SVI JOURNAL

Die Zeitschrift des Spectra-Video-Club Austria

SVI-328: Doublesided Disk-BASIC

Unser Test: SVI-738 X'press

Heft 9/85

MSX

S 15.-

Lieber SVI-Journal-Leser!

Im Inneren dieser Ausgabe finden Sie die Auswertung unserer Fragebogenaktion. Leider sind nur 20 Bögen in unserer Redaktion eingetrudelt. Es ist auf der einen Seite natürlich enttäuschend, wenn es nur 10 % der Clubmitglieder der Mühe wert finden, ihre Meinung kundzutun. Andererseits erfreut es uns, daß die meisten Leser mit dem Journal halbwegs zufrieden sein müssen. Erfahrungsgemäß werden nämlich die Fragebögen eher von kritischeren Mitgliedern retourniert. Wenn nun so wenige zurückkommen, gibt es offensichtlich wenig auszusetzen (oder ist die Mehrzahl der Leser in Puncto Qualität schon so deprimiert, daß sie sich gar nicht mehr zu schreiben getraut?). Als weiteres Lob sehen wir an, daß von eben diesen kritischen Lesern das Echo über unser Journal überwiegend positiv war. Wir werden uns jedenfalls bemühen, die wichtigsten Punkte der Kritik zu beherzigen. Alles weitere können Sie im Inneren des Heftes lesen.

Wir werden übrigens eine neue Aktion starten. Wie im BASIC gibt es nun auch für's CP/M Befehlserweiterungen. Wir wollen damit einige Nachteile des CP/M aus der Welt schaffen. Das einfache Belegen der Funktionstasten und andere Kleinigkeiten machen die Bedienung komfortabler. Wir wollen jedoch nicht in den Fehler verfallen, die einzelnen Programme wirr auf der Disk zu verstreuen. Daher werden wir alle Erweiterungen in einem File unterbringen, das wir COMMAND.COM nennen. Dadurch ist das einfache "Transportieren" der Erweiterungen von einer Disk auf die andere gewährleistet.

Kommen wir nun zu einem weniger erfreulichen Kapitel. Nach drei Aufrufen zu aktiverem Clubgeschehen ist nichts passiert! Natürlich gibt es keinerlei Verpflichtung, die Clubabende zu besuchen, aber ist es nicht langweilig, immer nur passiv zu konsumieren?

Wir haben ein ziemlich dominantes Cluborgan. Das SVI-Journal, dessen sind wir uns bewußt, zielt darauf ab, allen Mitgliedern möglichst viel Information zu liefern. Doch das soll niemanden davon abhalten, selber seine Ideen zu verwirklichen. Aus der Fragebogenaktion geht hervor, daß alle Befragten selber programmieren. Warum tun sich nicht einmal einige unserer Mitglieder zusammen und programmieren im Teamwork?

Ich spreche nun genau Sie an! Wenn Sie diesen Text lesen, im Fauteuil sitzen oder am Schreibtisch arbeiten. Wenn Sie nicht gerade 25 Stunden am Tag arbeiten, dann kommen Sie einmal zu uns. Denken Sie sich nicht, andere werden es für Sie tun. Wir brauchen genau Sie! Am nächsten Clubabend können Sie sich entweder an unsern Clubobmann, Herrn Rotschek, an unseren Kassier, Herrn Schmid, oder an meine Wenigkeit wenden.

Qualifikationen werden sowieso nicht verlangt. Es gilt auch nicht, innerhalb von Rekordzeit ein Betriebssystem zu erstellen, das mit GEM oder PC-DOS konkurrieren soll. Aber es muß doch möglich sein, daß man von 130 Clubmitgliedern ein paar findet, die miteinander arbeiten wollen. Und wenn Sie keine konkreten Vorstellungen haben, das macht nichts, wir versorgen Sie mit Kontakten und Ideen. Sie werden sehen, es lohnt sich!

Ihr SVI-Journal-Chefredakteur Gerhard Fally!

Inhalt: Seite

Die nächsten Clubtermine:

Mi,	25. September	1985, ab 19 Uhr
Di,	1. Oktober	1985, ab 19 Uhr
Sa,	12. Oktober	1985, ab 17 Uhr
Mi,	16. Oktober	1985, ab 19 Uhr
Di,	22. Oktober	1985, ab 19 Uhr
Sa,	9. November	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.

Spectravideo-BASIC

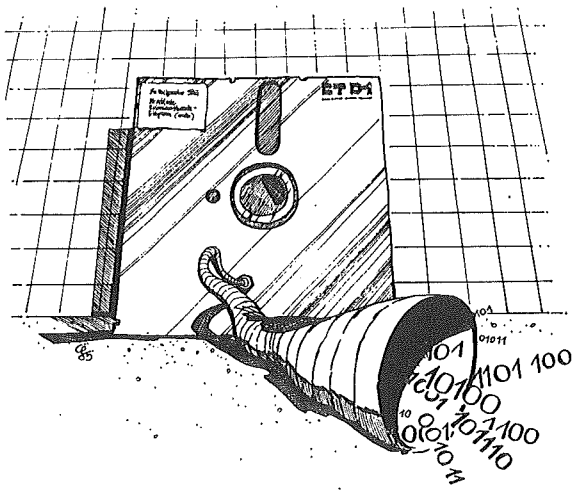
Nach 15 Folgen sind wir nun zum letzten Kapitel unseres Kurses gekommen. Dieses ist dafür sehr lang. Quasi als Abschluß wollen wir nämlich noch etwas in die große Welt der Datenverarbeitung "hineinschnuppern".

Zuerst müssen wir einmal klären, was man unter Dateien versteht und welche Typen für uns wichtig sind. Zu diesem Zweck ist es sinnvoll, wenn man sich die Schwierigkeiten des Abspeicherns von Daten auf einem Massenspeicher vor Augen hält.

Es gibt im Computer verschiedene Arten von Daten. Die uns bekannteste ist ohne Zweifel das BASIC-Programm. Wir können ein Programm mittels einfacher Befehle von Diskette (oder Kassette) laden oder speichern. Dabei muß der Computer jedoch einen bestimmten Rahmen einhalten. Wir dürfen ja keine Daten wild auf die Disk schreiben. Sie müssen gebündelt mit einer Kennung versehen auf einen Platz gespeichert werden, der auch später vom Computer wiedergefunden werden kann. Dies erledigen die oben genannten Befehle für uns. Auch Maschincodeprogramme werden in einem speziellen Rahmen abgespeichert.

Um nun eigene Daten vom Speicher auf Disk oder Kassette zu speichern, können wir uns zweier Möglichkeiten bedienen. Wir können die Daten in einen definierten Speicherbereich poken und als MC-Code-Programm mit BSAVE abspeichern. Diese Methode ist jedoch äußerst unelegant. Deshalb gibt es im BASIC noch eine andere Möglichkeit, Daten zu sichern:

Softwarekanäle auf die Disk!



Das BASIC beinhaltet spezielle Befehle, mit denen man ein File (Datei) "händisch" manipulieren kann. Dazu wird mit einem Befehl ein "Softwarekanal" vom Speicher zum Peripheriegerät geöffnet. Das Peripheriegerät kann eine Diskstation, ein Kassettenrekorder, der Bildschirm, die Tastatur oder jede andere Schnittstelle sein, die vom BASIC her angesprochen werden kann. Durch diesen Kanal kann man dann, bildlich gesprochen, Daten in den Speicher holen oder zur Peripherie schicken. BASIC sorgt dann dafür, daß diese Daten richtig verstaut werden. Natürlich kann man Kanäle auch wieder schließen (man denke nur an die Schleusen vorm Donaukanal), daher muß auch im BASIC eine analoge Funktion zur Verfügung stehen.

Die Softwareschleusen werden durch OPEN geöffnet. Die Syntax ist unterhalb zu sehen. Nach dem Befehlswort wird ein Dateiname angegeben, danach folgt ein Ausdruck, den wir später klären wollen. Zuletzt muß man noch eine Nummer anschreiben, die angibt, welcher Kanal geöffnet werden soll. 15 Kanäle dürfen gleichzeitig offen sein (für Wien wäre das ein gewaltiger Hochwasserschutz!).

```
OPEN 'Dateiname' FOR 'Ausdruck' AS 'Nr.'
```

Schließen kann man einen Dateikanal mit CLOSE. Hinter CLOSE kommt eine Nummer, die angibt, welcher Kanal "dichtgemacht" wird.

```
CLOSE 'Nr.'
```

Wir haben das Manipulieren der Datei bewußt ausgelassen, da es hier zwei Möglichkeiten gibt:

Sequentielle Dateien

Wie vieles in der Computertechnik ist auch die Dateitechnik in zwei Möglichkeiten geteilt. Eine sehr flexible (teure), die hohe Anforderungen an das Material stellt und eine einfache Möglichkeit, die billig aber nicht so komfortabel ist.

Sequentielle Dateien können nur Satz für Satz ausgelesen werden. Wenn man Satz 5 haben möchte, muß man auch die ersten vier einlesen. Dies ist die einfache Möglichkeit. Sie ist in vielen Fällen ausreichend.

Für komplexere Anwendungen sind Random-Dateien besser. Mit ihnen kann man Sätze einzeln anwählen. Man kann zuerst den 5., dann den 12. und zuletzt den ersten einlesen. Diesen Vorteil erkauft man sich durch geringfügig mehr Programmieraufwand und leistungsfähigere Peripherie.

Nun können wir auch verstehen, was es mit dem Ausdruck des OPEN-Befehls auf sich hat. Es gibt hier nämlich drei Möglichkeiten. "FOR INPUT" bedeutet, daß man nur aus der Datei lesen kann. Ein gleichzeitiges Speichern ist nicht möglich. Will man speichern, muß man die gleiche Datei unter einer anderen Nummer öffnen. "FOR OUTPUT" ist das Gleiche, nur darf nichts gelesen werden, es wird nur geschrieben. Die ersten beiden Modi sind sowohl für Random-Dateien als auch für sequentielle Dateien möglich. "FOR APPEND" ist nur für Random-Dateien wichtig. Wir werden diesen Mode später kennenlernen.

Zuerst wollen wir uns mit den sequentiellen Dateien beschäftigen:

Sequentielle Dateien benötigen auf jeden Fall hinter OPEN die Angabe eines Modus. Je

nachdem, welcher angegeben wurde, stehen dann folgende Befehle zum Datenmanipulieren zur Verfügung:

INPUT #x,'Variablenliste': Vom Peripheriegerät wird ein Datensatz in die angegebene Variablenliste gelesen. Die Variablen können sowohl numerisch als auch alphanumerisch sein. Es gelten die Regeln wie für INPUT. x ist die Dateinummer.

LINEINPUT #x,'Variable': Reagiert wie LINE INPUT, mit dem Unterschied, daß nicht von der Tastatur auf eine Eingabe gewartet wird, sondern daß sich der Computer die Eingabe vom angesprochenen Kanal holt!

PRINT #x,'Ausdruck': Wie PRINT arbeitet diese Anweisung, sie schreibt einen Datensatz zum Device. x ist wieder die Dateinummer. Es ist übrigens ratsam, einen Ausdruck immer in Anführungszeichen abzuspeichern, wenn man im Vorhinein nicht genau weiß, welche Zeichen alle abgespeichert werden. Der Computer trennt einen String ohne Anführungszeichen nämlich beim Einlesen durch INPUT bei jedem Strichpunkt oder Beistrich ab. Dies ist meistens nicht erwünscht. Man kann diesen Effekt dadurch umgehen, daß man - wie gesagt - Anführungszeichen mitspeichert oder den Text mit LINEINPUT einliest (was nicht immer möglich ist). Beispiele folgen später! Selbstverständlich kann man auch PRINT #x,USING verwenden.

Zwei Funktionen zur Dateiverwaltung gibt es natürlich auch:

EOF('Dateinummer') zeigt an, ob daß Dateiende erreicht wurde. Wenn der letzte Satz gelesen wurde, ergibt der Test '-1'.

LOC('Dateinummer'): Gibt die Position an, bei der gerade gearbeitet wird. Konkret liefert LOC die Anzahl an Sektoren, die schon gelesen oder geschrieben wurden. Ein Sektor hat eine Länge von 256 Bytes.

Soweit der theoretische Teil, nun werden wir einige sequentielle Dateien erstellen. Wir werden dabei bedacht darauf nehmen, daß sequentielle Dateien nur bedingt zur Datenverarbeitung verwendet werden können. "Richtige Dateien" werden erst mit den Random-Files aufgebaut.

Als erstes Beispiel wollen wir eine Reihe von Zahlenwerten abspeichern. Dies kann zum Beispiel dann interessant sein, wenn man einen Zufallsgenerator testen möchte und Zahlenreihen erzeugt, die man nachher auswerten möchte. Wir werden daher Zufallszahlen auf ihre Zufälligkeit überprüfen.

```
10 CLS:DEFSNG Z:AN=0:DS=0
20 OPEN "1:ZAHLEN.RND" FOR OUTPUT AS 1
25 PRINT "TESTREIHE LAEUFT!"
30 Z=RND(-TIME)
40 FOR I=1 TO 2000:Z=RND(Z)
50 PRINT #1,Z:NEXT
60 PRINT "TESTVORGANG BEENDET!"
70 CLOSE 1
80 OPEN "1:ZAHLEN.RND" FOR INPUT AS 1
90 IF EOF(1) THEN 120
100 INPUT #1,Z
110 DS=DS+Z:AN=AN+1:GOTO 90
120 CLOSE 1:DS=DS/AN
130 PRINT "DER DURCHSCHNITT BETRAEGT"DS
140 END
```

Das Programm selber ist nicht sehr praxisnah. Es soll jedoch zeigen, daß man Zahlenreihen abspeichern und später weiterverarbeiten kann. In unserem Fall wird nur der Durchschnitt gebildet (den hätte man schon beim ersten Durchgang bilden können, aber dann hätten wir keine Aufgabe für den zweiten Arbeitsgang gehabt).

Zuerst wird die Datei geöffnet, dann schreiben wir 2000 Zahlenproben auf die Diskette. Wenn dies geschafft ist, schließen wir die Datei und öffnen sie zum Lesen.

Dann wird vor jedem Lesen eines Datensatzes dezent überprüft, ob schon der letzte gelesen worden sei. Dadurch ersparen wir uns eventuelle negative Überaschungen, wie zum Beispiel die Meldung "Input past end", was soviel wie "Leseversuch nach dem Dateiende" bedeutet. Die Funktion EOF ist für diese Abfrage verantwortlich. Dann wird mit der Anweisung INPUT # eine Zahl nach der anderen eingelesen und verarbeitet.

Wir sehen uns nun das Ganze in der Praxis an: Nach einem RUN greift der SVI auf die Disk zu und öffnet das File. Daß es jetzt schon auf der Diskette existiert, kann man nachprüfen, indem man direkt hinter dem OPEN ein CLOSE und ein STOP einfügt.

```
20 OPEN "1:ZAHLEN.RND" FOR OUTPUT AS 1:CLOSE
1:STOP
```

Wenn wir nun ein FILES machen, erkennen wir ein "ZAHLEN RND" im Directory. Ein CLOSE ist übrigens nach jeder Bearbeitung von größter Wichtigkeit. Erst durch das CLOSE wird nämlich das Directory auf den aktuellen Stand gebracht. Wenn wir nur ein STOP ohne CLOSE verwenden, sieht das Directory nämlich anders aus. Nach einem KILL und einem RUN findet sich kein "ZAHLEN RND" im Directory.

