MZ-Verlag Harald Schicke • Postfach • D-2110 Buchholz 5 ISSN 0931-8860 • Nr. 5/6 • September/Dezember 1988

Einzelpreis DM 15,--

MagaZin 700/800

Das Profi-Magazin für alle Anwender von MZ-700/800

Wir wünschen Ihnen ein erfolgreiches neues Jahr 1989!

Gemeinsam gehen wir in den dritten Jahrgang unseres MagaZin 700/800!

Wir wünschen Ihnen und Ihrer Familie ein frohes Weihnachtsfest!

Editorial

Die Tage werden kürzer, oft ist es draußen ungemütlich. Da hat man mehr Zeit als gewöhnlich, um sich seinem Computer zu widmen.

Wäre es nicht eine gute Idee, jetzt ein tolles neues Programm zu schreiben und es zu veröffentlichen? Schauen Sie Ihr Schatzkästlein doch einmal genau durch! Sicherlich werden Sie manches finden, daß anderen gut helfen würde, eine Programmieraufgabe zu bewältigen.

Einer, der ganz bestimmt helfen wird, ist Claus Becker. Er hat ein Herz für die Anfänger! Er hat sich eine Wahnsinnsarbeit gemacht und eine tolle Anleitung für Anfänger geschrieben, wie man mit seinem MZ-700/800 programmieren kann. Da Herr Becker diese Arbeit ohne Honorar geleistet hat, wird seine Broschüre auch nicht verkauft. Wir machen es wie bei den Freiprogrammen. Sie zahlen nur eine geringe Gebühr für die Kopier- und Versandkosten, nämlich DM 10,--. Die Broschüre liegt versandbereit hier im MZ-Verlag. Näheres erfahren Sie auf Seite 3.

Computerzeit kann auch Bastelzeit sein! Deshalb veröffentlichen wir in diesem Heft gleich zwei Bauanleitungen. Besonders angetan sollten Sie von der Anleitung zum Bau eines Floppy-Controllers sein. So kommen Sie für nur ca. DM 90,-- an einen Controller, an den Sie bis zu vier IBM-kompatible Laufwerke anschließen können. So preiswert war der Weg zur eigenen Floppy noch nie!

Mit der eigenen Floppy-Disk ist der Weg endlich frei zu CP/M und zu Eumel/Elan! Damit ist auf dem MZ-800 professionelles Arbeiten möglich! Erst mit einer Floppy kann der MZ-800 alle seine Qualitäten ausspielen.

Um Ihnen den Einstieg in CP/M ein bißchen schmackhaft zu machen, veröffentlichen wir in diesem Heft

Verleger Harald Schicke

das Listing für einen Full Screen Editor in Pascal unter CP/M.

Lars Hanke, der Autor dieses Programmes, würde gern noch weitere Programme zur Verfügung stellen. Aber dafür brauchen wir erst mehr CP/M-Anwender. Der Weg dahin ist durch die Bauanleitung für den Floppy-Controller endlich geebnet. Also, packen Sie´s an!

Was Sie auch noch anpacken sollten, das ist das neue Buch für den MZ-800 von Edgar Lefgrün. Alle vorbestellten Exemplare sind bereits ausgeliefert worden. Zu Weihnachten wäre es gerade das richtige! Das Inhaltsverzeichnis des Buches bringen wir auf Seite 4, damit Sie sich ausführlich informieren können. Bitte denken Sie daran, daß wir bei diesem Buch gelernt haben und deshalb nur eine kleine Auflage drucken ließen. Bitte bestellen Sie unter der Bestellnummer B 187. Wenn dieses Buch gut läuft, wird es sicherlich nicht das letzte zum MZ-800 gewesen sein.

Damit wir wieder pünktlich werden, haben wir eine Doppelnummer gemacht. Im nächsten Jahr wird es weitergehen. Aber eins brauchen wir dafür von Ihnen: Bitte schicken Sie uns weitere Artikel zu oder geben Sie uns ein paar Anregungen, was Sie noch interessieren würde, damit MagaZin 700/800 auch weiterhin interessant für Sie bleibt!

Impressum

Redaktion: Harald Schicke
Mitarbeiter dieser Ausgabe: Claus Becker, Hans Werner und Bernd Birkenbach, Oliver Braun, Matthias Großmann, Dirk Grube, Lars Hanke, Hans-Jürgen Hillgemann, Axel Lücking, Rafael Metz, Volker Möllenhoff
Anzeigen: Harald Schicke
Vertrieb: MZ-Verlag
Druck: Wemcard GmbH, 3226 Sibbesse 3
Verlag:
MZ-Verlag Harald Schicke, Postfach
(für Pakete: Lindenweg 18),
D-2110 Buchholz 5
☎ 0 41 87/65 33
Telex: 051933521 dmbox g
ref: box:dm4:mz-verlag
BTX: 041876533

MagaZin 700/800 ist eine unabhängige Zeitschrift und nicht SHARP Electronics angegliedert.
MagaZin 700/800 erscheint sechs mal im Jahr. Der Einzelpreis beträgt DM 7,50. Im Abonnement kostet es DM 36,-- pro Jahr (Ausland DM 42,--). Das Abonnement gilt grundsätzlich für ein Kalenderjahr und verlängert sich automatisch um ein Jahr, wenn es nicht bis sechs Wochen vor Ende des Kalenderjahres gekündigt wird.

ISSN 0931-8860

Hilfe? – Hilfe!

Wie oft haben wir Kritik einstecken müssen von fortgeschrittenen Computer-Anwendern, die die Artikel im MagaZin 700/800 zu leicht fanden. Manch einer hat deshalb sein Abo gekündigt. Gleich viel Kritik kam von den Anfängern, die die Artikel und Listings wiederum als viel zu schwierig empfanden.

Den Fortgeschrittenen haben wir inzwischen wohl einiges geboten. Jetzt sind die Anfänger dran! Bitte lesen Sie, was Claus Becker dazu geschrieben hat:

„Kein Meister ist je vom Himmel gefallen!

Ein Gerät vereint uns alle: Der SHARP-Computer, MZ-Serie.

Wir alle verfügen über einen Computertyp, der im Rahmen der 8-Bit-Technologie ein hervorragendes Leistungssprektrum benutzerfreundlich bietet. Weniger benutzerfreundlich erscheint mir allerdings die Tatsache, daß uns der Hersteller nur geringe Benutzerpflege angedeihen läßt. Da gibt es nur ein Mittel: Selbsthilfe jeglicher Art. Die Selbsthilfeform gelingt nur durch Zusammenschluß aller Benutzer, sich also zu organisieren. Als Autor dieser Schrift bin ich einer von Tausenden. Anfänglich stand ich dem MZ-Verlag kritisch gegenüber. Meine Einstellung hat sich geändert und ich versuche, im Rahmen meiner Möglichkeiten einen kleinen Beitrag zu liefern zum Wohle aller Benutzer. Denn eines dürfte sich herumgesprochen haben: Der MZ-Verlag hat aus Eigeninitiative diese Organisationsform geschaffen.

Da ich selbst spät an den Computer geriet, und ich kaum Möglichkeiten des Gedankenaustausches pflegen kann und konnte, sind mir die Probleme eines „blutigen Anfängers" sehr wohl bekannt. Aus Eigenerfahrung möchte ich anderen Kollegen in ähnlicher Situation helfen. Ich erhebe nicht den Anspruch, bis ins Kleinste alle Basic-Anweisungen zu kennen. Das mögen berufenere Kollegen tun. Auch ich bin, genaugenommen, Anfänger, wenn man bedenkt, was ich nicht weiß. Aber was ich weiß, gebe ich gern weiter mit dieser kleinen Zusammenstellung zur Erläuterung des Handbuches. Doch man wächst mit seinen Aufgaben, wie es so schön heißt. Meine mir selbstgestellte Aufgabe lautet: Den Kollegen zu helfen, die mit den Beispielen im Handbuch bei grundlegenden Anweisungen nicht zurecht kommen. Es kann nicht Sinn dieser Zusammenstellung sein, das Handbuch neu zu bearbeiten. Jedoch, so meine ich, ist es sinnvoll, weitere Hinweise zum Handbuch zu geben.

Für Kritik bin ich dankbar. Schreiben Sie ruhig, wie es Ihnen gefällt. Juristische Ahndungen sind nicht zu befürchten. Ihre Zuschriften werden als „persönlich-kollegiale Bemerkungen" gewertet. Und mein Humor dürfte einiges verkraften.

Gleichzeitig aber fordere ich alle Kollegen auf, im Rahmen der Möglichkeiten Schwerpunkte zu setzen und ebenfalls mitzuarbeiten. Dann sind uns kaum Grenzen gesetzt und mancher Besitzer eines anderen Computertyps wird uns beneiden.

Packen wir's an!

Euer Kollege Claus Becker"

So bestellen Sie:
Überweisen Sie DM 10,-- auf unser Postgirokonto Hamburg,
Harald Schicke Verlag,
Konto-Nr. 3595 74-203,
BLZ 200 100 20,
mit dem Stichwort „Anfänger". Bitte vergessen Sie nicht, Ihre Adresse anzugeben! Viel Erfolg und viel Spaß mit dieser tollen Anleitung bei der Programmierung!

Inhalt

Tips, Tricks + Informationen

Inhaltsverzeichnis Seite

Tips, Tricks + Informationen Das neue Buch zum MZ-800 und seiner Umgebung

von Edgar Lefgrün

Wer hätte das gedacht? Der MZ-800 ist kaum noch zu haben (deshalb steigen schon wieder die Preise: MZ-811 DM 248,--, MZ-821 DM 298,--), manch einer hat ihn längst tot gesagt und trotzdem erscheint ein neues Buch!

Edgar Lefgrün und der MZ-Verlag sind dabei ein hohes Risiko eingegangen. Herr Lefgrün hat wahnsinnig viel Zeit investiert und der MZ-Verlag viel Geld, denn Miniauflagen sind immer besonders teuer.

Herr Lefgrün hat das Buch natürlich mit dem MZ-800 und Wordstar erfaßt. So konnte er die Assemblerroutinen direkt einlesen. Wir haben dann das die Diskette mit einem AT-kompatiblen gelesen und über unser Netzwerk in die Mac´s eingelesen. In den Mac´s haben wir dann die Umlaute konvertiert und das Layout erstellt.

Das hat zwei wichtige Folgen: Erstens kann es in den Assemblerlistings keine Tippfehler geben! Alle Listings müßten also problemlos laufen (wenn Sie keinen Eingabefehler machen!). Zweitens konnten wir die Seitenzahl so von knapp 200 auf 103 reduzieren. Dadurch ergab sich ein gerade noch vertretbarer Herstellungspreis.

Links haben wir das Inhaltsverzeichnis abgebildet, damit Sie sich einen Überblick über den reichhaltigen Inhalt verschaffen können. Alle Kapitel enthalten nützliche Beispiele, die Sie sofort in die Praxis umsetzen können oder als Vorbild für eigene Routinen verwenden.

Bitte bestellen Sie das Buch unter der Bestellnummer B 187. Es kostet DM 39,80.

MZ-Forth für MZ-800

Bislang haben wir uns im MagaZin 700/800 fast ausschließlich auf BASIC konzentriert. Es gab auch einige Assemblerlistings und vereinzelt Pascal.

Der Grund ist natürlich ganz einfach: Das BASIC wurde als Grundausstattung zusammen mit dem Computer geliefert. Also fängt man damit an – und bleibt meistens dabei, weil man sonst noch etwas anderes lernen müßte.

Dabei vergibt man aber viele Chancen, denn vieles ist mit BASIC nicht zu machen, weil es z.B. zu langsam ist oder zuviel Speicherplatz benötigt.

Ein Pascal auf Kassette ist schon geraume Zeit nicht mehr verfügbar. Da bietet sich der MZ-Forth Compiler/Interpreter an, der enorme Vorteile aufzuweisen hat.

Die Programmiersprache Forth ist in vieler Hinsicht revolutionär, denn

- Forth ist sehr schnell. Im Schnitt laufen Forth-Programme mehr als 10 mal schneller als vergleichbare BASIC-Programme.
- Forth erfordert sehr wenig Speicherplatz. Die meisten Compiler blähen die Programmlänge enorm auf. Aufgrund der manchmal unglaublichen Codeeffizienz von Forth lassen sich sehr leistungsfähige Programme auf relativ wenig Speicherplatz unterbringen (in einer Projektstudie wurde festgestellt, daß ein kompiliertes Fortran-Programm von 64K in Forth nur etwa 16K benötigte).
- Forth ist strukturiert. Wie alle moderneren Hochsprachen (Pascal, C) enthält Forth schachtelbare Strukturen für Verzweigungen und Schleifenbildung.
- Forth ist sowohl eine Interpreter- als auch eine Compilersprache. Die Programme können genauso einfach interaktiv entwickelt werden wie mit einem BASIC-Interpreter. Der genial einfache Forth-Compiler sorgt dennoch für eine wesentlich höhere Ablaufgeschwindigkeit.
- Forth ist maschinennah. Daher kann Forth in vielen Fällen die Assemblerprogrammierung vollständig ersetzen.
- Forth kann in beliebiger Richtung erweitert und so als problemorientierte Sprache für das jeweilige Anwendungsfeld optimiert werden.
- Forth ist eine offene Sprache. Da Forth dem Programmierer keine Zwangsjacken anlegt, können nahezu beliebige Daten- und Programmstrukturen verwirklicht werden.

Allerdings erfordert die Programmierung in Forth ein gründliches Umdenken. Deshalb wird manchem (mir auch) zu Anfang Forth als schwierig erscheinen. Sobald man sich jedoch daran gewöhnt hat, kann man wesentlich produktiver und kreativer arbeiten als zuvor in BASIC.

Das hier vorgestellte MZ-Forth bietet eine ganze Reihe von Vorteilen:

- Eine eingebaute RAM-Disk ermöglicht auch ohne Diskettenstation eine Arbeitsweise, die vollständig den Forth-Systemen mit Diskettenstation entspricht. Durch wirkungsvolle Kompressionsmechanismen wird dabei erreicht, daß in der RAM-Disk im Durchschnitt zwei- bis dreimal soviel Platz für Programmtexte zur Verfügung steht, wie an Speicherplatz eigentlich vorhanden ist.
- Ein komfortabler bildschirmorientierter Editor erleichtert das Schreiben und Ändern von Programmen.
- Klare und eindeutige Fehlermeldungen in deutscher Sprache erleichtern die Fehlersuche. Tritt bei der Kompilierung ein Fehler auf, so kann der Editor aufgerufen werden, der den Cursor automatisch an der Fehlerstelle plaziert.
- Ein ausgefeiltes Ein- und Ausgabekonzept ermöglicht es, in einfacher Weise die Ein- und Ausgaben des Forth-Systems auf unterschiedliche Kanäle umzuleiten, eigene (in Forth geschriebene) Ein-/Ausgabetreiber zu installieren und Filterprogramme in die Ein- und Ausgabekanäle einzufügen. Ein Komfort, der sonst nur von moderneren Betriebssystemen wie UNIX oder MS-DOS geboten wird.

MZ-Forth wird auf Kassette mit ausführlichem Handbuch (120 Seiten) in einer Plastikbox geliefert. MZ-Forth kostet DM 98,--. Wir haben einen kleinen Vorrat, so daß es bei schneller Bestellung sofort lieferbar ist.

Bitte beachten Sie, daß Sie für das Arbeiten mit MZ-Forth einen Monitor benötigen, da die Ausgabe im 80-Zeichen-Modus erfolgt. Da ist es sicherlich eine gute Idee, sich neben MZ-Forth auch den noch lieferbaren Farbmonitor MZ-1D05 unter den Weihnachtsbaum legen zu lassen. Er ist mit DM 448,-- zuzüglich DM 10,-- Versandkosten erstaunlich preisgünstig!

MZ-1R12 Preiswerte Floppies

Speichererweiterung für den MZ-700

Angeblich ist die Speichererweiterung MZ-1R12 für den MZ-700 nie ausgeliefert worden. Trotzdem haben wir nun fünf Stück ergattert.

Die Speicherweiterung umfaßt 32K und war in erster Linie dafür gedacht, Systemprogramme, z.B. BASIC, statt von Kassette direkt aus dem Zusatz-RAM zu laden. Damit das funktioniert, muß das RAM natürlich batteriegepuffert sein. Dazu dient ein Akku, der bei vollem Ladezustand bis zu 1,5 Monate die Daten frisch hält. Wird der MZ-700 benutzt, lädt sich der Akku automatisch nach. Leider wird für den Betrieb die Erweiterungsbox MZ-1U06 benötigt.

Beim Einschalten des MZ-700 startet BASIC automatisch, wenn es zuvor einmal von Band in die Speichererweiterung geladen worden ist. Man kann allerdings auch andere Programme oder Daten einladen. Das geschieht über I/O-Ports.

Alle Möglichkeiten werden in der englischen Bedienungsanleitung mit Beispielen erklärt, so daß die Benutzung eigentlich kein Problem sein dürfte.

Wahrscheinlich kann man die Erweiterung auch am MZ-800 benutzen. Für das 800er BASIC ist die Erweiterung jedoch zu klein.

MZ-1R12 kann ab sofort bestellt werden. Es kostet DM 98,--. Eine Garantie kann jedoch nicht übernommen werden.■

Teure Floppies für den MZ-800?

Das muß nicht sein. Der Floppycontroller für den MZ besitzt den Standardausgang. Es läßt sich also problemlos jedes Standardlaufwerk anschließen. Dazu kann man ein hochwertiges Laufwerk verwenden, wie das von Karasch Datentechnik verwendete SD-521 von Epson oder aber jedes 360K Laufwerk für den XT. Solche Laufwerke gibt es in qualitativ akzeptabler Bauweise z.B. von TEAC ab DM 120,--.

Zum Anschluß eines Laufwerkes legt man einfach die vom Controller ankommenden Leitungen an den 34-poligen Stecker des Laufwerks an. Es empfiehlt sich, Flachbandkabel und Stecker in Schneid-Klemm-Technik zu verwenden. Es ist darauf zu achten, daß Pin 1 der Pfostenleiste des Controllers mit Pin 1 des Kartensteckers verbunden wird. Um Störeinflüsse gering zu halten, sollte das Kabel eine Länge von 1m nicht wesentlich überschreiten.

Die Spannungsversorgung wird durch ein Netzteil bereitgestellt, das 5 V bei 500 nA und 12 V bei 1 A bereitstellt. Bei Stabilisierung, Siebung und Verdrosselung sollte man für einen fehlerfreien Betrieb nicht sparen. Zur Stabilisierung genügt jedoch ein Spannungskonstanthalter aus der 78xx Serie, zur Siebung schlage ich 4700 µF und zur Verdrosselung zwei Ringkerndrosseln von jeweils 50 mH oder mehr vor. Zur Vermeidung von Schwingungen und Rauschen sollte man Trafo (sekundär), den 78er beidseitig wie auch den Ausgang mit 10 nF gegen Masse sieben. Für den Anschluß mehrerer Laufwerke muß die Leistungsfähigkeit des Netzteils entsprechend erhöht werden. Der Punkt 1 des 4-poligen Versorgungssteckers wird mit 12 V und der Punkt 4 mit 5 V verbunden. Die beiden Mittleren sind Masse.

