C8 DD 36 06 00 CD 2A CA ..6...*.
CA78: DD 36 06 00 18 EC CD A7 .6......
CA80: CA DD 77 07 18 0B CD A7 ..w.....
CA88: CA DD BE 07 DD 36 06 00 .....6..
CA90: C8 DD 36 06 01 CD C8 C9 ..6.....
CA98: 18 EC CD 5B C8 CD 72 C8 ...[..r.
CAA0: CD 8B C8 CD 95 64 C9 CD .....d..
CAA8: B5 CA CD C9 CA C9 CD B5 ........
CAB0: CA CD C8 CA C9 08 2A 16 ......*.
CAB8: C8 7D DD B6 00 6F 08 C9 .}...o..
CAC0: CD DE CA 7E CD D0 CA C9 ...~....
CAC8: CD DE CA 77 CD D0 CA C9 ...w....
CAD0: 08 3E 0F D3 88 DB 90 CB .>......
CAD8: CF D3 8C 08 FB C9 F3 08 ........
CAE0: 3E 0F D3 88 DB 90 CB 8F >.......
CAE8: D3 8C 08 C9 F5 1F 1F 1F ........
CAF0: 1F CD F5 CA F1 E6 0F C6 ........
CAF8: 90 27 CE 40 27 C3 18 00 .'.@'...
CB00: 2A 03 FA 67 C3 3E 39 12 *..g.>9.
```

```
**********************************************
*                                            *
*       3D-Graphik-Nachlese   Teil 1         *
*                                            *
**********************************************
```

In dieser Serie werden wir uns etwas mit Mathematik beschäftigen. Wir wollen anhand des Programms aus Heft 2/85 zeigen, wie man die tollsten Ebenen auf den Bildschirm zaubern kann. In der ersten Folge werden wir Grundsätzliches klären. Zuerst wollen wir uns die Verhältnisse ansehen, die in diesem Programm geschaffen wurden. Wir wollen anhand des untenstehenden Bildes die ersten Eindrücke sammeln. Um diese karierte Ebene zu bekommen, muß man folgendes eingeben:

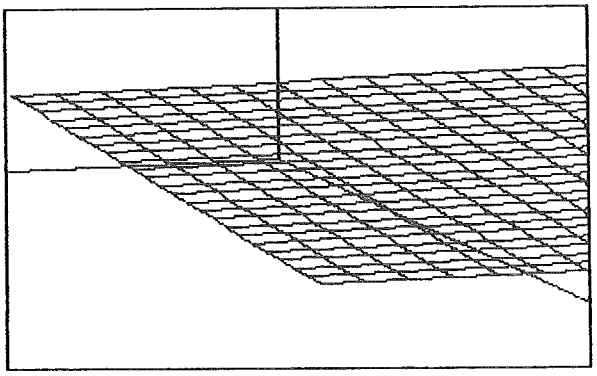
```
Zeile 120 wie folgt ändern:
120 z=1-y/20
```

Nun geben wir RUN ein und tippen hintereinander folgende Werte ein: 120 'CR', 80 'CR', 3 'CR', 30 'CR', 2 'CR', 4 'CR', 1, 2 und .5 'CR'. Nun bildet sich innerhalb der nächsten zwei Minuten die Ebene.

Im Bild sehen wir im Hintergrund drei Linien. Diese Linien zeigen die Richtungen der drei Koordinatenachsen. Der Raum hat, wie algemein bekannt ist, drei Dimensionen (deshalb heißt es ja auch 3D-Graphik). Jede Dimension hat eine Achse, die in "natura" alle normal aufeinander stehen (im Winkel 90 Grad, wie die einzelnen Wände in einer Ecke eines Zimmers). Da der Bildschirm zweidimensional ist, muß man die drei Dimensionen etwas verzerrt darstellen. Die drei Linien geben die jeweiligen Winkel an, in denen die Achsen auf dem Schirm abgebildet sind.

Alle Ebenen werden in diesem Programm in Form von Liniengittern dargestellt. Im Bild sind diese Gitter deutlich erkennbar. Man nennt diese Netze auch Raster. Die Feinheit dieses Rasters läßt sich übrigens im Programm ändern (letzter Wert der Eingabe).

In der nächsten Folge widmen wir uns der ersten Eingabeart im Programm, den "kartesischen Koordinaten".



```
*****************************************
*                                       *
*          Disk-BASIC-Erweiterung       *
*                                       *
*****************************************
```

Nun haben mein Freund und ich auch das Disk-BASIC näher betrachtet und uns Einblick in seine Arbeitsweise verschafft. Am einfachsten zu durchschauen sind die Routinen, in denen die Ein- und Ausgabe von Daten durchgeführt werden. So fielen unseren neugierigen Blicken sowohl die Disk-I/O-Routinen als auch die Modem- und 80-Zeichenkarten-Subroutinen zum Opfer. Soweit die Vorgeschichte, doch was hat das mit der Erweiterung zu tun?

Nun, es wird glücklichen Besitzern von Diskettenstationen bestimmt schon aufgefallen sein, daß ein nicht vorhandenes Diskettenlaufwerk oder eine nicht eingelegte Diskette den Computer zur Selbstaufgabe zwingen können. Wenn Sie zum Beispiel FILES eingeben und sich im Laufwerk 1 keine Diskette befindet, wartet der Computer, bis Sie eine einlegen. Das ist ja nicht weiter schlimm, aber wenn Sie FILES 2 eingeben und dabei gar keine zweite Diskettenstation angeschlossen ist, wartet der Computer, bis eines angeschlossen ist (entweder Sie kaufen nun schnell ein zweites Laufwerk und schließen es an, oder Sie schalten den Computer aus). Zum Teufel mit der schönsten Entwicklung, wenn Sie bei nur einem Laufwerk irrtümlicherweise SAVE "2:..." eingeben.

Um diesem Übel Abhilfe zu schaffen, erkundigten wir uns beim Autor der Serie "Das CP/M-Betriebssystem am SVI-328", welche Möglichkeiten man hätte zu überprüfen, ob das angewählte Laufwerk überhaupt vorhanden sei. Mit seiner Unterstützung gelang es dann auch, ein kurzes Maschincodeprogramm zu entwickeln, das bei jedem Diskettenanwählen testet, ob die Floppy-Station auch wirklich vorhanden ist.

Bei der Entwicklung des Maschincodeprogramms ergab sich die Frage, wohin mit ihm, und die war nicht leicht zu beantworten. Weiters fanden wir heraus, daß man das MC-Programm zweimal aus dem Disk-BASIC aufrufen muß. Wir sahen uns gezwungen, ein CALL und ein JP zu unserem Programm einzubauen, und siehe da, es funktionierte. Der Computer meldete sich bei einem nicht vorhandenen Laufwerk oder einer fehlenden Diskette mit einem sauberen 'Disc not mounted'.

Leider gibt es im ganzen Computer kein Bit, das Auskunft darüber gibt, ob die gewünschte Diskettenstation eine Diskette beinhaltet oder nicht. Also mußte ein anderes System herhalten. Es besteht aus zwei Teilen:

1) Dem Floppy-Diskcontroller (FDC) den Befehl zu geben, einen Interrupt zu generieren, wenn das Indexloch der Diskette erkannt wird,

2) dann auf den Interrupt zu warten. Kommt der innerhalb einer Sekunde nicht, so kann mit Sicherheit gesagt werden, daß keine Diskette (oder Laufwerk) vorhanden ist. Andernfalls, wenn der Interrupt kommt, kehrt das System zum aufrufenden Programm zurück.

Die Diskette dreht sich mit 300 Umdrehungen in der Minute im Laufwerk und deshalb kommt alle 0.2 Sekunden das Indexloch vorbei (bei vorhandener Diskette). Daraus ergibt sich eine maximale Verzögerung von 0.2 Sekunden, wenn Sie auf Diskette zugreifen, und das ist ja hoffentlich noch zu ertragen.

Zu dem Interrupt ist noch hinzuzufügen, daß es eigentlich kein Interrupt ist, der den Z-80 zwingt, die Interruptbehandlungsroutine bei &H38 aufzurufen. Dieser Interrupt des FDC wird in einem Register desselbigen angezeigt. Das siebente Bit, INTRQ genannt, wird mehrfach verwendet, um Zustände während Schreib-Leseoperationen oder dergleichen anzuzeigen. Im Heft 3/85 haben Sie die Bedeutung einiger Bits auf den Seiten 7 und 8 aufgelistet.

Das BASIC-Programm implementiert diese kurze Erweiterung direkt auf den Systemspuren der Diskette im Laufwerk 1. Deshalb sind sie in der Lage, das Programm einmal zu laden und danach eine Disk-BASIC-Systemdiskette nach der anderen ins Laufwerk 1 zu stecken und jeweils mit RUN zu starten. Nach dem 'Ok...' ist das MC-Programm implementiert.

Wenn Sie nun eine solcherart behandelte Systemdiskette booten, dann Laufwerk 1 entriegeln (den Laufwerksverschluß öffnen) und FILES eingeben, dürfte nach zirka einer Sekunde eine 'Disc not mounted'-Meldung kommen. Ansonsten war irgendetwas falsch, und Sie sollten das Implementierungsprogramm genau mit der Vorlage aus dem Heft vergleichen.

Das Maschincode-Programm liefert gegebenenfalls den Error 65, 'Disc not mounted'. Diese Meldung ist nicht selbstgestrickt, sondern kommt direkt aus dem Disk-BASIC. Sie wird nur nie angesprungen (es sei denn, man verwendet ein ERROR 65). Es gibt eine weitere Fehlermeldung, 'Disc offline', die ebenso nicht verwendet wird. Anscheinend kannten die Programmierer des Disk-BASIC keine Methoden, solche Fehler zu entdecken. Dafür wissen wir und Sie es nun.

```
10 ' Dieses Programm erweitert das
11 ' Discbasic um eine Errormeldung:
12 ' Disk not mounted
13 ' Sie tritt auf, wenn
14 ' a) das Laufwerk nicht vorhanden,
15 ' b) keine Diskette im Lw., oder
16 ' c) das Lw. nicht verriegelt ist
17 ' Philipp Ott         05.04.1985
18 CLEAR512:FIELD #0,128 AS A$,128 AS E$
19 R$=DSKI$(1,0,12)
20 MID$(A$,&H22)=CHR$(&HCD)+CHR$(&H3C)+CHR$(
&HF3)+CHR$(0)+CHR$(0)
21 MID$(A$,&H3F)=CHR$(&HC3)+CHR$(&H27)+CHR$(
&HF3)
22 DSKO$ 1,0,12:R$=DSKI$(1,2,6):B$="":RESTOR
E26
23 R$=DSKI$(1,2,6)
24 READC$:IFC$<>"*"THENB$=B$+CHR$(VAL("&H"+C
$)):GOTO24
25 MID$(A$,&H28)=B$:MID$(A$,&H28)=B$:DSKO$1,
2,6:PRINT"Ok..."
26 DATA D3,34,3E,D4,D3,30,21,00,00,DB,34
27 DATA 87,D8,2B,7C,B5,20,F7,C3,E6,EA,3A
28 DATA F4,D9,BB,C0,E1,18,E5,*
```