Wir löschen jedoch wieder alle Änderungen, die wir angebracht haben und starten unsere Tests. Nach etwa einer Minute hat der Computer alle Daten gespeichert und beginnt mit der Auswertung. Wenn diese fertig ist, kann man den Durchschnittswert bewundern. Wir könnten nun mehrere Male verschiedene Durchschnitte bilden und andere Datenverarbeitungen durchführen, die Zufallszahlen dienen uns, solange wir wollen. Wir können übrigens ohne Schwierigkeiten wieder ein RUN eintippen, der Computer überschreibt dann die alten Daten und ersetzt sie durch neue. Die verwendete Formel ist übrigens sehr ausgeglichen, der Durchschnitt liegt ziemlich genau bei .5 (Messungen lagen zwischen .482 und .501). So kann man die optimale Zufallszahlenverteilung bestimmen und diese dann später immer einsetzen.

Doch nicht nur Zufallszahlen, auch Bildschirmspeicherabschnitte (sofern nicht durch SAVE "...",S machbar) oder selbstentworfene Zeichensätze können so abgespeichert werden. Auch in der seriellen Datenübertragung kann eine sequentielle Datei von Nutzen sein. In der Dateiverwaltung werden sequentielle Dateien eigentlich nur dann verwendet, wenn man mit Kassettenrekordern arbeitet. Im Normalfall wird im Diskettenbetrieb eine Random-Datei erstellt.

In den nächsten Folgen werden wir noch ein kleines Dateiprogramm für Kassette erstellen, dann geht es ab zu den Random-Dateien.

Das CP/M-Betriebssystem am SVI-328

4.2.3. Der Consol-Command-Prozessor

In den folgenden CP/M-Beiträgen werden jene Speicherbereiche Gegenstand unserer Betrachtungen sein, die den eigentlichen Programmcode des CP/M-Betriebssystems ausmachen.

Allgemeines

Aufgrund der tatsächlichen Aufteilung im Speicher beginnen wir mit dem an die TPA anschließenden Programmteil CCP. Der auch als Kommando-Interpreter bezeichnete Teil des CP/M-Betriebssystems ist das einzige Modul, daß von vornherein Befehle in Textform über die Tastatur versteht. Alle anderen Module, wie BDOS und BIOS benötigen Speicherzellen und CPU-Register als Übergabeort für Aufträge an das CP/M.

Der CCP spielt, wenn auch zeitlich begrenzt, eine sehr zentrale Rolle. Er ist es, der je nach gewähltem Laufwerk das Bereitschaftszeichen "A>" bis "P>" auf den Bildschirm bringt und ab diesem Zeitpunkt, wenn auch nicht viele, dafür aber sehr effiziente Befehle entgegennimmt und interpretiert.

Während des allerersten Dialogs mit CP/M, sei es nach dem Einschalten des Computers oder nach einem Kalt- oder Warmstart, kommt man um den CCP nicht herum. Man wird ihm regelmäßig begegnen und ihn, vor allem dann, wenn er uns durch eingebaute Funktionen mühsame Tipparbeit zu ersparen beginnt, schätzen lernen. Das bedingt aber, wie bei allen "Betriebssystemen", genaues Kennenlernen der Eigenheiten des CCP's.

Eigentlich stellt der CCP ein in sich geschlossenes "Mini-Betriebssystem" dar, das die ersten Eingabevorgänge, meistens bis ein gewünschtes Programm von der Diskette läuft, kontrolliert und ausführt.

Solche Programmodule werden auch gerne als "Monitor" bezeichnet. Die Bezeichnung "Interpreter im Direktmodus", wie wir sie vom BASIC kennen, kommt hier auch in Frage. Schließlich setzt er von der Tastatur kommende Befehle sofort in eine Operation um.

Aufgabengebiet

Die Hauptaufgaben des CCP's bestehen im Entgegennehmen bestimmter Befehle und dem Laden von Programmen in die TPA und deren Aufruf. Weiters sorgt der CCP noch für die Übergabe von eventuell angegebenen weiteren Filenamen und Parametern in bestimmte Puffer für ein gerade aufgerufenes Programm.

Eine Automatisierung von wiederkehrenden gleichlautenden Eingabeprozessen, Bedienungsvereinfachungen für den Benutzer usw., ergibt sich durch die Möglichkeit, Anweisungen an den CCP vom CCP selbst aus einer Datei lesen, und diese durch ihn hintereinander abarbeiten zu lassen.

In weiterer Folge, unter Zuhilfenahme eines Hilfsmoduls, das unterhalb des CCP's plaziert wird und ihn gleichzeitig unterstützt, ist auch eine automatische Kommunikation durch den CCP direkt mit Programmen, in deren individueller Befehlssprache, erlaubt.

Diese zur Verfügung stehenden CCP-Fähigkeiten geben dem Programmierer die Möglichkeit, zum Einen den Endbenutzer vor oft sehr komplizierten Eingabevorgängen zu verschonen, zum Anderen oft wiederkehrende Anweisungsprozesse, wo sich meist nur Namen von Dateien ändern, auf minimale Aufrufe über die Tastatur zu beschränken.

Da uns der CCP nicht vor dem "Ausschauhalten" nach einem solchen zuletzt genannten Arbeitsauftrag zur Verfügung steht, beginnen wir der ablaufmäßigen Reihenfolge entsprechend mit der Stapelverarbeitung und heben uns die Befehlsbeschreibung und anderes für später auf, zumal die Bedienung des CCP's sowieso schon den meisten CP/M-Anwendern geläufig ist.

Stapelverarbeitung

Bevor überhaupt eine direkte Kommunikation zwischen Benutzer und CP/M stattfinden kann, überprüft der CCP immer nach einem Kalt- oder Warmstart das Vorhandensein einer Datei mit der Bezeichnung "$$$.SUB" auf der Diskette. Das Erkennen einer so genannten Datei veranlaßt den CCP die darin in einem genau definierten Format befindlichen Befehle der Reihe nach ohne Zutun des Benutzers abzuarbeiten.

So eine Datei, in der Regel erst von laufenden Programmen erzeugt, wird, wie wir sehen werden, in der Praxis durch verschiedene Arbeitsvorgänge erzeugt. Einerseits kann so ein Arbeitsauftrag an den CCP in Form einer "$$$.SUB"-Datei durch jedes CP/M-Programm, aber auch durch ein Hilfsprogramm unter Zuhilfenahme einer Hilfsdatei erstellt werden. Beide Möglichkeiten werden hier untersucht.

Bedingung für die Einleitung eines Stapelauftrages, auch Batch-Betrieb genannt, ist hiebei immer das Vorhandensein einer Textdatei namens "$$$.SUB".

Diese Datei kann bis maximal 128 Befehlszeilen mit je 128 Zeichen beinhalten. Jede Zeile muß genau das Format besitzen, wie wir direkt über die Tastatur mit dem CCP zu "sprechen" gewohnt sind. Allerdings muß, genau wie beim später zu beschreibenden CCP-Befehlsbuffer, ein Zeichenzähler vor die Zeichenkette, und eine den Befehlstext abschließende Null hinzugefügt werden. Das ergibt somit eine maximale Länge der Eingabezeile von 126 Zeichen, die aber in der Praxis gar nicht ausgenützt wird.

Die Eingabezeile inklusive Zählerbyte darf, auch wenn sie kürzer als das Maximum ist, nur am Beginn eines 128 Byte-Blocks stehen. Das entspricht einem logischen Sektor von CP/M (Record). Dieser Umstand ist vor allem zu berücksichtigen, wenn man selbst durch ein Programm meherere Eingabezeilen generieren will. Zu berücksichtigen ist, daß der CCP einen Stapelauftrag aus organisatorischen Gründen von "hinten nach vorne" bearbeitet. Der Programmierer muß deshalb mit der ersten Eingabezeile am Beginn des letz-

ten Records der "$$$.SUB"-Datei beginnen. Bei einer einzigen Befehlszeile, was bei Programmsteuerung die Regel ist, stellt sich dieses Problem nicht.

An dieser Stelle sei darauf hingewiesen, daß zwischen zwei Kalt- bzw. Warmstartvorgängen, immer nur eine Befehlszeile abgearbeitet wird. Zu diesem Zweck wird der Recordzähler von "$$$.SUB" im Directory um 1 reduziert und somit verhindert, daß der CCP eine Zeile mehrmals interpretiert. Somit ist die "$$$.SUB"-Datei vor dem "Abstottern" der nächsten Befehlszeile einfach um einen Record kürzer. Das erklärt auch, warum von hinten mit dem Abarbeiten begonnen wird. Um den Stapelauftrag zu Ende zu bringen, werden also genauso viele Kalt- oder Warmstarts benötigt, wieviele Befehlszeilen sich anfänglich in der Arbeitsdatei befinden.

Diese Befehlszeilen werden unmittelbar nach Auffinden einer solchen Datei vom CCP interpretiert und abgearbeitet. Unterbrochen kann dieser Vorgang der Stapelverarbeitung jederzeit durch Drücken irgendeiner Taste, allerdings nur zwischen der "Befehlsholphase", werden.

Direkte Erzeugung von "$$$.SUB"

Will man zum Beispiel aus einem Programm ein anderes Programm von der Diskette zur Ausführung bringen, um aber nach Abarbeitung dieses wieder automatisch das ursprüngliche Programm zu starten, muß das aufrufende Programm eine Datei namens "$$$.SUB" auf die Diskette schreiben. Diese Datei darf dabei lediglich den Namen des gewünschten Programms beinhalten.

Wird nun das zuletzt aufgerufene Programm durch einen Kalt- oder Warmstart verlassen, findet der neu in den Speicher geladene CCP eine Datei mit der Bezeichnung "$$$.SUB" vor. Da der Inhalt dieser Datei nur, dem Aufruf eines beliebiegen Programms entsprechend, einen Namen beinhaltet, wird dieses, so als wäre gerade eine entsprechende Anweisung über die Tastatur gekommen, in den Speicher gelesen und zur Ausführung gebracht.

Vorher wird aber noch die Datei "$$$.SUB", beinhaltet sie, was hier der Fall ist, keine weiteren Befehlszeilen mehr, gelöscht, um dem CCP bei Bedarf einen Neubeginn eines sogenannten Stapelauftrags zu ermöglichen.

Das Utility-Programm "POWER" wendet diese Methode an. Es erlaubt, mit dem Befehl "RUN" ein beliebieges anderes Programm von Diskette zu starten. Nach Abarbeitung dieses, übernimmt "POWER" wieder die Kontrolle über den Computer.

Um das zu "erzwingen", wird vor dem Start des gewünschten Programms von "POWER" eine Datei "$$$.SUB" erzeugt, die lediglich im einzigen Record den Programmnamen "POWER" beinhaltet, der den CCP zum Laden dieses Programms in die TPA und dessen Start veranlasst.

Der Inhalt der von "POWER" erzeugten "$$$.SUB"-Datei sieht wie folgt aus:

```
0000 05 50 4F 57 45 52 00 .....   .POWER....
```

Das erste Byte im Sektor (128 Byte) beinhaltet immer die Länge der gültigen Eingabezeile. Das abschließende Byte muß eine Null sein. Der Inhalt des Records nach der Null ist nicht mehr von Bedeutung.

Bei Betrachtung mit dem Disk-Editor kann der Rest der "$$$.SUB"-Datei wirre Zeichen beinhalten, je nach dem ob der Disk-Buffer beim Erzeugen der Datei vor dem Beschreiben vollständig gelöscht wurde.

Beinhaltet eine solche Arbeitsdatei "$$$.SUB" jedoch mehrere Eingabezeilen, wird vor der Abarbeitung der aktuellen Befehlszeile der Recordzähler um 1 vermindert. Das ist jener Zeiger im Directory, der Aussage über die Größe einer Datei macht.

Ein neuerlicher Warm- oder Kaltstart, der praktisch durch jede Beendigung eines Programmablaufs ausgelöst wird, veranlaßt den CCP neuerlich nach "$$$.SUB" zu suchen.

Zu diesem Zeitpunkt ist die Arbeitsdatei bereits um einen Record kürzer geworden, sodaß die "nächstfolgende" Befehlszeile als Auftrag herangezogen wird. Dieser Vorgang wiederholt sich solange, bis keine Anweisungen vorgefunden werden können, also der Recordzähler Null erreicht hat. Abschließend wird der Name der Arbeitsdatei aus dem Directory auf der Diskette gelöscht und der Stapelauftrag gilt als beendet.

Ein vorzeitiger Abbruch erfolgt, wie schon oben erwähnt, durch Druck auf eine beliebige Taste, aber auch bei fehlerhaften Anweisungen. Der Vorgang der Stapelverarbeitung kann optisch mitverfolgt werden, da der CCP alle von der Arbeitsdatei empfangenen Befehle zuerst auf der Konsole (Bildschirm) ausgibt.

Da der CCP Eingabezeilen, die zum Beispiel mit einem Sonderzeichen wie Doppelpunkt, Gleichheitszeichen, Fragezeichen, Punkt und anderen Zeichen beginnen, nicht als Befehlszeile aber auch nicht als Fehler interpretiert werden, kann sich der Programmierer diese Tatsache zunutze machen, um dem Benutzer Meldungen oder Anweisungen über die Konsole zu geben.

Nachteil dieser einfachsten Form der Stapelverarbeitung ist, die ja auch nur von einem Programm direkt genutzt wird, daß bei jeder Eingabezeile neben dem CCP-Befehl auch schon der Name der jeweils anzusprechenden Datei bekannt sein muß.

Diese Einschränkungen ändern sich schlagartig mit folgendem auf der CP/M-Betriebssystemdiskette mitgelieferten Programm, das eine Erweiterung der CCP-Fähigkeiten darstellt. Es wird wie ein normales Programm mit der Angabe einer vorbereiteten Hilfsdatei und entsprechenden Parametern aufgerufen.

SUBMIT

Im Zusammenhang mit diesem Programm ist es nun zu jedem Zeitpunkt möglich von Hand aus die vorhin beschriebene "$$$.SUB"-Datei durch erst beim Aufruf bekannte Variablen zu erweitern. Diese sind in der Regel Laufwerksbezeichnungen und Teile von Dateinamen, die erst kurz vor dem Abarbeitungsvorgang bekannt sind. Damit ergibt sich eine wesent-

lich flexiblere Nutzung, vor allem dann, wenn man selbst eine bestimmte Stapelverarbeitung auszulösen beabsichtigt.

Um diesen erweiterten Stapelverarbeitungsprozeß zu ermöglichen, muß eine Datei mit der Dateitypkennzeichnung "SUB", nicht zu verwechseln mit "$$$.SUB", auf der Diskette abgelegt sein. Diese wird durch einen "ganz normalen" Texteditor erstellt.

In dieser Datei werden wie gewohnt alle gewünschten CCP-Anweisungen festgehalten. Allerdings braucht hier nicht mehr ein Zeichenzähler und eine abschließende Null eingetragen werden. Diese Arbeit erledigt "SUBMIT" beim Erstellen der nach wie vor für den CCP erforderlich "$$$.SUB"-Datei.

Ist beim Festlegen der jeweiligen Befehlszeile bewußt noch nicht klar, welche Datei hier in den Verarbeitungsprozess eingebunden werden soll, wird an dieser Stelle einfach ein "$" und eine nachfolgende Nummer als Variable eingesetzt.

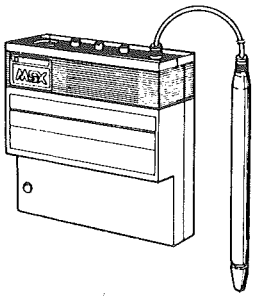
Das Vorfinden dieser Information teilt dem Programm "SUBMIT" beim Erstellen der für den CCP notwendigen Datei "$$$.SUB" mit, daß er hier der Nummerierung der Variablen entsprechend beim Aufruf von "SUBMIT" mitgegebene Parameter einzusetzen hat.

Diese wesentliche Erweiterung unter Zuhilfenahme des Programms "SUBMIT", eines "SUB-Files" und der Angabe von jeweils aktuellen Parametern, kann dem Benutzer eine erhebliche Zahl von CCP-Anweisungen ersparen. Vor allem dann, wenn es sich bei der Aufgabenstellung um sich immer wiederholende Programmabläufe handelt, wo sich lediglich die zu bearbeiteten Programmnamen ändern.