Die Laufwerksnummer wählt man mit einem 4- oder 5-stelligen Jumper, der meist in der Nähe des Kartensteckers ist. Die Positionen 1 bis 4 entsprechen den jeweiligen Laufwerksnummern und die Position 5 kennzeichnet einen unbedingten Zugriff.

Sollen mehrere Laufwerke angeschlossen werden, so ist dies sehr einfach. Man klemmt lediglich einen weiteren Stecker auf die Datenleitung und entfernt aus den Zweitlaufwerken das Pull-up-Widerstandsarray, welches sich meist gesockelt in der Nähe des Kartensteckers befindet. Übrigens ist das Netzteil eines Originallauffwerkes in der Lage, ein zweites mitzuversorgen. Es ist darauf zu achten, daß jedem Laufwerk mit dem Jumper eine andere Nummer zugeordnet wird. Ein unbedingter Zugriff ist nicht zulässig!

Und nun noch ein Bonbon. Von Seiten des Controllers bestehen keine Bedenken auch ein 80-Track-Laufwerk, z.B. für das P-CP/M SDS-768 Format anzuschließen. Da es sich um eine Standardschnittstelle handelt, sollten dafür auch die 1,2 MB Laufwerke für den AT verwendbar sein.

Für den Controller habe ich bis jetzt keinen Ersatz gefunden, was jedoch mehr daran liegt, daß ich bis dato keinen gesucht habe. Es sollte jedoch nicht schwer sein, einen Standardcontroller für 8-Bit-Systeme, z.B. von Western Digital, an den MZ anzuschließen.

Alle Angaben, die ich über Laufwerke gemacht habe, waren am SD-521 von Epson orientiert. Ich kann nicht garantieren, daß andere Laufwerke in Werten und Positionen übereinstimmen. Sollten Zweifel bestehen, so sollte man sich die Unterlagen über das jeweilige Laufwerk vom Hersteller anfordern. In den meisten Fällen wird diesem Wunsch sogar unentgeldlich nachgekommen.

Ein Gehäuse für die Laufwerke ist im Elektronik-Fachhandel für durchschnittlich DM 25,-- erhältlich.

Ich hoffe, daß dieser Artikel einigen MZ-Benutzern die Scheu vor teuren Laufwerken und damit vor CP/M genommen hat. **Für diese vergleichsweise kleine Investition erschließt man sich nicht nur neuen Bedienungskomfort unter BASIC, sondern auch die gewaltige Softwarewelt von CP/M** und damit auch eine große Menge äußerst interessanter Programme (Sprachen, Spiele, Utilities, Geschäftsprogramme, ...) aus Public Domain Beständen.

Lars Hanke

HUKAMA.00

Testen Sie die Geschwindigkeit von Ihrem MZ-800 und Ihre Intelligenz! Mit diesem Programm möchte Oliver Braun, Kettelerstr. 28, D-6612 Schmelz, ☎ 06887/4883 zur Gestaltung des MagaZin 700/800 beitragen. Viel Spaß bei Ihrem Test!

```
10 INIT "LPT:M2,S2,B7,10"
20 CLS
30 REM HUND,KATZE,MAUS
40 PRINT "AUFGABE"
50 PRINT "-------"
60 PRINT "PETER HAT 500 MARK UND"
70 PRINT "ER SOLL GENAU 500 TIERE KAUFEN;          MINDESTENS JE 1 STUECK"
80 PRINT"ES DARF KEIN GELD UEBRIG BLEIBEN"
90 PRINT
100 PRINT"1 HUND  KOSTET 10 MARK"
110 PRINT"1 KATZE KOSTET  1 MARK"
120 PRINT"1 MAUS  KOSTET 25 PFENNIG"
130 PRINT
140 PRINT"WILLST DU EINE DRUCKERAUSGABE J/N :";:INPUT J$
150 IF J$="N" THEN 190
160 IF J$="J" THEN 200
170 IF J$<>"J" OR J$<>"N" THEN 140
180 PRINT
190 PRINT "MOEGLICHE LOESUNGEN:"
200 PRINT "-------------------"
210 PRINT
220 PRINT"ANZAHL  HUNDE   KATZEN   MAEUSE    ZEIT"
230 PRINT"-------------------------------------"
240 A$="###)        ##        ###       ####"
241 A=1
250 TI$="000000"
260 H$="##"
270 K$="###"
280 M$="####"
290 FOR H=1 TO 33
300 FOR K=1 TO(500-15*H-1)
310 LET M=500-H-K
320 IF(15*H)+(1*K)+(.25*M)=500THEN PRINTUSING A$;A,H,K,M;:PRINTTAB(35);MID$(TI$,A
,1);":";MID$(TI$,5,2):A=A+1
330 NEXT K
340 NEXT H
350 CURSOR 2,24:PRINT:PRINT"RECHENZEIT: ";MID$(TI$,3,2);" MINUTE(N) ";MID$(TI$,5
,2);" SEKUNDEN"
360 IF J$="N" THEN "END"
370 REM * DRUCKERAUSGABE *
380 PRINT/P "AUFGABE"
390 PRINT/P "-------"
400 PRINT/P "PETER HAT 500 MARK UND"
410 PRINT/P "ER SOLL GENAU 500 TIERE KAUFEN; MINDESTENS JE 1 STUECK"
420 PRINT/P"ES DARF KEIN GELD UEBRIG BLEIBEN"
430 PRINT/P
440 PRINT/P"1 HUND  KOSTET 10 MARK"
450 PRINT/P"1 KATZE KOSTET  1 MARK"
460 PRINT/P"1 MAUS  KOSTET 25 PFENNIG"
470 PRINT/P:PRINT/P
480 PRINT/P"MOEGLICHE LOESUNGEN:"
490 PRINT/P"-------------------"
500 TI$="000000"
510 PRINT
520 X=1
530 FOR H=1 TO 33
540 FOR K=1 TO(500-15*H-1)
550 LET M=500-H-K
560 IF(15*H)+(1*K)+(.25*M)=500THEN PRINT/PX;")";H;" HUNDE";K;" KATZEN";M;" MAEUS
E":PRINT/P"TAB(35)"ZEIT=";MID$(TI$,4,1);":";MID$(TI$,5,2):X=X+1
570 NEXT K
580 NEXT H
590 PRINT/P:PRINT/P"RECHENZEIT GESAMT : ";MID$(TI$,3,2);" MINUTE(N) ";MID$(TI$,5
,2);" SEKUNDEN"
600 LABEL"END"
```

Wir bauen einen Floppy-Controller

Floppy-Controller-Karte für MZ-800

Hier ist eine Schaltung für die MZ-800 Besitzer, die gern selbst etwas bauen wollen. Das Herz der Schaltung ist der Controller IC 1791 von Siemens oder Western Digital. Gegenüber dem SHARP Controller hat der 1791 noch ein Pin für +12 V. Da man für die Floppy-Laufwerke selbst +12 V und +5 V braucht, muß man ein externes Netzgerät haben. Mit diesem Netzgerät kann man auch die Floppy-Controller-Karte versorgen. Bitte nicht versuchen, die +5V aus dem MZ-800 zu nehmen und nur die +12V in einem Netzgerät zu erzeugen. Nach meiner Erfahrung müssen vermutlich beide Spannungen gleichzeitig am 1791 anliegen. Die Stromaufnahme der Karte beträgt bei +5 V ca. 0,8 A und bei +12 V 0,2 A. Die gesamte Schaltung baut man am besten auf einer Lochraster-Platine auf und bringt sie im Laufwerksgehäuse unter. Es ist natürlich etwas schwierig, einen Stecker für die I/O-Erweiterungsbuchse zu beschaffen. Man kann sich aber aus einer doppelseitig kupferkaschierten Epoxy-Platte einen Stecker selbst ätzen. Wer will, kann auch eine Platine herstellen, die direkt in den I/O-Erweiterungsslot eingesetzt werden kann.

Die Adressierung für den 1791 und die Speicher läuft über die Portadressen 0D8H bis 0DEH. Dazu dienen die beiden 74LS138 und ein paar NAND- und NOR-Gatter. Die 8x3 kΩ Widerstände zwischen dem Datenbustreiber und dem Controller sollen Störungen auf dem Datenbus vermeiden. Der 1791 hat einen invertierten Datenbus. Das bedeutet, daß die Befehle invertiert ausgegeben werden. Das ist im Monitor und im Disk-BASIC schon berücksichtigt, da ja der SHARP-Controller auch einen invertierten Bus hat. Es ist nur dann zu beachten, wenn man selbst mit dem Controller arbeiten will. Das ist aber sehr schwierig, da der Controller einen sehr komplizierten Befehlssatz hat. In dem Buch „Die Prozessoren 68000 und 68008" von Rolf-Dieter Klein, Franzis-Verlag, ist eine Controller-Karte mit dem Befehlssatz sehr gut beschrieben. Der Datenseparator hat zum Einstellen der Präkompensation drei Pins: P0, P1 und P3. Ich habe sie auf die kleinste Zeit (0 nSek) festgelegt. Der Takteingang am Pin 11 benötigt einen Takt von 16 MHz. Den Takt kann man mit der gezeigten Schaltung oder mit einem Quarz-Oszillator erzeugen.

Die gesamte Platine kostet ca. DM 90,--. Das teuerste sind die beiden IC 1791 und 9229B. Beide habe ich zusammen für ca. DM 50,-- bei der Firma Frank Electronik GmbH in Nürnberg bekommen. Die 5,25" Laufwerke müssen IBM PC kompatibel sein, das heißt:

- zwei Köpfe
- 40 Spuren pro Seite
- Aufzeichnungsart MFM
- Kapazität 360 KByte
- Shugartbus

Es können bis zu vier Laufwerke an den Bus angeschlossen werden. Die Pullup Widerstände in den Laufwerken für die Leitungen: MOTOR ON, SEITE, STEP, WG, DIRC und WDATA dürfen nur beim letzten Laufwerk auf dem Bus liegen. Bei den anderen Laufwerken werden sie durch Brücken oder Schalter entfernt. Die Brücken für die Laufwerksleitungen 1 bis 4 müssen auf jedem Laufwerk, je nach Zuordnung, gesteckt werden. Was jetzt noch fehlt, ist ein Netzgerät und das Disk-BASIC MZ-2Z046. Das BASIC bekommen Sie beim MZ-Verlag (DM 138,--). Und nun können Sie anfangen zu bauen. Der Autor lehnt, wie bei solchen Schaltungen üblich, für eventuelle Schäden jede Haftung ab.

MZ-700

Noch eine Bemerkung für die MZ-700 Besitzer. Die Floppy-Controller-Karte kann natürlich auch am MZ-700 betrieben werden. Nur kann das 700er Disk-BASIC MZ-5Z008 oder MZ-2Z009 nicht von der Diskette geladen werden, da der IPL-Lader im 700er Monitor nicht vorhanden ist. Wenn aber das Disk-BASIC von der Kassette eingeladen ist, kann man nun mit den Laufwerken arbeiten. Auf der SHARP-Controller-Karte MZ-1E05 sitzt dazu ein 4K EPROM. In diesem EPROM befindet sich der IPL-Lader, der beim Einschalten in den Speicherbereich ab Adresse 0F00H eingeblendet wird.

EPROMs + RAM-Karte

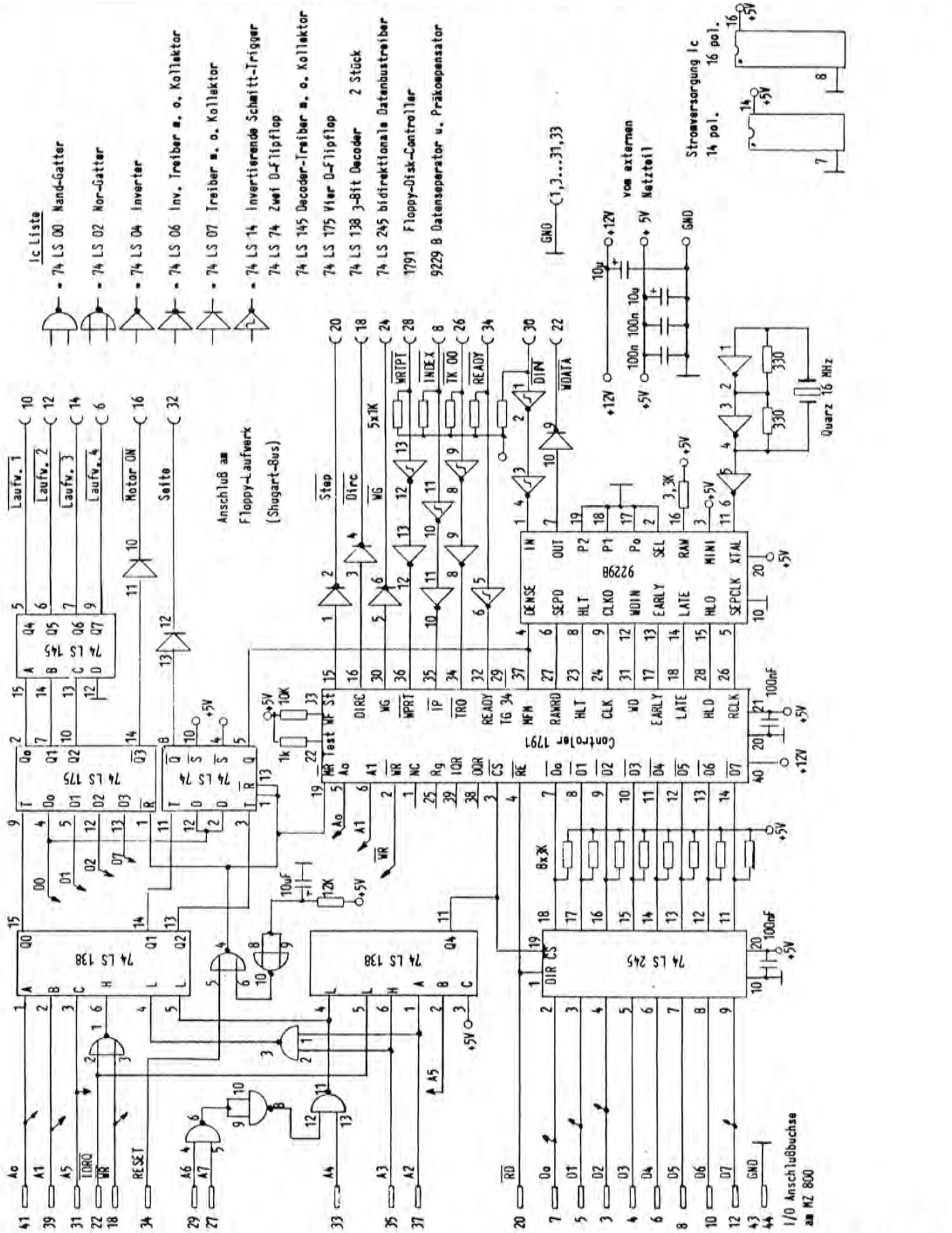
Hier noch ein Wort zu den Programmiersprachen. Bei mir werden die Sprachen auf folgende, diskettenschonende und zeitsparende Weise geladen:

Ich habe alle Programmiersprachen auf EPROMs gebrannt. Die zwei EPROM-Karten mit den EPROMs habe ich, unter anderen Karten, in einer Erweiterungsbox untergebracht. Das Monitor-EPROM wurde mit einem zusätzlichen Ladeprogramm neu gebrannt. Mit einem Schalter kann ich die einzelnen Sprachen anwählen und dann vom Monitor in Sekunden einladen. Ich habe auch eine 64K Ramkarte mit Static-RAM aufgebaut. Wenn Interesse besteht, kann ich die Schaltung mal vorstellen.

Die Schaltung finden Sie auf Seite 9.

Hans Werner und Bernd Birkenbach
Wiener Str. 31, 6000 Frankfurt 70

Wir bauen einen Floppy-Controller



Turbovoc - der Vokabeltrainer/M2-800

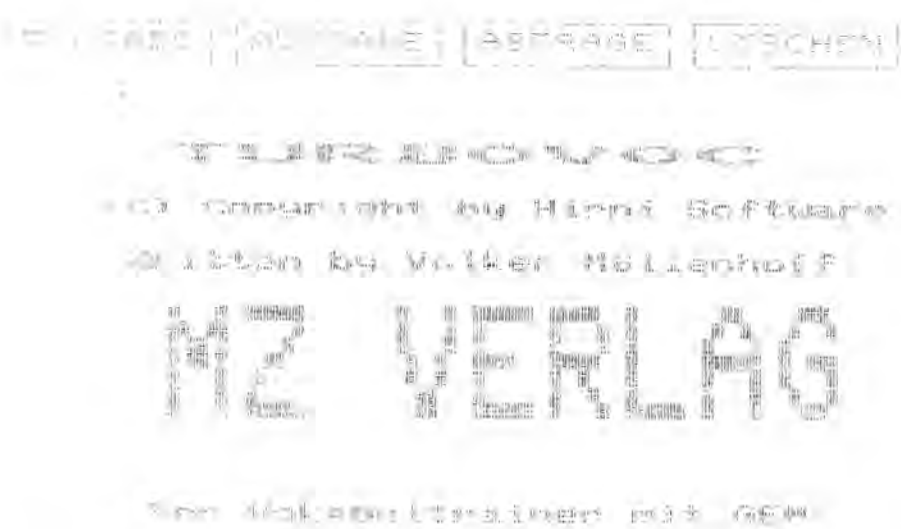
```
10 REM ---- TURBOVOC  V.1 ----
20 REM ---Volker Möllenhoff---
30 REM ----Weidenstraße 19----
40 REM -----8011 BALDHAM------
50 INIT"CRT:M1"
60 LABEL"GDAT"
70 DATA 4
80 DATA "EINGABE",1,1
90 DATA "AUSGABE",10,1
100 DATA "ABFRAGE",19,1
110 DATA "LöSCHEN",28,1
120 SYMBOL 50,50,"TURBOVOC",3,1
130 SYMBOL 1,90,"     Written by Volker Möllenhoff",1,1
140 SYMBOL 2,90,"     Written by Volker Möllenhoff",1,1
150 SYMBOL 0,184,"      Der Vokabeltrainer mit GEM ",1,1
160 SYMBOL 1,184,"      Der Vokabeltrainer mit GEM ",1,1
170 SYMBOL 52,110,"MZ VERLAG",3,6
180 SYMBOL 53,110,"MZ VERLAG",3,6
190 SYMBOL 53,50,"TURBOVOC",3,1
200 SYMBOL 1,70,"    (C) Copyright by Hirni Software",1,1
210 SYMBOL 2,70,"    (C) Copyright by Hirni Software",1,1:GOSUB"GEM"
220 IF B$="EINGABE"THEN B=1 ELSE IF B$="AUSGABE" THEN B=2 ELSE IF B$="ABFRAGE" T
HEN B=3 ELSE IF B$="LöSCHEN" THEN B=4
230 ON B GOSUB "EINGABE","AUSGABE","ABFRAGE","LöSCHEN"
240 CLS:GOTO 50
250 LABEL"EINGABE"
260 CLS
270 SYMBOL 0,0,"EINGABE",3,1
280 SYMBOL 3,0,"EINGABE",3,1
290 CURSOR 5,5:INPUT"Vokabel Nummer:";VN
300 CURSOR 5,7:INPUT"Form in Sprache 1:";A$
310 CURSOR 5,9:INPUT"Form in Sprache 2:";B$
320 CURSOR 5,11:PRINT"Alles Ok ?"
330 GET F$:IF F$="N" THEN 250 ELSE IF F$="J" THEN "AOK"
340 GOTO 330
350 LABEL"AUSGABE":CLS
360 SYMBOL 10,10,"AUSGABE",3,1
370 SYMBOL 13,10,"AUSGABE",3,1
380 CURSOR 5,5:INPUT"Wieviele Vokabeln:";AVN
390 CURSOR 5,7:PRINT"Mit Space = Stop"
400 RESTORE"STORE":BEEP:PRINT:PRINT
410 ON ERROR GOTO 460:FOR T=1 TO AVN
420 READ V$,M$:PRINT
430 PRINT TAB(0);V$;TAB(17);"=";TAB(22);M$
440 IF R$=" " THEN GOSUB"WAIT"
450 NEXTT
460 GET R$:IF R$="" THEN 460 ELSE RETURN
470 LABEL"ABFRAGE"
480 CLS
490 SYMBOL 10,10,"ABFRAGE",3,1
500 SYMBOL 13,10,"ABFRAGE",3,1
510 CURSOR 5,5:PRINT"A> Sprache 1 'F Sprache 2"
520 CURSOR 5,7:PRINT"B> Sprache 1 _' Sprache 2"
530 GET R$:IF R$="" THEN 530
5540 IF R$<>"A" AND R$<>"B" THEN GOTO 530
550 VM$=R$
560 RESTORE"STORE":BEEP
570 ON ERROR GOTO "EOF"
580 F=0:T=1:REM GTS MIT T
590 IF VM$="A" THEN READ A$,B$
600 IF VM$="B" THEN READ B$,A$
```