```
 1:        org  f327h
 2:        ;Hier erfolgt der erste
 3:        out  (34h),a    ;Einsprung
 4: check  ld   a,d4h
 5:        ;Schicke dem FDC den Befehl,
 6:        ;eine Interrupt zu generieren
 7:        out  (30h),a
 8:        ld   hl,0000    ;HL dient
 9:        ;zum Zeitzaehlen,nun warte
10: wait   in   a,(34h)    ;auf INTRQ
11:        add  a,a        ;Bit 7 ins Carry
12:        ret  c          ;ok, INTRQ gesetzt
13:        dec  hl         ;Zeit=Zeit-1
14:        ld   a,h
15:        or   l
16:        jr   nz,wait    ;bis Zeit=0
17:        jp   eae6h      ;jp Error
18:        ;Hier erfolgt der zweite Ein-
19:        ld   a,(d9f4h) ;sprung
20:        cp   e          ;Alte Disk ?
21:        ret  nz         ;Nein, zurueck
22:        pop  hl       ;CALL neutralisieren
23:        jr   check    ;Disk pruefen
24:        end
```

ifabo 85 — wir sind dabei

Anfang Mai wird wie jedes Jahr auch heuer wieder die "IFABO"-Messe eröffnet werden. Und in diesem Jahr werden wir dabei sein. Hier eine kleine Vorschau über das, was wir brauchen, was wir vorhaben und wo wir zu finden sind.

Voriges Jahr im November nahmen wir zum ersten Mal an einer Messe teil. In einigen Wochen stellen wir zum zweiten Mal aus. Diesmal hat uns jedoch nicht der "itm-praktiker" sondern eine Wiener Firma "Unterschlupf" gewährt. Die Wehsner Ges.m.b.H. überläßt uns einen großen Teil ihres Standes auf der diesjährigen "ifabo"-Messe. Das Problem des Platzes ist somit gelöst, doch andere Probleme müssen von uns selbst gelöst werden.

Zum Ersten brauchen wir wieder allerhand nützliche Software. Einige Spiele von Spectravideo werden auf der "ifabo" in Verwendung kommen, doch wir wollen den Spectravideo ja nicht als Spielcomputer abqualifizieren. Sollten Sie also Programme haben, die einiges können, dann kommen Sie zu uns und zeigen Sie uns Ihre Entwicklung. Selbstverständlich ist auch nützliche Hardware immer willkommen. Letztes Mal hatten wir ja auch einen Sprachsynthesizer installiert. Wer weiß, vielleicht hat jemand schon ein "Gesangsmodul" gebaut?

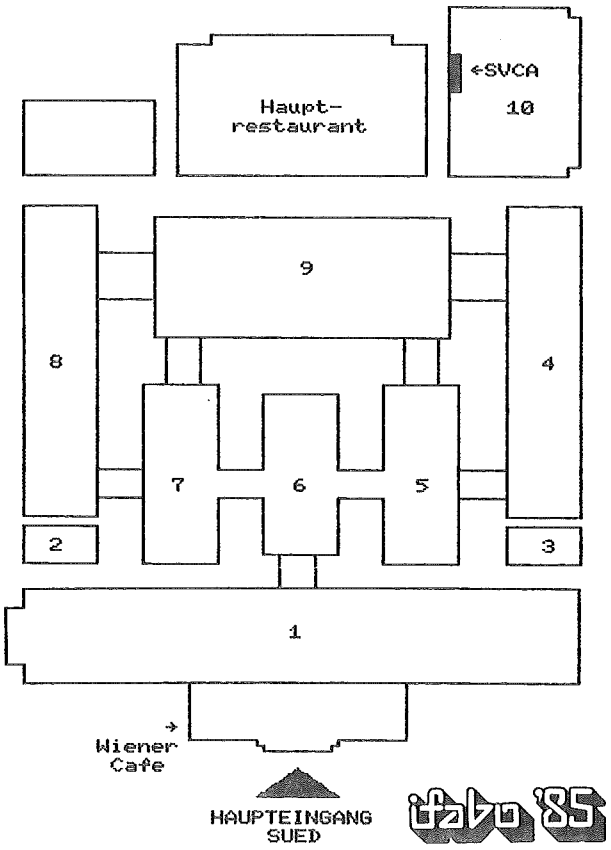
Zum Zweiten brauchen wir Helfer. Auf der letzten Messe hatten wir regen Zulauf an Besuchern. Diesem Andrang an interessierten Messebesuchern müssen wir ja mit gut informierten Clubmitgliedern entgegentreten. Sollten Sie also Zeit haben und den Club vertreten wollen, dann sagen Sie uns dies.

Was haben wir bis jetzt geplant?

Sehen wir uns zuerst die Platzverhältnisse und die Position unseres Standes an. In der Zeichnung kann man das Messegelände von oben bewundern. Dies ist freilich nur ein kleiner Ausschnitt des ganzen Territoriums. Wir sind in Halle 10 untergebracht, der schwarz "angemalte" Teil gibt die Position in der Halle an.

Von diesem Stand dürfen wir ungefähr das obere Drittel in Anspruch nehmen. Glücklicherweise können wir die Tische, auf denen die Computer stehen werden, nun an die Wand stellen. Platz haben wir für ungefähr fünf Computer. Dementsprechend beladen werden wir auch dort aufkreuzen. Neben den fünf Konsolen wird wieder nützliche Peripherie mitgenommen. So lassen sich neben Expandern und Monitoren auch noch Kassettenrekorder finden. Diese Utensilien wiegen allerdings etwas und haben auch ein bestimmtes Volumen. Deshalb werden wir auch arbeitsame "Träger" brauchen, die mit uns die Hardware auf den rechten Platz schaffen. Sonntag oder Montag vor der Messe wird das ganze über die Bühne gehen. Nähere Informationen darüber im Clublokal!

Natürlich tragen wir dafür Sorge, daß unser Club auch als solcher gekennzeichnet ist. Daher werden wir einige Transparente herstellen. Außerdem wollen wir wie letztes Mal wieder Flugblätter drucken.



An Computersystemen werden wir die ganze Spectravideo-Palette ausstellen. Ein SVI-318 wird der Vollständigkeit halber auch dabei sein. Er wird voraussichtlich im Spielbereich verwendet werden. Zwei SVI-728 bilden eine Brücke zum inzwischen vielbeachteten MSX-Standard. Von vielen Seiten wird übrigens der Spectravideo gelobt. Er sei der Beste unter den MSX-Computer heißt es. Kein Wunder, hat er doch die beste Tastatur, den größten Speicher und vieles andere.

Nach wie vor das Flaggschiff unter den Computern ist der SVI-328. Wir werden zwei seiner Sorte ausstellen. Am SVI-328 werden wir wieder CP/M fahren, denn schließlich müssen die Leute überzeugt werden, daß der Spectravideo SVI-328 ein vollwertiger PC ist.

Am Betriebssystem CP/M sind ja viele sehr gute Programme lauffähig. Seit einiger Zeit gibt es das Turbo-Pascal für unsere Computer. Das Turbo-Pascal wird neben einigen anderen Sprachen auf CP/M auch auf der Messe vertreten sein.

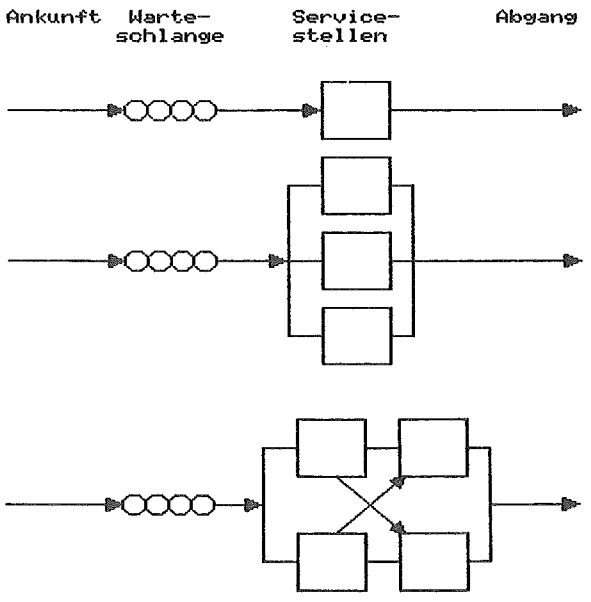
Die Messe fängt am 7. November an und hört am 11. Mai auf. Bis dahin können sich nun alle Clubmitglieder, die aktiv mittun wollen, bei uns melden. Doch auch an die Mitglieder, die kein Interesse an aktiver Teilnahme hegen, sei appelliert. Kommen Sie bei Ihrem Messebesuch an unseren Stand, sehen Sie sich kritisch alles an und sagen Sie uns dann ihre Meinung. Wir wollen ja bei jeder Messe dazulernen.

Wie oft hat wohl jeder von uns in einer Warteschlange, ob diszipliniert oder nicht, warten müssen? Sicherlich sehr oft, zu unserem Leidwesen! Doch Warteschlangen können aus unserem täglichen Leben nicht weggedacht werden. Deshalb ist es einen Versuch wert, sie etwas unter die Lupe zu nehmen.

Als erstes taucht die Frage auf: Was sind Warteschlangen, und wie kommen sie zustande?

Eine Warteschlange (queue) ist eine Folge von gleichartigen Elementen, bei der Elemente nur an einem Ende (Kopf (head) der Schlange) eingefügt und am zweiten Ende (Schwanz (tail) der Schlange) herausgenommen werden können. Bezeichnenderweise benutzen die Amerikaner in der EDV für Warteschlangen das Kürzel FIFO (First In First Out = als erstes Element in die Schlange hinein, als erstes Element wieder aus der Queue heraus). Warteschlangen kommen im täglichen Leben sehr oft vor. Denken wir nur, um einige triviale Beispiele zu nennen, an die Warteschlangen bei den Bushaltestellen, vor der Theaterkasse, bei der nächsten Verkehrsampel oder gar am Supermarkt. Die Person, die zuerst bei der Theaterkasse war, kommt als erste an die Reihe, und nach der Bedienung verläßt sie als erste die Kasse. Der Volksmund sagt ja auch: "Wer zuerst kommt, mahlt zuerst".

Die untenstehenden Zeichnungen illustrieren Warteschlangen verschiedener Modelle, manche mit mehr als einer Servicestelle, doch alle haben nur eine Ankunftsstelle (arrival) und nur eine Abgangsstelle (departure).



Bis jetzt haben wir bewußt verschwiegen, wozu man die Datenstruktur Warteschlange in der "Elektronischen Datenverarbeitung" braucht und wie sie dargestellt werden kann.

Die Warteschlange ist eine sehr wichtige Form der Datenverarbeitung und wird am einfachsten durch Arrays oder durch Verweislisten (linked lists) erzeugt. Wenn einfache Arrays benutzt werden, ist es am zweckmäßigsten, kreisförmige Arrays (circular lists) zu verwenden. Das sind ganz normale lineare Listen, welche so behandelt werden, als wären sie kreisförmig (siehe Beispiel). Benutzt werden Warteschlangen, um Modelle zu konstruieren, welche reale Prozesse simulieren.

Doch welche Vorteile bringt die Simulation eines Prozesses (was immer dieser Prozeß darstellen mag) durch den Computer? Die Frage ist einfach zu beantworten: nämlich viele. Einer der größten Vorteile der Simulation von Prozessen, Systemen oder Ereignissen durch den Computer besteht darin, daß ein System einem Test unterzogen werden kann, ehe viel Geld und Zeit in ein korrespondierendes reales System investiert wurde.

Zusätzlich kann bei der Simulation eines Prozesses durch den Computer ein sehr wichtiger Faktor, nämlich die Zeit, geändert, angehalten und sogar reversiert werden, Vorgänge welche bei einem realen System praktisch unmöglich sind. Außerdem erlaubt eine Simulation ein Testen unter extremen Bedingungen, welche das Verhalten des künftigen realen Systems unter höchst ungewöhnlichen Umständen prüft. Solche extremen Umstände hätten bei einem realen System vielleicht katastrophale Folgen. Doch an dieser Stelle soll nicht verschwiegen werden, daß komplexe Probleme sehr schwer zu simulieren sind und infolge dessen viel Geld für die Erschaffung der Simulationssoftware bezahlt werden muß.

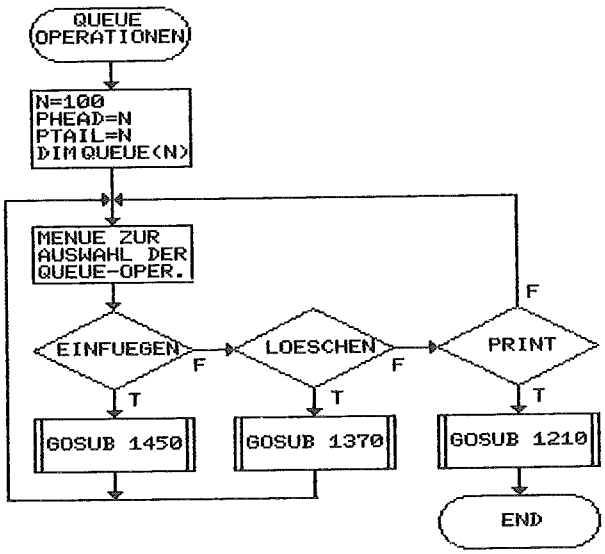
Eine wichtige Rolle bei der Simulation spielt die nachfolgende Interpretation (Analyse) der Resultate, welche ja den jeweiligen Annahmen unterliegen. Diese Annahmen charakterisieren das System in der Behandlung und reflektieren dieses.

Es gibt eigentlich drei primitive Operationen an Warteschlangen: Das Einfügen (insert), Löschen (remove) und die Abfrage, ob die Schlange leer (empty, underflow) ist. Hinzu kommt bei der Repräsentation der Warteschlange durch ein Array die Abfrage, ob das Array voll (overflow) ist. Der erste Algorithmus handelt von den oben erwähnten Warteschlangen-Operationen. Er ist sehr einfach und braucht wenig Erklärung. Er benutzt ein kreisförmiges Array (=Feld). Elemente werden durch Inkrementieren (=Erhöhen) des PTAIL-Zeigers eingefügt und durch Inkrementieren des PHEAD-Zeigers ausgefügt. Anfangs zeigen beide Zeiger auf das letzte Element der Liste PHEAD=PTAIL=N, was zugleich bedeutet, daß die Queue leer ist. PHEAD wird zu jeder Zeit auf eine Position vor dem ersten Element gegen den Uhrzeigersinn zeigen.

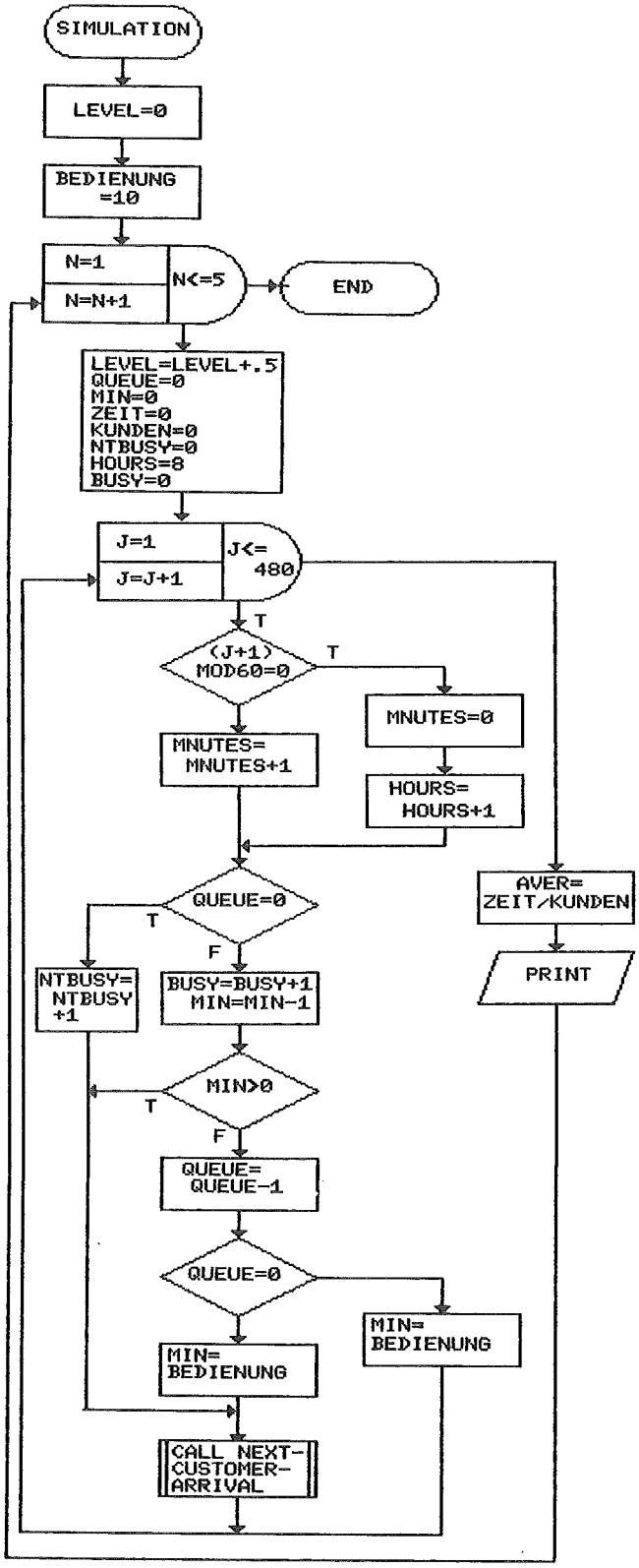
Der zweite der Algorithmen zeigt eine einfache Simulation einer Warteschlange. Es wird für eine Zeitspanne von 8 Stunden (480 Minuten) der Betrieb einer Autowaschstraße simuliert. Das Ganze findet unter fünf verschiedenen Bedingungen (Verteilung nach der Wahrscheinlichkeitstheorie) für Kundenankunft statt. Die Hauptkomponenten des Programms sind die getroffenen Annahmen wie

Bedienungszeit und zufällige Ankunft eines Kunden. Für die Ankunft haben wir keinen speziellen Algorithmus entwickelt, obwohl vieles dafür gesprochen hätte, sondern die RND-Funktion des SVI-Computers benutzt. Wir haben aber, je nach Versuchsebene, die Grenzen eingeschränkt. Im Endeffekt zeigt das Programm die Ankunftszeit eines jeden Kunden, den Zustand der Warteschlange und letztlich die Versuchsebene, die freie Zeit, die Betriebszeit, die Anzahl der Kunden, die Gesamtzeit und die mittlere Wartezeit.

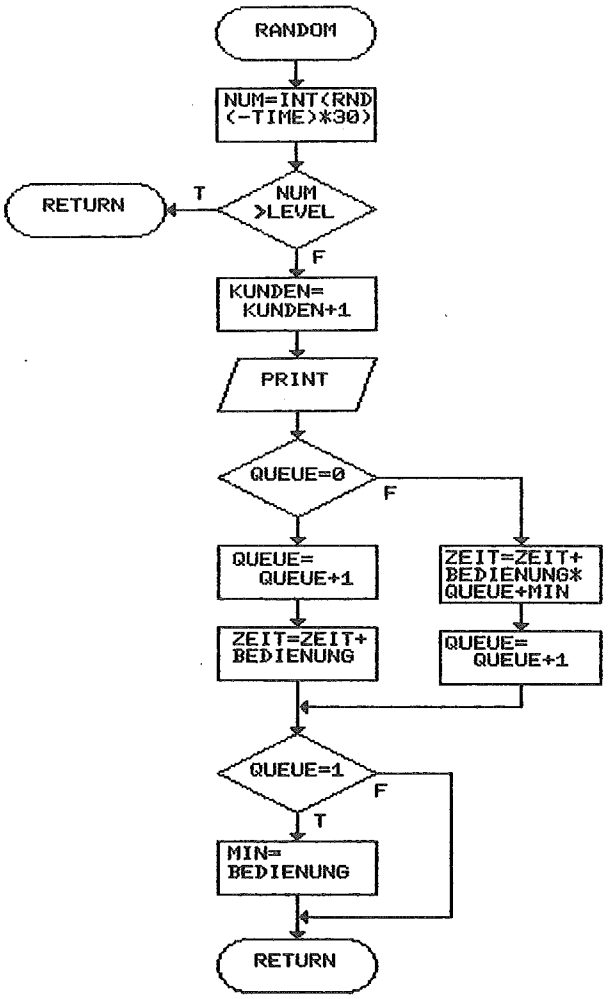
```
1000 '''''''''''''''''''''''''''
1010 ' Operationen an Queues    '
1020 ' written by C.Gaganas     '
1030 '        Maerz 1985        '
1035 '                          '
1040 '''''''''''''''''''''''''''
1050 CLS
1060 N=100    'Max Anzahl der Elemente
1070 PHEAD=N 'Zeiger auf kopf
1080 PTAIL=N 'Zeiger auf Schwanz
1090 DIM QUEUE(N)
1100 CLS
1105 PRINT "Queue Operationen":
     PRINT "@@@@@@@@@@@@@@@@@@"
1110 PRINT "E .... Einfuegen "
1120 PRINT
1130 PRINT "L .... Loeschen  "
1140 PRINT
1150 PRINT "P .... Print Inhalt"
1160 PRINT
1170 '''''
1180 A$=INKEY$
1190 IF A$="L" OR A$="l"
       THEN GOSUB 1370:GOTO 1100
     ELSE IF A$="E" OR A$="e"
    THEN GOSUB 1450:GOTO 1100
   ELSE IF A$="P" OR A$="p"
  THEN GOSUB 1210:
ELSE GOTO 1180
1200 END
1210 '''''''''''''''''''''''''''
1220 ' Listen des Inhalts       '
1225 '                          '
1230 '''''''''''''''''''''''''''
1240 GOSUB 1320
1250 IF EMPTY=1
       THEN PRINT "Empty queue":END
1270 IF PHEAD=N
       THEN PHEAD=0
1275 PRINT "Queue Inhalt"
1276 PRINT "@@@@@@@@@@@@@"
1280 FOR I=0 TO PTAIL
1290    IF QUEUE(I)=0
          THEN PRINT I;": ";"leer"
         ELSE PRINT I;": ";QUEUE(I)
1300 NEXT I
1305 PRINT "Phead=";PHEAD;
           "Ptail=";PTAIL
1310 RETURN
1320 '''''''''''''''''''''''''''''''
1330 ' Abfrage ob Queue leer ist    '
1335 '                             '
1340 '''''''''''''''''''''''''''''''
1350 IF PHEAD=PTAIL
       THEN EMPTY=1
      ELSE EMPTY=0 '1=leer,0=nichtl
1360 RETURN
1370 '''''''''''''''''''''''''''''''
1380 'Loeschen; X=geloeschte Zahl   '
1385 '                             '
1390 '''''''''''''''''''''''''''''''
1400 GOSUB 1320 'frage ob queue empty
1410 IF EMPTY=1
       THEN PRINT "Queue underflow."+
            " Queue empty.":END
1420 IF PHEAD=N
       THEN PHEAD=0
      ELSE PHEAD=PHEAD+1
1430 X=QUEUE(PHEAD):QUEUE(PHEAD)=0:
     PRINT "Geloescht ";X:
   PRINT "Any key to continue":
  C$=INPUT$(1)
1440 RETURN
1450 '''''''''''''''''''''''''''''''
1460 'Einfuegen;x=einzufuegende zahl '
1465 '                              '
1470 '''''''''''''''''''''''''''''''
1480 IF PTAIL=N
       THEN PTAIL=0
      ELSE PTAIL=PTAIL+1
1490 IF PHEAD=PTAIL
       THEN CLS:
            PRINT "Queue overflow.",
           "Kein Einfuegen moeglich":
         PHEAD=0:PTAIL=N:
        GOSUB 1210:END
1500 INPUT "Einfuegen";X
1510 QUEUE(PTAIL)=X
1520 RETURN
```