Im nächsten Beitrag generieren wir als Beispiel eine Datei "COMP.SUB", die hier für einen Compilierprozeß von Maschinencode-Programmen vorbereitet werden soll.

```
*****************************************
*                                       *
*      LICHTGRIFFEL FÜR MSX-COMPUTER    *
*                                       *
*****************************************
```

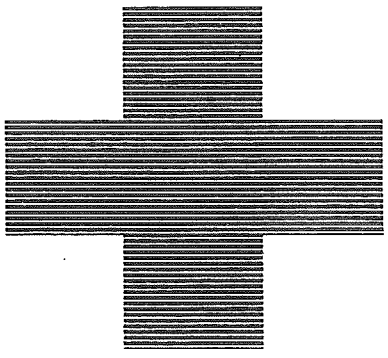
Auf der Wiener Herbstmesse war als Premiere in Österreich zum ersten Mal ein Lichtgriffel für MSX zu sehen. Das SVI-Journal hat sich gleich um ein erstes Exemplar dieses Gerätes bemüht, ein ausführlicher Testbericht folgt in der nächsten Nummer.



Angeschlossen wird der Lichtgriffel wie ein Modul im Modulschacht. Dieses Modul beherbergt neben der Lichtgriffel-Hardware noch etwa 7 Kbyte ROM, in dem neben einem komfortablen Zeichenprogramm noch einige BASIC-Erweiterungen abgespeichert sind. So wird zum Beispiel das Zeichnen von dicken Linien ermöglicht und es gibt ein sogenanntes "SUPERPAINT", bei dem die Farbe, mit der man ausmalen will, nicht mit der Randfarbe der auszufüllenden Figur übereinstimmen muß.

Das Zeichenprogramm bietet neben zahlreichen Befehlen, wie z.B. Zeichnen von Linien, Rechtecken und Kreisen, Scrollen von Bildern, Vergrößern von Bildauschnitten auch die sehr interessante Möglichkeit, Bilder als BASIC-Programme abzuspeichern.

Erstmals in Österreich



Österreichisches Rotes Kreuz stellt "Stand-by"-Teams für Katastrophenfälle im Ausland

Bei Naturkatastrophen ist es unbedingt notwendig, qualifiziertes Personal ins Katastrophengebiet zu entsenden, um an Ort und Stelle das Ausmaß der Katastrophe festzustellen und so rasch wie möglich Hilfsaktionen für die Opfer einzuleiten.

Die Liga der Rotkeuzgesellschaften hat deshalb kürzlich das System der "Stand-by"-Teams ins Leben gerufen, um bei Katastrophenfällen rasche Hilfeleistung zu gewährleisten. 12 nationale Rotkreuzgesellschaften sind übereingekommen, während einer Zeitspanne von einem Monat gut vorbereitete Teams ständig in Bereitschaft zu halten. Im September wird vom Österreichischen Roten Kreuz dieses Team gebildet. Ein vollständiges Team besteht aus einem Mannschaftsführer, einem Arzt, einem Logistiker und einem Informationsdelegierten. Im Ernstfall hat dieses Team unter anderem zwei wesentliche Aufgaben zu verfolgen:

Die Mannschaft muß sich sofort in das Katastrophengebiet begeben, um an Ort und Stelle einen Überblick über das Schadensausmaß zu bekommen und in Zusammenarbeit mit der Liga in Genf erste Hilfsmaßnahmen in die Wege zu leiten. Ferner hat das Team die Aufgabe, die Planung und Anschaffung von Hilfsgütern und deren Transport sowie Lagerung und Verteilung zu unterstützen.

```
*****************************************
*                                       *
*                PILOT                  *
*                                       *
*****************************************
```

TURBO-Pascal hat sowohl durch seine Leistung als auch dadurch, daß die Werbetrommel dafür kräftig gerührt wurde, die anderen unter CP/M für die SVI-Computer erhältlichen Programmiersprachen ziemlich in den Hintergrund gedrängt. FORTRAN, COBOL, C und LISP sind wenigstens dem Namen nach den Meisten geläufig, ein Aschenbrödeldasein führt jedoch PILOT.

Was dahinter steckt

PILOT ist - wie so viele andere Namen in der Computerei auch - ein Akronym. Und zwar steht es für:

Programmed
Inquiry
Learning
Or
Teaching

was auf Deutsch soviel heißt wie "Programmierte Befragung, Programmiertes Lernen oder Unterrichten". PILOT wurde von John A. Starkweather - einem Doktor der Philosophie - an der Universität von Kalifornien in San Francisco entwickelt und liegt unter CP/M bereits in der Version 5.0 (!) vor. Es wird von der amerikanischen Firma Ellis Computing vertrieben.

PILOT ist eine stark text-orientierte Programmiersprache, die es Leuten ohne vorherige Computererfahrung möglich machen soll, interaktive Programme zur Wissensvermittlung und zur Überprüfung des Verstehens dieses Wissens zu schreiben. Es gibt eine Vielzahl möglicher Programmthemen: Ein PILOT-Programm könnte zum Beispiel eine mathematische Aufgabe präsentieren, wobei das Programm dem Schüler weiterhelfen kann, wenn er steckenbleibt. Ebenso ist ein Vokabelquiz möglich, oder sogar ein Programm, das das Programmieren in PILOT lehrt! Selbst Spiele mit Text-Schwerpunkt, zum Beispiel das bekannte Psychiaterprogramm ELIZA oder ein Adventure können leicht erstellt werden (ein Spiel, bei dem ähnlichklingende - englische - Wörter aus einem Buchstabenquadrat herausgefunden werden sollen, befindet sich auf der PILOT-Diskette).

Wie BASIC ist PILOT eine Interpreter-Sprache. Sie verfügt über einen eigenen bildschirmorientierten Editor, es ist somit nicht wie zum Beispiel bei Assembler oder FORTRAN notwendig, zuerst den Programmtext mit einem separaten Editor zu erstellen und ihn erst dann ausführen zu lassen, sondern die Programme können sofort getestet und ohne mühseliges Programmwechseln geändert werden.

Die Installation von PILOT ...

ist für SVI-Besitzer sehr einfach. Es wird ein Installationsprogramm mitgeliefert, das nicht nur die Anpassung des Programmes an einen bestimmten Bildschirm vornimmt (beim SVI nicht nötig, da bereits von Haus aus auf ein Zenith Z-19 Terminal - entspricht 80 Zeichenkarte - vorinstalliert), sondern auch die Auswahl der Autor- oder der Student-Version und beim DIR-Kommando die Anzeige entweder aller Dateien oder nur solcher mit der Extension .PLT ermöglicht (Der Unterschied zwischen der Autor- und der Student-Version liegt darin, daß in letzterer Editier- sowie möglicherweise zerstörend wirkende oder - zur falschen Zeit - "wissenserweiternde" direkte Kommandos nicht erlaubt sind). Mein Mitgefühl gehört denjenigen, die PILOT für einen im Auswahlmenü nicht aufgeführten Bildschirm installieren müssen, da die Änderung eines falsch eingegeben Wertes gleichbedeutend ist mit "Und jetzt das Ganze nochmal von vorne".

Es gibt noch ein weiteres Installationsprogramm, das für die Tastenbelegungen der EDIT-Modus-Funktionen zuständig ist. Der Original-PILOT ist nicht hundertprozentig WORDSTAR-kompatibel, das kann aber mit Hilfe dieses Programmes leicht behoben werden.

Die dritte Quelle für Änderungen am PILOT-Interpreter bildet die Datei "NVPILOT.PRN". Sie enthält wichtige Adressen, die mit Hilfe eines DDT oder ZSID dem vorhandenen System beziehungsweise eigenen Vorstellungen angepasst werden können. Unter anderem sind dies die Kontrolltasten für die Editor-Steuerung, Escape-Sequenzen für den Bildschirm und die Fehlermeldungen.

Der Kern der Sache

Es gibt drei verschiedene Modi in PILOT. Der erste ist der Direkt-Modus, in dem zum Beispiel das Inhaltsverzeichnis einer Diskette angezeigt, eine Datei geladen, oder ein Programm ausgeführt werden kann. Der zweite ist der Editier-Modus, in dem die PILOT-Programme entwickelt werden, und der dritte ist der Programmodus. In diesem werden die nachfolgend beschriebenen Befehle ausgeführt.

Die wichtigsten Anweisungen in PILOT (die sogenannten "Core-Instructions", also die "Kernanweisungen") werden alle durch ihre Anfangsbuchstaben abgekürzt, um Schreibarbeit zu sparen.

Und zwar sind dies:

```
T (TYPE):      Text auf den Schirm schreiben
A (ACCEPT AN ANSWER): auf Antwort warten
M (MATCH):     Antwort mit vorgebenen Werten
               vergleichen
J (JUMP):      Sprungbefehl
U (USE):       Unterprogrammaufruf
E (END):       Beenden eines Unter- oder des
               Hauptprogramms
C (COMPUTE):   Berechnungsanweisung
R (REMARK):    Programmkommentar
```

Alle PILOT-Anweisungen werden durch einen Doppelpunkt abgeschlossen. An einige der Befehle können zusätzliche Buchstaben angehängt werden, die die Befehle abwandeln, erweitern oder als logische Operatoren wirken. (So gibt zum Beispiel der Befehl TY: seinen Text nur dann aus, wenn ein zuvor durchgeführter Vergleich erfolgreich war, das heißt, eine eingegebene Antwort der Erwartung entsprach). Diese Anweisungen bilden die am häufigsten gebrauchten Befehle jedes PILOT-Programms. Ein kleines (banales) Beispiel:

```
T:Lesen Sie gerne?
A:
M:nein,nicht
TY:Das ist schade!
JY:ENDE
TN:Was lesen Sie am liebsten?
A:
TN:Aha!
*ENDE
E:
```

Zuerst wird die Frage angezeigt und dann auf eine Eingabe gewartet. Diese wird auf das Auftreten der Worte "Nein" und "Nicht" geprüft, es werden Antworten wie "Nicht sehr", "Nicht besonders" oder "Gar nicht" ebenfalls erkannt, da sie alle "Nicht" beinhalten (die Eingabe wird automatisch in Großbuchstaben übersetzt, "nicht" und "NICHT" werden daher gleich behandelt). Sogar "Mitnichten" wird gefunden (Nicht gefunden werden hingegen Antworten wie "Nur Comics"). Wenn also eines der Worte in der Antwort gefunden wird, dann wird der hinter TY: stehende Text ausgegeben, und anschließend das Programm durch den Sprungbefehl JY:ENDE, der zum Label *ENDE springt, beendet. Ansonsten wird vor dem Programmende noch ein zusätzlicher Kommentar ausgegeben. Es ist auch möglich, einen Text auszugeben, wenn die Antwort keines der in der M-Anweisung stehenden Worte enthält. In diesem Fall benutzt man TN: (Alternativ kann man statt TY: und TN: Y: und N: verwenden).

Befehle wie FOR/NEXT oder REPEAT/UNTIL wird man in PILOT vergeblich suchen. Aufgrund der Konzeption dieser Sprache kann auf sie jedoch weitgehend verzichtet werden. Die Kontrolle des Programmflußes kann von den "Anhängseln" des P:- bzw. U:-Befehles gesteuert werden. So springt zum Beispiel J:§A zum letzten A: zurück.

Auch IF/THEN ist nicht vorhanden. Soll ein Befehl nur dann ausgeführt werden, wenn eine Bedingung erfüllt ist, so hängt man diese einfach an den Befehl an. Trifft der Interpreter zum Beispiel auf die Zeile

```
U(A>6):*STEP1
```

so wird das Unterprogramm STEP1 nur dann ausgeführt, wenn A größer als 6 ist.

PILOT bietet zwei Arten von Variablen, numerische und alphanumerische. Zu den Stringvariablen ist an sich nicht sehr viel zu sagen, außer daß ihre Länge durch den INMAX-Befehl festgelegt werden kann (INMAX 1 würde daher der BASIC-Anweisung INKEY$ entsprechen). Die numerischen Variablen belegen je zwei Bytes im Speicher, das heißt, Rechenoperationen beschränken sich auf ganze Zahlen im Bereich von -32768 bis 32767 (warum schrecken eigentlich so viele Softwarehersteller davor zurück, für ihre Programmiersprachen Floating-Point Arithmetikroutinen zu schreiben?), an Rechenfunktionen stehen die Grundrechnungsarten und die Rest-Funktion zur Verfügung. Umfangreicher sind die Vergleichsoperationen. Sie beinhalten AND, OR, NOT sowie die Größer-, Kleiner- und Gleich-Operatoren und deren Kombinationen.

In PILOT ebenfalls möglich ist das Arbeiten mit Protokolldateien. In diesen können Antworten und Punktewerte des Lernenden sowohl zur Selbstkontrolle als auch zur Information des Lehrers abgespeichert werden.

Einen wichtigen Rang nehmen die Funktionen zur Ein- und Ausgabe sowie zur Gestaltung des Bildschirmes ein. Ein Text kann zweifellos leichter aufgenommen werden, wenn er übersichtlich präsentiert wird, als wenn Teile von früheren und aktuellen Fragen bunt durcheinandergemischt sind. PILOT bietet hier verschiedene Befehle zum vollständigen oder zeilenweisen Löschen des Bildschirms sowie zum Positionieren des Cursors an eine bestimmte Stelle an. Der Editier-Modus enthält einen speziellen Steuerbefehl (Ctrl O) zur inversen Darstellung von Texten (In diesem Zusammenhang ist auch erwähnenswert, daß PILOT über Anweisungen zur Steuerung eines - speziellen - Videorekorders bzw. eines Bildplattenspielers verfügt. Im Handbuch befindet sich eine Beschreibung, wie diese Geräte mit dem Computer zu verbinden sind.).

Die "Spezialfunktionen" habe ich mir für den Schluß aufgehoben. Zu diesen gehören XI:, das eine Stringvariable als PILOT-Befehl interpretiert und ausführt, XS:, mit dem COM-Dateien aufgerufen werden können, SET:, das die Einstellung der Anzeigegeschwindigkeit, die optionale Ausgabe des Bildschirminhaltes an den Drucker und die Angabe der Größe des für PILOT zur Verfügung stehenden freien Speichers erlaubt. PEEK und POKE brauchen, glaube ich, nicht erklärt zu werden, und mit CALL: kann ein Maschinencode-Programm im Speicher aufgerufen werden (diese Funktionen haben allerdings einen Nachteil: aufgrund der zum Rechnen verwendeten Zahlendarstellung müssen die Speicheradressen im "signed two's complement mode" angegeben werden. Will man daher eine Speicheradresse größer als 32767 - 7FFF hexadezimal - als Argument verwenden - z. B. 53419, dieser Wert ist allerdings willkürlich gewählt - so muß man zuerst 65536 abziehen und den negativen Wert - in unserem Fall -12117 - als Argument angeben. Die Verwendung von Papier und Bleistift oder einem Taschenrechner bleibt also nicht erspart).

Abgesehen von den bereits oben erwähnten Programmen befinden sich noch die folgenden auf der PILOT-Diskette:

WAPP, ein in PILOT geschriebener PILOT-Programmgenerator, der im Dialog mit dem Anwender ein sogenanntes "Multiple Choice"-Programm erzeugt (Multiple Choice bedeutet, daß eine Frage sowie eine bestimmte Anzahl von möglichen Antworten vorgegeben wird - bei diesem Generator bis zu drei - von denen jedoch nur eine richtig ist). Es ist möglich, zu den einzelnen Antworten jeweils einen erklärenden Kommentar auszugeben; weiters eine Sammlung von Programmen, mit denen das Funktionieren des Interpreters getestet werden kann.