Turbovoc - der Vokabeltrainer/M2-800

```
610 CLS
620 CURSOR 5,5:PRINT"Frage Nummer ";T
630 CURSOR 5,7:PRINT "Was heisst ";A$;" ?":T=T+1
640 CURSOR 5,9:INPUT"";F$
650 IF F$=B$ THEN CURSOR 5,11:PRINT"Die Antwort stimmt."
660 IF F$<>B$ THEN CURSOR 5,13:PRINT"Die Antwort stimmt nicht.":F=F+1:CURSOR 5,2
1:PRINT A$;" heisst ";B$
670 CURSOR 5,15:PRINT"Weiter mit <CR>"
680 GETR$:IFR$<>CHR$(13)THEN680
690 GOTO 590
700 LABEL"EOF":CLS:PRINTCHR$(17,17,17,17)
710 PRINT"  Hier ihr Ergebniss:"
720 PRINT
730 PRINT"  Von ";T-1;" Vokabeln wussten Sie ";T-F-1;"."
740 PRINTCHR$(17);"  Das sind ganze ";F;" Fehler."
750 GETR$:IFR$=""THEN 750
760 RETURN
770 LABEL"LÖSCHEN":CLS
780 SYMBOL 10,10,"LÖSCHEN",3,1
790 SYMBOL 13,10,"LÖSCHEN",3,1
800 PRINTCHR$(17,17,17,17,17,17,17,17);" Sind Sie sich sicher ?"
810 GET R$:IF R$="" THEN 810
820 IF R$="N" THEN RETURN
830 IF R$="J" THEN 850
840 GOTO 810
850 PRINT CHR$(17,17);"RUN = START":DELETE 30010-
860 LABEL"WAIT"
870 GET R$:IFR$=" " THEN RETURN ELSE GOTO 870
880 LABEL"GEM":RESTORE"GDAT":READ Z:S=5:PAL2,15:PAL1,15
890 DIMW$(Z),A(Z),B(Z)
900 FORT=1TOZ:READW$(T),A(T),B(T):NEXTT
910 FORT=1TOZ
920 CURSORA(T),B(T):PRINTW$(T)
930 BOXA(T)*8-4,B(T)*8-4,A(T)*8+(LEN(W$(T))*8)+4,B(T)*8+8+4
940 'BOXA(T)*8-2,B(T)*8-2,A(T)*8+(LEN(W$(T))*8)+2,B(T)*8+8+2
950 NEXTT:X=10:Y=10
960 LINEA2,10X,Y,X+5,Y+5:LINEA2,10X,Y,X+2,Y:LINEA2,10X,Y,X,Y+2
970 GET R$:A=STICK(0):B=STRIG(0):IFA=0 AND B=0 AND R$="" THEN 970
980 BLINEA2,10X,Y,X+5,Y+5:BLINEA2,10X,Y,X+2,Y:BLINEA2,10X,Y,X,Y+2
990 IF A=1 THEN Y=Y-S
1000 IF A=2 THEN Y=Y-S:X=X+S
1010 IF A=3 THEN X=X+S
1020 IF A=4 THEN X=X+S:Y=Y+S
1030 IF A=5 THEN Y=Y+S
1040 IF A=6 THEN Y=Y+S:X=X-S
1050 IF A=7 THEN X=X-S
1060 IF A=8 THEN Y=Y-S:X=X-S
1070 IF X<0 THEN X=0
1080 IF Y<0 THEN Y=0
1090 IF Y>40 THEN Y=40
1100 IF X>312 THEN X=312
1110 IF VAL(R$)<>0 THEN S=VAL(R$)
1120 IF B=1 THEN GOTO "PRF"
1130 GOTO960
1140 LABEL"PRF"
1150 FORT=1TOZ
1160 Q=A(T)*8-4
1170 W=B(T)*8-4
1180 V=A(T)*8+LEN(W$(T))*8+4
1190 M=B(T)*8+8
1200 IF (X<=V AND X=>M) AND (Y<=M AND Y=>W) THEN SOUND 70,1:GOTO"#"
```

```
1210 NEXT T
1220 GOTO 960
1230 LABEL"#"
1240 IF W$(T)="" THEN 960
1250 B$=W$(T):RETURN
1260 LABEL"AOK":P$=STR$(30000+(VN*10))+" DATA "+A$+","+B$
1270 POKE $5FEF,$C9:USR($58B4,P$):POKE$5FEF,$CA
1280 RETURN
1290 LABEL"STORE"
30010 DATA AA,BB
```

Harddopy:



Anleitung und Hinweise: Vokabeltrainer Turbovoc V.1

Dieses Programm ermöglicht die Eingabe und Abfrage beliebiger Vokabeln. Weiterhin können die Vokabeln auf ein Speichermedium gespeichert werden. Nicht benötigte Datensätze sind löschbar.

Dieser Trainer besitzt zwei besondere Merkmale:

1. Die Eingabe beim Menü erfolgt über die Cursortasten (grafisch orientierte Benutzeroberfläche).
2. Das Programm erweitert sich bei der Dateneingabe selbständig. Die neuen Vokabeldatensätze werden an das Programm angehängt. Wollen Sie Ihre Vokabeln abspeichern, so saven Sie das Hauptprogramm einfach unter einer neuen Versionsnummer ab.

Doch nun zu den einzelnen Menüfunktionen:

1. Eingabe: Sie klicken das Feld EINGABE an. Wenn nach dem Druck der SPACE-Taste nicht sofort das gewünschte Erscheint, dann können Sie durch Druck auf eine der Zahlentasten die Geschwindigkeit des Pfeils verändern (1= langsam bis 9= schnell). Nun werden Sie nach der Vokabelnummer gefragt. Jeder Vokabel ist eine Vokabelnummer zugeordnet. Für die erste Vokabel tippen Sie die 1 ein, für die zweite die 2 usw. Wenn Sie die Sprache ENGLISCH als erste Sprache nehmen wollen, quittieren Siie die nächste Frage mit 1. Bei der Frage nach der zweiten Form der Vokabel logischerweise die deutsche.
2. Ausgabe: Diese Funktion gibt alle im Speicher befindlichen Vokabeln aus. Sie können die Obergrenze dieser Menge durch vorherige Wahl bestimmen.
3. Abfrage: Die Abfrage fragt zuerst nach der Abfragerichtung (z. B. englisch - deutsch oder deutsch - englisch). Der Rest erklärt sich von selber.
4. Löschen: Der Aufruf dieser Funktion löscht den im Speicher befindlichen Datensatz. Vorher wird noch gefragt, ob Sie wirklich löschen wollen.

Hinweise zum Abtippen des Listings:

Achten Sie darauf, daß Sie alle Umlaute übernehmen. Im Falle der Zeilen 110, 220, 230 ist dieses sehr wichtig. ö ist CHR$($A8).

In den Zeilen 510 und 520 tritt ein anderer Druckerfehler auf: Mit den Zeichen „<P" ist natürlich „->" gemeint. Ebenso in der nächsten Zeile. Hier muß es „<-" statt „<" heißen. Weiterhin hat der Drucker Probleme mit der Ausgabe der eckigen Klammern. Dieses Problem tritt in den Zeilen 960 und 980 auf. Das „Ä" steht für Klammer auf, das „Ü" für Klammer zu.

Beachten Sie noch, daß ab der Zeile 1290 keine weitere Programmzeile kommen darf, da ab hier die Vokabeldaten abgelegt werden. Die Zeile 30010 hat folglich auch nichts zu sagen und kann weggelassen werden. Sie stellt nur ein Beispiel für das Format einer Vokabelablage dar.

Power-Box M2-700/800

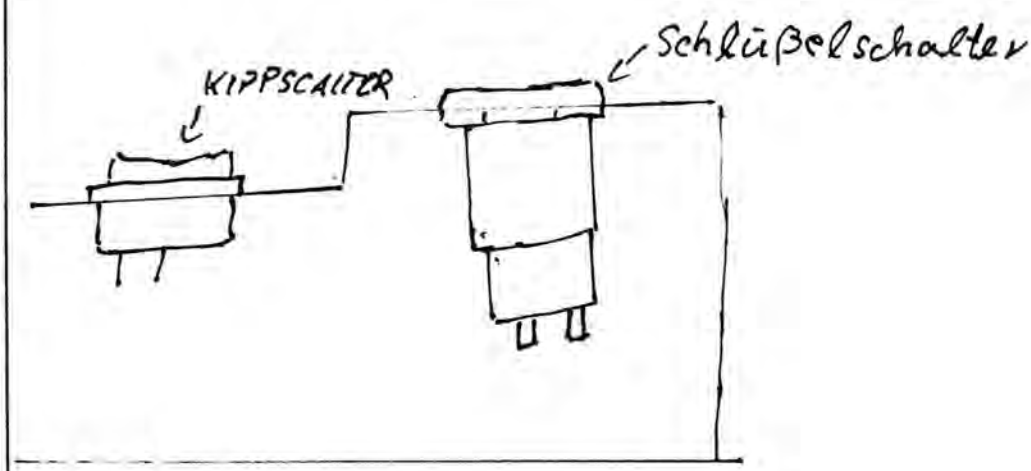
Das ist uns allen wohlbekannt: Das Anschalten der Hardware. Man tastet an der Rückseite nach dem POWER-Schalter. Hier wird dieses „Problem" einfach gelöst.

Eine kleine „Power-Box" von der aus alle Geräte angeschaltet werden können. Ein Schlüsselschalter dient dazu, die ganze Stromversorgung abzuschalten und somit die Geräte vor unerlaubter Inbetriebnahme zu sichern.

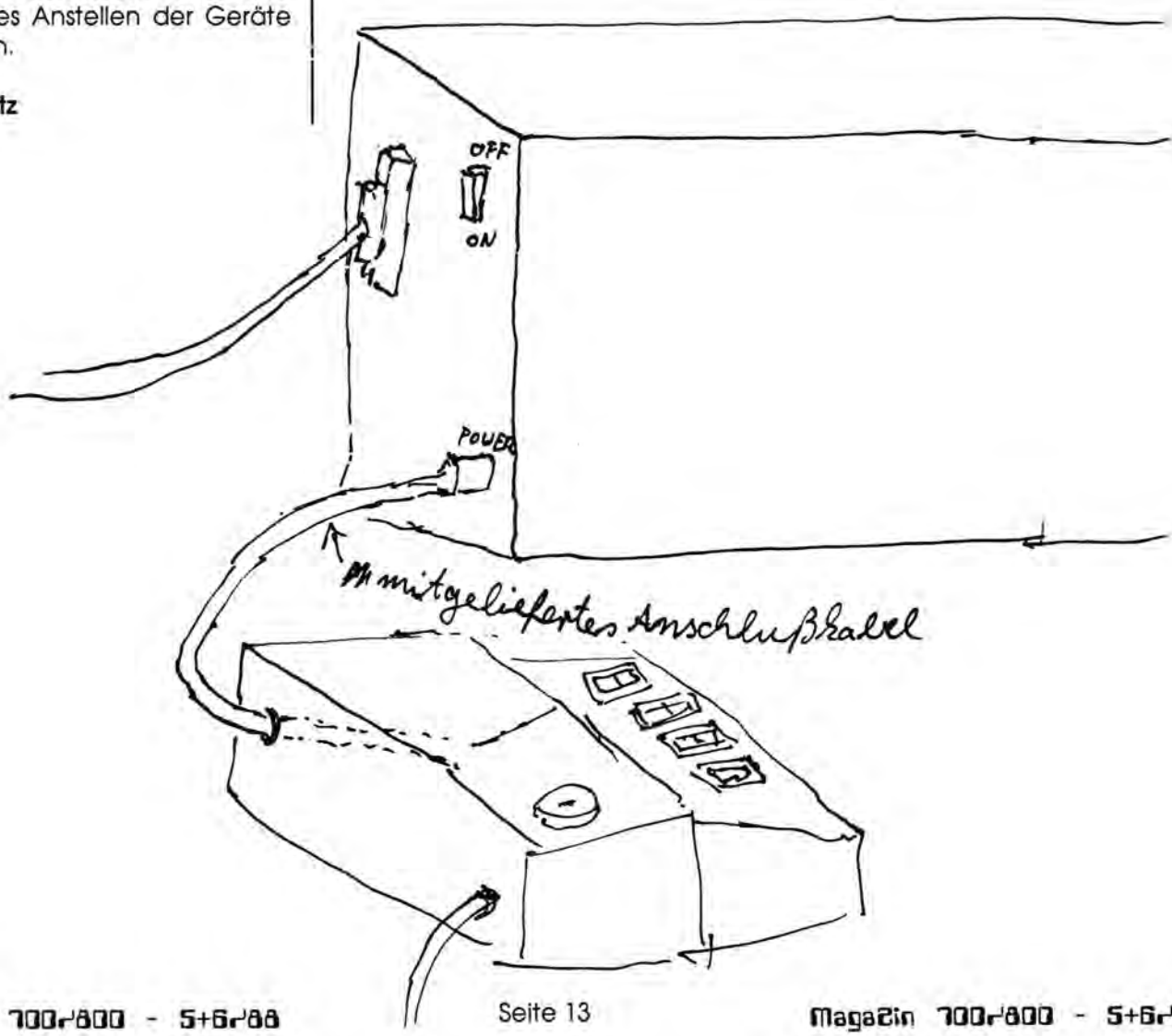
Bauteile:
1 x ABS-Gehäuse
4 x Kippschalter
1 x Schlüsselschalter

Die ON/OFF-Schalter der einzelnen Hardware sind auf ON zu stellen. Durch den Schlüsselschalter wird unerlaubtes Anstellen der Geräte vermieden.