```
1000 '''''''''''''''''''''''''''''''
1010 '  warteschlangen-simulation    '
1020 '                               '
1030 'written by C.Gaganas,Maerz 1985'
1040 '                               '
1050 '''''''''''''''''''''''''''''''
1060 CLS
1070 LEVEL=0
1080 BEDIENUNG=10 'Bedienungszeit
1090 FOR N=1 TO 5 'Anzahl der Versuche
1100   LEVEL=LEVEL+.5
1110   QUEUE=0   ' Warteschlange leer
1120   MIN=0     ' Minuten
1130   ZEIT=0    ' Gesamtzeit
1140   KUNDEN=0 ' Anzahl der Kunden
1150   NTBUSY=0 ' Freie Zeit
1160   HOURS=8   ' Oeffnungsstunde
1170   BUSY=0    ' Es wird gearbeitet
1180   FOR J=1 TO 480 '8-Stunden Simul
1190     IF (J+1) MOD 60=0
           THEN HOURS=HOURS+1:
                MNUTES=0
         ELSE MNUTES=MNUTES+1
1200     IF QUEUE=0
           THEN NTBUSY=NTBUSY+1:
                GOSUB 1320:GOTO 1230
         ELSE BUSY=BUSY+1:MIN=MIN-1
1210     IF MIN = 0
           THEN QUEUE=QUEUE-1
           ELSE GOSUB 1320:GOTO 1230
1220     IF QUEUE=0
           THEN MIN=BEDIENUNG:
                GOSUB 1320:GOTO 1230
         ELSE MIN=BEDIENUNG
1230     ' naechste minute
1240   NEXT J
```