Die Literatur zu PILOT

Ein Büchlein (es ist zu dünn, um Buch genannt zu werden), das sich zwar hauptsächlich mit computerunterstütztem Unterricht beschäftigt, jedoch auch PILOT ein paar Seiten widmet, ist

Martyn Sibley
Computer Assisted Learning
Century Communications

Fortsetzung auf Seite 15

Doublesided Disk-BASIC

Alle diejenigen unter Ihnen, werte Leser, die keinen SVI-605B mit dopppelseitigen Laufwerken ihr eigen nennen, können weiterblättern.

Da Sie nun in diesem Absatz angelangt sind, haben Sie wahrscheinlich einen SVI-605B oder sind eine Leseratte.

Das CP/M 2.23 des SVI-605B kann ja bekanntlich 80 Tracks zu je 17 physikalischen oder 34 logischen Sektoren verwalten. Somit sind etwa 320 Kbyte unter CP/M frei verfügbar. (17 * 256 * 80 / 1024 = 340, 4 Systemtracks abgezogen, ergibt 323 KByte.)

In Verbindung mit dem Textverarbeitungsprogramm WORDSTAR verbleiben dann noch etwa 260 Kbyte auf der Disk, im Gegensatz zu mageren 100 Kbyte bei einseitigen Disketten. Und 100 Kbyte sind schnell verbraucht, zumal WS auch Sicherheitskopien von Dateien anfertigt.

Doch unter Disk-BASIC hatte sich nichts geändert, es kam kein erweitertes Disk-BASIC, das 80 Tracks verwalten kann. Also waren unter BASIC wieder nur maximal etwa 600 Datensaetze zu 256 Bytes möglich.

Anfang dieses Sommers hatte ich dann die Gelegenheit, unser Disk-BASIC zu "zerpflügen" und seine Arbeitsweise zu durchschauen. Anhand meiner Unterlagen gelang es mir dann zu Schulbeginn, das Disk-BASIC zu erweitern. Probeweise hatte ich eine Testdatei mit 1200 Datensätzen erzeugt oder 76, allerdings nur jeweils einen Track lange Programme auf die Diskette überspielt. Alle diese Tests endeten positiv.

Wie wird nun ein BASIC-System doppelseitig? Dazu sollten wir uns das Installationsprogramm zu Gemüte führen!

Das Installationsprogramm besteht aus mehren wichtigen Teilen. Die Beschreibungen der Routinen beziehen sich auf die PRINT-Befehle im Programm.

1) "Verbiege Zeiger von Diskblock 2"

Innerhalb des Disk-BASIC sind für jede Diskettenstation 45 Bytes grosse Buffer freigehalten, ingesamt also 90 Bytes. Fünf Bytes von jedem der zwei Buffer enthalten Informationen über die Diskette, die restlichen 40 Bytes bilden den Allocation Table, der die Trackbelegung wiedergibt. Er wird immer aktualisiert und gegebenenfalls auf die Sektoren 15, 16 und 17 der Spur 20 geschrieben oder von dort gelesen.

Um nun 80 Track verwalten zu können, muss der Buffer von 45 auf 85 Bytes vergrössert werden. Allerdings liegen die 90 Bytes innerhalb des Disk-BASIC, woraus resultiert, daß der zweite Buffer verschoben werden muß. Dazu wird der Zeiger vom zweiten Buffer "verbogen" sodaß der Buffer nun bei &HF380 liegt.

2) "Lösche Diskparameterblock 2"

Hier werden die Informationen des Diskparameterblocks - die 5 Bytes von oben - so initialisiert, daß zum Beispiel bei einem OPEN "2:" der AT (Allocation Table) neu gelesen wird. Somit wird verhindert, dass der interne AT mit dem auf der Diskette 2 nicht übereinstimmt.

3) "Sprung auf Density/Select"

Das Disk-BASIC bedient sich nur einer Routine zur direkten Ein- und Ausgabe auf Disketten. In dieser Routine wird dann letztendlich die Disknummer, Track und Sektor eingestellt und dann die Operation durchgeführt. Bei &HD9C1 liegt die Routine, um den Track einzustellen.

Hier baue ich einen Sprung in eine eigene "Track Select"-Routine ein, die dann in der Lage ist, bei einer Trackzahl grösser 39 die zweite Seite einzustellen.

4) "Density/Track Selectroutine"

Hier wird die eigentliche Routine auf die Systemspuren geschrieben. Die Routine wird ab &HF350 in den Speicher geladen, sie behindert die Erweiterung aus Heft 4/85 nicht.

5) "Trackzähler = .. + 40"

Im Disk-BASIC gibt es einige Zähler mit den Werten 39 oder 40. Insofern Sie mit Diskettenbehandlung etwas zu tun haben, müssen Sie natürlich auf 79 beziehungsweise auf 80 erweitert werden.

6) "Erweitere AT um 40 freie Tracks"

Hier werden auf alle Fälle die Tracks auf der zweiten Seite freigegeben. Diese Änderung muss auf den Sektoren 15, 16 und 17 vorgenommen werden.

7) Nach dem "Alles okay" ist die Diskette im Laufwerk 1 für doppelseitigen Betrieb geeignet.

Um nun eine Diskette doppelseitig verwenden zu können, müssen Sie folgende Schritte unternehmen:

1) Eine Diskette unter CP/M mit FORMAT formatieren.

2) Unter CP/M mit SYSGEN ein herkömmliches Disk-BASIC überspielen.

3) Unter Disk-BASIC neue Diskette im Laufwerk 2 mit dem in BASIC geschriebenen FORMAT-Programm logisch formatieren. Also RUN "1:format" und dann 2 eingeben.

4) Mein Installationsprogramm laden, Diskette aus Laufwerk 2 in Laufwerk 1 legen und RUN eingeben.

5) Danach haben Sie im Laufwerk 1 eine fertige doppelseitige Systemdiskette.

Wenn Sie nun diese Diskette booten, und dann einmal probeweise ?DSKF(1) eingeben, sollten Sie den Wert 76 erhalten.

Um doppelseitige Disketten neu zu formatieren, sollten Sie das FORMAT-Programm des BASIC-Systems wie folgt ändern:

```
load "1:format"
360 LSET A$= STRING$(3,254)+STRING$(17,255)+
CHR$(254)+STRING$(60,255)
save "1:format"
```

Dann müssen Sie die zu formatierende Diskette ins Laufwerk 2 stecken und formatieren.

Wenn Sie nun bisher mit COPY 2 FROM 1 Ihre Disketten kopiert haben, so können Sie dies natürlich auch unter dem neuen Disk-BASIC tun. Es dauert nur länger.

Auch die Trackangaben bei COPY und DSK.$ können jetzt von 0 bis 79 variieren.

Das neue Disk-BASIC ist natürlich kompatibel zum alten, Sie können ohne weiteres Programme von einseitigen auf doppelseitige Disketten überspielen. Das funktioniert so:

1) Doppelseitiges Disk-BASIC booten.

2) Einseitige Diskette ins Laufwerk 2 stecken und unbedingt FILES 2 eingeben, um dem Computer mitzuteilen, daß nun eine einseitige Diskette im Laufwerk 2 ist.

3) Mit COPY-Befehl die Dateien von 2 auf 1 kopieren.

Nur eines dürfen Sie nie: Mit einem einseitigen Disk-BASIC auf eine doppelseitige Diskette speichern oder umgekehrt. Nur Lesen vom anderen Format ist möglich, Speichern ist verboten.

Es gibt allerdings eine Ausnahme: Wenn Sie ein kleines Programm im Speicher haben und doppelseitiges Disk-BASIC fahren, so können Sie durchaus auch auf eine einseitige Diskette speichern insofern auf der noch genügend Platz ist.

Ansonsten versucht der Computer dann ja auf die zweite Seite zu schreiben und das geht nicht, weil diese ja nicht formatiert ist. Oder aber er stürzt ab, weil im zu kurzen AT eventuell widersprüchliche Informationen stehen.

Ohne Ausnahme dürfen Sie aber nie unter einseitigem Disk-BASIC auf doppelseitige Disketten speichern, da dann die Informationen über die zweite Seite im AT verloren gehen und dann das doppelseitige Disk-BASIC einen 'Bad allocation'-Fehler erzeugen würde.

Unten ist das komplette Listing des Install-Programms zu sehen. Falls Sie sich die Tipparbeit ersparen möchten, können Sie sich natürlich das Programm von einer Clubdiskette kopieren.

```
100 '
110 ' 80 Tracks unter Disk-BASIC
120 ' fuer SV-605B
130 '
140 ' (c) 1985 by Philipp Ott
150 '
160 CLEAR 1000:DEFINT A-Z:RESTORE
170 FIELD #0,128 AS S$(1),128 AS S$(2)
180 PRINT
190 PRINT" Verbiege Zeiger von Diskblock
2
200 PRINT
210 TR=0:SE=12:GOSUB 900
220 MID$(S$(1),125)=CHR$(&H83)
230 MID$(S$(1),126)=CHR$(&HF3)
240 GOSUB 940
250 PRINT
260 PRINT" Lösche Diskparameterblock 2
270 PRINT
280 TR=2:SE=6:GOSUB 900
290 MID$(S$(2),1)=STRING$(2,0)
300 MID$(S$(2),4)=CHR$(255)
310 MID$(S$(2),5)=STRING$(3,0)
320 GOSUB 940
330 PRINT
340 PRINT" Sprung auf Density/Track Sele
ct
350 PRINT
360 TR=0:SE=12:GOSUB 900
370 MID$(S$(1),69)=CHR$(&HC3)
380 MID$(S$(1),70)=CHR$(&H50)
390 MID$(S$(1),71)=CHR$(&HF3)
400 GOSUB 940
410 PRINT
420 PRINT" Density/Track Selectroutine
430 PRINT
440 TR=2:SE=6:GOSUB 900
450 I=81
460 READ B$:IF B$="*" THEN 490
470 MID$(S$(1),I,1)=CHR$(VAL("&H"+B$))
480 I=I+1:GOTO 460
490 GOSUB 940
500 PRINT
510 PRINT" Trackzähler = .. + 40
520 PRINT
530 READ TR,SE,BY
540 IF TR=255 THEN 610
550 GOSUB 900
560 IF BY>128 THEN BY=BY-128:S=2 ELSE S=
1
570 B$=MID$(S$(S),BY,1):IF ASC(B$)³41 TH
EN PRINT ">"TR;SE;BY+(S-1)*128:GOTO 530
580 MID$(S$(S),BY)=CHR$(ASC(B$)+40)
590 GOSUB 940
600 GOTO 530
610 PRINT
620 PRINT" Erweitere AT um 40 freie Trac
ks
630 PRINT
640 TR=20:FOR SE=15 TO 17:GOSUB 900
650 MID$(S$(1),41)=STRING$(40,255)
660 GOSUB 940:NEXT
670 PRINT
680 PRINT" Alles okay
690 PRINT
700 END
710 '
720 ' Density/Select MC-Programm
730 '
740 DATA 6,0,3A,F5,D9,F5,FE,28,3F,CB,10
750 DATA FE,1,CB,10,78,D3,38,F1,D6,28
760 DATA 30,3,3A,F5,D9,47,DB,31,B8,C8
770 DATA 78,D3,33,3E,12,D3,30,C3,DB,D9
780 DATA *
790 DATA 1,1,197
800 DATA 1,2,79
810 DATA 1,11,48
820 DATA 1,12,18
830 DATA 1,16,206
840 DATA 1,17,37
850 DATA 1,17,162
860 DATA 255,0,0
870 '
880 ' Sektor lesen
890 '
900 DU$=DSKI$(1,TR,SE):RETURN
910 '
920 ' Sektor schreiben
930 '
940 DSKO$ 1,TR,SE:RETURN
```

SVI-738 X'press

Die Ankündigung

Frühjahr 1985
Auf der CES (Consumer Electronics Show) in Las Vegas sticht ein Computer aus der Menge der Geräte hervor. Sein Name ist "X'Press", sein Produzent Spectravideo. Das vorgestellte Gerät hat 64 KByte RAM (+ 16 KByte zusätzliches VRAM), vier eingebaute Anwenderprogramme und ein (ebenfalls eingebautes) 3.5 Zoll Diskettenlaufwerk. Sein Aussehen gleicht dem des Apple IIc.

Das Ergebnis

27. August 1985
Der neue tragbare MSX-Computer ist eingetroffen. Sein endgültiger Name ist SVI-738 X'press. Im großen Verpackungskarton befinden sich neben den obligaten Handbüchern (selbstverständlich in Englisch) noch zwei Disketten und eine Tragetasche. Mitgeliefert wird jedoch auch das deutsche SVI-728-Handbuch.

Das Zubehör

In dieser (fast eleganten) Tasche ist neben dem Computer auch noch Platz für das Netzteil, Monitorkabel und diverse Notizzettel. Die beiden Kabel des mitgelieferten Netzteils sind insgesamt 3.5 Meter lang, wobei die beiden Anschlußkabel so ziemlich gleich lang sind. Die Tasche kann sowohl wie ein Koffer als auch wie eine Umhängtasche getragen werden. Der Umhänggurt wird nach dem bereits bei Fototaschen bewährten Prinzip fixiert. Insgesamt macht die Tasche einen sehr stabilen Eindruck. Wenn Sie diese Tasche bei der Fahrt mit öffentlichen Verkehrsmitteln nicht benutzen, sollten Sie sich auch nicht über die neugierigen Blicke Ihrer Mitfahrer wundern.

Die Software

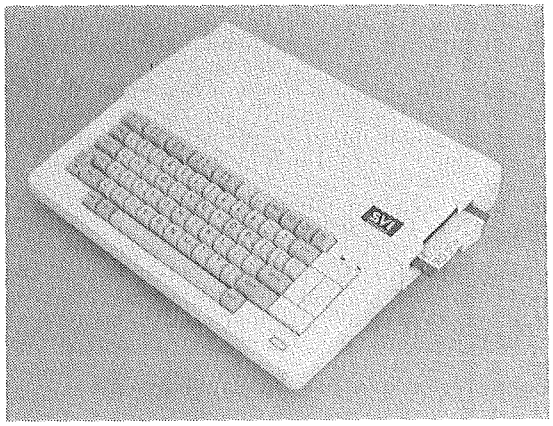
Auf den beiden mitgelieferten 3.5-Zoll-Disketten befinden sich das CP/M-System und MSX-DOS. Außer den üblichen CP/M-Programmen befinden sich noch folgende (praktische) Programme auf der Diskette:

FILECOPY.COM:
Mit diesem Programm können Sie einzelne Files (auch mit nur einem Laufwerk) kopieren. Es ist übrigens auch möglich, MSX-DOS-Programme zu kopieren.

BACKUP.COM:
Dieses Programm erleichtert das Erstellen von Backup-Disketten. Wiederum wird nur ein Laufwerk benötigt, wobei allerdings die Diskette mehrmals gewechselt werden muß. Mit zwei Laufwerken entfällt diese Arbeit selbstverständlich.

EDITFKEY.COM und LOADFKEY.COM
Diese beiden Programme dienen zur Veränderung der Funktionstastenbelegung. Zur Eingabe der neuen Belegung(en) ist ein kleiner Editor eingebaut. Gespeichert (geladen) können die Änderungen auf (von) der Disk oder im (aus dem) RAM werden.

RS232.COM:
Das Programm RS232.COM ermöglicht den Datenaustausch über die serielle Schnittstelle beziehungsweise über einen Akustikkoppler. Nach dem Aufruf dieses Programms gelangen Sie in ein Menü und können zwischen "file transfer mode", "conversation mode" und "parameter select" wählen. Auch bei den erste beiden Modi müssen noch einige Parameter eingestellt werden. Für besonders interessant halte ich den "conversation mode". Hierbei wird der Bildschirm in zwei Teile (Windows) geteilt. Auf der einen Hälfte werden die zu übermittelnden und auf der anderen Hälfte die übermittelten Daten angezeigt. In diesem Modus gibt es auch noch eine (lächerliche) Helpfunktion. Beendet wird der Dialog mit Control-C (^C).