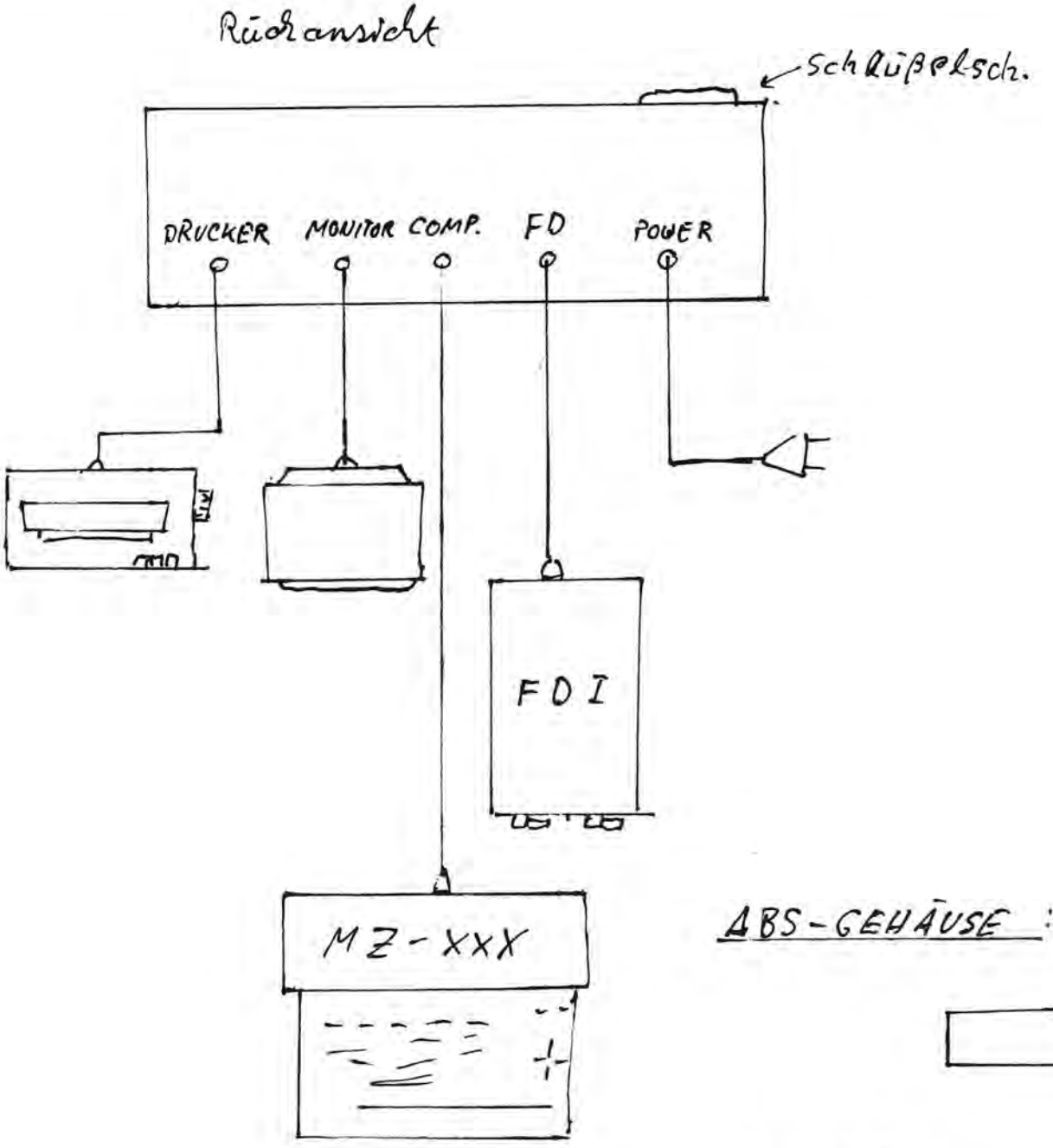
Rafael Metz



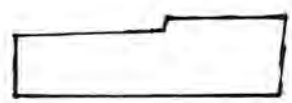
Seitenansicht



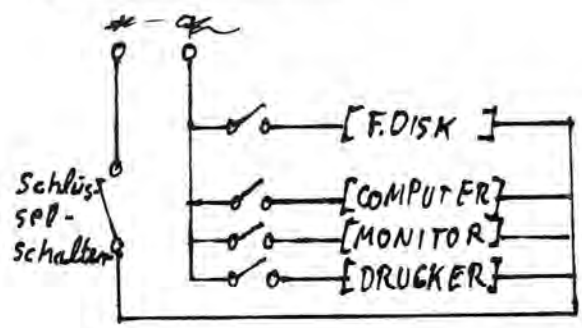
Power-Box MZ-700/800



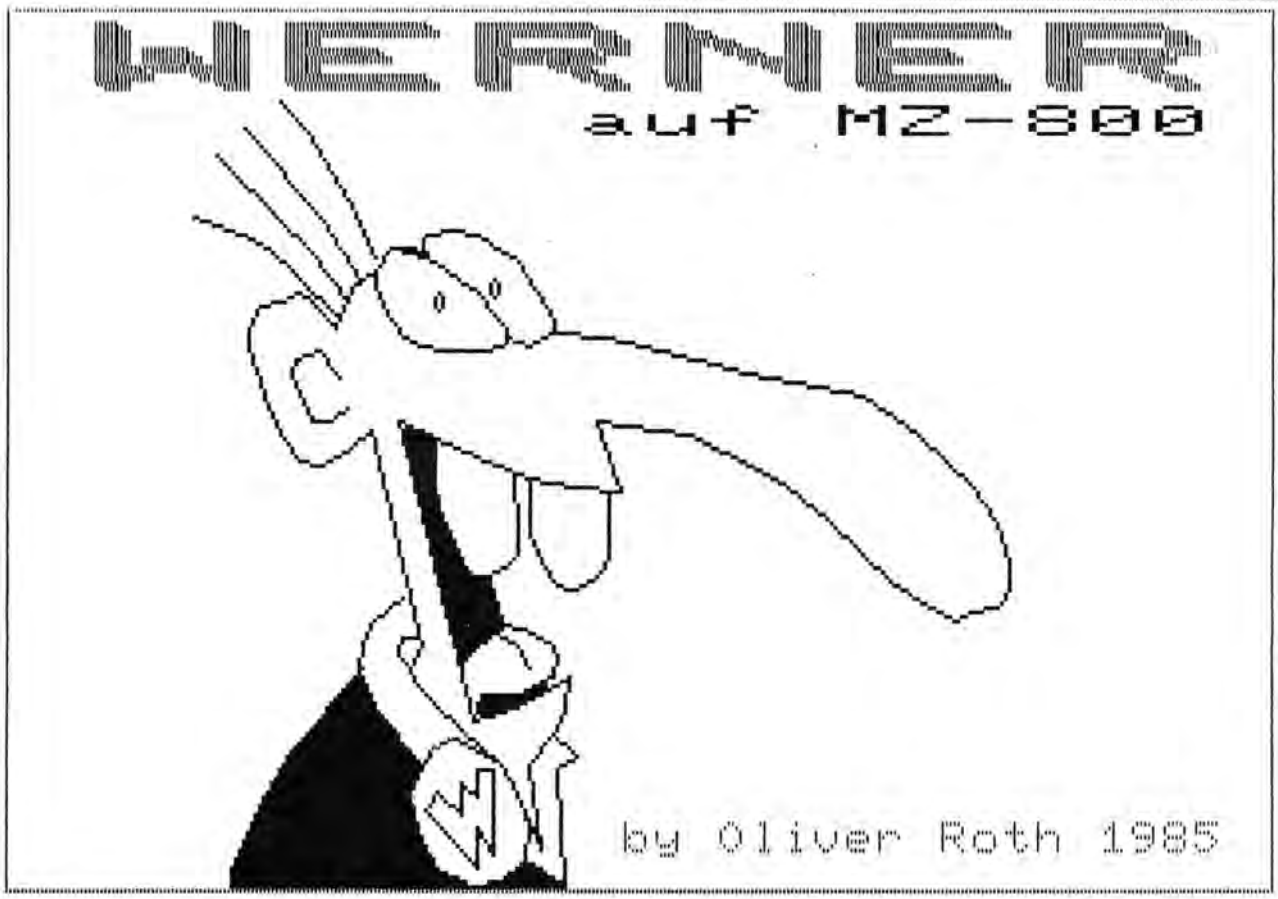
ABS-GEHÄUSE:



SCHALTPLAN:



auto-run mit HCOPY




```
10 DEF KEY( 1)="RUN    "+CHR$(13)
20 DEF KEY( 2)="LIST"
30 DEF KEY( 3)="AUTO"
40 DEF KEY( 4)="RENUM"
50 DEF KEY( 5)="DIR "
60 DEF KEY( 6)="CHR$("
70 DEF KEY( 7)="DEF KEY("
80 DEF KEY( 8)="CONT"
90 DEF KEY( 9)="SAVE  "
100 DEF KEY(10)="LOAD  "
110 FORI=0TO224:READA:POKE$56A4+I,A:NEXTI
120 DATA$CD,$BA,$0,$11,$1E,$57,$CD,$43,$57,$3A,$7A,$13,$FE,$3,$30,$D
130 DATA$3E,$23,$32,$3C,$57,$21,$0,$0,$22,$E6,$56,$18,$A,$AF,$32,$3C
140 DATA$57,$21,$18,$F,$22,$E6,$56,$3E,$32,$DD,$21,$0,$80,$F5,$CD,$75
150 DATA$57,$3E,$4,$F5,$3A,$6B,$13,$21,$D0,$27,$F5,$3E,$1,$D3,$CD,$CD
160 DATA$2C,$57,$18,$F,$3E,$2,$D3,$CD,$7D,$D6,$F,$6F,$30,$1,$25,$CD
170 DATA $2C,$57,$2B,$DD,$23,$F1,$3D,$20,$E1,$F1,$3D,$20,$D6,$CD,$54,$57
180 DATA$F1,$CB,$4F,$28,$7,$47,$CD,$1E,$0,$28,$4,$78,$3D,$20,$BE,$3E
190 DATA$1B,$CD,$F0,$14,$3E,$40,$CD,$F0,$14,$C9,$1B,$40
200 DATA$1B,$33,$18,$0D
210 DATA$32,$D,$1B,$2A,$4,$80,$2,$D,$DB,$E0,$3E,$8,$DD,$46,$0,$CB
220 DATA$18,$F5,$CB,$16,$F1,$CB,$16,$23,$0,$3D,$20,$F3,$DB,$E1,$C9,$F5
230 DATA$1A,$13,$FE,$D,$CA,$51,$57,$CD,$F0,$14,$C3,$44,$57,$F1,$F3,$C9
240 DATA$C5,$F5,$11,$26,$57,$CD,$43,$57,$1,$80,$2,$11,$D0,$27,$1A,$CD
250 DATA$F0,$14,$13,$B,$78,$B1,$20,$F6,$3E,$D,$CD,$F0,$14,$F1,$C1,$F3
260 DATA$C9,$1,$C0,$3,$11,$D0,$27,$AF,$12,$13,$B,$78,$B1,$C2,$7B,$57
270 DATA $C9
280 POKE$5D53,$A4,$56
290 NEW
```

Das AUTO RUN von Hans Peter Aul.Gekürtzt von Hans-Jürgen Hillgeman.
Das Programm druckt eine Copy des Bildsschirmes aus.Hardcopy ist somit im 800 CMT und QD Basic fest eingebaut.Das Bild oben wurde mit dem Programm ausgedruckt Zeile 200 muß je nach Druckertyp geändert werden.Dort befinden sich die Steuerchen für den Zeilenvorschub.Hier für Panasonic KX-P1083

Textdateien ⇒ Wordstar-Format

Ubertragung von alten Textdateien auf Wordstar-Format

In Heft 5+6/88 hatte ich auf den Seiten 12-14 ein Verfahren vorgestellt, um alte Dateien, die z.B. unter Sharp-BASIC geschrieben wurden, in das Betriebssystem CP/M zu übertragen. Das Programm SHPASC.PAS konvertiert den Text dann anschließend in korrekten ASCII-Code; nun erst kann man mit der neuen Datei auf anderen Computern etwas anfangen.

Mit kleinen Änderungen kann man das Programm auch zu anderen Zwecken gut gebrauchen. Wer möchte nicht beispielsweise seine alten Textdateien nun auch unter CP/M oder gar MS-DOS weiterverwenden?

Bis der MZ-800 auf dem Markt erschien, hatten sich bei mir inzwischen weit über 10 MegaByte an Textdateien angesammelt, die ich eigentlich auch heute immer noch brauche, wenigstens auszugsweise. Sie sind überwiegend mit dem Textverarbeitungsprogramm CROWOOD-Text geschrieben, ein recht ausgereiftes Prográmmchen mit folgenden Nachteilen:

- 40 Zeichen-Bildschirm
- Speicherung nur auf Cassette, spätere Version auch auf Quick-Disk, jedoch nicht auf Diskette
- kein Betriebssystem im Hintergrund, daher kein Dialog mit anderen Programmen

Die Reihe der Nachteile läßt sich fortsetzen, etwa mit der fehlenden softwaremäßigen Tastaturentprellung usw., immerhin ist das Programm aber doch vergleichsweise leistungsfähig, denn es hat Blocksatz, Formatierungsbefehle, Suchfunktion und Ähnliches.

Eine Besonderheit ist der merkwürdige Zeichensatz. CROWOOD verwendet nämlich den Bildschirm-Zeichensatz, wie er im 700er-Handbuch wiedergegeben ist. Man kann sich nach Laden eines Textes davon überzeugen, indem man sich den Speicherinhalt oberhalb 3000 anschaut (das ist der Textpuffer): Alles Grafikzeichen!

Nun will ich die einzelnen Arbeitsschritte aufzeigen, die notwendig sind, um eine CROWOOD-Datei zu transferieren und zu konvertieren. Es hört sich komplizierer an als es ist. Wenn man alles genau befolgt, geht es schneller als eine neue Texteingabe.

1. Übertragung der Cassettendatei in das Betriebssystem CP/M

Das bereits vorgestellte Programm 'CMT.COM' ermöglicht es uns auch hier wieder, eine Cassettendatei in das CP/M und auf die Diskette zu übertragen. Ich habe es mir für DM 30.-- bei Honisch-Datentechnik, Wartburgstr. 1, 4100 Duisburg 25 besorgt.

CMT.COM lädt also auch "Fremddateien", und dabei spielt die Ladeadresse keine Rolle, was ja sonst immer zu erheblichen Problemen führt wegen der besonderen Lage der TPA unter CP/M im Gegensatz zur "normalen" Speicheraufteilung nach Einschalten des Computers.

Nun gibt es allerdings ein Ärgernis: CMT.COM interpretiert alle groaen "Z" als Dateiende (EOF oder ^Z) und bricht die Übertragung ab, sobald dieses Zeichen eingelesen wird. Das bedeutet, daß man das Programm CMT.COM ändern müßte. Das ist schwierig, und ich habe es anders gemacht: CROWOOD verfügt wie schon gesagt über eine Suchfunktion. Damit kann man die großen "Z" herausfiltern und ersetzen. Am besten durch große "X" denn glücklicherweise kommt der Buchstabe nur selten vor, so daß es nicht allzu viel Mühe macht. WORDSTAR hat mit dem umgekehrten Vorgang, wie wir noch sehen werden, gar keine Probleme.

Die derart vorbereitete CROWOOD-Datei wird also mit CMT.COM auf eine Diskette transferiert, auf der möglichst nur Turbo-PASCAL drauf ist. Mit den ganzen Zwischendateien, die noch entstehen werden, kann man den Platz gut gebrauchen.

2. Konvertierung der Datei in eine ASCII-Datei

Dazu laden wir Turbo und anschließend CROASC.PAS. Das Programm enthält nur die notwendigsten Routinen. Die Schleifengröße in Zeile 3 gibt man am besten per Hand ein: sie ist identisch mit der Dateigröße der zu übersetzenden Datei, da byteweise übersetzt wird. Die Größe läßt sich leicht errechnen aus der Differenz, die CROWOOD bei freiem und bei belegtem Speicher im Hauptmenü anzeigt (h-Taste). Die Schleifenkonstruktion mußte ich mit Real-Zahl konstruieren, weil der Compiler bei Integer nur bis MAXINT arbeitet, und meine Dateien haben fast alle mehr als 37 kByte. Wer kleinere Dateien konvertieren will, kann dies mit FOR TO DO machen, es geht selbstverständlich schneller.

Noch ein Tip zu den Laufwerken. Es ist zweckmäßig, der Zieldatei das Laufwerksattribut E: voranzustellen, die neue Datei also in die RAM-Floppy zu schieben, damit die Schreib-Leseköpfe der Floppy-Laufwerke bei zwei geöffneten Dateien nicht immer zwischen den Spuren hin- und herspringen müssen. Anschließend kann man die neue Datei mit PIP von E nach A oder B übertragen. Obwohl ein Arbeitsschritt mehr erforderlich ist als bei der Umsetzung Diskette/Diskette ist die Methode Diskette/RAM-Floppy/Diskette schneller und schont die Laufwerke. Hat man CMT.COM vorher

Textdateien ⇒ Wordstar-Format

schon in die RAM-Floppy geladen und Laufwerk E eingestellt, kann man auch noch diesen Zwischenschritt einsparen. Der Transfer erfolgt dann direkt von der Cassette auf die RAM-Floppy. Als weiterer positiver Nebeneffekt stellt sich ein, daß das Laufwerk während der Cassettenübertragung nicht ständig mitläuft - eine Unzulänglichkeit von CMT.COM. Aber es läuft wie gesagt auch ohne RAM-Floppy!

3. weitere Bearbeitung der ASCII-Datei

Jetzt kann die Datei mit Option D (Document-Modus) bearbeitet werden. Bei mir hat sich die nachfolgend beschriebene Reihenfolge bewährt.

Schaut man sich die Datei an, stellt man fest, daß am rechten Rand das "<" - Zeichen steht, das ist die harte Worttrennung. Will man das nicht haben, ruft man die Austauschfunktion ^Q^A auf. Entfernt werden soll ^N (dies Zeichen so eingeben), also muß es durch "Nichts" ersetzt werden, und das ist SPACE (also SPACE-Taste einmal drücken, dann CR). Als Option geben wir dann noch ein "NG999". "N" bedeutet Austausch, ohne daß WORDSTAR nachfragt, "G" im gesamten Text und 999 die Anzahl der Wiederholungen (hier also nur eine genügend große Zahl nehmen).

Auf diese Weise zieht WORDSTAR nun die gesamte Datei zu einer zusammenhängenden Zeile zusammen. Das kann bei großen Dateien sehr lange dauern, so daß es von Vorteil sein kann, wenn man schon bei Beginn die Gesamtdatei in einzelne Blöcke aufteilt in TEIL1.TXT TEIL2.TXT usw. Das geht sehr schnell mit den Blockmarkierungen ^K^B und ^K^K und anschließendem ^K^W. Nach der Bearbeitung fügt man alles wieder zusammen mit dem Blocklesebefehl ^K^R.

Die aus einer einzigen Zeile bestehende Datei sollte man nun erst einmal vorformatieren, damit sie überschaubar wird. Das geht, indem man zunchst mit ^O^H den Vorschlag für Silbentrennung ausschaltet (^O^H ist ein sog. TOGGLE, d.h. mit demselben Befehl schaltet man auch später wieder ein). In der Regel ist WORDSTAR so installiert, daß der automatische Trennvorschlag eingeschaltet ist. Danach ^Q^Q^B eingeben und die ganze Datei wird auf die voreingestellte Zeilenbreite formatiert.

Jetzt können die großen "X" wieder zurückverwandelt werden (^Q^A "X" CR "Z" CR NG999) und WORDSTAR kommt in Wallung. Es ist schon eine Freude, zuzuschauen. Anschließend die CROWOOD-Steuerzeichen entfernen, z.B. LL oder ähnliche. Auch das macht ^Q^A, ich will nicht alles wiederholen.

Bei Such- und Austauschfunktionen (^Q^F / ^Q^A) ist es zweckmäßig, bei Erreichen des Dateiendes nicht immer wieder an den Anfang zu gehen. Das kostet Zeit, und WORDSTAR legt bei langen Texten immer eine Zwischendatei an. Einfacher ist es, die Option "B" einzugeben, und die Suche geht von hinten nach vorn.

Zum Schluß schalten wir mit ^O^H die Silbentrennung wieder ein, formatieren durch, und die Sache ist fertig.