```
1250    AVER=ZEIT/KUNDEN:
        ' SY=KUNDEN*BEDIENUNG
1260    CLS
1265    PRINT " AUTO WASCHSTRASSE"+
              " 8-STUNDEN BETRIEB"
1270    PRINT CHR$(27)+"p";
        " Angenomm.Bedienungszeit ist";
        BEDIENUNG;" Min. ":
        PRINT CHR$(27)+"q"
1280    PRINT CHR$(27)+"p";
        " Vers. Fzeit BZeit Kunden GZeit
Wzeit ":PRINT CHR$(27)+"q";
1290    PRINT USING "##### ###### ###### ###
##   ######    ###.#";N;NTBYSY;BUSY;
   KUNDEN;ZEIT;AVER
1300    PRINT ",,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,"
1310 NEXT N
1315 END
1320 '''''''''''''''''''''''''''''''''
1321 ' Random Ankunft eines Kunden   '
1322 '                               '
1323 '''''''''''''''''''''''''''''''''
1330 NUM=INT(RND(-TIME)*30)
1340 IF NUM>LEVEL
       THEN RETURN 'kein Kunde
1350 KUNDEN=KUNDEN+1:
     PRINT "Neuer Kunde um -->";
     HOURS;":";MNUTES;"queue= ";QUEUE
1360 IF QUEUE=0
       THEN QUEUE=QUEUE+1:
            ZEIT=ZEIT+BEDIENUNG
     ELSE ZEIT=ZEIT+BEDIENUNG*
          QUEUE+MIN:QUEUE=QUEUE+1
1370 IF QUEUE=1
       THEN MIN=BEDIENUNG
1380 RETURN
```



```
*****************************************
*                                       *
* Auf der Suche nach Zeichen im Video-RAM *
*                                       *
*****************************************
```

Es ist oft notwendig zu wissen, wo bestimmte Zeichen im Bildschirmspeicher abgelegt sind. Manchmal möchte man ja die Matrizen von Buchstaben oder Zeichen verändern oder für andere Effekte verwenden.

Ein Leser schickte uns zu diesem Thema ein Programm, das mit sehr viel Bedienerkomfort den Bildschirmspeicher (und auch das "normale" RAM) des MSX-Computers SVI-728 durchsucht.

Wie bei solchen Programmen üblich, muß man nach dem RUN erst einmal eine Startadresse eingeben, der Computer möchte ja wissen, wo er mit der Durchforstung des Speichers beginnen soll. Nun darf man folgende Informationen für jede ausgelesene Speicherzelle bewundern:

1. Speicheradresse dezimal
2. Speicheradresse hexadezimal
3. Datenbyte dezimal
4. Datenbyte hexadezimal
5. Datenbyte binär

Um eine Einteilung in dieser Datenflut zu erhalten, wird nach jeder achten Adresse eine Linie eingefügt, die als Trennlinie fungiert.

Nach 16 getesteten Speicherwerten erscheint die Frage, ob man die am Bildschirm vorhandenen Daten auswerten möchte, oder aus dem Programm aussteigen will. Drückt man ENTER, werden die nächsten 16 Speicherzellen am Bildschirm dargestellt.

Man drückt im Normalfall also die "P"-Taste (für "Prüfung"). Nun werden dem User die binären Datenwerte besser vor Augen geführt. Die Nullen wandeln sich in Leerzeichen um und die Einsen in Sterne. Dadurch kommen eventuelle Zeichenmatrizen in der Binärdarstellung besser zur Geltung. Hat man sich am neuen Bild sattgesehen, darf man ENTER oder die Funktionstaste F5 drücken. Nullen und Einsen werden wieder aktiviert und die nächsten 16 Speicherzellen werden ausgegeben.

Das Programm ist relativ unkompliziert aufgebaut. Bis Zeile 170 befindet sich der Hauptteil, der für die Anzeige der Werte verantwortlich ist. Danach folgen die Fehlermeldungssubroutine und der Ausstieg aus dem Programm. Ab 300 werden im Video-RAM die notwendigen Änderungen für die Umwandlung der Binärdarstellung getroffen. Der Computer wechselt ganz einfach die beiden Matrizen der Zeichen (0 und Space, 1 und *) aus. Durch diese Art der Umwandlung werden jedoch alle Einser und Nullen in Leerzeichen und Sterne verwandelt. Dies kann beim Lesen der anderen Daten am Bildschirm störend sein.

Das Programm wurde für den SVI-728 geschrieben. SVI-328-User müssen die DATA-Zeilen ab 300 ändern und die richtigen VRAM-Adressen eintragen (der 728 hat andere Zeichencodes). Will man nicht den Bildschirmspeicher sondern das normale RAM (oder ROM) durchforsten, kann man in Zeile 50 die V?-Befehle umändern. Natürlich müssen dann auch die Adressen auf 65536 erweitert werden.

```
20 SCREEN0:COLOR 15,1,15:KEY OFF
30 WIDTH40
40 INPUT "Startadresse";A
50 IF A<0 OR A>16383 THEN GOSUB 180
60 CLS
70 PRINTA;
80 PRINTTAB(7);STRING$(4-LEN(HEX$(A)),"0")+H
EX$(A);
90 PRINTTAB(15);VPEEK(A);
100 PRINTTAB(20);STRING$(2-LEN(HEX$(VPEEK(A)
)),"0")+HEX$(VPEEK(A));
110 PRINTTAB(28);STRING$(8-LEN(BIN$(VPEEK(A)
)),"0")+BIN$(VPEEK(A))
120 A=A+1
130 IF A/8=INT(A/8) THEN PRINTSTRING$(37,"-"
)
140 IF A/16=INT(A/16) THEN INPUT "Pruefung?
oder Programm beenden? (P/E)";F$
150 IF F$="P" OR F$="p" THEN GOSUB 300
160 IF F$="E" OR F$="e" THEN GOTO 210
170 GOTO 70
180 PRINT:PRINTTAB(10);"Falsche Eingabe"
190 FOR T=1 TO 1500:NEXT:CLS
200 RETURN
210 END
300 FOR N=1 TO 4
310 READ C,D
320 FOR M=0 TO 7
330 VPOKE C+M,VPEEK(D+M)
340 DATA 4096,2440
350 DATA 2440,2384
360 DATA 4104,2432
370 DATA 2432,2048
380 NEXTM,N
390 INPUT "Weiter";F$:F$=INKEY$
400 FOR N=1 TO 4
410 READ C,D
420 FOR M=0 TO 7
430 VPOKE C+M,VPEEK(D+M)
440 DATA 2384,2440
450 DATA 2440,4096
460 DATA 2048,2432
470 DATA 2432,4104
480 NEXT M,N
500 RESTORE
510 RETURN 70
```

```
*****************************************
*                                       *
*    Ansteuerung der ersten 31 Zeichen  *
*                                       *
*****************************************
```

Die MSX-Computer sind bekannt dafür, daß sie einen hervorragenden Zeichensatz haben. Der Spectravideo SVI-728 hat sogar so einen großen, daß 31 Zeichen nur über Umwege angesprochen werden können. Man darf ja auf die ersten 31 ASCII-Codes keine Zeichen setzen, da diese Codes für Steuerfunktionen reserviert sind. Durch einen Trick kann man aber trotzdem über CHR$ alle Graphikzeichen ansprechen.

Beim SVI-728 wird der ASCII-Code 1 für eine Spezialaufgabe aufgehoben. Man darf hinter dem Zeichen noch einen weiteren Code anhängen. Dieser Code muß zwischen 1 und 31 liegen und spricht die ersten Graphikzeichen an. Mit ?CHR$(1);CHR$(1) erscheint zum Beispiel das "Gesicht" am Bildschirm.

Um jedoch das gesamte Zeichen sehen zu können, muß man sich in den SCREEN 1 bemühen. Im SCREEN 0 werden nämlich nur die ersten sechs Punkte pro Zeile eines Zeichens dargestellt. SCREEN 1 hingegen bietet alle 8 an.

```
*****************************************
*                                       *
*    Neuer Cursor für 80 Zeichen-Karte  *
*                                       *
*****************************************
```

Mit dem von mir geschriebenen Programm CURSOR ist es unter CP/M nun möglich verschiedene Cursor-Betriebsarten auszuwählen. Ich habe dabei einen etwas ungewohnten Weg beschritten, da CURSOR direkt in die Kontrollregister der 80 Zeichen-Karte schreibt.

Insgesamt kann aus sieben verschiedenen Betriebsarten ausgewählt werden, wobei die notwendigen Parameter direkt beim Programmaufruf angegeben werden müssen.

Vom Programm wird auf folgende Parameter geprüft:

0	Kein Cursor
F	FULL Cursor
U	UNDERLINE Cursor (voreingestellt; braucht nicht angegeben zu werden)
1	Kein Blinken
2	Langsam Blinken (voreingestellt; braucht nicht angegeben zu werden)
3	Schnell Blinken

Dadurch ergeben sich folgende Kombinationsmöglichkeiten:

0, U1, U2, U3, F1, F2, F3
und keine Angabe

Wie bereits oben erwähnt, erreiche ich diese Betriebsarten durch eine direkte Ausgabe an die 80 Zeichen-Karte, wobei nur das Kontrollregister 0AH benötigt wird. Dieses Register enthält in den Bits 0 bis 4 die Zeilenstartadresse des Cursors sowie in den Bits 5 und 6 den Blinkmodus; Bit 7 wird nicht benötigt. Im folgenden nun eine Übersicht der möglichen Konfigurationen:

Bit 6 5

0 1	Cursor OFF
0 0	Kein Blinken
1 1	Langsam Blinken
1 0	Schnell Blinken

In den Bits 0 bis 4 sollte der Binärwert für 8 nicht überschritten werden, da die Zeilenendadresse des Cursors im Kontrollregister 0BH auf 8 eingestellt ist (dieser Wert kann, falls es gewünscht ist, ebenfalls geändert werden).

Angesprochen werden diese Register über die Befehle