Auf der MSX-DOS-Diskette befindet sich neben MSXDOS.SYS und COMMAND.COM noch ein Programm mit dem "alles sagenden" Namen MP04.COM. Hinter dieser Bezeichnung (Name ist hier das falsche Wort!) verbergen sich vier Utilities von denen drei in der CES-Version noch im ROM steckten. Eingebaut sind ein Memowriter, ein Spreadsheet und ein Programm zur Diskettenbearbeitung. Das vierte Programm ist das oben beschriebene RS232.COM. Die Steuerung erfolgt wieder über ein Menü oder über die Funktionstasten in den einzelnen Programmen. Diese Progamme können natürlich keinen WordStar oder kein Multiplan ersetzen, doch sind Sie durchaus für kleine Anwendungen geeignet. Beim Spreadsheet sind zum Beispiel theoretisch 245x48 Felder möglich.

Wie Sie sehen, wird eine ganze Menge (guter) Software mitgeliefert. Doch nützt die beste Software ohne die passende Hardware absolut nichts (und umgekehrt!). Aber auch die Hardware kann sich sehen lassen.

Die Hardware

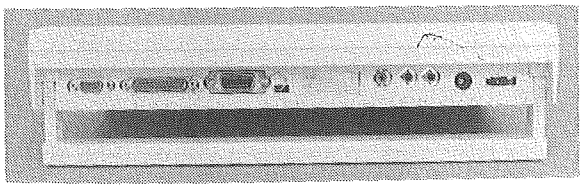
Der SVI-738 X'press hat bereits eine parallele (identisch mit der des SVI- 728) und eine serielle Schnittstelle (gleicht einem Joystickstecker) eingebaut. Neben diesen Anschlüssen befinden sich auch noch der Einbeziehungsweise Ausgang für ein zweites Diskettenlaufwerk (25 polig), ein Fernsehanschluß, die beiden Buchsen für die Monitorkabel (Audio und Video) und der Anschluß für das Netzteil unter dem schützenden Griff. Der Griff hat insgesamt drei Funktionen:

1.) Tragegriff

2.) Schutz bei der Lagerung oder beim Transport in der Tasche

3.) Stütze beim Arbeiten

Außer diesen vielen Anschlüssen sind noch der Ein/Ausschalter und ein Namensschild(!!) auf der Rückseite des Geräts zu finden.



Auf der rechten Seite vor dem Diskettenlaufwerk befinden sich noch die Schnittstelle für den Kassettenrekorder und zwei Joystickanschlüsse.

Wenn man den Computer von der Vorderseite betrachtet, fallen sofort die (überdimensionalen) Cursortasten auf. Die Tastatur gleicht der des SVI-328 beziehungsweise der des SVI-728, nur der Zehnerblock fehlt. Leider haben sich die Designer diesmal den (mir unverständlichen) Wunsch nach sehr großen Cursortasten zu Herzen genommen. Diese Tasten sind aber vollkommen übertrieben, Cursortasten wie beim SVI-328 hätten auch genügt. Beim Scrollen ergeben sich nämlich dadurch Probleme. Der Anschlag der Tastatur ist diesmal etwas weicher.

Bisher hatte Spectravideo immer auf die (noch) verbreiteteren 5.25"-Laufwerke gesetzt. Mit dem eingebauten 3.5-Zoll-Laufwerk schloß sich SVI dem momentanen Trend an. Die Vor- und Nachteile dieser Disketten werden in einem eigenen Bericht in diesem Heft aufgezeigt. Das Laufwerk arbeitet äußerst leise und präzise. Zum Formatieren einer CP/M-Diskette werden ungefähr 80 Sekunden und für eine MSX-DOS-Diskette nur ungefähr 60 Sekunden benötigt. Nach dem Formatieren sind 360 KByte auf der Diskette frei. Die Daten werden mit einer Geschwindigkeit von 250 KBits/sek übertragen, was ungefähr 64 KByte in 8 Sekunden entspricht.

Die Firmware

Neben den üblichen MSX-Bausteinen und dem MSX-ROM sind weiters noch 8 KByte ROM mit RS232 Firmware und 16 KByte ROM mit Diskfirmware vollgepackt. Insgesamt sind wieder 80 KByte RAM (in der gewohnten Verteilung) eingebaut.

Angesprochen wird die Firmware unter MSX-BASIC mit folgenden Befehlen:

CALL COMHELP
Nach diesem Aufruf werden sämtliche einzustellende Parameter angezeigt.

CALL COMINI 'parameter'
Durch diesem Aufruf können Sie die oben angezeigten Parameter verändern.

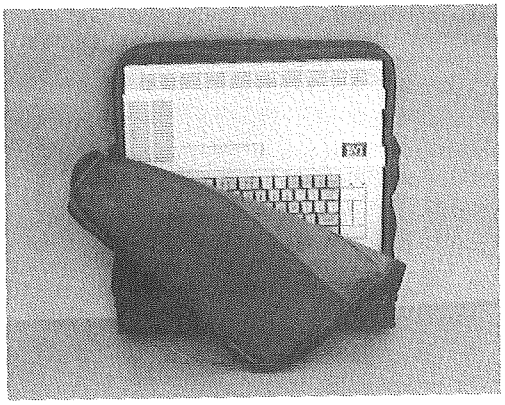
CALL COMTERM
Nach dieser Eingabe erscheinen sämtliche Eingaben, die auf Ihrem Computer gemacht werden, auf dem Bildschirm Ihres "Gesprächspartners" und umgekehrt.

Files werden übertragen, indem Sie zuerst die seriellen Schnittstellen auf den beiden kommunizierenden Computern eröffnen (OPEN "COMO:" FOR INPUT (OUTPUT) AS #1) und danach das File mit SAVE"COMO:"(LOAD"COMO:) abspeichern (laden). Im "User's Manual" sind zu diesem Thema viele Beispiele abgedruckt. Es wird die Datenfernübertragung unter CP/M und unter MSX-BASIC genau erklärt und beschrieben. (Bastler finden übrigens die genaue Anschlußbelegungen sämtlicher Schnittstellen im Anhang.)

Der Modulschacht ist diesmal von der Platine hermetisch abgeriegelt. Es ist also nicht möglich, die Platine irgendwie zu beschädigen. Der vom SVI-728 her bekannte Microswitch, der beim Öffnen des Modulschachts automatisch den Computer abschaltet, fehlt aber.

Mit eingebauter 80 Zeichenkarte

Besonders zu erwähnen ist die eingebaute 80 Zeichenkarte. Bei dem Testmodell, das mir zur Vefügung stand, war diese zwar noch nicht eingebaut (Mein Testmodell hatte die Seriennummer 49!), laut Auskunft der Generalvertretung soll sich dies aber bei den für den Verkauf bestimmten Computern ändern. Somit ist der SVI-738 X'press der erste MSX-Computer mit eingebauter 80-Zeichenkarte. Angeblich wurde hierfür extra ein neuer VRAM-Baustein entwickelt, der zu den in anderen MSX-Computern verwendeten absolut kompatibel sein soll.



Durch die Portabilität ist der X'press besonders gut für Leute geeignet, die den Computer sowohl im Büro als auch zu Hause verwenden wollen. Auch der Austausch von "WordStar"-Texten mit einem größeren Bürocomputer sollte kein Problem darstellen. Der Anschluß an das neue Spectravideo-Netzwerk ist selbstverständlich auch möglich. Die benötigte Erweiterungskarte trägt die Bezeichnung SVI-709.

Allerdings ist der X'press auch für Anwender, die keinen Wert auf Portabilität legen, sehr interessant. Durch das eingebaute Diskettenlaufwerk wird er nämlich zu einem der preiswertesten MSX-Computer, die momentan am europäischen Markt erhältlich sind.

Der Preis für den SVI-738 X'press mit Tragetasche und Software wird voraussichtlich knapp 13000 Schilling betragen. In der nächsten Zeit dürfte auch das Angebot an Soft-

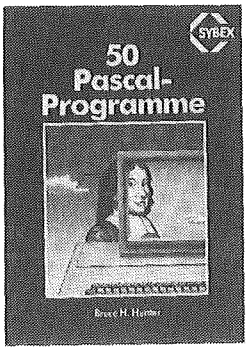
Fortsetzung auf Seite 14

****
*
* **Buchecke** *
*
****

In den letzten Wochen kamen vier besonders interessante Werke in den Handel. Die ersten beiden stammen vom Vogel-Verlag aus der Serie CHIP-SPECIAL.

In "Anwender-Programme Turbo-Pascal" finden Sie eine Unzahl interessanter Programme und Prozeduren für Turbo-Pascal. Das komplexeste Programm ist PLIST. Dieses Programm stellt alle anderen Lister in den Schatten. Angefangen mit Zeilennummerierung, Markierung reservierter Wörter und Ausdruck von Include-Files bis hin zur Angabe der Schachtelungstiefe sind alle nur erdenklichen Optionen eingebaut. Auch Crossreferenzen und Bündelung der Textabschnitte sind möglich. Weiters sind ein Grafikprogramm für Drucker, ein Superdirectory, ein DFÜ-Programm, eine Adressverwaltung und ein paar Spiele in diesem ausgezeichneten Buch zu finden. Auch Baumstrukturen werden durch ein kleines Programm erklärt. Abgeschlossen wird das Buch mit einigen Buch- und Programmiertips. Der Preis beträgt 230 ÖS. (DFÜ = Datenfernübertragung, d.h. Datenübertragung über Akkustikkoppler oder Modem, Anm. der Redaktion)

Das zweite Buch aus dem Vogel-Verlag nennt sich "MSX-Computer". In diesem CHIP-SPECIAL wird die Anwendung der Funktionstasten, die Verwendung des eingebauten Timers und das Erstellen von komfortablen Menüs erklärt und durch Programme verdeutlicht. Weiters ist noch ein Vokabellernprogramm abgedruckt. Tips für die Erstellung einer Datenbank und von Tabellen dürfen selbstverständlich auch nicht fehlen. Die bereits obligat gewordenen Spiele, Grafik- und Musikprogramme sind auch wieder abgedruckt. Der Preis für dieses CHIP-SPECIAL beträgt 195 Schilling.



In "50 Pascal Programme" von Bruce H. Hunter aus dem Sybex-Verlag finden Sie eine große Sammlung sehr nützlicher und teilweise auch lehrreicher Programme. Obwohl die Programme zwar nicht extra für TURBO PASCAL geschrieben sind, sollten sie ohne große Probleme und Veränderungen laufen. Jedes Programm ist ausführlich dokumentiert und gut strukturiert, sodaß das Verständnis der einzelnen Programme erheblich erleichtert wird. Neben verschiedenen Ein-/Ausgabe-Techniken, der Programmierung von Funktionen und der Erzeugung verschiedenster Datentypen wird Ihnen auch noch gezeigt, wie man sequentielle oder Random-Dateien erzeugt. Weiters werden noch das Sortieren, das Suchen und sonstige Arbeiten mit Dateifeldern erklärt. Das Buch ist besonders Anfängern zu empfehlen, da es durch die genauen Beschreibungen und den schön strukturierten Aufbau der Programme das Erlernen der Sprache PASCAL erleichtert. Auf diese Art und Weise gerät man erst gar nicht in Versuchung, den von BASIC her bekannten "Spaghetti-Code" zu programmieren. Das Buch hat einen Umfang von 317 Seiten und kostet 374 ÖS.

Nach dem "Trainingsbuch zu TURBO PASCAL" erschien nun auch "Tips und Tricks für TURBO PASCAL" bei DATA BECKER. Das Buch stammt wieder vom erfolgreichen Autorenduo Sgonina/Warner. Wie der Titel bereits verrät, soll dieses Buch mehr als eine reine Programmsammlung sein. Den Inhalt kann man in neun Schwerpunkte teilen: Sortierverfahren, Bäume, TURBO und das Betriebssystem, Bildschirmausgabe, Eingabefunktionen, Maskengenerator, Diskettenverwaltung, Zeit und Datum, Utilities. Die Bildschirmausgaberoutinen und der Schwerpunkt "Zeit und Datum" sind nur für MS-DOS gedacht. Alles andere ist für CP/M und MS-DOS.

Als Utilities sind ein Programmlister, ein Tracer und ein Programm zur Erstellung von Crossreferenzen abgedruckt. Sehr eingehend und auch (verhältnismäßig) leicht verständlich wurden verschiedene Sortierverfahren und Bäume erklärt. Allen Lesern, die sich mit Baumstrukturen und Sortierverfahren beschäftigen wollen, empfehle ich dieses Buch. Wer lieber nur Programme abtippen möchte, ist mit dem CHIP-SPECIAL vielleicht besser bedient. Das DATA-BECKER-Buch kostet 382 ÖS.

Testbericht: X'press
Fortsetzung von Seite 13

ware auf 3.5"-Disketten steigen. Zum Beispiel ist bereits eine Version von UCSD-Pascal für MSX auf diesem Diskettenformat angekündigt.

Zuletzt noch eine Aufstellung mit den technischen Daten des neuen Computers:

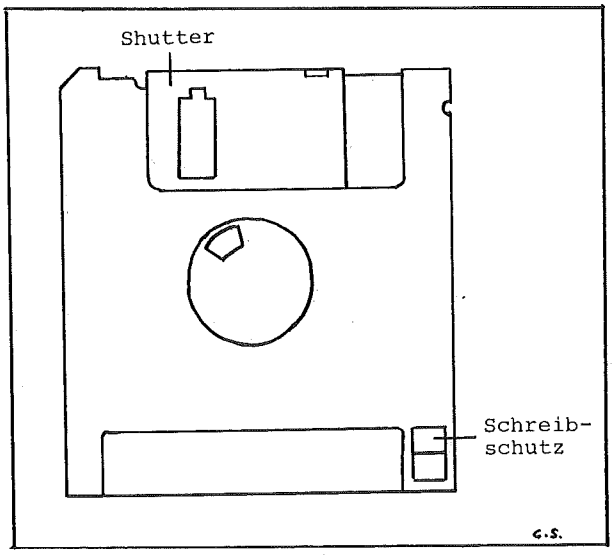
Name:	SVI 738 X'press
Arbeitsspeicher:	64 Kbyte RAM
Bildschirmspeicher:	16 Kbyte RAM
MSX-BASIC:	32 Kbyte ROM
Firmware:	16 Kbyte für Disk 8 Kbyte für DFÜ
Zeichen pro Zeile:	40 bzw. 80
Grafikauflösung:	256 x 192
Farben:	16
Sound:	3 Kanäle
Schnittstellen:	Centronics RS232 I/O-Port für 2 Laufwerk ein Modulschacht 2 Joystickports TV-, Monitor- und Kassettenrekorderanschluß
Tastatur:	74 Tasten
Diskettenlaufwerk:	3.5 Zoll (Microfloppy) 360 Kbyte (formatiert) 250 Kbit/sek. 80 Tracks 9 Sektoren

****
*
* **Was sind 3.5-Zoll-Disketten** *
*
****

Trotz der geringen Größe von nur 3.5 Zoll haben diese Disketten die selbe Speicherkapazität wie die "großen" 5.25-Zoll-Disketten. Da MSX-Computer die unter MS-DOS aufgezeichneten Daten lesen können sollen, wurde ein absolut IBM-kompatibles Aufzeichnungsformat gewählt. Das bedeutet also 360 KByte (formatiert) pro Diskette.