Dirk Grube, Ulmenstr. 31, 2320 Plön
☎ 04522 / 8714

```
PROGRAM croasc;

(*  Dirk Grube
   Ulmenstr.31
   2320 Ploen

   04522 / 8714  *)

CONST zaehler = 3.1E+03;  (*
Dateigroesse der zu ueber-
                tragenden Datei,
kleiner
                      Zuschlag
schadet nicht *)

VAR source, ziel : text;
  sname, zname :
STRING[40];
  ch : char;
  i : real;           (*
siehe Text *)

PROCEDURE p(c:byte);

BEGIN
   write(ziel,chr(c));
END;

PROCEDURE kodiereum (VAR
ch:char);
VAR z:byte;

BEGIN                 (* ko-
diereum *)

   z:=ord(ch);
    IF (z>0) AND (z<27)
THEN p(z+64);
    IF (z>128) AND (z<155)
THEN p(z-32);
    IF (z>31) AND (z<42)
THEN p(z+16);
    IF (z>96) AND (z<106)
THEN p(z+64) ELSE BEGIN

    CASE z OF
       $F0 : BEGIN
p($0D);p($0A); END;
       $AA : p($7E);
       $AB : p($7D);
       $AC : p($7C);
       $AD : p($5D);
       $AE : p($5B);
       $AF : p($5C);
       $00 : p($20);
       $2A : p($2D);
       $2B : p($3D);
       $2C : p($3B);
       $2D : p($2F);
       $2E : p($2E);
       $2F : p($2C);
       $49 : p($3F);
       $4F : p($3A);
       $57 : p($3E);
       $51 : p($3C);
       $65 : p($25);
       $67 : p($29);
```

Textdateien ⇒ Wordstar-Format

```
      $68 : p($28);
      $6A : p($2B);
      $6B : p($2A);
      $9B : p($7B)

     END;
   END;
END;

BEGIN                  (* main
*)
write('Name des Quellpro-
gramms: '); readln(sname);
assign(source,sname);
reset(source);
write('Name der Zieldatei:
'); readln(zname);
assign(ziel,zname);
rewrite(ziel);

i:=1;

REPEAT
   read(source,ch);
   kodiereum(ch);
   i:=i+1;

UNTIL i>zaehler;

write (ziel,EOF);
close(ziel);

END.


PROGRAM bbgasc.pas;

(*  Dirk Grube
   Ulmenstr. 31
   2320 Ploen
          •
   04522 / 8714  *)


CONST zaehler = 3.0E+04;

VAR source, ziel : text;
  sname, zname :
STRING[40];
  ch : char;
  i : real;

PROCEDURE p(c:byte);
BEGIN
   write(ziel,chr(c));
END;


PROCEDURE kodiereum (VAR
ch:char);
VAR z:byte;

BEGIN
   z:=ord(ch);
    IF (z>31) AND (z<94)
THEN p(z);

     BEGIN
      CASE z OF
       $0D : BEGIN
p($0D);p($0A); END;
       $AA : p($66);
       $AB : p($76);
       $AC : p($2C);
       $AF : p($6A);
       $00 : p($32);
       $92 : p($65);
       $93 : p($60);
       $94 : p($FE);
       $96 : p($74);
       $97 : p($67);
       $98 : p($68);
       $9A : p($62);
       $9B : p($78);
       $9C : p($64);
       $9D : p($72);
       $9E : p($70);
       $9F : p($63);
       $A0 : p($71);
       $A1 : p($61);
       $A2 : p($7A);
       $A3 : p($77);
       $A4 : p($73);
       $A5 : p($75);
       $A6 : p($69);
       $A7 : p($60);
       $A8 : p($5D);
       $A9 : p($6B);
       $AA : p($66);
       $AB : p($76);
       $AC : p($2C);
       $AF : p($6A);
       $B0 : p($6E);
       $B2 : p($DC);
       $BB : p($7B);
       $BC : p($3C);
       $BD : p($79);
       $B8 : p($6C);
       $B9 : p($DB);
       $B3 : p($6D);
       $B7 : p($6F);
       $BA : p($FC);
       $AD : p($FD);
       $AE : p($FE)

     END;
    END;
END;

BEGIN
write('Name des Quellpro-
gramms: '); readln(sname);
assign(source,sname);
reset(source);
write('Name der Zieldatei:
'); readln(zname);
assign(ziel,zname);
rewrite(ziel);

i:=1;

REPEAT
   read(source,ch);
   kodiereum(ch);
   i:=i+1;

UNTIL i>zaehler;

write (ziel,EOF);
close(ziel);
END.
```

HiSoft-PASCAL ⇒ Turbo-PASCAL

Von HiSoft-PASCAL nach Turbo-PASCAL

Das im Heft 5+6/88 auf den Seiten 12-14 von mir beschriebene Verfahren kann man auch dazu benutzen, um HiSoft-PASCAL-Programme in Turbo-PASCAL-Programme zu verwandeln.

Zwei wesentliche Voraussetzungen waren erforderlich, die hier noch einmal wiederholt werden:

1. Sharp-BASIC bietet die Möglichkeit, mit der Option SAVE "Progname",A ein Programm in ASCII-Format und nicht im komprimierten Sharp-Format abzuspeichern. Letzteres zeichnet sich dadurch aus, daß zum einen das Aufzeichnungsformat zu anderen inkompatibel ist, als auch die Befehle in Token-Form abgelegt sind. Das nennt sich dann etwas großspurig "Halbcompiler".

Leider umfaßt ASCII bei Sharp nur Großbuchstaben und die wichtigsten Zeichen. Doch damit ist beim Transfer schon das meiste gewonnen. Die Lauffähigkeit ist sicherlich gegeben, der reine Text kann nachgebessert werden.

2. Die andere Voraussetzung war ein Programm, das aus dem CP/M heraus die sonst nicht unterstützte Kassette ansprechen zu können. 'CMT.COM' tut dies.

Als weitere Programmiersprache für den MZ-700 und MZ-800 brachte die Firma HiSoft eine Implementierung von PASCAL heraus, die in der letzten Version leider auch nur die Kassette, maximal die Quickdisk unterstützte.

Obwohl HiSoft-PASCAL in vielen Routinen wesentlich schneller läuft als Turbo-PASCAL, ist es doch mit etlichen Unzulänglichkeiten behaftet und entspricht nicht dem Umfang des Wirth'schen Standards.

Irgendwann einmal fliegen schließlich doch die Kassetten vom Tisch. Wenn man aber die alten Programme noch ins CP/M hinüberretten kann, dann fällt der Schritt zur Diskette erheblich leichter. Und darum geht es in diesem Beitrag.

Die von mir für BASIC vorgeschlagene Methode versagt bei HiSoft-PASCAL leider, weil die Möglichkeit des Abspeicherns in ASCII nicht gegeben ist. Alle Programme und Dateien werden in einem für CP/M nicht lesbaren Zeichenformat abgespeichert.

Es gibt aber dennoch im HiSoft-PASCAL eine Routine, die ASCII-Code erzeugt, und das ist die Druckroutine, weil der Drucker eben ASCII verlangt. Dies brachte meinen Clubkameraden Bruno Volkmer auf die Idee, die Druckerausgabe 'umzubiegen' und für unsere Zwecke weiterzuverwenden. Und das geht so:

1. HiSoft-PASCAL-Programme auf Kassette bereitstellen, so daß sie nach Möglichkeit nicht mehr als 4kByte lang sind. Warum wird später deutlich werden. Notfalls ein wenig zerlegen. Das geht einfach mit dem P-Befehl und den gewünschten Zeilennummern als Option.

2. HiSoft-PASCAL laden und mit 'B' und zweimal 'CR' in den Monitor zurückspringen.

3. Mit dem M-Befehl auf die Adresse 13FB einen Sprung nach 1400 schreiben. Der Z-80-Befehl lautet C3 00 14 .

4. Auf die Adresse 140C ist nun ein Call nach 1002 zu schreiben, (wieder mit dem M-Befehl): CD 02 10 F1 C9 (POP AF, RET) .

5. Nun wird ab 1002 folgendes eingetragen: E5 2A 00 10 77 23 FE 0D 20 03 36 0A 23 22 00 10 E1 C9 .

6. Mit 'J 1221' HiSoft wieder warm starten.

7. Programm laden, nicht starten, nicht drucken.

8. Programm compilieren und die Anfangsadresse aufschreiben. Nehmen wir an, es sei 7451. Wegen der besseren Rechenbarkeit erhöhen wir nun die Adresse auf den nächsten Tausen der, also 8000 Hex.

9. Mit B,CR,CR wieder in den Monitor springen und 8000 auf Adresse 1000 eingeben: M 1000 00 1001 80 (im Z-80 Code sind die beiden Bytes bekanntlich vertauscht.)

10. J 1221 (Warmstart)

11. L und gleich danach ohne CR ^P (CONTROL-P) eingeben. Es tut sich scheinbar nichts, aber nach ca. 20s Kunstpause ist das Wunder vollbracht: Der ASCII-Code wurde erzeugt und steht gewissermaaen als zweite, modifizierte Datei im RAM. Daher muß beachtet werden, daß sie auch wirklich hineinpaßt. Also Programmlänge beachten!

12. Die Endadresse steht jetzt auf 1000 und kann mit D 1000 angeschaut werden. Als Beispiel 71 85, das würde dann 8571 bedeuten.

13. Am einfachsten ist es nun, sich den neu erzeugten Programmtext im Speicher anzusehen. Mit D8000FFFF läuft der Speicherinhalt durch, das erste Wort muß 'PROGRAM' sein, damit beginnt jedes PASCAL-Programm. Analog steht weiter hinten irgendwo 'END.' (END mit Punkt). Die Speicheradresse, an der der Punkt steht aufschreiben.

14. Mit dem S-Befehl den Speicherinhalt aufzeichnen: Nach Eingabe von S,CR wird nach dem

Programmnamen gefragt. Hier muß irgendetwas eingetragen werden, zweckmäßigerweise in Großbuchstaben. Nach CR wird die Anfangsadresse abgefragt; das war 8000, danach die von uns ermittelte Endadresse (die mit dem Punkt von 'END.') Die Startadresse ist unerheblich, geben Sie 8000 ein.

15. Nun folgt das Einlesen in CP/M ! CP/M laden, Programm 'CMT.COM' laden. Mit 'CMT L' wird das Kassettenprogramm eingelesen und auf die Diskette übertragen. Als Programmnamen hatten Sie beispielsweise 'MEINPROGRAMM' gewählt. Dann werden Sie es unter dem neuen Namen 'MEINPROG.OBJ' auf der Diskette wiederfinden. 8 Zeichen verwendet CP/M, 'OBJ' wird durch das Programm 'CMT.COM' selbständig angehängt.

Der Rest ist schnell getan und noch einfacher, als ein transferiertes BASIC-Programm lauffähig zu machen. Mit einem Editor, am besten gleich dem Turbo-Editor werden die Zeilen-Nummern gelöscht. Es kann sein, daß das Programm dann schon läuft, weil die Syntax in PASCAL sehr streng ist und HiSoft schon fast kompatibel ist. Allenfalls muß noch ein wenig nachgearbeitet werden, aber dann hat sich die Mühe schließlich gelohnt. Wer die 30.- DM für das Programm CMT.COM bei Honisch-Datentechnik, Duisburg, Wartburgstraße 1 nicht ausgeben will, der sollte folgendes versuchen: Nach Punkt 13 eine Speicherverschiebung vornehmen, so daß unser Text anschließend in der TPA des CP/M steht, so daß mit dem CP/M-SAVE-Befehl unter Angabe der Sektorenzahl die Datei auf die Diskette gezogen werden kann. Das müßte funktionieren. Ich selbst habe es noch nicht ausprobiert, weil der oben beschriebene Weg einfacher ist. ■

Übertragung von CP/M-Textdateien auf MS-DOS-Format

Als Anwender auf den Systemen MZ-700, MZ-800, MZ-800 unter CP/M und MS-DOS (286er AT) gleichzeitig stehe ich häufig vor der Aufgabe, Dateien auch mal auf das jeweils andere System übertragen zu müssen.

Eine immer wiederkehrende Angelegenheit ist der Transfer von Texten. Auf Probleme bei der Verwendung unterschiedlicher Textsysteme hat die Redaktion der MZ-Zeitung auf Seite 4 des Heftes 5+6/88 bereits hingewiesen und auch Lösungsmöglichkeiten aufgezeigt.

Es ist für die Bedienbarkeit sehr nützlich, daß viele CP/M-Programme auch für MS-DOS verfügbar sind. So nutze ich selbst auch noch viel den 800er, obwohl doch eigentlich vom IBM-kompatiblen AT verwöhnt. Aber man braucht sich erstens nicht umzugewöhnen, wenn man morgens auf diesem, abends auf jenem Rechner arbeiten muß, etwa Texte schreiben. Und die Textverarbeitung läuft auf dem AT bis auf einige Routinen nicht entscheidend schneller.

Nehmen wir als Beispiel den guten alten WORDSTAR, so hat er sich zwar unter MS-DOS inzwischen mit den Versionen 4 oder 2000 sehr gemausert, aber die alten CP/M-Texte laufen auch dort mit einer kleinen Einschränkung. MS-DOS verläßt bei den deutschen Umlauten den ANSI-Standard, bzw. ASCII.

Zeichen	ASCII	IBM
$	64	21
Ä	91	142
Ö	92	153
Ü	93	154
ä	123	132
ö	124	148
ü	125	129
ß	126	225

Ist eine Datei mithilfe von MSCOPY.COM oder eines anderen Verfahrens auf eine MS-DOS-Diskette übertragen worden, kann man sich den Text sofort auf dem anderen Rechner ansehen. Nur die Umlaute stimmen noch nicht.

Mit dem WORDSTAR-Editor läßt sich das leicht nachbessern mit der Austauschfunktion ^Q^A. WORDSTAR fragt, welches Zeichen ausgetauscht werden soll. Das IBM-Sonderzeichen, das auf unserem MZ-800 ein bildschönes 'ä' war, ist jetzt zu einer geschweiften Klammer geworden. Auf der deutschen Tastatur suchen wir das Zeichen natürlich vergeblich. Aber es läßt sich doch eintippen, indem man die ALT-Taste gedrückt hält und danach auf dem Nummernfeld rechts den entsprechenden Dezimalwert des 'ä' eingibt, in unserem Falle 123 (siehe Tabelle). Nach RETURN wird gefragt, wogegen getauscht werden soll. Die Antwort ist das normale IBM-Tastatur-'ä'. Als Option wählen wir NG, worauf ohne Nachfrage alle entsprechenden Sonderzeichen im Text durch das 'ä' ersetzt werden.

Befinden wir uns nun am Ende des Textes, fahren wir mit dem Backslash analog fort und geben als Option allerdings NGB ein (B für 'back') und die Sache läuft rückwrts ab, ohne daß man nochmal zwischendurch an den Textanfang gehen müßte. Nach dem 7.Schritt, dem 'ß' ist die Arbeit relativ schnell beendet.

Da der Trick mit der ALT-Taste unter CP/M leider nicht funktioniert, muß man beim Transfer MS-DOS nach CP/M die Änderung nach Möglichkeit schon auf dem MS-DOS-Rechner vornehmen, wenn man vermeiden will, jeden einzelnen Umlaut manuell nachbessern zu müssen.

Wem dieses manuelle Verfahren mißfällt, dem empfehle ich, mein PASCAL-Programm 'SHPASC.PAS'

Textdateien CP/M ⇒ MS-DOS

auf Seite 14 des Heftes 5+6/88 abzuwandeln: Die Symboltabelle muß mit den oben angegebenen Werten geändert werden.

Wer einen etwas älteren Drucker sein eigen nennt, wird möglicherweise feststellen, daß dieser die deutschen Sonderzeichen unter WORDSTAR oder anderen Programmen klaglos ausdruckt, mit dem DOS-Befehl 'TYPE' jedoch wieder die unerwünschten Sonderzeichen. Das kann daran liegen, daß die DIP-Schalter zum Anwählen des IBM-Zeichensatzes falsch stehen oder aber der Drucker als reiner ASCII-Drucker - wie früher üblich - den erweiterten IBM-Zeichensatz gar nicht beherrscht.

Dem Übel braucht man nicht, wie mir mal vorgeschlagen wurde durch Neukauf abzuhelfen - ein Treiberprogramm tut's auch, wenn es nur um die Umlaute geht. Es würde den Rahmen und den Sinn dieser Zeitung sprengen, ihn hier abzudrucken. Wer Probleme hat, darf sich an mich wenden.

Dirk Grube, Ulmenstraße 31, 2320 Plön, ☎ 04522 / 8714

```
PROGRAM shpasc;

(*  Dirk Grube
    Ulmenstr. 31
    2320 Ploen

    04522 / 8714


  Das Programm setzt den speziellen Sharp-
    Zeichensatz um auf korrekten ASCII-Code *)


VAR source, ziel : text;
  sname, zname : STRING[40];
  ch : char;
  i : real;
  a,zaehler : integer;

PROCEDURE p(c:byte);
BEGIN
   write(ziel,chr(c));
END;

PROCEDURE   kodiereum   (VAR
ch:char);
VAR z:byte;

BEGIN
   z:=ord(ch);
    IF (z>=$30) AND (z<=$5D)
THEN p(z);

     BEGIN
      CASE z OF
                $0D   :   BEGIN
p($0D);p($0A); END;
       $20 : p($20);
       $21 : p($21);
       $22 : p($22);
       $23 : p($23);
       $24 : p($24);
       $25 : p($25);
       $27 : p($27);
       $28 : p($28);
       $29 : p($29);
       $2A : p($2A);
       $2C : p($2C);
       $2B : p($76);
       $1A : p($62);
       $22 : p($7a);
       $3d : p($79);
       $9b : p($78);
       $23 : p($77);
       $Ab : p($76);
       $25 : p($75);
       $96 : p($74);
       $24 : p($73);
       $9d : p($72);
       $20 : p($71);
       $9e : p($70);
       $37 : p($6f);
       $b0 : p($6e);
       $33 : p($6d);
       $38 : p($6c);
       $22 : p($6b);
       $2f : p($6a);
       $26 : p($69);
       $98 : p($68);
       $97 : p($67);
       $2a : p($66);
       $12 : p($65);
       $9c : p($64);
    $9f : p($63);
    $9a : p($62);
    $21 : p($61);
    $93 : p($60)

     END;
    END;
END;

BEGIN

write('Laenge des Quellprogramms in Byte: ');
readln(zaehler);
write('Name des Quellprogramms: '); readln(sname);
assign(source,sname);
reset(source);
write('Name der Zieldatei: '); readln(zname);
assign(ziel,zname);
rewrite(ziel);

for a:=1 to zaehler do begin;

   read(source,ch);
   kodiereum(ch);

end; (* do *)

write (ziel,EOF);
close(ziel);
END.
```

Meßdatenerfassung mit MZ-700/MZ-800

Die Preise für Mikrocomputersysteme fallen zur Zeit rapide. Der MZ-800 kostete als Auslaufmodell zuletzt DM 198.--. Den MZ-700 dürfte man zu vernünftigen Preisen kaum noch an den Mann bringen können. Bevor man ihn also mit Wertverlust verkauft, sollte man prüfen, ob man ihn nicht für bestimmte Zwecke noch verwenden kann.

Eine gute Anwendungsmöglichkeit ist die Meßdatenerfassung, die merkwürdigerweise mit den Modellen MZ-7xx / MZ-8xx nicht sehr verbreitet ist, offensichtlich wegen der fehlenden seriellen Schnittstelle, denn nur mit dieser kann man im Prinzip Daten einlesen. Auf recht einfache Weise läßt sich jedoch auch ohne diese eine Meßdatenerfassung realisieren, und zwar über den I/O-Port. Doch zunächst eine kurze Betrachtung, warum überhaupt Meßdatenerfassung mit dem Computer?

Die Vorteile:

- Schnell ablaufende Vorgänge lassen sich wie mit einem Speicheroszilloskop erfassen und später gedehnt und beliebig oft wiedergeben
- Für die verschiedensten physikalischen, technischen und sonstigen Größen gibt es heute schon preiswerte Sensoren, die fast alle den Meßwert in eine Spannung umsetzen, so daß sich das Problem auf eine Spannungsmessung reduziert. Oft ist die Spannung aber dem tatsächlichen Wert nicht proportional. Die Nicht-Linearität läßt sich gerade aber mit dem Computer softwaremäßig viel besser korrigieren und meistens auch billiger als mit aufwendigen elektronischen Korrekturschaltungen.
- Die Meßbereiche lassen sich flexibler variieren.
- Es lassen sich mehrere Meßdaten – im einfachsten Falle bis zu 8 – fast zeitgleich erfassen. Dabei kann die Zeit als Systemvariable zusätzlich mitprotokolliert werden.