```
OUT   (50H),Registeradresse
OUT   (51H),Wert
```

Nun zum Programm selbst:

Im Programmkopf erfolgt eine Definition der notwendigen Assembleroptionen und der verwendeten Konstanten.

In der Routine START wird das Register B mit dem Wert für UNDERLINE-Cursor und langsam Blinken geladen, sowie der Übergabebuffer auf zumindest ein Zeichen getestet. Wurde kein Parameter angegeben so werden die voreingestellten Werte (Register B) beibehalten und es wird sofort zur Routine C_OUT verzweigt.

In TEST1 wird auf Cursor OFF, FULL- beziehungsweise UNDERLINE-Cursor geprüft. Trifft Cursor OFF zu, wird nur der Blinkmodus geändert, da in diesem Fall der Cursormodus nebensächlich ist. Bei FULL-Cursor werden in der Routine C_MODE die entsprechenden Bits für den Darstellungsmodus geändert, sowie auf eine weitere Parameterangabe getestet. Trifft keine dieser Möglichkeiten zu, so wird UNDERLINE-Cursor beibehalten und sofort nach TEST2 verzweigt.

Mit der Routine TEST2 erfolgt noch die Prüfung auf die möglichen Blinkgeschwindigkeiten. Sollte hier eine falsche Option angegeben sein, so wird "Langsam Blinken" aus der Voreinstellung beibehalten und zur Ausgaberoutine verzweigt.

In B_MODE werden sodann die Bits des Registers B entsprechend dem neuen Blinkmodus verknüpft, und in C_OUT erfolgt die Ausgabe an die 80 Zeichen-Karte.

Als Besonderheit darf wohl auch noch die mehrmals vorkommende Anweisung DB 11H angesehen werden. An und für sich dient die Anweisung DB ‹Liste› zum Reservieren von Daten-Bytes mit den in der Liste angegebenen Werten. Ich habe diese Funktion nun dazu verwendet, damit der Z80 in bestimmten Fällen den nächsten Befehl ignoriert.

Im speziellen Fall funktioniert dies folgendermaßen: Wird vom Programm das Label FAST angesprungen, so wird zuerst der Befehl LD C,C_FAST abgearbeitet. Danach soll die Routine B_MODE ausgeführt werden. Allerdings stehen dazwischen noch die Anweisungen LD C,C_CONS und LD C,C_OFF. Damit diese Befehle nun nicht zur Ausführung kommen, fügte ich die Anweisungen DB 11H ein. Dies bewirkt, daß der Z80 bei Ansprung des Labels FAST folgende Befehle findet:

```
0133   0E 40        FAST:   LD      C,C_FAST
0135   11 0E  00             LD      DE,000E
0138   11 0E  20             LD      DE,400E
013B                 B_MODE:
```

Bei einem Einsprung beim Label CONST beziehungsweise OFF findet der Z80 allerdings die für das Programm richtigen Befehle.

Es wäre selbstverständlich auch möglich gewesen, die Anweisungen LD C,n direkt nach dem Befehl CP (HL) einzufügen und die Zieladresse des nachfolgenden JR Z,n auf die Verknüpfungsroutine zu ändern. Dies hätte im speziellen Fall allerdings nur eine Ersparnis von zwei Bytes bewirkt. Weiters hat die von mir gezeigte Variante (diese habe ich aus dem BASIC-Betriebssystem nachvollzogen) einen Vorteil; es ist damit nämlich möglich, richtige Tabellen anzulegen, welche sich im Bedarfsfall neuen Bedürfnissen leicht anpassen lassen.

Für Experimentierfreudige besteht natürlich die Möglichkeit noch weitere Optionen einzubauen und auch die Zeilenendadresse des Cursors zu verändern.

Zum Schluß wäre nur noch zu beachten, daß die Ausgabe einer Escape Sequenz für den Cursor-Modus an die 80 Zeichen-Karte eine Änderung der eingestellten Werte bewirkt. Weiters ist eine Verwendung unter Disk-BASIC stark eingeschränkt, da nach jedem CR im Direktmodus vom BASIC-Betriebssystem auf FULL-Cursor umgeschalten wird. Da das Programm nur relative Sprünge durchführt, ist es in jedem Speicherbereich lauffähig. Für eine Anpassung an das Disk-BASIC wäre dadurch nur eine Änderung der Parameterübergabe notwendig.

Rotschek Wolfgang

```
                        TITLE   CURSOR Version 1.0

                                .Z80                ; Z80 Assemblercode
0000'                           ASEG                ; Absoluten Maschinencode erzeugen
                                ORG     0100H       ; Beginn des Programmbereichs


                        ; CP/M Systemadressen

005D                    CCPBUF  EQU     005DH       ; Übergabebuffer des CCP

                        ; IO-Adressen für 80-Zeichenkarte

0050                    V_RS    EQU     50H         ; Controll Register select
0051                    V_W     EQU     51H         ; Write Controllregister

                        ; Cursor Optionen (Bitmuster)

0000                    C_CONS  EQU     00000000B   ; Blinken aus
0020                    C_OFF   EQU     00100000B   ; Cursor-OFF
0040                    C_FAST  EQU     01000000B   ; Schnell Blinken
0060                    C_SLOW  EQU     01100000B   ; Langsam Blinken

0000                    C_FULL  EQU     00000000B   ; FULL-Cursor
0008                    C_UNDER EQU     00001000B   ; UNDERLINE-Cursor


0100                    START:                      ; Programmbeginn
0100    06 68                   LD      B,01101000B ; Lade B mit Bitmuster für Underline-Cursor
                                                    ; und "Langsam Blinken"
0102    21 005D                 LD      HL,CCPBUF   ; Lade HL mit Bufferbeginn
0105    3E 20                   LD      A,20H       ; Teste auf Space
0107    BE                      CP      (HL)        ; (Länge gleich Null?)
0108    28 36                   JR      Z,C_OUT     ; Ja; Ausgabe der voreingestellten Werte
                                                    ; Nein; Weiter
010A                    TEST1:
010A    3E 30                   LD      A,'0'       ; Teste auf SPEED 0
010C    BE                      CP      (HL)        ; (Cursor-OFF)
010D    28 2A                   JR      Z,OFF       ; Ja; Cursor ausschalten und Programmende
                                                    ; Nein; Weiter
010F    3E 46                   LD      A,'F'       ; Teste auf FULL-Cursor
0111    BE                      CP      (HL)
0112    28 08                   JR      Z,FULL      ; Ja; Verzweige
                                                    ; Nein; Weiter
0114    3E 55                   LD      A,'U'       ; Teste auf UNDERLINE-Cursor
0116    BE                      CP      (HL)
0117    20 10                   JR      NZ,TEST2    ; Trifft keine der möglichen Optionen zu,
                                                    ; dann wird zum Test auf Cursor stationär,
                                                    ; langsam bzw. schnell blinkend verzweigt.
0119    23                      INC     HL          ; Da Underline-Cursor voreingestellt ist,
011A    18 0D                   JR      TEST2       ; brauchen die entsprechenden Bits nicht ge-
                                                    ; setzt zu werden; es braucht lediglich der
                                                    ; Bufferpointer erhöht und zur Routine
                                                    ; TEST2 verzweigt zu werden.

011C    0E 00           FULL:   LD      C,C_FULL    ; Lade C mit Bitmuster für Cursor-FULL

011E                    C_MODE:                     ; Setzen des Cursor-Modus
011E    78                      LD      A,B         ; Lade Akku mit voreingestellten Werten
011F    E6 60                   AND     01100000B   ; Lösche voreingestellte Bits für Cursor-Modus
                                                    ; und lasse Blink-Modus unverändert
0121    B1                      OR      C           ; Setzen des neuen Modus
0122    47                      LD      B,A         ; Speichern der gesetzten Werte in B
```

```
0123  23                      INC     HL          ; Zeiger auf Buffer um 1 erhöhen
0124  3E 20                   LD      A,20H       ; Teste auf Space
0126  BE                      CP      (HL)        ; (Bufferende)
0127  28 17                   JR      Z,C_OUT     ; Ja; Verzweige zur Ausgaberoutine
                                                  ; Nein; Weiter

0129                    TEST2:
0129  3E 31                   LD      A,'1'       ; Teste auf "Blinken aus"
012B  BE                      CP      (HL)        ; (SPEED 1)
012C  28 08                   JR      Z,CONST     ; Ja; Verzweige
                                                  ; Nein; Weiter

                              ; Langsam blinken voreingestellt
                              ; (SPEED 2)

012E  3E 33                   LD      A,'3'       ; Teste auf "Schnell Blinken"
0130  BE                      CP      (HL)        ; (SPEED 3)
0131  20 0D                   JR      NZ,C_OUT    ; Trifft keine der möglichen Blinkgeschwindig-
                                                  ; keiten zu, so wird "Langsam Blinken" aus der
                                                  ; Voreinstellung beibehalten und zur Ausgabe-
                                                  ; routine verzweigt

0133  0E 40                FAST:   LD      C,C_FAST    ; Lade C mit Bitmuster für "Schnell Blinken"
0135  11                      DB      11H         ; Setze Datenbyte mit 11H
                                                  ; (Ignoriere nächsten Befehl)

0136  0E 00                CONST:  LD      C,C_CONS    ; Lade C mit Bitmuster für kein Blinken
0138  11                      DB      11H         ; Setze Datenbyte mit 11H
                                                  ; (Ignoriere nächsten Befehl)

0139  0E 20                OFF:    LD      C,C_OFF     ; Lade C mit Bitmuster für Cursor OFF

013B                       B_MODE:                    ; Setzen des Blinkmodus
013B  78                      LD      A,B         ; Lade Akku mit voreingestellten Werten
013C  E6 1F                   AND     00011111B   ; Lösche Bits für Blink-Modus und lasse
                                                  ; Bits für Cursor-Modus unverändert
013E  B1                      OR      C           ; Setze Blink-Modus
013F  47                      LD      B,A         ; Speichere Cursor-Optionen in B

0140                       C_OUT:                     ; Ausgaberoutine
0140  3E 0A                   LD      A,0AH       ; Lade Akku mit Kontrollregisteradresse
0142  D3 50                   OUT     (V_RS),A    ; Ausgabe an 80 Zeichen-Karte
0144  78                      LD      A,B         ; Lade Akku mit Cursor-Optionen
0145  D3 51                   OUT     (V_W),A     ; Ausgabe an selektiertes Kontrollregister
0147  C9                      RET                 ; RETURN zum Betriebssystem

                              END
```

Macros:

Symbols:

```
B_MODE  013B    CCPBUF  005D    CONST   0136    C_CONS  0000
C_FAST  0040    C_FULL  0000    C_MODE  011E    C_OFF   0020
C_OUT   0140    C_SLOW  0060    C_UNDE  0008    FAST    0133
FULL    011C    OFF     0139    START   0100    TEST1   010A
TEST2   0129    V_RS    0050    V_W     0051
```

No Fatal error(s)

```
*****************************************
*                                       *
*      Dreidimensionales Tic-Tac-Toe    *
*                                       *
*****************************************
```

Das beliebte Logikspiel "Tic-Tac-Toe" wurde für Spectravideo um eine Dimension erweitert. Nun ist es möglich, am SVI-328 in einem Raum von 4*4*4 sein Glück zu versuchen. Ziel des Spieles ist es nämlich, 4 gleichfarbige Steine in einer Reihe zu erhalten. Als Reihe gilt dabei alles, was nur irgendwie als einheitliche Linie aufgefaßt werden kann. Neben allen denkbaren Seitenlinien werden auch Diagonalen auf einer Seitenfläche oder im Raum gebilligt.

Nach dem Start des Programms kann der Spieler entscheiden, ob er mit dem Computer oder gegen andere Personen spielen möchte. Ist ein "menschlicher" Gegner vorhanden, gibt man einfach seienen Namen ein. Drückt man jedoch als Name eines Spielers einfach ENTER, übernimmt der Computer den Part dieses Spielers. Die Spielstärke des Computers läßt sich übrigens zwischen "leichter" und "schwerer" differenzieren.

Hat man erst einmal alle Fragen beantwortet, sieht man den zukünftigen "Kampfplatz". 16 große Quadrate geben die Plätze an, auf die Steine gesetzt werden können. Die dritte Dimension erreicht man, wenn man einfach auf einen schon vorhandenen Stein einen weiteren setzt. Neben diesem Feld findet sich ein verkleinertes Abbild des Quadrates, in dem ein Cursor blinkt. Dieses Feld wird gebraucht, um seinen jeweiligen Zug zu justieren. Wie geht das Spiel nun vor sich?

Ziel des Ganzen ist es, wie schon oben erwähnt, 4 Steine von der Farbe, mit der man spielt, in irgendeine gerade Reihe zu bringen. Gleichzeitig muß man natürlich verhindern, daß der andere Spieler diese vier Blöcke zuerst in die richtige Position setzen kann. Weggenommen dürfen keine Steine werden, ebenso darf man keinen mehr verschieben. Die Richtung der Reihe ist vollkommen egal, doch muß sie eine Linie bilden (über zwei, drei Ecken darf sie daher nicht gehen).

Man führt nun einen Zug aus, indem man mit dem Cursor im kleinen Feld die richtige Position bestimmt. Man kann beliebig lange umherfahren, erst wenn die Space-Taste gedrückt wird, ist der Zug unwiderruflich gemacht. Sollte sich auf dem ausgewählten Feld schon ein Stein befinden, wird über diesen ersten der zweite gelegt. So kann man auch Säulen von bis zu vier Steinchen erzeugen. Wird ein Feld angesprochen, auf dem schon vier Steine untergebracht sind, wird die Space-Taste ignoriert und man kann weiterwählen. Unter dem großen Quadrat steht übrigens immer der Spieler, der am Zug ist.

Wenn man nicht mehr weiter weiß, oder im verlieren ist, kann man auch den Computer um Rat fragen. Wenn F1 gedrückt wird, schlägt der Computer einen Zug vor. Man kann dann noch immer eigene Ideen verfolgen oder die Space-Taste drücken. Ist die Situation schon aussichtslos, hilft ein Druck auf F2 weiter. Dann wendet sich nämlich das Blatt, die Seiten sind getauscht.

Hat eine Person gewonnen, zeigt das Programm die Reihe mit den vier Steinen durch Blinken an. Will man nun weiterspielen, drückt man nach einer kurzen Zeit ein "j" (Achtung, kein großes "J", es geht nur ein kleines). Interessant ist vielleicht noch, daß im ganzen Spiel nie die hochauflösende Graphik verwendet wird. Alles wird mit Graphikzeichen im Textmodus erledigt.