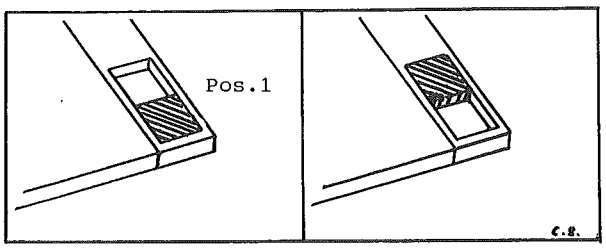
Einer der grundlegenden Unterschiede zwischen 5.25- und 3.5-Zoll-Disketten besteht im verwendeten Material. Wenn Sie schon einmal eine 3.5"-Diskette in der Hand gehalten haben, wird Ihnen sicher aufgefallen sein, daß diese nicht biegsam ist. Sie können somit nicht mehr so leicht geknickt oder verbogen werden.

Abbildung 1



Das als Shutter bezeichnete Stück der Disk verhindert, daß die Magnetschicht durch unvorsichtige Handhabung beschädigt oder gar zerstört wird. Dieser Shutter wird erst im Diskettenlaufwerk entfernt. Beim Entnehmen der Diskette schützt er wieder die empfindliche Magnetschicht. Fingerabdrücke auf der Magnetschicht sind dadurch so gut wie unmöglich.

Abbildung 2 Abbildung 3



schreibgeschützt beschreibbar

Auch der Schreibschutz hat sich geändert. Bei den 5.25 Zoll Disketten mußte man einen Aufkleber über eine kleine Kerbe kleben. Was war aber, wenn man keinen hatte? Das kann jetzt nicht mehr passieren. Jetzt ist der Schreibschutz nämlich eingebaut (siehe Abbildung). Ein kleiner Stift muß nur verschoben werden, um die Diskette zu schützen oder um sie wieder beschreibbar zu machen.

Wenn sich der Stift in der Position 1 (Abbildung 2) befindet, ist die Diskette schreibgeschützt. Wenn das kleine Loch verschlossen ist (Abbildung 3), so können Sie die Diskette beschreiben.

Der einzige Nachteil der neuen 3.5"-Disketten ist der Preis. Eine Diskette kostet momentan ungefähr 110 Schilling.

Programmiersprachen: PILOT
Fortsetzung von Seite 9

Etwas ausführlicher auf die obengenannten Themen geht

Graham Beech
Computer Based Learning
Practical Microcomputer Methods
Sigma Technical Press

ein. Die ersten Kapitel sind dem Thema "Computerunterstützter Unterricht" generell gewidmet, der Autor erläutert verschiedene Soft- und Hardwaresysteme, unter anderem das weithin bekannte System PLATO der Control Data Corporation. Die Beispielprogramme der PILOT-Kapitel sind in SuperPILOT für den Apple II geschrieben, sollten aber auch unter NEVADA-PILOT laufen, wenn Abstriche in Punkto Grafik, Ton und sonstigen, nicht standardmäßigen Erweiterungen gemacht werden (eines dieser Beispielprogramme ist ein in PILOT geschriebener PILOT-Generator a la WAPP).

Die dritte Quelle für Informationen über PILOT stellt der Sammelband des

Dr. Dobbs Journal of Computer Calisthenics & Orthodontia
Volume 2
Hayden Books

dar. Hier finden sich Artikel über PILOT, unter anderem von dessen "Vater", Dr. Starkweather, sowie ein 8080-Assemblerlisting für die Maschinensprachprogrammierer unter unseren Lesern (nebenbei bemerkt sind die Bücher dieser Reihe eine Goldgrube für jeden Personalcomputeranwender). Das Interesse an PILOT war (ist?) doch so groß, daß auch noch in späteren Bänden Beiträge über diese Programmiersprache erschienen sind. (Eine Voraussetzung ist für das Arbeiten mit PILOT jedoch unbedingt notwendig, nämlich die Kenntnis der englischen Sprache).

Ein Nachtrag für Schulen: Als interessante Kombination dürften sich PILOT und der neue Superexpander SVI-609 mit Harddisk und Netzwerkfähigkeit erweisen, über den in einem der nächsten Hefte ein ausführlicher Testbericht folgen wird.

Turbo-PASCAL, von Anfang an Folge 8

In dieser Folge gibt es wieder ein Listing unseres Editors, das allerdings noch nicht alle geplanten Features beinhaltet. Bevor aber dieses Listing besprochen wird, muß noch etwas zu dem Programm aus der vorigen Folge gesagt werden.

Das Programm Filedemo zeigt, wie man ein File eröffnet und es bearbeitet. Welche Befehle hat TURBO, um Files zu bearbeiten?

Zuerst muß man das File in der Variablendefinition vereinbaren. Das geschieht mit folgendem Befehl:

```
filebezeichner: FILE OF variablentyp;
```

Durch dieses Statement wird ein File definiert, in dem Daten mit dem Typ "variablentyp" gespeichert sind. Alle in PASCAL möglichen Typen können in einem File abgespeichert werden. In dem Programm Filedemo sind es Strings mit der Länge 30. Nach der Vereinbarung muß dem File ein Name auf der Diskette zugeordnet werden. Der Name "filebezeichner" aus der Vereinbarung ist kein echter Filename, er ist nur eine interne Bezeichnung für das File. Einen Namen auf der Diskette erhält das File mit dem Befehl:

```
ASSIGN(filebezeichner,filename);
```

Jetzt ist das File aber nur als Directoryeintrag auf der Diskette existent. Um es zu bearbeiten, muß das File noch eröffnet werden. Wenn das File neu ist, das heißt, daß es auf der Diskette noch nicht vorhanden ist, muß der Befehl

```
REWRITE(filebezeichner);
```

verwendet werden. Durch dieses Kommando wird das File auf der Diskette geschaffen. Wichtig ist, daß man dieses Kommando nicht auf bereits existierende Files anwendet, da diese mit diesem Befehl gelöscht werden. Wie man erkennt, ob ein File bereits existiert, oder nicht, wird in einer späteren Folge erklärt. Für bereits bestehende Files verwendet man das Kommando

```
RESET(filebezeichner);
```

Jetzt ist das File eröffnet und kann angesprochen werden. Das geschieht mit den Statements

```
READ(filebezeichner,variable);
```

für das Lesen aus einem File und

```
WRITE(filebezeichner,variable);
```

wenn man in das File schreiben möchte. Bevor man aber eine Operation durchführen will, muß man PASCAL noch mitteilen, auf welchen Teil des Files man zugreifen möchte. Das geschieht, indem man den Filezeiger auf den Record stellt, auf den man zugreifen möchte. Das geschieht mit dem Befehl

```
SEEK(filebezeichner,recordnummer);
```

Die aktuelle Position des Filezeigers kann man mit der Funktion

```
FILEPOS(filebezeichner);
```

abrufen. Nach jedem READ- oder WRITE-Zugriff wird der Filezeiger um Eins erhöht. Wenn man ein File eröffnet, wird der Zeiger auf den ersten Record gestellt. Beim Verstellen des Filezeigers mit SEEK muß man nur darauf aufpassen, daß man nicht nach dem Dateiende zugreift. Die Nummer des letzten Records kann man mit der Funktion

```
FILESIZE(filebezeichner);
```

ermitteln. Nachdem man ein File bearbeitet hat, muß man es wieder schließen. Das geschieht mit dem Statement

```
CLOSE(filebezeichner);
```

Das waren die wichtigsten Befehle zur Bearbeitung von Files. Mit diesen Befehlen kann man beliebige Files bearbeiten. Mehr über Files, und wie man zum Beispiel verschiedene Daten in einem File (z.B zwei INTEGER und eine STRING-Variable) ablegen kann, in der nächsten Folge.

Doch nun zu unserem Editor. In dieser Folge gibt es ein Listing unserer Entwicklung. Wegen der Länge des Programmes sind noch nicht alle in Folge 6 zusammengefaßten Features verwirklicht worden. Doch die Grundzüge sind bereits vorhanden. Die restlichen Features können Sie sicher leicht selbst implementieren. In einer der nächsten Folgen werde ich noch besprechen, wie man den Text auf Diskette abspeichern kann, und wie man den Editor erweitern kann, daß nicht nur Texte von der Länge einer Seite, sondern beliebig lange Texte bearbeitet werden können. Wenn Fragen bezüglich des Editors auftauchen, können Sie sich an mich wenden.

In der nächsten Folge geht es dann mit Files und mit den sogenannten strukturierten Daten - den RECORDS - weiter.

RAFAEL RAZIM

```
PROGRAM editor;

CONST breite=80;
      laenge=22;

VAR
   inhalt:ARRAY(.1..laenge.)
                          OF STRING(.breite.);
   veraendert:ARRAY(.1..laenge.) OF BOOLEAN;
   rand:ARRAY(.1..laenge.) OF INTEGER;
   spalte,zeile:INTEGER;
   eingabe:CHAR;
   inserton:BOOLEAN;

PROCEDURE clearline(zeile:INTEGER);
VAR f:INTEGER;
 BEGIN
  inhalt(.zeile.):='';
  FOR f:=1 TO breite DO
   inhalt(.zeile.):=inhalt(.zeile.)+' ';
  veraendert(.zeile.):=TRUE;
  spalte:=1;
  rand(.zeile.):=1;
 END;
```

```
PROCEDURE initvar;
VAR f,g:INTEGER;
 BEGIN
  spalte:=1;zeile:=1;
  inserton:=FALSE;
 END;

PROCEDURE clear_screen;
VAR f:INTEGER;
 BEGIN
  CLRSCR;
  FOR f:=1 TO laenge DO
   BEGIN
    clearline(f);
   END;
  GOTOXY(5,1);WRITE('SPALTE:');
  GOTOXY(20,1);WRITE('ZEILE:');
  GOTOXY(30,1);WRITE('INSERTO');
 END;

PROCEDURE print_headline;
 BEGIN
  GOTOXY(12,1);
  WRITE(spalte,' ');
  GOTOXY(26,1);
  WRITE(zeile,' ');
  GOTOXY(37,1);
  CASE inserton OF
   TRUE  : WRITE('N ');
   FALSE : WRITE('FF');
  END;
 END;

PROCEDURE print(zeile:INTEGER);
 BEGIN
  GOTOXY(1,zeile+1);
  WRITE(inhalt(.zeile.));
  veraendert(.zeile.):=FALSE;
 END;

PROCEDURE print_scr;
VAR pzeile:INTEGER;
 BEGIN
  WRITE(#27,'x5');
  FOR pzeile:=1 TO laenge DO
  IF veraendert(.pzeile.)
   THEN print (pzeile);
  GOTOXY(spalte,zeile+1);
  WRITE(#27,'y5');
 END;

PROCEDURE koordinaten_richtigstellen;
 BEGIN
  IF spalte=0
   THEN
    BEGIN
     zeile:=zeile-1;
     IF zeile>0
      THEN spalte:=rand(.zeile.)
      ELSE spalte:=1;
    END;
  IF spalte>rand(.zeile.)
   THEN
    BEGIN
     spalte:=1;
     zeile:=zeile+1;
    END;
  IF zeile<=0
   THEN zeile:=1;
  IF zeile>=laenge+1
   THEN zeile:=laenge;
 END;

PROCEDURE printe_buchstabe;
 BEGIN
  IF inserton
   THEN
    BEGIN
     IF rand(.zeile.)<80
      THEN rand(.zeile.):=rand(.zeile.)+1;
    END
   ELSE
    BEGIN
     IF (spalte=rand(.zeile.))
                        AND (rand(.zeile.)<80)
      THEN rand(.zeile.):=rand(.zeile.)+1;
     DELETE (inhalt(.zeile.),spalte,1);
    END;
  INSERT(eingabe,inhalt(.zeile.),spalte);
  veraendert(.zeile.):=TRUE;
  spalte:=spalte+1;
 END;

PROCEDURE delete_char;
 BEGIN
  veraendert(.zeile.):=TRUE;
  IF spalte<>1
   THEN
    BEGIN
     DELETE(inhalt(.zeile.),spalte,1);
     inhalt(.zeile.):=inhalt(.zeile.)+' ';
     IF rand(.zeile.)>spalte
      THEN rand(.zeile.):=rand(.zeile.)-1;
    END
   ELSE
    BEGIN
     clearline(zeile);
     spalte:=0;
    END;
 END;

PROCEDURE delete_left;
 BEGIN
  spalte:=spalte-1;
  IF spalte=0
   THEN
    BEGIN
     rand(.zeile.):=1;
     zeile:=zeile-1;
     IF zeile>0
      THEN spalte:=rand(.zeile.)
      ELSE
       BEGIN
        zeile:=1;
        spalte:=1;
       END;
    END
   ELSE
    rand(.zeile.):=rand(.zeile.)-1;
  DELETE(inhalt(.zeile.),spalte,1);
  inhalt(.zeile.):=inhalt(.zeile.)+' ';
  veraendert(.zeile.):=TRUE;
 END;

PROCEDURE carrige_return;
 BEGIN
  zeile:=zeile+1;
  IF spalte>rand(.zeile.)
   THEN spalte:=rand(.zeile.);
 END;

PROCEDURE insert_line;
VAR pzeile:INTEGER;
 BEGIN
  FOR pzeile:= laenge DOWNTO zeile+1 DO
   BEGIN
    inhalt(.pzeile.):=inhalt(.pzeile-1.);
    veraendert(.pzeile.):=TRUE;
   END;
  clearline(zeile);
 END;

BEGIN
 initvar;
 clear_screen;
  REPEAT
   print_scr;
   koordinaten_richtigstellen;
```

Fortsetzung auf Seite 18

```
*****************************************
*                                       *
*       Utilities für TURBO-PASCAL      *
*                                       *
*****************************************
```

Ungefähr ein Jahr nach dem Erscheinen von TURBO-PASCAL sind schon eine Reihe von Programmen verfügbar, die speziell für TURBO-PASCAL geschrieben worden sind. So gibt es neben einer Graphik-Erweiterung eine Einführung in TURBO und eine Datenbank. Zwei Programme für TURBO habe ich getestet:

TURBO-TUTOR und die TOOL-BOX

TURBO-TUTOR wird auf einer Diskette im SVI-Format und einem deutschen Handbuch geliefert. Dieses Programm (eigentlich sind es ja viele Programme) will eine einfache Einführung in TURBO sein, mit vielen Beispielen auf Diskette. Das wirklich ausgezeichnete Handbuch ist ein guter Lehrgang für TURBO.

So gut das Handbuch ist, so sehr ist man vom Inhalt der Diskette enttäuscht. Wenn man sich das Directory ansieht, stechen sofort zwei Files ins Auge: Die Programme "READ.ME" und "LIES.ME". "READ.ME" ist eine englische Zusammenfassung der Erweiterungen von TURBO 3.0 und hat nur bedingt mit dem übrigen Inhalt der Diskette zu tun. LIES.ME dagegen ist in deutscher Sprache abgefaßt und dürfte vom deutschen Importeur stammen. Darin steht, welche Beispielprogramme auf der Diskette sind, und daß zusätzlich zu den im Handbuch erwähnten Programmen noch ein Liste für PASCAL-Programme auf der Diskette ist. Wenn man sich die übrigen 24 Beispielprogramme ansieht, kommt man bald darauf, daß nur 8 davon in CP/M 2.2 laufen. Alle anderen sind speziell für den IBM-PC und Kompatible geschrieben. Anscheinend hält man bei BORLAND die "8-Bit'ler" schon für tot, sonst hätte man mehr CP/M Programme auf die Diskette gegeben. Denn die wenigen auf dem SVI lauffähigen Programme sind nicht sehr informativ. Die wirklich interessanten Programme, die zeigen, wie man Betriebssystem-Routinen von PASCAL aus aufruft oder wie man ASSEMBLER mit PASCAL kombiniert, sind nur auf MS-DOS lauffähig. Nur zwei CP/M-Programme, nämlich eins, das das Directory einer Diskette auslesen kann, und ein anderes, das die System- und Diskparameter ausgibt, sind von wirklichem Interesse. Etwas entschädigt wird man durch das Source-Listing des Listers für PASCAL-Programme, der das alte "TLIST" von der TURBO-Diskette ersetzt. Dieser Lister ist ein Programm, das zeigt, wie man längere Programme in PASCAL übersichtlich schreiben kann. Außerdem kann man den Lister den individuellen Bedürfnissen anpassen.