Der Nachteil:

- Der Meßaufbau ist etwas klobiger und netzabhängig. Letzteres läßt sich jedoch bei Bedarf auch noch beseitigen.

Der Aufbau:

Auf dem Markt sind zahlreiche Schaltungen und Bausätze vorhanden, die das tun, was wir wollen, nämlich analoge Daten (Spannungen) in digitale Signale (für den Computer lesbar) umwandeln, nämlich sog. A/D-Wandler.

Ich selbst habe den von 'Elektor' aus Heft 5/85 verwendet. Es gibt ihn bei verschiedenen Firmen (Inserate lesen) für etwa DM 70.- mit Platine und Steckdose, bei Selbstbau wird es billiger, die Teile sind leicht zu beschaffen.

Der Aufbau der Platine selbst ist so einfach, daß ich darauf nicht näher eingehe. Die Schaltung kann hier aus urheberrechtlichen Gründen nicht abgedruckt werden. Der Baustein hat schon fast alle Signalanschlüsse, die auch der Sharp-I/O-Port bereits zur Verfügung stellt. Ausnahme: Beim MZ-7xx die +5V-Versorgung, die man sich vom Joystick-Port holt. **Vorsicht**, *im Handbuch ist die Anschlußbelegung des Joy-Ports falsch herum abgedruckt, jedenfalls in meiner Ausgabe.*

An der Besonderheit des R/W-Signals braucht man sich nicht zu stören, WR wird direkt an R/W angeschlossen.

Der A/D-Baustein von Elektor ist eigentlich einer von Vieren, die in den im selben Heft beschriebenen Universellen I/O-Port eingesteckt werden soll. Daher gibt es noch das Signal SS (SLOT SELECT). Über einen IC (z.B. 7425), den wir freifliegend einschleifen, gehen wir an den Computer. Man kann sich dazu eine Mini-"Platine" aus einer Lochrasterplatine herstellen wie unten abgebildet, dann lassen sich die IC-Beinchen besser mit den Kabeln verbinden.

Wie bei allen Parallel-Datenleitungen muß darauf geachtet werden, daß sie nicht zu lang werden. Das ganze gehört in ein vernünftiges Gehäuse. Am besten ist es, wenn man für die Meßeingänge Klinkenbuchsen (3,5 mm) mit Schaltkontakt verwendet, dann liegen nämlich automatisch die eventuell nicht verwendeten Eingänge auf Masse, was immer empfehlenswert ist.

Der Abgleich ist einfach, am besten mit +5V, und ist im Elektor-Heft genauer beschrieben. Dann liegt am Port der Wert 255 oder FFH an, den das Programm dann umrechnen muß. Größere Spannungen müssen über Vorteiler herangeführt werden.

Ein Meßprogramm ist ganz einfach geschrieben, z.B. in BASIC:

```
10 CLS
20 REM Prüfen der Eingänge
(MZ-700)
30 GET X$
40 REM Die Eingänge werden
durch die entsprechende Taste
 des Computers angewählt
50 IF X$ = "" GOTO 50
60 X = ASC(X$)-49
70 OUT#16,X
80 INP#16,A
90 PRINT A 100 GOTO 30
```

Beim 800er heißen die Befehle etwas anders, das PASCAL-Programm dürfte auch keine Schwierigkeiten bereiten. Wenn – wie bereits angesprochen – der verwendete Sensor keine lineare Kurve liefert, muß softwaremäßig nach-

geholfen werden. Je nachdem, wie krumm die Kurve ist, gibt es dazu verschiedene Möglichkeiten:

Im schlimmsten Falle muß eine Wertetabelle von Hand eingegeben werden, von der sich der Computer den entsprechenden zweiten Wert als korrigierten Wert holt (DATA-Zahlenpaare und READ- Anweisung).

In minder schweren Fällen wird es genügen, den Bereich aufzuteilen und entsprechende Korrekturwerte zu addieren, zu multiplizieren – dazu muß man sich die Kurve im Einzelfall anschauen.

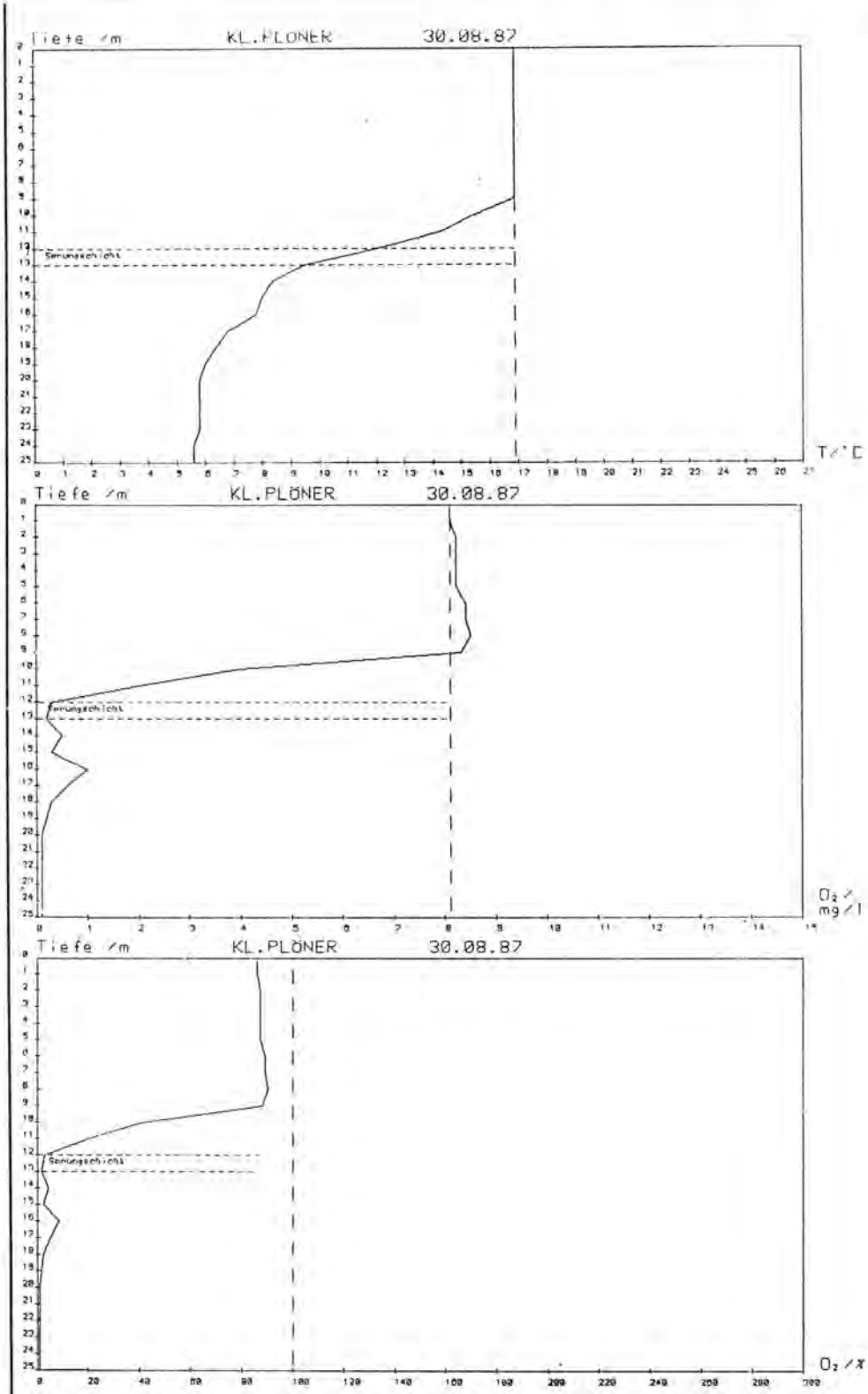
Am einfachsten geht es, wenn man in der Abweichung einen formelmäßigen Zusammenhang erkennen kann, den man in Form einer Umrechnungsfunktion eingeben kann.

In den meisten Fällen wird das bei den heute gebräuchlichen Sensoren aber gar nicht mehr nötig sein, weil sie schon eine interne Kompensation und Offset-Korrektur eingebaut haben.

Die ganze Angelegenheit hört sich viel komplizierter an, als sie ist, ein paar Stunden sorgfältige Lötarbeit - und der Computer hat ein sinnvolles neues Arbeitsgebiet.

Wer Schwierigkeiten hat, darf sich gern an mich wenden.

Dirk Grube, Ulmenstr. 31, 2320 Plön
☎04522/8714



Erklärungen zu den Meßdaten finden Sie auf Seite 24.

Beispiele für Meßdatendarstellung

Aufnahme erfolgte wie beschrieben mit dem MZ-700.

1. (Seite 23) Temperatur- und Sauerstoffverteilung im Kleinen Plöner See am 30.08.87

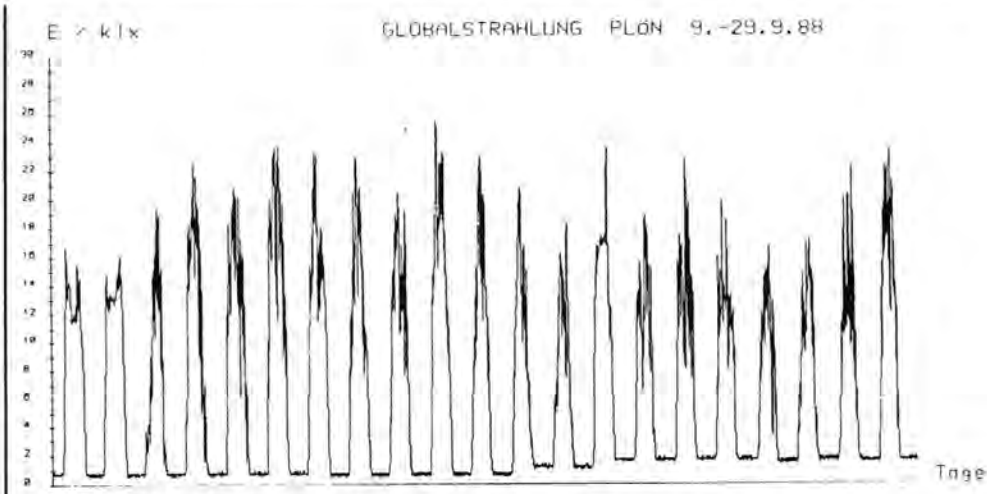
   - Die drei Diagramme zeigen auf der Y-Achse die Wassertiefe an, und zwar wie in der Natur von der Wasseroberfläche (0) bis 25 m.

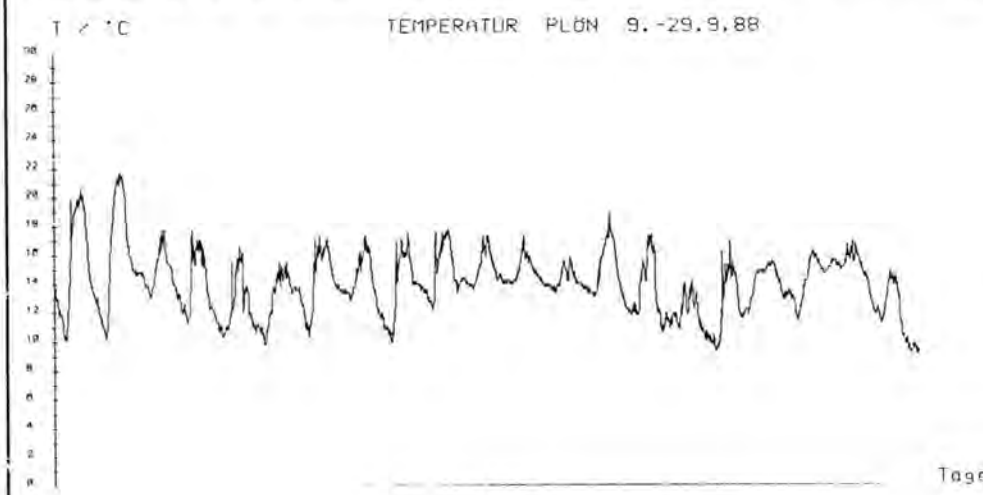
   - Das erste Diagramm zeigt die vertikale Temperaturverteilung mit einer Oberflächentemperatur von 16,9° C bis in 10 m Tiefe, nachfolgender Sprungschicht (Thermokline) und 5,6° C im Sediment.

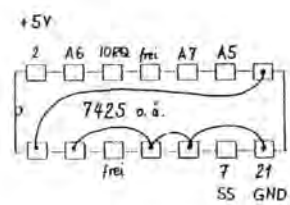
   - Darunter Sauerstoffverteilung in mg/l und in %. Deutlich sichtbar das Sauerstoffdefizit unterhalb der Sprungschicht durch bakterielle Zehrung, verbunden mit Nicht-Mischung der beiden Wasserkörper durch zu großen Temperaturunterschied.

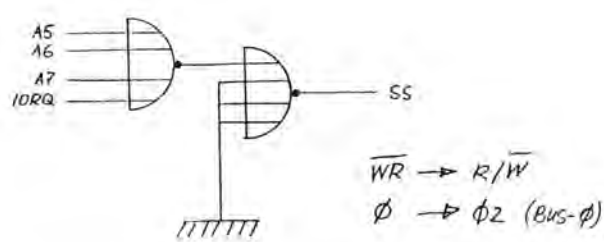
2. - Globalstrahlung (direkte Sonnenstrahlung + diffuse Himmelsstrahlung) an der Wasseroberfläche als Hauptparameter für das Pflanzenwachstum. Deutlich erkennbar die Tages-Peaks über einen Zeitraum von 21 Tagen.

   - Darunter Zeitgleich der Temperaturverlauf (Luft in 1 m über Wasseroberfläche (Uferbereich).









Eingangs-Signalanpassung

Tastaturabfrage beim MZ-700/800

Die Tastaturabfrage beim MZ-700 und 800 und das Einbinden dieser ML-Routine in eine Basiczeile

Für mich war die Tastaturabfrage in Maschinensprache lange Zeit ein „Buch mit 7 Siegeln", vor allem deswegen, da sich das 800er Handbuch völlig darüber ausschweigt. Im 700er Handbuch ist wenigstens die Tastaturmatrix (Seite 142) abgedruckt, eine genaue Erklärung fehlt allerdings auch dort. Ebenso rätselhaft war es mir, wie es möglich ist, zu prüfen ob zwei Tasten gleichzeitig gedrückt sind, z.B. ↑ und → für diagonal (mit GET aus Basic nicht zu realisieren). Probleme gibt es auch bei der Erzeugung des AUTOREPEATS (Taste kann bei GET gedrückt gehalten werden anstatt sie jedesmal neu zu drücken): entweder man kennt die für jedes Basic unterschiedlichen POKE-Adressen oder „nichts geht mehr". Übrigens sind mir folgende Adressen bekannt:

MZ-2Z046 (FD-800):
Autorepeat bei GET:
Dauer: `POKE $6879,0`
mit blinkendem Cursor:
`POKE $6879,1`
normal: `POKE $6879,255`

MZ-SZ008 (QD/FD-700):
Dauer: `POKE $4FFC,0`
mit blinkendem Cursor:
`POKE $4FFC,1`
normal: `POKE $4FFC,255`

MZ-1Z0138 (S-Basic):
Dauer: `POKE $59,240`

SOLO-Basic
normal: `POKE $59,83`

Die Adressen für folgende Basics sind mir leider nicht bekannt, falls sie jemand kennt, kann er sie ja veröffentlichen:
800: MZ-5Z009, MZ 1Z016, K&P-MZ-800 D-Basic
700: MZ-2Z009E, Gischel-Basic, K&P-MZ-700 D-Basic

In jedem Basic gibt es zwar schon Maschinenroutinen zur Tastaturabfrage, die vom User aufgerufen werden können, aber diese liegen bei jedem Basic an verschiedenen Adressen und sind mir zum größten Teil unbekannt. Dabei ist das Prinzip der Tastaturabfrage relativ einfach:

Der 700er und 800er benutzt zum Abtasten der Tastatur die programmierbare Schnittstelle (PIO) 8255 (über diese Schnittstelle wird u.a. auch der Kassettenrekorder und die Blinkfrequenz des Cursors gesteuert).

Im 700er Modus werden diesem Baustein die I/O-Adressen $E000 bis $E003, in dem 800er Modus die Portadressen $D0 bis $D3 zugeordnet. Zur Tastaturabfrage benötigen wir die Ports A und B (also $E000/$E001 bzw. $D0/$D1), wobei Port A als Ausgang und Port B als Eingang geschaltet ist. (Dies kann über den Control-Port $E003/$D3 geändert werden, was aber normalerweise nie gemacht wird).

Zum Scannen der Tastatur wird über Port A nacheinander jede Spalte (0-9) der Matrix angesprochen (mit LD ($E000), Spaltennummer bzw. OUT ($D0), Spaltennummer) und damit ein LOW-Signal gesendet. Nun wird überprüft, ob dieses LOW-Signal an den Anschlüssen 11 bis 18 der Tastaturmatrix angekommen ist und zwar über Port B (mit LD A,($E001) bzw. IN A,($D1).

Wenn ja, so läßt sich aus dem Kreuzungspunkt der Spalten- und Zeilenleitung die gedrückte Taste ermitteln. In diesem Programm wird nur ein bestimmter Bereich der Matrix gescannt und zwar Spalte 7 (Cursor) und Spalte 6 (SPACE):

`LD A,$07`	Akkumulator mit der Spaltennummer laden
`OUT ($D0),A`	Der Inhalt des Akkus wird an Port $D0 (700: LD ($E000),A) ausgegeben
`IN A ($D1)`	Der an den Leitung 11-18 anstehende Bit-Code wird in den Akku eingelesen (700: LD A,($E001))

Im Akku steht nun der Bit-Code und kann ausgewertet werden. Ist keine Taste gedrückt, so enthält der Akku $FF, also alle Bits sind HIGH.

11	12	13	14	15	16	17	18
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	1	1	1	1	1	1

Wurde z.B. ↑ gedrückt, so enthält der Akku $DF, also:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	1	1	1	1	1

Legt man dieses Bit-Muster (D7 bis D0) senkrecht an die Anschlüsse 11 bis 18 der Tastaturmatrix, so kann man festellenn, daß die ↑-Taste gedrückt wurde.

Sind z.B. ↑ und → gleichzeitig gedrückt, so enthält der Akku

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	1	0	1	1	1

So kann also geprüft werden, ob mehrere Tasten gedrückt wurden.