```
100 '  ******************************
110 '  *                            *
120 '  *      3D-VIER GEWINNT       *
130 '  *                            *
140 '  *       fuer SVI-328         *
150 '  *      by Norbert Exler      *
160 '  *                            *
170 '  ******************************
180 '
190 'ON STOP GOSUB 3770:STOP ON
200 SCREEN0,0: GOSUB 2850: GOSUB 530 ' Initi
alisierung
210 ZG=0: PL=3: GW=0
220 PL=5-PL: ZG=ZG+1
230 IF (GW<>0) OR (ZG>64) THEN 2600
240      PRINTCO$(PL):GOSUB2520
250 H=(20-LEN(SP$(PL)))/2: X0=1: Y0=24
260 X$=LEFT$(BL$,H)+SP$(PL)+LEFT$(BL$,10):GO
SUB 2490
270 IF SP$(PL)="SVI-328" THEN 480
280 ' **** Spielerzug ****
290 X=0 :Y=0
300 LOCATE 2*(X+15)+1,27-2*(Y+5)
310 '
320 KEY ON:ON KEY GOSUB 440,460:STRIG(0) ON:
ON STRIG GOSUB 410
330 ST=STICK(0)
340 IF ST=1 OR ST=2 OR ST=8 THEN Y=Y+1
350 IF ST=6 OR ST=5 OR ST=4 THEN Y=Y-1
360 IF ST=8 OR ST=7 OR ST=6 THEN X=X-1
370 IF ST=2 OR ST=3 OR ST=4 THEN X=X+1
380 IFX<0 THEN X=3 ELSE IFX>3 THEN X=0
390 IFY<0 THEN Y=3 ELSE IFY>3 THEN Y=0
400  GOTO 300
410 KEY OFF:STRIG(0) OFF: RETURN 420
420 IF BE(X,Y)=4 THEN  GOTO 310 ELSE GOSUB 1
570 '  Zug ausfuehren
430  GOTO 220
440 KEY OFF:STRIG(0) OFF: RETURN 450
450 GOSUB 1190: X=ZX: Y=ZY: GOTO 300
460 KEY OFF:STRIG(0) OFF: RETURN 470
470 SWAP SP$(2),SP$(3): GOTO 240
480 ' **** Computerzug ****
490 GOSUB 1190
500 X=ZX: Y=ZY
510 GOSUB 1570
520  GOTO 220
530 IF RU=1 THEN M=0: GOTO  860 ELSE RU=1:CL
S
540 DIM FE(3,3,3),BE(3,3),BW(3,3),W(3,3)
550 DIM RX(3,3),RY(3,3),RZ(3,3),MX(15)
560 DIM UX(3),UY(3),UZ(3),OX(3),OY(3),OZ(3)
570 DIM X$(2,1),SP$(3),CO$(3),CL$(3)
580 DIM X(3),Y(3),Z(3),SS(3),CO(3)
590 DIM PL$(1,1)
600 X$(0,0)=":II"
610 X$(1,0)="⟩II"
620 X$(2,0)="⟨=;"
630 X$(0,1)="?RR"
640 X$(1,1)="CRR"
650 X$(2,1)="AB@"
660 PL$(0,0)="o414cde"
670 PL$(1,0)="14o4co3ao4co3g"
680 PL$(0,1)="18cdefec"
690 PL$(1,1)="18cdefec"
700 BL$="
    "
710 P$="116o5er64er64edcr64cr64cr64o4bagr64g
r64gr64fedr64o314al16o3br32br8"
720 PLAY P$
730 PRINT LEFT$(BL$,12)"3D-VIER-GEWINNT"LEFT
$(BL$,13)
740 PRINT:PRINT:PRINT:PRINT " BITTE WAEHLEN
SIE:"
750 PRINT:PRINT" 1 = SCHWARZWEISSFERNSEHER"
760 PRINT:PRINT" 2 = FARBFERNSEHER"
770 PLAYP$
780 K=VAL(INPUT$(1))
790 ON K GOTO 810,830
800  GOTO 780
810   COLOR 15,1,5
820  GOTO 840
830   COLOR 15,4,5
840 SP$(2)="SVI-328":SP$(3)="SVI-328"
850  GOTO 910
860 SWAP SP$(2),SP$(3)
870 FOR I=0 TO 3: FOR J=0 TO 3
880 BE(I,J)=0
890 FOR K=0 TO 3:FE(I,J,K)=0
```

```
900 NEXT K,J,I
910 '
920 GOSUB 970: PRINT"[F1] ZUGVORSCHLAG";LEFT
$(BL$,4);"[F2] SEITENWECHSEL"
930 FOR I=1 TO 4:PRINT:PRINT:PRINT
940 PRINT" 4(     4(     4(     4("
950 PRINT" )*     )*     )*     )*":NEXT
960 GOTO 1080
970 INPUT "Bitte Name der Spieler angeben
      ";SP$(2):INPUT" ";SP$(3):PRINT
980 FOR I=2 TO3
990 PRINTCO$(I)"SPIELER";I-1;"  "SP$(I)
1000 SS(I)=1
1010 IF SP$(I)<>"SVI-328" THEN 1050
1020 PRINT"SPIELSTAERKE 1/2"
1030 K=VAL(INPUT$(1)): IF K=2 THEN SS(I)=2:
GOTO 1050
1040 IF K<>1 THEN 1030
1050 '
1060 PRINTLEFT$(BL$,30)
1070 NEXT:FOR W=0 TO 1000:NEXT: RETURN
1080 FOR I=0 TO 3: FOR J=0 TO 3
1090 RX(I,J)=1/16:RY(I,J)=1/16:RZ(I,J)=1/8:F
E(I,J,0)=1
1100 NEXT:NEXT
1110 FOR I=0 TO3
1120 RX(I,0)=1:RY(I,0)=1
1130 UX(I)=1/8:UY(I)=1/8:UZ(I)=1/16
1140 OX(I)=1/8:OY(I)=1/8:OZ(I)=1/16
1150 NEXT
1160 UZ(0)=1:OZ(0)=1
1170 D1=1/8:D2=D1:D3=D1:D4=D1
1180 RETURN
1190 XX=0:GOSUB1370
1200 IF SS(PL)=1 THEN 1280
1210 FOR I=0 TO 3: FOR J=0 TO 3
1220 W(I,J)=BW(I,J):NEXT J,I
1230 XX=1:GOSUB 1370
1240 FOR I=0 TO 3: FOR J=0 TO 3
1250 BW=BW(I,J):BW(I,J)=W(I,J)
1260 IF (W(I,J)<64) AND (BW>0) THEN BW(I,J)=
BW(I,J)-BW/2
1270 NEXT :NEXT
1280 MUX=-5000:H=0
1290 FOR I=0 TO 3: FOR J=0 TO 3
1300 BW=BW(I,J)
1310 IF BW=MUX THEN MX(H)=10*I+J:H=H+1
1320 IF BW>MUX THEN H=1: MX(0)=10*I+J: MUX=B
W
1330 NEXT:NEXT
1340 ZZ=INT(RND(1)*(H))
1350 ZX=INT(MX(ZZ)/10):ZY=MX(ZZ)-ZX*10
1360 RETURN
1370 FOR Y=0 TO 3: FOR X=0 TO 3: BW=0
1380 Z=BE(X,Y)+XX:LOCATE 31+2*X,17-2*Y
1390 IF Z>3 THEN BW=-10000: GOTO 1540
1400 BW=BW+RX(Y,Z)+RY(X,Z)+RZ(X,Y)
1410 IF Y=Z THEN BW=BW+UX(X)
1420 IF X=Z THEN BW=BW+UY(Y)
1430 IF X=Y THEN BW=BW+UZ(Z)
1440 IF Y=3-Z THEN BW=BW+OX(X)
1450 IF X=3-Z THEN BW=BW+OY(Y)
1460 IF X=3-Y THEN BW=BW+OZ(Z)
1470 IF (X=Y) AND (X=Z) THEN BW=BW+D1
1480 IF (X=3-Y) AND (X=Z) THEN BW=BW+D2
1490 IF (X=Y) AND (X=3-Z) THEN BW=BW+D3
1500 IF (Y=Z) AND (X=3-Z) THEN BW=BW+D4
1510 HO=BW*10000!-INT(BW*10000!)
1520 IFBW>64 THEN BW=64
1530 IF ABS(HO-PL/10)<.05 THEN BW=65
1540 BW(X,Y)=BW
1550 NEXT:NEXT
1560 RETURN
1570 Z=BE(X,Y):BE(X,Y)=Z+1
1580 GOSUB 2830
1590 FE(X,Y,Z)=PL
1600 IF Z<>3 THEN FE(X,Y,Z+1)=1
1610 PRINT CO$(PL):X$=CHR$(PL+65):GOSUB2460
1620 H=1:Q=0: FOR I=0 TO 3
1630 H9=FE(I,Y,Z):X(I)=I:Y(I)=Y:Z(I)=Z
1640 GOSUB 2380: NEXT
1650 GOSUB 2420: RX(Y,Z)=H
1660 H=1: Q=0: FOR I=0 TO 3
1670 H9=FE(X,I,Z):X(I)=X:Y(I)=I:Z(I)=Z
1680 GOSUB 2380: NEXT
1690 GOSUB 2420:RY(X,Z)=H
1700 H=1: Q=0: FOR I=0 TO 3
1710 H9=FE(X,Y,I):X(I)=X:Y(I)=Y:Z(I)=I
1720 GOSUB 2380: NEXT
1730 GOSUB 2420: RZ(X,Y)=H
1740 IF Y<>Z THEN 1790
1750 H=1: Q=0: FOR I=0 TO 3
1760 H9=FE(X,I,I):X(I)=X:Y(I)=I:Z(I)=I
1770 GOSUB 2380: NEXT
1780 GOSUB 2420: UX(X)=H
1790 IF X<>Z THEN 1840
1800 H=1: Q=0: FOR I=0 TO 3
1810 H9=FE(I,Y,I):X(I)=I:Y(I)=Y:Z(I)=I
1820 GOSUB 2380: NEXT
1830 GOSUB 2420: UY(Y)=H
1840 IF X<>Y THEN 1890
1850 H=1: Q=0: FOR I=0 TO 3
1860 H9=FE(I,I,Z):X(I)=I:Y(I)=I:Z(I)=Z
1870 GOSUB 2380: NEXT
1880 GOSUB 2420: UZ(Z)=H
1890 IF Y<>3-Z THEN 1940
1900 H=1: Q=0: FOR I=0 TO 3
1910 H9=FE(X,I,3-I):X(I)=X:Y(I)=I:Z(I)=3-I
1920 GOSUB 2380: NEXT
1930 GOSUB 2420: OX(X)=H
1940 IF X<>3-Z THEN 1990
1950 H=1: Q=0: FOR I=0 TO 3
1960 H9=FE(I,Y,3-I):X(I)=I:Y(I)=Y:Z(I)=3-I
1970 GOSUB 2380: NEXT
1980 GOSUB 2420: OY(Y)=H
1990 IF X<>3-Y THEN 2040
2000 H=1: Q=0: FOR I=0 TO 3
2010 H9=FE(I,3-I,Z):X(I)=I:Y(I)=3-I:Z(I)=Z
2020 GOSUB 2380: NEXT
2030 GOSUB 2420: OZ(Z)=H
2040 IF (X<>Y) OR (X<>Z) THEN 2090
2050 H=1: Q=0: FOR I=0 TO 3
2060 H9=FE(I,I,I):X(I)=I:Y(I)=I:Z(I)=I
2070 GOSUB 2380: NEXT
2080 GOSUB 2420: D1=H
2090 IF (X<>3-Y) OR (X<>Z) THEN 2140
2100 H=1: Q=0: FOR I=0 TO 3
2110 H9=FE(I,3-I,I):X(I)=I:Y(I)=3-I:Z(I)=I
2120 GOSUB 2380: NEXT
2130 GOSUB 2420: D2=H
2140 IF (X<>Y) OR (X<>3-Z) THEN 2190
2150 H=1: Q=0: FOR I=0 TO 3
2160 H9=FE(I,I,3-I):X(I)=I:Y(I)=I:Z(I)=3-I
2170 GOSUB 2380: NEXT
2180 GOSUB 2420: D3=H
2190 IF (Y<>Z) OR (X<>3-Z) THEN 2240
2200 H=1: Q=0: FOR I=0 TO 3
2210 H9=FE(3-I,I,I):X(I)=3-I:Y(I)=I:Z(I)=I
2220 GOSUB 2380: NEXT
2230 GOSUB 2420: D4=H
2240 Z=Z+1: IF Z=4 THEN 2370
2250 RX(Y,Z)=RX(Y,Z)*2
2260 RY(X,Z)=RY(X,Z)*2
2270 IF Y=Z THEN UX(X)=UX(X)*2
2280 IF X=Z THEN UY(Y)=UY(Y)*2
2290 IF X=Y THEN UZ(Z)=UZ(Z)*2
2300 IF Y=3-Z THEN OX(X)=OX(X)*2
2310 IF X=3-Z THEN OY(Y)=OY(Y)*2
2320 IF X=3-Y THEN OZ(Z)=OZ(Z)*2
2330 IF (X=Y) AND (X=Z) THEN D1=D1*2
2340 IF (X=3-Y) AND (X=Z) THEN D2=D2*2
2350 IF (X=Y) AND (X=3-Z) THEN D3=D3*2
2360 IF (Y=Z) AND (X=3-Z) THEN D4=D4*2
2370 RETURN
2380 IF H9=0 THEN H=H/2: RETURN
2390 IF H9=1 THEN RETURN
2400 IF (Q<2) OR (H9=Q) THEN H=H*4: Q=H9: RE
TURN
2410 H=0: RETURN
2420 IF H=64 THEN H=H+PL/100000!
2430 IF H<>256 THEN RETURN
2440 FOR J=0 TO 3: GX(J)=X(J):GY(J)=Y(J):GZ(
J)=Z(J):NEXT
2450 GW=PL:RETURN
2460 XO=6*X+Z+1:YO=19-5*Y-Z
2470 FOR GR=0 TO 2: LOCATE XO,YO+GR: PRINTX$
(GR,PL-2): NEXT
2480 RETURN
2490 LOCATEXO,YO
2500 PRINT X$;:RETURN
2510 RETURN
2520 '
2530 LOCATE 30,10:PRINT"0Q6Q6Q6Q$"
2540 LOCATE 30,11:PRINT"S S S S S"
2550 FOR I=1 TO 5 STEP 2
2560 LOCATE 30,11+I:PRINT" Q2Q2Q2Q#"
2570 LOCATE 30,12+I:PRINT"S S S S S": NEXT
2580 LOCATE 30,18:PRINT"9Q7Q7Q7Q""
```