Zusammenfassend kann man sagen, daß TURBO-TUTOR für den IBM-PC-Besitzer empfohlen werden kann, für uns CP/M-User ist er nur bedingt nützlich. Das Handbuch ist zwar eine ausgezeichnete Einführung, die von BASIC ausgehend, alle Gebiete von TURBO auch dem Anfänger näherbringt. Derjenige, dem nicht sehr viel an dem Lister und den zwei wirklich interessanten Programmen liegt, ist wohl mehr mit einem der billigeren, und bereits zahlreich erschienen Bücher über TURBO bedient. Trotzdem kann der TURBO-TUTOR für einige Leute, die Beispielprogramme nicht eintippen wollen, wegen seines nicht allzuhohen Preises von 980 Schilling eine nützliche Einführung in TURBO sein.

Das zweite von mir geteste Programm nennt sich TOOL-BOX und bietet eine Sammlung von Unterprogrammen, die den Aufbau von leistungsfähigen Datenbanken in TURBO sehr erleichtern.

TOOL-BOX besteht aus drei Teilen:

TURBO-ACCESS, das aus einer Sammlung von Unterprogrammen besteht, mit denen man eine Datenbank, mit der Daten nach dem B-TREE Verfahren verwaltet werden (siehe mehr über diese Art von Datenverwaltung in der Serie "DATENSTRUKTUREN UND IHRE PROGRAMMIERUNG"),

TURBO-SORT, mit denen Dateien beliebiger Art nach dem QUICK-SORT-Verfahren (siehe ebenfalls die oben genannte Serie im SVI-Journal) sortiert werden kann

und TURBO-GINST, ein universelles Installationsprogramm, mit dem jedes Programm, daß in TURBO geschrieben wurde, an jedes Terminal angepasst werden kann. Das funktioniert so wie die Anpassung von TURBO an verschiedene Terminals.

Für die ganze TOOL-BOX gilt, daß sie geschrieben wurde, um das "Wiedererfinden des Rades"-Syndrom zu verhindern, das heißt, daß man sich nicht die Arbeit machen muß, schon längst gelöste Probleme neu ausarbeiten zu müssen. Die sehr leistungsfähigen Routinen von TURBO-ACCESS erlauben einen einfachen Aufbau von Datenbanken. Natürlich ist eine Einarbeitungszeit notwendig, bis man herausgefunden hat, wie die einzelnen Teile miteinander zu verbinden sind, um eine individuelle Datenbank zu erstellen. Zwei Beispielprogramme auf der Diskette erleichtern aber den Einstieg. Damit ist TURBO-ACCESS eine echte Alternative zu dem viel teureren DBASE II-Programm, mit dem ebenfalls Datenbanken individuell erstellt werden können. Gegenüber Fix- und Fertiglösungen bietet TURBO-ACCESS den Vorteil, daß man sich seine "eigene" Datenbank programmieren kann.

TURBO-SORT ist auch aus Unterprogrammen aufgebaut und erlaubt ein schnelles Sortieren von Dateien nach unterschiedlichen

Fortsetzung auf Seite 21

Turbo-Pascal, von Anfang an
Fortsetzung von Seite 17

```
    print_headline;
    GOTOXY(spalte,zeile+1);
    eingabe:=chr(bios(2));
    CASE eingabe OF
      ^E,^^    : zeile:=zeile-1;
      ^X,^     : zeile:=zeile+1;
      ^S,^Ü    : spalte:=spalte-1;
      ^D,^Ö    : spalte:=spalte+1;
      ^V       : inserton:=NOT inserton;
      ^Y       : clearline(zeile);
      ^G       : delete_char;
      ^M       : carrige_return;
      ^N       : insert_line;
       #127,^H : delete_left;
     ELSE printe_buchstabe;
    END;
  UNTIL eingabe = ^K;
END.
```

```
*****************************************
*                                       *
*    Funktionstasten mit 'CHR$' listen  *
*                                       *
*****************************************
```

Der Spectravideo gibt uns die Möglichkeit, 10 häufig verwendete Befehle oder Zeichenketten auf Tastendruck zu generieren. Meistens verwendet man dann für diese Funktionstasten Zeichenketten, die viele undruckbare Zeichen enthalten. Nun kann es passieren, daß man eine Funktionstastenbelegung geringfügig ändern möchte, zum Beispiel noch ein Cursorkommando einbauen will. In diesem Fall muß man den ganzen Befehl noch einmal eingeben. Es sei denn, man schreibt sich ein Programm, welches einem den KEY-Inhalt so ausdruckt, daß er vom BASIC-Interpreter auch wieder anerkannt wird. Dazu gehört, daß nicht druckbare Zeichen als CHR$-Sequenzen ausgegeben werden. Druckbare werden der Übersichtlichkeit halber als direkte Zeichen ausgegeben. Die einzige Konzession an den Programmumfang ist die Ausgabe der CHR$-Sequenz. Es werden die Zahlen nämlich nicht dezimal sondern hexadezimal ausgegeben. Da dies nur einen geringen Verlust an Komfort darstellt, die Programmierung aber erheblich vereinfacht, wurde diese Lösung gewählt.

Um den Inhalt einer Funktionstaste zu ermitteln, braucht man lediglich LIST KEY 'Nummer' eingeben. Dies darf man nicht mit dem Befehl KEY LIST verwechseln, der alle Belegungen auflistet, aber auf nicht druckbare Zeichen keine Rücksicht nimmt. Mit diesem MC-Programm kann man zwar nicht alle Tasten auf einmal ausdrucken, dafür bekommt man aber einen genaueren Ausdruck.

Initialisiert wird das Programm durch eine kleine Routine ab D100H. Um das Laden möglichst komfortabel zu gestalten, speichert man das Programm am besten mit BSAVE "LSTKEY",&HCFF0,&HD100 ab. Geladen wird dann mit BLOAD "CAS:",R.

Sehen wir uns nun den Programmaufbau etwas genauer an. Wir haben uns bemüht, das Programm so übersichtlich wie möglich zu gestalten und gut zu dokumentieren.

Am Anfang wird auf die Token LIST KEY überprüft. Zum Schluß wird noch eine Zahl eingelesen, die - wie gehabt - in DE abgelagert wird. Da wir nur Zahlen zwischen 1 und 10 brauchen, wird in der nächsten Routine darauf geprüft, daß DE keine Zahl außerhalb dieses Bereichs enthält. Wird kein Fehler gefunden, arbeitet der Computer mit einer Subroutine weiter, die die Meldung "KEY x," ausdruckt. Danach stellt der Computer HL auf die angesprochene Funktionstaste ein. Nun ist man mit der Initialisierung fertig und kann anfangen, die einzelnen Zeichen in den Computer zu laden und zu testen. Dies wird durch die Hauptschleife erledigt. Sie geht eigentlich von Zeile 156 bis 177. Ein Teil wird jedoch nicht immmer abgearbeitet. Das ist die Ausgabe der druckbaren Zeichen. Sie wird während der CHR$-Sequenzen umgangen.

Die Ausgabe der Zeichen ist an bestimmte Prüfungen gebunden. Der Test wird auf folgende Weise durchgeführt:

Ganz am Anfang wird auf eine Null - dem Ende der KEY-Anweisung - getestet. Erst danach folgen die eigentlichen Tests:

Zuerst testet der Computer auf ein '"'. Danach wird A auf das Leerzeichen gestellt. Durch diesen Test fallen alle Zeichen unter 20H zu den CHR$-Sequenzen ab. Es folgt die Einstellung auf das Zeichen hinter dem letzten Graphikzeichen. So werden die Zeichen 224-255 eliminiert. Durch die Zahl 160 teilt sich der Weg in Graphikzeichen, die ausgegeben werden und anderen, die vielleicht ihren Weg zur direkten Ausgabe finden.

'ev_chr' sorgt für die letzte Abfrage des DEL-Zeichens (hinter dem letzten druckbaren 'normalen' Zeichen).

Die Ausgabe der Zeichen verlangt folgende Regeln: Ein direktes Zeichen wird durch RST 18H ausgegeben. Um jedoch sicherzustellen, daß das Zeichen auch wirklich zwischen zwei Anführungszeichen steht, wird mit der Routine 'test' die Anzahl der bisher verwendeten Anführungszeichen getestet. Ist die Anzahl gerade, wird eines gesetzt. Zusätzlich wird vor diesem Zeichen ein "+" hingestellt. Nur wenn das Anführungszeichen ganz am Anfang steht, verzichtet der Computer aufs Plus.

Eine CHR$-Sequenz muß etwas mehr Tests bestehen. Zuerst wird wieder auf die Anzahl der Anführungszeichen getestet und bei Bedarf ein direkter String geschlossen. Außerdem darf man nicht gleich "blind" ein Plus vor das CHR setzen, da ein CHR$ auch am Anfang des Programms stehen kann. Daher wird auch dies geprüft. Hinter dem CHR$ steht kein Plus. Es muß bei Bedarf vom nächsten Zeichen erzeugt werden.

Aus diesen Bedingungen ergeben sich die restlichen Routinen. Lediglich am Schluß befindet sich noch 'keyend'. Es sorgt dafür, daß die logische Zeile vom Ende der KEY-Anweisung an bis zum Ende der logischen Zeile gesäubert wird. Ein CR und LF wird auch noch nachgeschickt.

```
        org c000h
;   base: Anfang der Keytabelle
;   cr_lf: Ausdruck von CR und LF
;   illeg: Einsprung zu 'Illegal
;          function call'
base    equ fa1eh
cr_lf   equ 6474h
illeg   equ 0f9eh
;
;   Test auf LIST KEY, bei Abwei-
;   chung zurück zum Interpreter
;
begin   push af
        push hl
        cp 93h          ; Token LIST
        jp nz,out
        rst 10h
        cp c7h          ; Token KEY
        jp nz,out
        call 1a98h
        pop bc
        pop bc
        pop bc
        dec hl
```

```
;
;   Test auf 1-10, bei Abweichung
;   zu 'Illegal function call'
;
        xor a
        cp d
        jp nz,illegl
        cp e
        jp z,illegl
        ld a,10
        cp e
        jp c,illegl
;
;   KEY x wird ausgedruckt
;   HL wird auf KEY x eingestellt
;
        call key
        push hl
        ld hl,base
        ld a,e
        dec a
        rlca
        rlca
        rlca
        rlca
        add a,l
        ld l,a
        ld b,16
;
;   Hauptschleife, Test auf nicht
;   druckbare Zeichen
;
loop    ld a,(hl)
        cp 0            ; Ende KEY x
        jr z,keyend
        cp 34           ; "-Zeichen
        jr z,chr
        cp " "
        jr c,chr
        cp 224
        jr nc,chr
        cp 160          ; 1. GRPH-Zeichen
        jr c,ev_chr
;
;   Ausgabe von druckbaren Zeichen
;
output push af
        call test
        and a
        call z,plusan
        pop af
        rst 18h
gemein inc hl
        djnz loop
;
;   PRINT ESC 'J' = Zeile bis Ende
;   löschen, danach zu Interpreter
;
keyend ld a,27
        rst 18h
        ld a,74
        rst 18h
        call cr_lf
        pop hl
        rst 10h
        ret
;
;   Test auf " "-"z", wenn nicht, chr
;
ev_chr cp 127
        jr c,output
;
;   Ausdruck von 'CHR$(&HXX)'
;   Test, ob "+ gebraucht wird
;
chr     push hl
        ld hl,chrstr
        push af
        call test
        and a    ;a=1 "+chr
        jr nz,yes
        inc hl
        ld a,16
        cp b
        jr nz,print1
        inc hl
        jr print1
yes     call inc_an
print1  call print
opende  pop af
        call sign
        ld a,")"
        rst 18h
        pop hl
        jr gemein
;
;   Hexzahl 0-255 ausdrucken
;
sign    push af
        rrca
        rrca
        rrca
        rrca
        call half
        pop af
half    and 15
        add a,90h
        daa
        adc a,40H
        daa
        rst 18h
        ret
;
;   Inkrementiere "-Zeichenzähler
;
inc_an push hl
        ld hl,temp
        inc (hl)
        pop hl
        ret
;
;   testet auf String offen
;   bei offen, A=1
;
test    ld a,(temp)
        and 1
        ret
;
;   Ausdruck von 'KEY x,'
;
key     push hl
        ld hl,keystr
        call print
        ld a,e
        call sign2
        ld a,","
        rst 18h
        xor a
        ld (temp),a
        pop hl
        ret
;
;   +" oder " drucken
;
plusan push af
        ld a,16
        cp b
        jr z, noplus
        ld a,"+"
        rst 18h
noplus ld a,34
        rst 18h
```

****
*
* SASA *
*
****

Die wohl interessantesten Spiele sind halbe Adventures mit exzellenter Graphik. Neben seinem Reaktionsvermögen, das getestet wird, sprechen solche Spiele auch die Phantasie und die Neugierde des Users an. Der Wunsch, endlich alle verschiedenen Modi zu kennen, treibt manche sogar dazu, nächtelang über diesen Adventures zu hocken und nervös die diversen Aufgaben zu lösen.

Zwei dieser Spiele wollen wir diesmal vorstellen, die nächsten zwei folgen in einer der nächsten Ausgaben.

SASA handelt von einem Astronauten, der sich durch einige Stationen durchkämpfen muß. Zuerst muß er eine Tür aufbrechen. Sie ist als schwarzes Viereck in einer Wand eingelassen, die mit Schüssen teilweise abbröckelt. Geschützt wird die Tür von Hubschraubern, die im einfachsten Fall nur durch Kollision gefährlich werden. In höheren Modi wird man auch noch mit Raketen beglückt, die auf einen abgeschossen werden.

Das Fortbewegungsprinzip dieses Astronauten und seine Verteidigungsmöglichkeiten sind einmalig. Am Boden kann man ihn wie jedes normale Männchen auch mit den Cursortasten steuern. Links und Rechts bewegen ihn nach links oder nach rechts. Geschossen wird mit überdimensionalen Kugeln. Doch der Astronaut muß auch fliegen können und da bedient er sich - man sollte es nicht glauben - der Rückstoßkraft seiner Kugeln. Will man also vom Grund abheben, muß man mit der Cursortaste "Unten" die Waffen samt Männchen nach unten richten und kräftig feuern. Dann hebt der Astronaut ab. Geflogen wird dann mit einem Konglomerat aus den verschiedensten Richtungen. "Power" für den Flug bekommt "Sasa" nur durch dosiertes Schießen.

Die Schwierigkeiten, die auftreten, liegen auf der Hand. Solange das Männchen am Boden tappt, liegen die Cursortasten auch gefühlsmäßig richtig. Doch sobald der Astronaut in der Luft ist, gilt das Rückstoßprinzip, das sich genau umgekehrt zur normalen Gehweise verhält. Daher passieren vor allem in der horizontalen Fortbewegung immer wieder frappierende Fehler, die "letal" enden können.

Daß man nicht unbegrenzt herumschießen kann, ist klar. Man hat einen gewissen Vorrat an Kugeln, der einem durch ein - zugegeben - etwas kompliziertes System vermittelt wird. Geht dieser zu Neige, was sich durch einen tiefen Ton bemerkbar macht, sollte man sich möglichst schnell in eines der Kraftfelder fallen lassen, das im Bild zu finden ist. Dann werden die Reserven wieder aufgefüllt. Aber Achtung! Es dürfen nur eine begrenzte Zahl an Auffrischungen durchgeführt werden, nach dieser Anzahl ist's mit dem Kugelsegen aus. Die Anzahl der mäglichen Auffrischungen wird in einem länglichen Kästchen im linken unteren Teil des Bildschirms angezeigt. Das Kraftfeld im ersten Modus ist übrigens in einem Traktor untergebracht. Es zeigt sich als hellweiße stark zackige Linie. Wenn man den ersten Modus geschafft hat, dann kommt man in ...