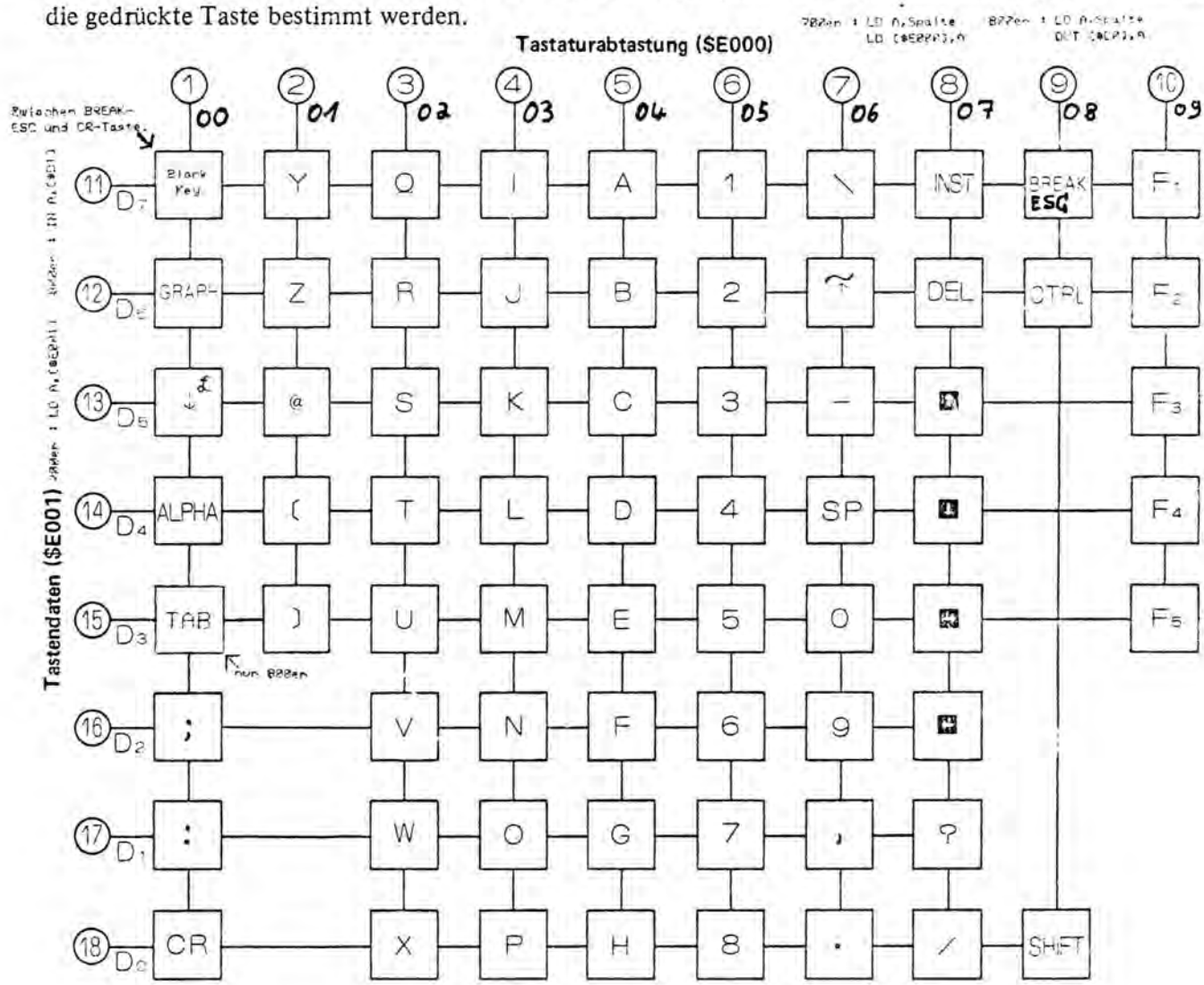
Um nun zu ermitteln welche Tastenkombination gedrückt wurde, wird in dem Programm eine Schleife durchlaufen, und der Inhalt des Akkumulators mit einer Tabelle verglichen. (Dies erledigt alles der Maschinenbefehl CPDR). Die Tabelle ist am Ende der Erklärung des Maschinenprogramms aufgelistet.

Zur Prüfung ob SPACE gedrückt ist, wird eine andere Methode angewendet: das für SPACE relevante Bit (Bit 4) wird durch 4 mal RLA in das Carry-Flag verschoben und dann geprüft ob das Carry-Flag gesetzt (1) ist oder nicht (0).

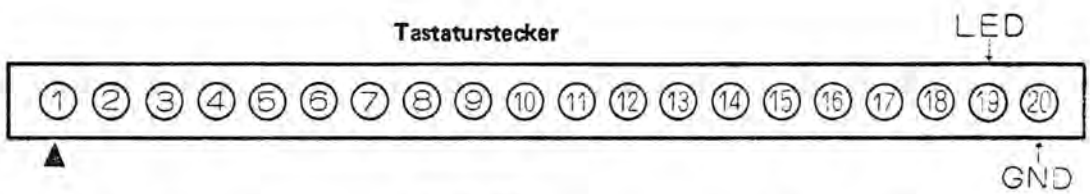
Tastaturabrage beim M2-700/800

[Tastaturmatrix]

Um feststellen zu können, welche Taste der Tastatur gedrückt wurde, erzeugt man über Port PA des Bausteins 8255 ein Signal und legt es an die unten dargestellte Tastaturmatrix an (Anschlüsse 1 bis 10). Ist nun eine Taste gedrückt, wird dieses Signal an die Anschlüsse 11 bis 18 weitergeleitet und wird dort von Port PB abgenommen. Durch Auswertung von angelegten und abgenommenen Signalen kann dann die gedrückte Taste bestimmt werden.



Ist keine Taste gedrückt so sind alle Bits im Akkumulator (D0-D7) HIGH gesetzt.
Der Akku (A) enthält also nach IN A,($E001) 255 (bzw $FF hex oder 11111111 bin).
Ist eine Taste gedrückt, so ist das entsprechende Bit der Zeile LOW (0).
Beispiel: Ist '3' gedrückt, wird der Akku mit $DF (223 dez) geladen.
Dies ist in Binär : Bit D7-D0
1101 1111
Bit 5 ist LOW, dies entspricht in der Tastaturmatrix der Taste '3'.



Tastaturabrage beim MZ-700/800

Die Startadresse des ML-Programms habe ich auf $C000 gelegt. Man kann das Programm natürlich an eine andere Adresse verschieben; dann müssen aber die unterstrichenen Adressen von den Labels EOTAB, CURPO und SPAPO entsprechend geändert werden. Aber Vorsicht beim Verschieben: der 800er nutzt den Bereich von $FE00 bis $FFFF als Arbeitsbereich und ab $F000 liegen oft Zeichensatzumkodierungstabellen (z.B. von CHAR DEF).

Beim 700er-Basic darf das Programm niemals zwischen $D000 und $FFFF geschrieben werden, da dieser Speicherbereich durch das OUT ($E3), A (beim 700-Basic unbedingt notwendig, um den Speicherbereich $D000 bis $FFFF auf V-RAM, Tastatur und Uhr zu schalten, siehe auch 700er-Handbuch (Seite 127)) „ausgebankt" wird. Der PC (Programmcounter) wird allerdings hardwaremäßig nicht mit verschoben und zeigt also wild in den Speicher hinein, was meistens einen Programmabsturz zur Folge hat. Der Urzustand wird wieder mit OUT ($E1),A hergestellt.

Bei Änderungen der Startadresse muß auch darauf geachtet werden, daß bei den Adressen der Labels immer zuerst der LOW-Anteil und dann der HIGH-Anteil der Adresse eingeben wird:

z. B.: `LD HL, $C03A`
(C0 = HI, 3A = LO)

wird hexadezimal wie folgt eingegeben: `21 3A C0`
(3A = LO, C0 = HI)

Nebenstehend das Assemblerlisting. Aufruf aus Basic: USR($C000)

Abfrage der Werte mit:
PEEK($C000+$3B) und
PEEK($C000+$3D)

```
SEITE    1
F ZLNR ADRS CODE       QUELLTEXT

                       ; Abfrage der CURSOR-Tasten und der SPACE-Taste
                       ; Ist eine der CURSOR-Tasten gedrueckt, so
                       ; wird die Adresse (Startadresse + 03BH) mit
                       ; dem Code (0-8) der gedrueckten Tasten,
                       ; entsprechend STICK(0) in Basic, geladen.
                       ; Ist SPACE gedrueckt, so steht in
                       ; Startadresse + 03DH eine 1 sonst 0
                       ; Um bei bei der Eingabe dieser Routine
                       ; die benoetigten Werte fuer die Label CURPO,SPAPO
                       ; und EOTAB zu ermitteln muessen zu der
                       ; von Ihnen gewaehlten Startadresse fuer diese Routine die
                       ; Adressen der Label addiert werden. Diese stehen am
                       ; Ende dieses Listings.
                       ; Wo sie diese Routine in Ihr Basic-Programm
                       ; einbauen, muessen sie selbst entscheiden.
                       ; Danach ermittelt sich dann auch die Startadresse.
                       ;
       C000                    ORG 0C000H
       C000 00                 NOP          ; D3 E3    OUT 0E3H,A
       C001 00                 NOP          ; beim 700er
       C002 E5                 PUSH HL
       C003 C5                 PUSH BC
       C004 3E00               LD   A,00H ; 0. Spalte
       C006 D3E0               OUT  (E0H),A ; 32 00 E0   LD (0E000H),A
       C008 00                 NOP               ; beim 700er
       C009 DBE1               IN   A,(E1H) ; 3A 01 E0   LD A,(0E001H)
       C00B 00                 NOP               ; beim 700er
       C00C 010300             LD   BC,0003H
       C00F 213AC0             LD   HL,EOTAB
       C012 EDB9               CPDR
       C014 ED433BC0           LD   (CURPO),BC
                               ; ab hier SPACE-Taste
       C018 3E09               LD   A,09H ; 9. Spalte
       C01A D3E0               OUT  (E0H),A ; 32 00 E0   LD (0E000H),A
       C01C 00                 NOP               ; beim 700er
       C01D DBE1               IN   A,(E1H) ; 3A 01 E0   LD A,(0E001H)
       C01F 00                 NOP               ; beim 700er
       C020 B7                 OR   A            ; CARRY-Flag loeschen
       C021 17                 RLA               ; Bit 7 durch linksrotieren
       C022 17                 RLA               ; in das CARRY-Flag
       C023 17                 RLA               ; bringen.
       C024 17                 RLA
       C025 3E00               LD   A,00H
       C027 3802               JR   C,02H
       C029 3E01               LD   A,01H
       C02B 323DC0             LD   (SPAPO),A
       C02E C1                 POP  BC
       C02F E1                 POP  HL
       C030 00                 NOP          ; D3 E1    OUT 0E1H,A
       C031 00                 NOP          ; beim 700er
       C032 C9                 RET
       C033 BF                 DEFB 0BFH
       C034 D7                 DEFB 0D7H
       C035 F7                 DEFB 0F7H
       C036 E7                 DEFB 0E7H
       C037 EF                 DEFB 0EFH
       C038 EB                 DEFB 0EBH
       C039 FB                 DEFB 0FBH
       C03A DB          EOTAB: DEFB 0DBH
       C03B 00          CURPO: NOP

SEITE    2
F ZLNR ADRS CODE       QUELLTEXT

       C03C 00                 NOP
       C03D 00          SPAPO: NOP
       C03E 00                 NOP
       C03F                    END

FEHLER    0
```

LABEL:

SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS	SYMBOL	ADRS
CURPO	C03B	EOTAB	C03A	SPAPO	C03D						

Tastaturabrage beim M2-700/800

Erklärung des Assemblerprogramms:

Zeile 18: Startadresse festlegen (hier $C000)

19-20: 800er-Version: NOP (No operation) NOP
700er-Version: OUT 0E3H,A (schaltet $0000 bis $FFFF auf V-Ram-Tastatur)
dies ist die Schreibweise meines Assemblers für HEX-Zahlen ($E3)

21-27 Retten der Doppelregister HL und BC auf dem Stack

23 Laden des Akkus mit der Spaltennummer die gescannt werden soll (hier Spalte 8, als $07)

24-25 Senden des Akkuinhalts über Port A der PIO
800-er: über Portadresse $D0
700er: über Speicherstelle $E000

26-27 Abgreifen der empfangenen Daten am Port B
800er: über Portadresse $D1
700er: über Speicherstelle $E001

28 Dopelregister BC mit $0009 laden (zum herunterzählen während des Schleifendurchlaufs)

29 HL wird mit der Adresse geladen, an der das Ende der Tabelle liegt) (E0TAB).

30 CPDR: folgende Schleife wird durchlaufen: wiederhole:
A (Akku) - Inhalt der Speicherstelle die durch HL adressiert wird (das Ergebnis wird nicht weiter berücksichtigt)
HL=HL-1
BC=BC-1
solange wie BC≠0 und A≠ dem Inhalt der Speicherstelle, die durch HL adressiert wird.
Ist diese Schleife abgearbeitet, dann enthält das Registerpaar BC die Nummer der gedrückten Tastenkombination (0-8).

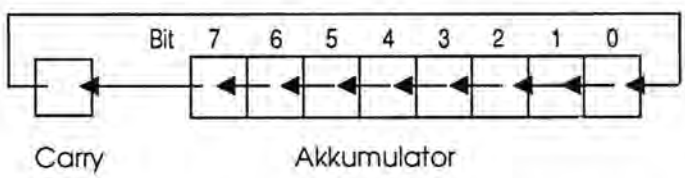
31 Der Inhalt von BC wird in die am Programmende dafür vorgesehene Speicherstelle (Startadresse + $3B) geladen.

33-37 Der gleiche Scann-Vorgang wie 23-27 nur mit Spalte 7

38 CARRY-Flag löschen (internes Register des Z80-Prozessors)

39-43 Durch mehrmaliges linksrotieren des Akkumulators durch das CARRY-Flag wird das für SPACE relevante Bit (Nr. 4, siehe auch Tastaturmatrix) in das CARRY-FLAG transportiert

43 Akku wird geladen



44 Falls das CARRY-Flag gesetzt ist, also SPACE nicht gedrückt wurde, (die von der Tastaturmatrix empfangenen Bits sind ohne Tastendruck alle HI (1)) wird ein relativer Sprung um 2 Bytes vorwärts ausgeführt, also nach Zeile 46

45 Akku wird mit 1 geladen (nur wenn SPACE gedrückt wurde, also das CARRY-Flag nicht gesetzt ist)

46 Die Speicherstelle am Ende des Programms (Startadresse +$3D) in der steht der SPACE gedrückt wurde oderr nicht, wird mit dem Inhalt des Akkumulators geladen

47-48 Die geretteten Registerpaare BC und HL werden wieder vom Stack geholt

49-50 800er: NOP
700er: OUT 0E1,H,A (schaltet $0000-$FFFF wieder auf Systembereich

51 Rücksprung in das aufgerufene Basic-Programm

52-59 Hier liegt die Tabelle der Tastaturmatrixwerte, die mit dem Inhalt des Akkus verglichen werden.

Tastencode	D7	D6	D5	D4	D3	D2	D1	D0	
	INST	DEL	↑	↓	→	←	?	/	
1≙ $DF =	1	1	0	1	1	1	1	1	↑
2≙ $D7 =	1	1	0	1	0	1	1	1	↑ +→
3≙ $F7 =	1	1	1	1	0	1	1	1	→
4≙ $E7 =	1	1	1	0	0	1	1	1	→ +↓
5≙ $EF =	1	1	1	0	1	1	1	1	↓
6≙ $EB =	1	1	1	0	1	0	1	1	↓ +←
7≙ $FB =	1	1	1	1	1	0	1	1	←
8≙ $DB =	1	1	0	1	1	0	1	1	← +↑

Nach dem Aufruf der ML Routine mit USRC (Startadresse) steht in der Adresse (Startadresse +$3B) die Codenummer der gedrückten Tastenkombination. In (Startadresse + $D3) steht 1, wenn SPACE gedrückt wurde, sonst 0. Für eine genauere Erklärung der einzelnen Maschinenbefehle ist das Buch „Programmierung des Z80" von Rodnay Zaks, erschienen im SYBEX-Verlag, empfehlenswert.

Hier wird nun die eleganteste Methode zum Einbau eines ML-Programmes oder anderer Daten in ein Basicprogramm vorgestellt: Die Routine wird unsichtbar in die erste Zeile des Basicprogramms eingebaut!

Zur Vereinfachung habe ich dazu ein genau auf unsere kleine Tastaturabfrageroutine zugeschnittenes Programm geschrieben, das die gesamte Installation selbstständig vornimmt. Dazu muß der Anfang des Ba-

Tastaturabrage beim MZ-700/800

sicprogrammspeichers, und ob es sich um ein 700er oder 800er Basic handelt, angegeben werden.

Die mir bekannten Anfangsadressen habe ich in dem Programm aufgelistet:

für **700er-Basic**:

S-Basic/SOLO-BASIC:	$6BCF
MZ-5Z008:	$85FF
MZ-2Z009E:	$85FE
K&P 700-D-Basic:	$7CA1

800er-Basic (hier muß NEWON beachtet werden):

	ohne NEW ON	mit NEW ON
MZ-1Z016:	$A3FA	$9F9E
MZ-5Z009:	$A471	$A015
MZ-2Z046:	$A473	$A017
K&P800D:	$AC2A	$A79E

Die nach Ablauf des Programms entstandene Zeile 0 kann nach dem Abspeichern in jedes beliebige Basicprogramm mit MERGE eingebunden werden, aber natürlich nur bei Programmen im gleichen Basictyp.

ACHTUNG: Das Programm unbedingt vor dem Starten abspeichern sonst ist alles schneller wieder weg als es gekommen ist!!!

Wichtig ist auch, daß z.B. ein Spielprogramm erst geschrieben wird, und dann erst Zeile 0 eingeladen wird. Danach sollte nichts mehr verändert werden.

Listing ☞ Seite 30/31

Wie wird ein Teil einer Basiczeile versteckt?

Dazu muß man an die Stelle, bis zu der die Zeile sichtbar sein soll, $00 gepokt werden, da $00 die Endkennung einer Basiczeile ist. (Dies geschieht im Basicprogramm in Zeile 1280). Befehle, die in dem unsichtbaren Teil stehen, werden allerdings nicht mehr ausgeführt.

Wie ist nun ein Basicprogramm im Speicher gespeichert?

z.B.: 10 REMuTEST (u=SPACE)
20 END

Im Speicher steht dann (MZ-2Z046 ohne NEW ON):

Adresse	Inhalt	Bedeutung
$A473	0B	Anzahl der Bytes in der Zeile (Low Byte)
$A474	00	Anzahl der Bytes in der Zeile (High Byte; immer 0, da nur 255 erlaubt sind)
$A475	0A	Zeilennummer (Low = 10) } bis $FFFF (65536)
$A476	00	Zeilennummer (High) }
$A477	97	Token für REM
$A478	20	SPACE
$A479	54	T
$A47A	45	E
$A47B	53	S
$A47C	54	T
$A47D	00	Ende der Basiczeile
$A47E	06	Anzahl der Bytes in der Zeile (Low Byte)
$A47F	00	Anzahl der Bytes in der Zeile (High Byte)
$A480	14	Zeilennummer (Low) = 20
$A481	00	Zeilennummer (High)
$A482	98	Token für END
$A483	00	Ende der Basiczeile
$A484	00	Ende Zeilenzähler = 0 > Ende des Programms
$A485	00	Ende Zeilennummer = 0

Wird nun bei diesem Programm in Adresse $A478 der Wert $00 gepokt, so wird diese Adresse als Zeilenende angenommen, obwohl die Speicherstelle für die Byteanzahl mehr Bytes angibt, nämlich $0B=11dezimal. Die Bytes zwischen dem neuen Zeilenende und dem tatsächlichen Zeilenende werden nun weder angezeigt noch abgearbeitet.

```
LIST
10 REM
20 END
```

Auf diese Weise kann man Text, z.B. einen Copyright-Vermerk oder Maschinenroutinen, verstecken.