```
2590 RETURN
2600 IF GW<>0 THEN 2630
2610 X$=LEFT$(BL$,13)+"UNENTSCHIEDEN!"+LEFT$
(BL$,13)
2620  GOTO 2670
2630 H$="SIEGER: "+SP$(GW):H=LEN(H$)
2640 IF H>40 THEN H$=LEFT$(H$,40):H=40
2650 H=(40-H)/2
2660 X$=LEFT$(BL$,H)+H$+LEFT$(BL$,H+.5)
2670 LOCATE 0,0 : PRINTX$:GOSUB 2830
2680 IF GW<>0 THEN 2700
2690 GOTO 200
2700 M=M+1: IF M>3 THEN 2710 ELSE GOSUB 2830
2710 '
2720 K$=INKEY$:IF K$="j" THEN 200
2730 '
2740 GOSUB 2790
2750 '
2760 K$=INKEY$:IF K$="j" THEN 200
2770 '
2780  GOTO 2700
2790 FOR BL=0 TO 1: PL=5-PL: FOR I=0 TO 3
2800 X=GX(I):Y=GY(I):Z=GZ(I)
2810 GOSUB 2460
2820 NEXT: NEXT :RETURN
2830 PLAY PL$(PL-2,(GW<>0)*(-1))
2840 RETURN
2850 RESTORE 2880: FOR I%=0 TO 79 :READ A$
2860 VPOKE 3792+I%,VAL("&H"+A$)
2870 NEXT I%: RETURN
2880 DATA 0,8,18,38,38,78,78,f8
2890 DATA 0,f8,f0,f0,e0,e0,c0,80
2900 DATA f8,f4,f4,ec,dc,bc,bc,7c
2910 DATA 0,fc,fc,fc,fc,fc,fc,fc
2920 DATA f8,f8,f8,f8,f8,f8,f8,f8
2930 DATA 0,8,18,28,28,48,48,88
2940 DATA 0,f8,8,10,20,20,40,80
2950 DATA 84,88,98,90,a0,a0,c0,fc
2960 DATA 0,fc,0,0,0,0,0,fc
2970 DATA 88,88,88,88,88,88,88,88
2980 SOO=SOO+1:IF SOO=2 THEN STOP ELSE RUN 1
00
2990 ' GRAPHIKZEICHEN (+ASCII-CODE)
3000 '    (ASCII: 160) = LEFT+A
3010 ' " (ASCII: 162) = LEFT+C
3020 ' # (ASCII: 163) = LEFT+D
3030 ' ( (ASCII: 168) = LEFT+I
3040 ' ) (ASCII: 169) = LEFT+J
3050 ' * (ASCII: 170) = LEFT+K
3060 ' 0 (ASCII: 176) = LEFT+Q
3070 ' 2 (ASCII: 178) = LEFT+S
3080 ' 4 (ASCII: 180) = LEFT+U
3090 ' 6 (ASCII: 182) = LEFT+W
3100 ' 7 (ASCII: 183) = LEFT+X
3110 ' 9 (ASCII: 185) = LEFT+Z
3120 ' : (ASCII: 186) = RIGHT+A
3130 ' ; (ASCII: 187) = RIGHT+B
3140 ' < (ASCII: 188) = RIGHT+C
3150 ' = (ASCII: 189) = RIGHT+D
3160 ' > (ASCII: 190) = RIGHT+E
3170 ' ? (ASCII: 191) = RIGHT+F
3180 ' @ (ASCII: 192) = RIGHT+G
3190 ' A (ASCII: 193) = RIGHT+H
3200 ' B (ASCII: 194) = RIGHT+I
3210 ' C (ASCII: 195) = RIGHT+J
3220 ' I (ASCII: 201) = RIGHT+P
3230 ' Q (ASCII: 209) = RIGHT+X
3240 ' R (ASCII: 210) = RIGHT+Y
3250 ' S (ASCII: 211) = RIGHT+Z
```

```
*****************************************
*                                       *
*              Kleinanzeigen            *
*                                       *
*****************************************
```

Spectravideo SVI-318 zu verkaufen!
9 Monate alt, unverb. Richtpreis S 3.500,-
Auskunft im Clublokal

Spectravideo SVI-328 zu verkaufen!
sehr günstig, Auskunft im Clublokal

Philips TX+, SW-Fernseher mit Fernsteuerung, Portable, ca. 37 cm Bildschirmdiagonale, gutes Bild, gut geeignet für Anschluß an Computer, Extras vorhanden. Auskunft im Clublokal

```
*****************************************
*                                       *
*      CLUBBIBLIOTHEK ERWÜNSCHT ??      *
*                                       *
*****************************************
```

Haben Sie sich auch schon geärgert, wenn Sie ein Programm von einem Clubmitglied wollten, das im Augenblick nicht zur Verfügung stand (weder das Programm noch das Clubmitglied)?

Wir haben uns geärgert!

Doch was nützt der größte Ärger, wenn er keine Abhilfe schafft? Deshalb haben wir einen Entschluß gefaßt, der hoffentlich allen Mitgliedern gefällt:

Wir wollen eine Clubbibliothek aufbauen, die

- die aktuellen Programme von der Zeitschrift übernimmt
- die bereits bestehenden Programme aus früheren Zeitschriften aufarbeitet (soferne sie verfügbar sind.)
- Programme, die uns Clubmitglieder zur Verfügung stellen, für den privaten Gebrauch anderer "Members" archiviert.

Dieses Service soll kostenlos zur Verfügung gestellt werden, wobei die Datenträger selbstverständlich beizustellen sind. Falls diese Idee Gefallen findet, werden wir für die Aufrechterhaltung der Bibliothek Sorge tragen. Für Leute mit Zeit und Initiative sind wir immer dankbar, sei es in Form von persönlichem Engagement oder von Programmen.

Unsere Vorstellung über eine Clubbibliothek wurde oben im Großen und Ganzen zusammengefaßt. Bei großer Anteilnahme der Clubmitglieder ist auch daran gedacht, die Bibliothek so zu führen, daß sie in verschiedene Bereiche gegliedert wird. Diese Aufgliederung soll die Auswahl der zur Verfügung stehenden Programme erleichtern, wobei hier nach Programmsprache (Betriebssystem), Anwendung (z.B Spiele) und so weiter unterteilt werden kann.

Darüber hinaus wäre es möglich, von Zeit zu Zeit, einen Auszug der vorhandenen Bibliotheksprogramme in der Clubzeitschrift zu bringen oder eventuell eine eigene Liste, welche natürlich nur den Clubmitgliedern zur Verfügung steht, anzulegen.

Unser Engagement und der Aufwand, mit dem diese Clubbibliothek betrieben werden soll, hängt in erster Linie vom hoffentlich positiven Echo aller angesprochenen Leute ab. In diesem Sinne hoffen wir, daß unsere Idee Anklang findet, und als Nebeneffekt erwarten wir uns ebenfalls eine Entlastung der bei den Clubabenden im Einsatz stehenden Computer. Wir sind nämlich der Meinung, daß ein Clubabend nicht nur aus "Programmkopiererei", wie dies zum Teil der Fall ist, bestehen sollte.

Melden Sie sich an einem der nächsten Clubabende bei Johann Kammerer oder Helmut Kvasnicka und sagen Sie uns ruhig Ihre Meinung (Betonung liegt auf "ruhig"!). Vielleicht ist es so in Zukunft eher möglich, lang ersehnte Programme ohne graue Haare zu bekommen.

JK,HK.