Mehr wollen wir nicht verraten, denn das Geheimnis um die nächsten Aufgaben ist ein Teil des Reizes, den dieses Spiel zu bieten. Verraten sei nur, daß man unter anderem noch mit bewaffneten Tintenfischen und "leicht abstrakten Lebensformen" zu kämpfen hat. Es ist nicht leicht herauszufinden, welche Aufgabe in welchem Modus bewältigt werden muß, aber mit Phantasie und Geduld ist es bei weitem nicht unmöglich. Alle, die ich kenne, die dieses Spiel näher unter die Lupe genommen haben, haben ohne Anleitung jeden Modus geschafft!

****
*
* Turboboat *
*
****

Das zweite Spiel ist auf Mutter Erde angesiedelt. Mit einem Boot geht man auf Eroberungsfahrt, dabei muß man Radarstationen fangen, vor Torpedoschnellbooten fliehen und andere Schnellboote abschießen.

Der erste Modus ist relativ leicht. Man befindet sich in einem ziemlich engen Fluß und muß Schnellboote abschießen, die Torpeos auf einen feuern. An den Ufern befinden sich Schnellfeuerbatterien, die im Laufe des Spiels aktiviert werden. Diese und Munitionsdepots kann man nebenbei auch abschießen.

Im zweiten Modus hat sich der Fluß verbreitert. Man muß Felsbrocken und Torpedos ausweichen und eine gewisse Anzahl an Radarstationen aufklauben, die auf Floßen unterwegs sind. Im dritten Modus muß man vor besagten Schnellbooten flüchten. Gibt man zu wenig Gas, dann wird man versenkt. Zuviel darf man jedoch auch nicht geben, da von der Gegenseite laufend Torpedos kommen, denen man ausweichen muß. Dies muß man eine zeitlang aushalten. Danach wird die ganze Dreierserie wiederholt, mit entsprechenden Erschwernissen.

Allen drei Modi gleich ist, daß der Fluß, auf dem man sich befindet, eine Gegenströmung erzeugt, die einen "vom Bild strömt". Der muß man durch gezielte Motorschübe mit der Spacetaste entgegenwirken. Unterstützt wird das Spiel durch effektvolle Geräuschuntermalung.

Beide Spiele sind für den SVI-328 programmiert. Erhältlich sind beide Games in allen einschlägigen besseren Fachgeschäften, sie sind als Kassetten- oder Diskversion vorhanden. Sie kosten sowohl auf Disk als auch auf Kassette 390 Schilling.

An alle Videospielfans!

Die von uns getesteten Spiele werden vom Autor in eigener Weise interpretiert. Es kann dadurch passieren, daß gewissen Teilen im Spiel ein anderer Name zugewiesen wird als von der jeweiligen Softwarefirma vorgesehen ist, je nachdem, welche Assoziationen der jeweilige Tester gerade hat. Wir halten

Fortsetzung auf Seite 23

```
        call inc_an
        pop af
        ret
;
;   Zahl 1-10 ausdrucken
;
sign2   cp 10
        jr c,lower
        ld a,31h
        rst 18h
        xor a
lower   add a,30h
        rst 18h
        ret
;
;   'Illegal function call'
;
illegl  pop hl
        pop af
        jp illeg
;
;   Printroutine, HL auf String
;   $ = Ende
;
print   ld a,(hl)
        cp 40h
        ret z
        rst 18h
        inc hl
        jr print
;
;   zurück zu Interpreter
;
out     pop hl
        pop af
        ret
temp    defb 0
chrstr  defb 34
        defm "+CHR$(&H$"
keystr  defm "key$"
;
;   Install-Routine
;
        org c100h
        ld hl,ff57h
        ld (hl),c3h
        ld hl,begin
        ld (ff58h),hl
        ret
        end
```

```
CFF0 : 21 57 FF 36 C3 21 00 D0  !W.6.!..
CFF8 : 22 58 FF C9 0C 0D 0E 0F  "X......
D000 : F5 E5 FE 93 C2 E8 D0 D7  ........
D008 : FE C7 C2 E8 D0 CD 98 1A  ........
D010 : C1 C1 C1 2B AF BA C2 DB  ...+....
D018 : D0 BB CA DB D0 3E 0A BB  .....³..
D020 : DA DB D0 CD AA D0 E5 21  .......!
D028 : 1E FA 7B 3D 07 07 07 07  ..ä=....
D030 : 85 6F 06 10 7E FE 00 28  .o..ß..(
D038 : 1D FE 22 28 29 FE 20 38  .."(). 8
D040 : 25 FE E0 30 21 FE A0 38  %..0!..8
D048 : 19 F5 CD A4 D0 A7 CC BE  ........
D050 : D0 F1 DF 23 10 DE 3E 1B  ...#..³.
D058 : DF 3E 4A DF CD 74 64 E1  .³J..td.
D060 : D7 C9 FE 7F 38 E3 E5 21  ....8..!
D068 : EC D0 F5 CD A4 D0 A7 20  .......
D070 : 09 23 3E 10 B8 20 06 23  .#³.. .#
D078 : 18 03 CD 9D D0 CD E0 D0  ........
D080 : F1 CD 8A D0 3E 29 DF E1  ....³)..
D088 : 18 C9 F5 0F 0F 0F 0F CD  ........
D090 : 93 D0 F1 E6 0F C6 90 27  .......'
D098 : CE 40 27 DF C9 E5 21 EB  .§'...!.
D0A0 : D0 34 E1 C9 3A EB D0 E6  .4..:...
D0A8 : 01 C9 E5 21 F6 D0 CD E0  ...!....
D0B0 : D0 7B CD CF D0 3E 2C DF  .ä...³,.
D0B8 : AF 32 EB D0 E1 C9 F5 3E  .2.....³
D0C0 : 10 B8 28 03 3E 2B DF 3E  ..(.³+.³
D0C8 : 22 DF CD 9D D0 F1 C9 FE  ".......
D0D0 : 0A 38 04 3E 31 DF AF C6  .8.³1...
D0D8 : 30 DF C9 E1 F1 C3 9E 0F  0.......
D0E0 : 7E FE 40 C8 DF 23 18 F8  ß.§..#..
D0E8 : E1 F1 C9 01 22 2B 43 48  ...."+CH
D0F0 : 52 24 28 26 48 40 6B 65  R$(&H§ke
D0F8 : 79 40 0B 0C 0D 0E 0F 10  y§......
```

KLEINANZEIGEN

Sharp PC-1500A, originalverpackt + 2 Handbücher, kaum gebraucht

2000 Schilling

SVI-105 Graphiktablett + Software, originalverpackt, kaum gebraucht

1000 Schilling

Televideo 802H + 20 MByte-Harddisk, kaum gebraucht, CP/M-Software!

VB 60000 Schilling

Achtung Kenner!
4-Bit-Steuerungscomputer abzugeben, frei programmierbar, mit Möglichkeiten zum eigenen Erweitern, Topzustand!

VB 2000 Schilling

Utilities für TURBO-PASCAL
Fortsetzung von Seite 18

Gesichtspunkten. Auch hier wird der Einstieg durch Beispielprogramme erleichtert. Somit bietet die TOOL-BOX eine billige Möglichkeit, gute Datenbanken auf einfache Weise zu programmieren. Die mit TOOL-BOX erstellten Programme können übrigens, vorausgesetzt die TOOL-BOX wurde ordnungsgemäß erstanden und nicht kopiert, uneingeschränkt weiterverkauft werden.

Das deutsche Handbuch zu der TOOL-BOX bietet eine knappe, aber alles umfassende, Einführung in die Systeme dieses Programmpakets. Mit den Beispielprogrammen und dem Handbuch ist der Anfang für etwas geübte Programmierer sicherlich nicht schwer.

TURBO-GINST ist nur eine kleine Draufgabe zur TOOL-BOX und ist nur für professionelle Programmierer nützlich. Wer hat schon zwei CP/M-Computer zu Hause, so daß er Programme uminstallieren muß. Für Leute, die ihre mit TURBO-PASCAL geschrieben Programme weiterverkaufen wollen, ist TURBO-GINST sehr nützlich, da es mit den zu installierenden Programmen weitergegeben werden kann (Immer vorausgesetzt den rechtmäßigen Kauf der TOOL-BOX).

TOOL-BOX ist sicher nichts für absolute Beginner, dazu ist es zu komplex. Für den fortgeschritten Programmierer ist die TOOL-BOX mit einem Preis vom 1890 Schilling sicher zu empfehlen.

RAFAEL RAZIM

****
*
* Ergebnis der Fragebogenaktion *
*
****

Um von unseren Lesern ihre Meinung über das SVI-Journal zu hören, haben wir etwa 170 Fragebögen versandt. Das war vor drei Monaten. Seit dieser Zeit sind 23 - ausgefüllt - wieder zurückgekommen. Dies entspricht einem Schnitt von über 10 %. Das ist nicht besonders viel, zumal wir ja einen repräsentativen Querschnitt haben wollten. Wir wollen unseren Lesern das Ergebnis nicht vorenthalten. Das Ergebnis trifft übrigens ungefähr die Linie, die wir von der Redaktion eingeschlagen haben:

62.5 % sind mit dem Umfang des SVI-Journals zufrieden, 37.5 % finden ihn zuwenig. Das Informationsangebot hingegen wird von 85 % als ausreichend empfunden. Je 5 % finden zuviel und zuwenig Information in den einzelnen Ausgaben. Aus diesen zwei Punkten schließen wir, daß wir die Info zu dicht packen, beziehungsweise nicht optimal an den Mann bringen. Schon in dieser Ausgabe haben wir, wie Sie vielleicht schon bemerkt haben, ein anderes Seitenformat gewählt. Daß wir den Umfang nicht steigern können, liegt daran, daß der Club auch nur ein begrenztes Budget hat. Allen denjenigen, die eine Erweiterung haben wollen, sei gesagt: Werben Sie mehr Mitglieder an, dann haben wir ein größeres Budget und können die Zeitschrift erweitern. Im übrigen sind wir im Umfang mit über 300 Seiten pro Jahr an der Spitze der Computerclubzeitschriften zu finden! Außerdem drucken wir mit verkleinerter Schrift, auch ein Kunstgriff, um mehr Information als andere unterzubringen!

Die Programmiersprachenkurse halten 75 % für ausreichend, die Verständlichkeit ist für 92.5 % der Leser optimal. Bei den Programmlistings BASIC und Assembler sind etwa 60 % der Meinung, sie seien ausreichend. 30 % finden zu wenig BASIC und Assembler in unseren Heften. 50:50 sieht es bei Pascal aus. Andere Programmiersprachen scheinen von uns etwas vernachlässigt worden zu sein. 35:40 geht der Vergleich zwischen ausreichend und zu wenig aus. 25 % haben keine Antwort gewußt. Die Kommentierung der Programmlistings wird von 50 % als sehr gut und von 45 % als ausreichend empfunden. 5 % sehen sie als schlecht an.

Etwas schlechter schneidet das Journal bei der System- und Hardwareinformation ab. 50 % sehen die HW-Info als ausreichend an, 45 % finden zuwenig Daten darüber. Über das System wird eindeutig zuwenig geschrieben. 65 % sehen zuwenig Fakten, 35 % finden die Information ausreichend. Die Programmtests sind für 65 % ausreichend, für 25 % zu viel.

Die Hintergrundberichte werden 55:35 % als ausreichend (35 % zuwenig) empfunden. 10 sehen zuviel Hintergrund. MSX-Information ist ebenso mit 55 % ausreichend, obwohl hier die Tendenz eher auf "zu viel" zeigt.

Interessant sind die Personen, die das SVI-Journal konsumieren. Die Alterstruktur der (aktiven) Leserschaft zeigt sich mit fast 90 % aller Befragten unter 40 Jahren sehr niedrig. 40 % sind unter 20 Jahren alt. Alle Befragten programmieren selber, 65 % verwenden kommerzielle Software. Der Computer wird von fast allen in der Freizeit, manchmal auch nur zur Unterstützung eines Hobbies, verwendet, 50 % haben auch im Beruf einen Verwendungszweck. Als Programmiersprachen gelten BASIC und Assembler als Favoriten, knapp gefolgt von Pascal. Alle anderen Sprachen sind weit abgeschlagen.

Wir sehen aus diesem Meinungsquerschnitt, daß unsere Leser großteils mit dem SVI-Journal zufrieden sind. Hardware- und System-Information werden wir ab nun forcieren. Ebenso wird auf CP/M etwas mehr Wert gelegt. Der Umfang - wie gesagt - kann von uns nicht grenzenlos gesteigert werden, doch wir hoffen, allen Wünschen möglichst gut nachzukommen.

Hier noch einige Vorschläge unserer Leser und unsere Antworten darauf:

"Informationen über allgemeine Entwicklungen am MC-Markt!" Da wir eine Spectravideo-MSX-Bondwell-Zeitschrift sind, wollen und können wir keine allgemeinen Informationen geben. Wir haben weder den entsprechenden Hintergrund noch die Kontakte dazu. Die diversen Computerzeitschriften geben hier jedoch genug Auskunft. Über MSX-Entwicklungen und Spectravideo-Geschehen werden wir Sie immer so gut wie möglich auf dem Laufenden halten!

"Literatur!" Wir bemühen uns schon seit einiger Zeit, auch etwas Literatur ins SVI-Journal einfließen zu lassen, doch auch hier ist unsere Zeitschrift in erster Linie nicht darauf konzipiert.

"Leserbriefkasten!" Die ganze Zeitschrift ist ein Leserbriefkasten, auch wenn es derzeit nicht so aussieht. Viele Berichte entstehen aus spontanen Anfragen. Leider bekommen wir allerdings sehr wenig Briefe von unseren Mitgliedern.

"Softewareinfo!" Wir beziehen diese Anfragen auf Profisoftware, da Spiele und Programme von Clubmitgliedern sowieso in der Programmecke vorgestellt werden. Wir werden es im Rahmen des Schwerpunktes CP/M berücksichtigen.

"Schaltpläne!" Sofern Mitglieder Schaltungen selber basteln (siehe Rafael Razim "Kassetteninterface"), räumen wir ihnen gerne Platz ein.

"Werbeprämien für Abonnenten!" Wir glauben, unser Budget besser anlegen zu können, als Mitgliedern mit den kleinen grünen Scheinchen winken zu müssen. Man sollte genug Idealismus besitzen, auch ohne Geld andere Leute von der Klasse des Spectravideo zu überzeugen und ihnen die Vorteile eines Clubs begreiflich zu machen.

Programmecke, Fortsetzung von Seite 22

es für besser, die eigene Phantasie mit Vorschlägen anzuregen als vorgekaute Stories brühwarm weiterzugeben. Im weitesten Sinne soll außerdem nur eine Einführung in das jeweilige Spiel samt Inhaltsangabe gegeben werden.

Die Redaktion

Spectra Video Club Austria

Was bietet Ihnen der Spectra Video Club Austria?

- regelmäßige Clubabende mit Gelegenheit zum Informaitonsaustausch
- 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 und über MSX
- kostenloser Bezug der monatlich erscheinenden Clubzeitschrift SVI-JOURNAL
- verbilligter Einkauf von Spectravideo-, MSX- und Bondwell-Produkten
- NEU! Bondwell- und MSX-Besitzer sind ebenfalls willkommen.

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, Christoph Sator, Heinz Schmid, Stephan Traxler, Georg Wolfbauer

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

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

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

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

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

Erscheinungsort Wien
Verlagspostamt 1040 Wien

Bankverbindung:
Creditanstalt: 0964-34147/00
(Spectra Video Club Austria)