```
10 '**********************************
11 '**********************************
12 '**********************************
13 '****                          ****
14 '****  BLACK JACK              ****
15 '****                          ****
16 '****  19850314  by  M.TOTH    ****
17 '****                          ****
18 '**********************************
19 '**********************************
20 '**********************************
21 '
22 'GRAFIK UND SPIELANLEITUNG
23 '
24 STOPON:ONSTOPGOSUB137
25 WIDTH39:TROFF:COLOR15,4,4:S$="V11T255CR64
CDFR64FECFCR64CR64FEL64"
26 SCREEN2,0:PRINT
27 PLAY$$:PLAY$$
28 FORI=1TO14:READA$
29 C=INT(RND(-TIME)*16):IFC=40RC=10RC=0THEN2
9ELSEIFA$=""THENPRINT" ";
30 COLORC:PRINTA$;:NEXT
31 DATA,B,L,A,C,K                    ,J,A,C
,K,,,,,
32 FORI=1TO1500:NEXT:SCREEN0,0:LOCATE,,0
33 COLOR15
34 PRINT"          * SPIELANLEITUNG *":PRINT
35 PRINT"BLACK JACK ist ein Glueckspiel, an
dem maximal 4 Spieler teilnehmen koennen.  D
as Ziel dieses Spiels ist, zu ver-    suchen
 mit den Karten moeglichst nahe  an 21 heran
zukommen, jedoch duerfen Sie21 nicht uebersc
hreiten, da Sie sonst  verloren";
36 PRINT" haben. Als weiterer Spieler   nimm
t auch der Computer teil. Zuerst   bekommt j
eder zwei Karten. Danach kann sich jeder Spi
eler noch weitere Karten bis maximal fuenf n
ehmen. Wenn Sie     gleich viele Punkte wie
der Computer   haben, hat ";
37 PRINT"dieser gewonnen. Innsgesamt sind 52
 Karten im Spiel: As,K,D,B,10,  9,8,7,6,5,4,
3 und 2,wobei Bub, Dame    und Koenig 10 zae
hlen. Ein As zaehlt   entweder 1 oder 11 als
 letzte Karte    oder bei einem BLACK JACK."
:PRINT
38 PRINT"    Druecken Sie bitte eine Taste"
39 IFINKEY$=""THEN39
40 CLS:PRINT"Einen BLACK JACK haben Sie dann
, wenn  Sie mit nur zwei Karten 21 erreicht
   haben."
41 PRINT:PRINT"          * Viel Glueck *":PR
INT:PRINT:PRINT"                  19850314
by  M. TOTH"
42 PRINT:PRINT:PRINT"Geben Sie die Anzahl de
r Spieler ein"
43 '
44 'EIGEBEN DER SPIELERNAMEN
45 '
46 DIMA$(52),F(52)
47 A$=INKEY$:IFA$=""THEN47
48 A=VAL(A$):IFA<1ORA>4THEN47ELSECLS
49 PRINT"Geben Sie die Namen der Spieler ein
:":PRINT:PRINT
50 FORS=1TOA:PRINT"Spieler";S;":";:PRINTTAB(
15);:LINEINPUTB$(S+1):PRINT:NEXT:B$(1)="COMP
UTER":SCREEN1,0
51 '
52 'MISCHEN DER KARTEN
53 '
54 FORM=1TO4:RESTORE58
55 FORB=1TO13
56 READA$((M-1)*13+B):F((M-1)*13+B)=M
57 NEXT:NEXT
58 DATA2,6,8,10,AS,3,K,5,9,D,4,B,7
59 FORX=1TO100
60 L=INT(RND(-TIME)*31)+1:M=INT(RND(-TIME)*3
1)+1
61 SWAPA$(L),A$(M):SWAPF(L),F(M)
62 NEXT
63 '
64 'JEDER SPIELER ERHAELT ZWEI KARTEN
65 '
66 CLS:COLOR15:FORA=1TO5
67 LOCATE25,A*38-20:PRINTLEFT$(B$(A),9)
68 NEXT
69 FORB=1TO2
70 FORF=STO2STEP-1
71 GOSUB130
72 NEXT:Y(B)=Y+1:Y=Y+1:NEXT
73 '
74 'WEITERE KARTEN
75 '
76 FORF=STO2STEP-1
77 B=2:IFC(F)=11AND(C$(F,1)="AS"ORC$(F,2)="A
S")THENR(F)=1:C(F)=100:GOTO87
78 B=B+1
79 COLOR15:LOCATE65+30*B,F*38-20:PRINT"J/N"
80 A$=INKEY$:IFA$=""THEN80
81 IFA$<>"J"ANDA$<>"j"ANDA$<>"N"ANDA$<>"n"TH
EN80
82 IFA$="N"ORA$="n"THEN86
83 GOSUB130
84 IFC(F)>=21THENB=5
85 IFB<5THEN78ELSE87
86 PLAY"O4C":LINE(63+30*B,F*38-33)-(85+30*B,
F*38-1),4,BF
87 NEXT
88 '
89 'COMPUTER NIMMT WEITERE KARTEN
90 '
91 H=Y:F=1:FORB=1TO2:Y=Y(B)-1:GOSUB130:NEXT:
Y=H:B=2
92 B=B+1
93 IFC(F)=11AND(C$(F,1)="AS"ORC$(F,2)="AS")T
HENR(F)=1:C(F)=100:GOTO105
94 IFC$(1,B-1)="AS"ANDC(1)+10<22THENC(1)=C(1
)+10:P=1
95 IFC(1)<16THENI=100ELSEIFC(1)>15ANDC(1)<18
THENI=13ELSEIFC(1)=18THENI=8ELSEIFC(1)=19THE
NI=4ELSEI=0
96 C=INT(RND(-TIME)*100)+1
97 IFC>ITHEN105
98 IFP=1THENC(1)=C(1)-10:P=0
99 GOSUB130:IFC(1)>21THENB=5
100 IFB=5THEN105
101 GOTO92
102 '
103 'ERMITTELN DES GEWINNERS
104 '
105 P=0:FORF=STO1STEP-1
106 I=0
107 I=I+1
108 IFC$(F,I)=""THENI=I-1ELSE107
109 IFC$(F,I)="AS"ANDR(F)=0THENS(F)=C(F)+10E
LSES(F)=C(F)
110 NEXT
111 FORF=2TOS
112 IFS(F)>GAND(S(F)<22ORS(F)>90)THENG=S(F)
113 NEXT
114 IFS(1)>=GAND(S(1)<22ORG=0ORS(1)=100)THEN
K=1:E(K)=1:GOTO119
115 FORF=2TOS
116 IFS(F)=GTHENK=K+1:E(K)=F
117 NEXT
118 IFK=0THENK=1:E(1)=1
119 COLOR15:FORB=1TOK:G(E(B))=G(E(B))+1
120 LOCATE85,38*E(B)-20:PRINT"*"
121 NEXT:PLAY$$:PLAY$$
122 FORTY=1TO2000:NEXT:CLS
123 FORF=1TOS
124 LOCATE25,F*38-20:PRINTLEFT$(B$(F),15):LO
CATE130,F*38-20:PRINTUSING"#####";G(F)
125 NEXT
126 FORI=0TO10:R(I)=0:C(I)=0:S(I)=0:FORC=1TO
10:C$(I,C)="":NEXT:NEXT:G=0:Y=0:K=0:GOTO54
127 '
128 'UNTERPROGRAMM: KARTEN ZEICHNEN
129 '
130 Y=Y+1:IFF(Y)/2=INT(F(Y)/2)THENC=1ELSEC=6
131 LINE(62+30*B,F*38-35)-(86+30*B,F*38),15,
BF:LOCATE69+30*B,F*38-20:COLORC:PRINTA$(Y):C
$(F,B)=A$(Y)
132 FORYI=1TO3:SOUND7,245:SOUND8,15:SOUND8,1
:SOUND7,250:NEXT
133 A$=C$(F,B)
134 IFA$="AS"THENI=1ELSEIFA$="K"ORA$="D"ORA$
="B"THENI=10ELSEI=VAL(A$)
135 C(F)=C(F)+I
136 RETURN
137 RUN
```

Herr Toth schickte uns das Programm "17 und 4". Ausgeprägte Spielernaturen werden ihre Freude daran haben.

****
*
* Programmecke
*
****

Wir wollen diesmal zwei Programme beschreiben, die schon leicht "antiquiert" sind. Sie waren wohl die ersten beiden Module, die bei uns auf den Markt gekommen sind. Alle Clubmitglieder, die von Anfang an beim Club dabei waren, kennen den "Frantic-Freddy" und "Cross-Force". Wir wollen für neu Hinzugekommene diese beiden Programme "aus der Schublade" holen.

"Frantic-Freddy" handelt von einer Art Käfer, der in einem Haus Feuer löschen muß. Am Anfang steht man vor der Hauswand und kann den Käfer links und rechts bewegen. Mit der Space-Taste wirft man nun Kugeln in die Feuer, die an manchen Fenstern brennen. Man braucht immer einige dieser "Löschkugeln", bis eine Flamme erloschen ist. Man kommt in den nächsten Modus, wenn alle Feuer "massakriert" wurden. Aufpassen muß man auf herabfallende Hausteile (oder Kugeln). Diese Teile kann man durch einen Schuß mit der "Löschkanone" beseitigen oder einfach ausweichen. Die Kanone schießt übrigens nur senkrecht nach oben.

Der zweite Modus spielt schon in den oberen Stockwerken. Der Käfer ist auf einer Art Balkon von Flammen eingeschlossen, die er zerstören muß. Neben den Tasten "Links" und "Rechts" darf man auch mit den restlichen zwei Cursor-Tasten an Regenrinnen auf- und abklettern. Die Feuerteile wandern übrigens nun dem Tierchen nach. Ziel dieser Aktion ist es, alle Flammen mit dem Feuerlöschgerät zu beseitigen und den Käfer in das rechte obere Eck zu manövrieren. Nach bestandenem zweiten Modus kommt wieder der erste an die Reihe.

Das zweite Modul nennt sich "Cross-Force". Wie kann es anders sein, es spielt im Weltall. Die Aufgabenstellung dürfte Ihnen bekannt sein! Man muß feindliche Raumschiffe abschießen. Das Lustige an diesem Spiel ist, daß Sie nicht nur ein Schiff sondern zwei sich synchron bewegende haben. Wahlweise kann man zwischen Diagonal- und Parallel-Bewegung umschalten. Für zwei Spieler läßt sich noch der Dual-Modus einstellen. Hier bewegt einer das obere und einer das untere Raumschiff. Der Clue an den beiden Fahrzeugen ist nämlich, daß sie sich an der Ober- und an der Unterseite des Bildschirms befinden. Die Schüsse werden immer in der Linie abgegeben, die sich zwischen den beiden Raumschiffen bildet. "Parallel" heißt nun, daß sich die Raumschiffe gleich bewegen (bei "Links" beide nach links und umgekehrt). Bei "Diagonal" führt man die zwei Gebilde genau entgegengesetzt.

Im Spiel kommen neun verschiedene Modi mit neun verschiedenen UFO-Typen vor. Diese UFOs unterscheiden sich vor allem durch ihre Flugbahn und Schußrichtung. Wir wollen hier nicht näher darauf eingehen, welches UFO welche Schüsse abgibt, denn weiß man einmal, wie diese "Dinger" anfliegen, hat man den Sieg schon in der Tasche. Sollte man einmal für einen Modus länger als erwartet brauchen, kommt ein Mutterschiff und läßt einen Fallschirmjäger auf die Unterseite des Schirms gleiten (Bitte beachten Sie! Im Weltall, das bekanntlich weder Schwerkraft noch eine Atmosphäre besitzt, werden Fallschirmspringer eingesetzt.). Diesen Jäger muß man auffangen, dann bekommt man neuen Treibstoff. Nach dem neunten Modus setzt das Spiel wieder mit dem ersten fort.

Beide Spiele sind im Fachhandel erhältlich. Sie werden in Modulen abgeboten und kosten je 790 Schilling.

Was bietet Ihnen der Spectra Video Club Austria?

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

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

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

IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Constantin Gaganas, Philipp Ott, Rafael Razim, Wolfgang Rotschek, Heinz Schmid, Stephan Traxler, Georg Wolfbauer

Titelfoto und Fotomontage: Klaus Herring, Klaus Ribing

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

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

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

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

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

Erscheinungsort Wien
Verlagspostamt 1040 Wien