Es ist auch möglich, die Zeilennummer zu Null zu machen (siehe Zeile 1290). Man muß nur in die Speicherstelle, die die Zeilennummer enthält, 0 poken. (Hier für die 1. Zeile $A475). Es ist auch möglich, mehrere Null-Zeilen hintereinander an den Anfang eines Basic-Programms zu stellen. Was nicht geändert werden darf, ist der Inhalt der Speicherstelle für die Anzahl der Bytes. Der hier stehende Wert muß immer mit dem tatsächlichen Zeilenende übereinstimmen, sonst passieren unter anderem recht erstaunliche Dinge.

Hier noch ein kleines Beispielprogramm um die erzeugte Zeile 0 zu testen: (die XXXX bei USR und PEEK müssen durch die in Zeile 0 angegebenen Adressen ersetzt werden.) Listing ☞ Seite 31

Erst eingeben und dann die vorher erzeugte Zeile 0 mit MERGE einbinden.

Axel Lücking, Mozartstraße 2, 4900 Herford

Tastaturabrage beim MZ-700/800

```
10 REM"Niemals ändern:USR($XXXX):$XXXX=C
URSOR:$XXXX=SPACE ********** Dies sind
ab dem 1. * 64 Zeichen ****************
**
20 ' Axel Lücking   Mozartstr. 2
     4900 Herford
30 ' Zeile 10 nicht ändern und keine
     Zeile davor schreiben !!! Dort
     liegt die Tastaturabfrageroutine !
40 ' Nach dem Starten schreibt sich die
     ML-Routine selbsttätig dort hinein
     und die Zeile bekommt die Nummer 0.
     Danach wird alles außer Zeile 0
50 ' gelöscht und Zeile 0 als File ab-
     gespeichert. Danach kann "ZEILE 0"
     in jedes beliebige Programm als
60 ' ERSTE Zeile eingebunden werden.
70 ' Die XXXX in Zeile 10 werden nach
     Programmablauf automatisch durch
     die entsprechenden Adressen ersetzt
80 ' PS  Eine Zeile mit der Nummer 0
         kann nicht einfach gelöscht
         oder überschrieben werden.
90 '     (Nur mit DELETE und NEW)
100 '    Allerdings wird sie mit RENUM
         neu durchnummeriert. Man sollte
         also bei RENUM darauf achten,
         daß durch die nachfolgenden
110 '    Parameter die erste Zeile aus-
         geschlossen wird zB RENUM 10,10
         Für die Funktion der Routine
         ist es unerheblich welche
120 '    Nummer die erste Zeile hat,
         solange es die ERSTE Zeile im
         Programm ist.
130 '
140 CLS:PRINT:PRINT" MZ-700er BASICs:■"
150 PRINT" S-BASIC / SOLO-BASIC : $6BCF"
160 PRINT" MZ-52008 : $85FF     MZ-22009E
 : $85FE"

170 PRINT" K&P MZ-700D-Basic : $7CA1"
180 PRINT:PRINT
190 PRINT" MZ-800er BASICs:     normal /
NEWON■"
200 PRINT" MZ-12016             $A3FA
$9F9E"
210 PRINT" MZ-52009 (QD)        $A471
$A015"
220 PRINT" MZ-22046 (FD)        $A473
$A017"
230 PRINT" K&P MZ-800D-Basic    $AC2A
$A79E"
240 INPUT "■■Anfang des Basicspeichers :
 $■----■■■■";ST$:ST=VAL("$"+ST$)
250 PRINT"■■■Ist dies ein 700 oder 800 B
asic [7/8]                  -       -"
260 GET A$:IF A$="7" THEN MZ=0:GOTO 290
270 IF A$="8" THEN MZ=1:GOTO 290
280 GOTO 260
290 '
300 ' 700 ---------------------------
310 DATA $D3,$E3     :' OUT  $E3,A
320 DATA $E5         :' PUSH HL
330 DATA $C5         :' PUSH BC
340 DATA $3E,$07     :' LD   A,$07
350 DATA $32,$00,$E0:' LD   ($E000),A
360 DATA $3A,$01,$E0:' LD   A,($E001)
370 DATA $1,$9,$0    :' LD   BC,$0009
380 DATA $21,$00,$00:' LD  HL,Start+$3A
390 '          LD  HI Startadresse + $3A
400 DATA $ED,$B9     :' CPDR
410 DATA $ED,$43,$00,$00:'LD(St+$3B),BC
420 '                LD  HI Startadr.+ $3B
430 DATA $3E,$06     :' LD   A,$06
440 DATA $32,$00,$E0:' LD   ($E000),A
450 DATA $3A,$01,$E0:' LD   A,($E001)
460 DATA $B7         :' OR   A
470 DATA $17         :' RLA
480 DATA $17         :' RLA
490 DATA $17         :' RLA

500 DATA $17         :' RLA
510 DATA $3E,$00     :' LD   A,$00
520 DATA $38,$02     :' JR   C,$02
530 DATA $3E,$01     :' LD   A,$01
540 DATA $32,$00,$00:' LD ($Start+$3D),A
550 '          LD  HI von Startadr.+ $3D
560 DATA $C1         :' POP  BC
570 DATA $E1         :' POP  HL
580 DATA $D3,$E1     :' OUT  $E1,A
590 DATA $C9         :' RET
600 DATA $DF,$D7,$F7,$E7,$EF,$EB,$FB,$DF
:'   (Tabelle der Matrixwerte)
610 DATA $0,$0
620 DATA $0,$0
630 '
640 ' 800 ---------------------------
650 DATA $00,$00     :' NOP  NOP
660 DATA $E5         :' PUSH HL
670 DATA $C5         :' PUSH BC
680 DATA $3E,$07     :' LD   A,$07
```

Tastaturabrage beim M2-700/800

```
690 DATA $D3,$D0      :' OUT  $D0,A
700 DATA $00          :' NOP
710 DATA $DB,$D1      :' IN   A,$D1
720 DATA $00          :' NOP
730 DATA $1,$9,$0     :' LD  BC,$0009
740 DATA $21,$00,$00:' LD  HL,Start+$3A
750 '          LO  HI Startadresse + $3A
760 DATA $ED,$B9      :' CPDR
770 DATA $ED,$43,$00,$00:'LD(St+$3B),BC
780 '               LO  HI Startadr.+ $3B
790 DATA $3E,$06      :' LD   A,$06
800 DATA $D3,$D0      :' OUT  $D0,A
810 DATA $00          :' NOP
820 DATA $DB,$D1      :' IN   A,$D1
830 DATA $00          :' NOP
840 DATA $B7          :' OR   A
850 DATA $17          :' RLA
860 DATA $17          :' RLA
870 DATA $17          :' RLA
880 DATA $17          :' RLA

890 DATA $3E,$00      :' LD   A,$00
900 DATA $38,$02      :' JR   C,$02
910 DATA $3E,$01      :' LD   A,$01
920 DATA $32,$00,$00:' LD (Start+$3D),A
930 '          LO  HI von Startadr.+ $3D
940 DATA $C1          :' POP  BC
950 DATA $E1          :' POP  HL
960 DATA $00          :' NOP
970 DATA $00          :' NOP
980 DATA $C9          :' RET
990 DATA $DF,$D7,$F7,$E7,$EF,$EB,$FB,$DB
:'   (Tabelle der Matrixwerte)
1000 DATA $0,$0
1010 DATA $0,$0
1020 IF M2=0 THEN RESTORE 310
1030 IF M2=1 THEN RESTORE 650
1040 FOR I=0 TO 62
1050 READ A$:POKE ST+I+57,VAL(A$)
1060 NEXT I
1070 ' Startadresse in 10 bei USR(XXXX)
       poken
1080 A$=HEX$(ST+57)
1090 FOR I=0 TO 3:POKE ST+26+I,ASC(MID$(
A$,I+1,1)):NEXT I
1100 ' Anfang der Tabelle in das ML-Pro-
       gramm poken
1110 A$=HEX$(ST+57+$3A)
1120 A1=VAL("$"+RIGHT$(A$,2))
1130 A2=VAL("$"+LEFT$(A$,2))
1140 POKE ST+57+16,A1,A2
1150 ' Adresse in die die Cursor-Kombi-
       nation gepokt wird in das ML-Pro-
       gramm und in Zeile 10 poken.
1160 A$=HEX$(ST+57+$3B)
1170 A1=VAL("$"+RIGHT$(A$,2))
1180 A2=VAL("$"+LEFT$(A$,2))
1190 POKE ST+57+22,A1,A2
1200 FOR I=0 TO 3:POKE ST+33+I,ASC(MID$(
A$,I+1,1)):NEXT I
1210 ' Adresse in die das SPACE-Flag

       gepokt wird in das ML-Programm
       und in Zeile 10 poken.
1220 A$=HEX$(ST+57+$3D)
1230 A1=VAL("$"+RIGHT$(A$,2))
1240 A2=VAL("$"+LEFT$(A$,2))
1250 POKE ST+57+44,A1,A2
1260 FOR I=0 TO 3:POKE ST+46+I,ASC(MID$(
A$,I+1,1)):NEXT I
1270 '
1280 POKE ST+56,0:' Zeile 10 unsichtbar,
       da 00 die Endkennung einer Basic-
       zeile ist.
1290 POKE ST+2,0:' aus Zeile 10 wird 0
1300 CLS
1310 PRINT"Drücken Sie [CR]..."
1320 PRINT
1330 PRINT"SAVE "+CHR$($22)+"ZEILE 0"+CH
R$($22)
1340 CURSOR 0,1
1350 DELETE 20-
```

```
10 ' TEST ----------------------------
20 CLS
30 X=20:Y=12:A=0:B=0
40 USR($XXXX):'   XXXX muß durch die
50 A=PEEK($XXXX):'entsprechenden Werte
60 B=PEEK($XXXX):'in 0 ersetzt werden.
70 CURSOR 15,0:PRINT A,B;
80 IF (A=0) * (B=0) THEN GOTO 220
90 CURSOR X,Y:PRINT " ";
100 IF A=1 THEN Y=Y-1
110 IF A=2 THEN X=X+1:Y=Y-1
120 IF A=3 THEN X=X+1
130 IF A=4 THEN X=X+1:Y=Y+1
140 IF A=5 THEN Y=Y+1
150 IF A=6 THEN X=X-1:Y=Y+1
160 IF A=7 THEN X=X-1
170 IF A=8 THEN X=X-1:Y=Y-1
180 IF X<1  THEN X=1
190 IF X>38 THEN X=38
200 IF Y<1  THEN Y=1
210 IF Y>23 THEN Y=23
220 CURSOR X,Y:IF B=0 THEN PRINT "?";:EL
SE PRINT "#";
230 GOTO 40
```

Full Screen EDitor unter CP/M

Anleitung zum FSED

Der FSED ist eine Full Screen Erweiterung des CP/M ED. Er orientiert sich an der Organisation des ED und der Bedienung des Turbo Pascal Editors. Die Arbeit mit FSED wird deshalb mit einer durch TUTASTEN geänderten Tastenbelegung erheblich vereinfacht. Der FSED vereinigt die Vorteile des ED mit dem Komfort moderner Full Screen Editoren. Er verarbeitet TAB Codes (^I), ist mit 14 kB kürzer als Turbo Pascal, bietet gute Editierfunktionen, legt BAK Files an und ist ziemlich benutzerfreundlich im Gegensatz zum ED. Nachteilig ist seine geringe Geschwindigkeit, die einen durchschnittlichen Schreiber bei längeren Zeilen oft zu Pausen zwingt.

Der FSED wird vom CCP aus aufgerufen. Dabei kann das zu bearbeitende File in der Kommandozeile angegeben werden. Ist dies nicht der Fall, so fragt FSED mit einem * nach. Wird keine Extension angegeben, so setzt FSED Z80 voraus. Eventuelle Laufwerkskennungen verarbeitet FSED korrekt. Existiert das zu bearbeitende File bereits, so wird es in File.BAK umbenannt und ein neues File eröffnet. Ein altes Backup wird dabei gelöscht. Ist das File nicht neu, fragt FSED mit >LOAD?, ob der Inhalt der alten Datei übernommen werden soll. Diese Frage wird mit Y oder N beantwortet. Nun übergibt FSED an den eigentlichen Editor.

Die Editierfunktionen

Der FSED bietet eine große Anzahl komfortabler Editierfunktionen, die – soweit implementiert – mit denen des Turbo Pascal Editors identisch sind.

Die Funktionen im einzelnen:

^S: ein Zeichen nach links
^D: ein Zeichen nach rechts
^E: eine Zeile nach oben
^X: eine Zeile nach unten
^A: ein Wort nach links; hält zwischen zwei Worten
^F: ein Wort nach rechts
^R: eine Seite nach oben
^C: eine Seite nach unten
^I: Tab Code setzen
^G: Zeichen unter Cursor löschen
^T: Löschen des dem Cursor folgenden Wortes
^Y: Zeile löschen
^N: Zeile einfügen

Erweiterte Codes mit ^Q:
^Q S: Anfang der Zeile
^Q D: Ende der Zeile
^Q R: Anfang des Textes
^Q C: Ende des Textes
^Q Y: Löschen bis Zeilenende
^Q I: springt zur letzten TAB Position

Erweiterte Codes mit ESC (^Ä):
^Ä E: FSED verlassen und neue Datei speichern
^Ä O: Rückkehr zur Originaldatei und Fortsetzen des Editierens
^Ä R: Seite neu ausgeben (Hilfsroutine bei der Entwicklung, wird nicht mehr benötigt).
^Ä S: aktuelles Speicherbild der Datei abspeichern
^Ä X: FSED verlassen ohne Update

Die Taste CR wird wie ein ^X interpretiert. Im Insert Modus jedoch wird an der neuen Cursorposition eine Zeile eingefügt.

Der FSED ist für den Gebrauch mit Assembler Quelltexten entwickelt worden und fügt deshalb zu Beginn einer neuen Zeile immer ein TAB Zeichen ein. Der Anfang der Zeile kann dann über ^Q I erreicht werden. Soll der FSED für andere Zwecke eingesetzt werden, so läßt sich dieses, wie auch die Standard Extension, im Quelltext leicht ändern.

Das Programm ist in Turbo Pascal geschrieben worden und teilt sich in zwei Files auf. FSED.PAS enthält die Definitionen und den Editor selbst. Die einzelnen Editierfunktionen werden hier zusammengestellt. FSED.INC enthält die Grundprozeduren, aus denen sich der Editor zusammensetzt und die Programminitialisierungen.

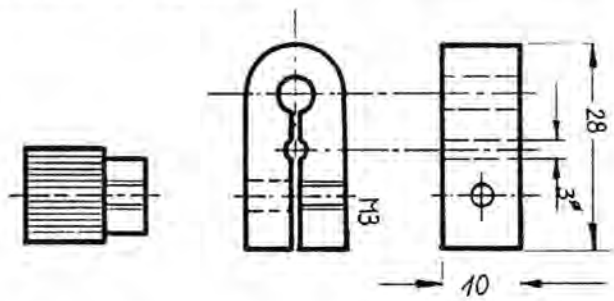
Der FSED ist nicht das Ende der Fahnenstange, sondern ein Entwicklungswerkzeug in der Version 1.0. Es ist ein Anstoß und eine Hilfe an alle entnervten ED Benutzer und ein gemeinsames Werkzeug für die Leser des MagaZin 700/800.

Der FSED ist für DM 15,-- als Quellcode und compiliert auf einer 5,25" Diskette in Standard P-CP/M erhältlich bei:

MGR-Software
Lars Hanke
Schlachthofstr. 67
4690 Herne 2

Viel Spaß bei der Arbeit und eventuell Weiterentwicklung des FSED!!!

Im Artikel „**Plotter-Tips**", MagaZin 700/800, Heft 4/88, Seite 5, haben wir leider versäumt, die Skizze der **Klemmvorrichtung** abzubilden. Das wollen wir hier nachholen. Maßstab 1:1, Material z. B. Leichtmetall, Stehbolzen nicht mitgezeichnet.



MagaZin 700/800 - 5+6/88

Full Screen EDitor unter CP/M

```
### Main File: FSED.pas

ä Full Screen EDitor saving ^I as tab sign ü
ä Programmed for assembler programming by Lars Hanke ü

program Full_Screen_Editor;

type lintyp = string[80ü;
     linpoi = ^inhalt;
     inhalt = record
                  line:array[1..80ü of char;
                  next:linpoi;
                  last:linpoi;
              end;

var multi,fil,workfil:text;
    line:lintyp;
    firstline,lastline,acline:linpoi;
    finish,old,insert,ok:boolean;
    col,scrow:byte;
    ans:char;

### Including: fsed.inc


function exist(name:lintyp):boolean;
         begin
              assign(multi,name);
              ä$I-ü
              reset(multi);
              ä$I+ü
              exist:=(ioresult=0);
          end;

function adapt(name:lintyp):lintyp;
         var help:lintyp;
             len,len1:byte;
         begin
              len:=pos(':',name);
              if len>1 then help:=copy(name,pred(len),80)
                 else if len=1 then help:=copy(name,2,80)
                 else help:=name;
              len1:=pos('.',help);
              if len1=0 then if len>1 then
                        begin
                             help:=copy(help,1,10);
                             help:=concat(help,'.ü80');
                        end else
                        begin
                             help:=copy(help,1,3);
                             help:=concat(help,'.ü80');
                        end
              else begin
                   if len>1 then len:=2 else len:=0;
                   help:=copy(help,1,12+len);
                   if pos('.',help)=0 then help:='';
              end;
              adapt:=help;
         end;

procedure init_files(name:lintyp;var older:boolean);
         var nam2:lintyp;
         begin
              if ok then
              begin
                   if exist(name) then
                   begin
                        older:=true;
                        assign(fil,name);
                        nam2:=concat(copy(name,1,(length(name)-3)),'bak');
                        close(fil);
                        if exist(nam2) then
                        begin
                             assign(multi,nam2);
                             erase(multi);
                        end;
                        rename(fil,nam2);
                        reset(fil);
                        writeln(' ### Old ',name,' renamed to ',nam2);
                        assign(workfil,name);
                        rewrite(workfil);
                   end else
                   begin
                        assign(workfil,name);
                        rewrite(workfil);
                        older:=false;
                   end;
              end;
         end;

procedure init_line(var zeile:linpoi);
         var lauf:byte;
         begin
              new(zeile);
              for lauf:=2 to 80 do zeile^.line[lauf]:=chr(0);
              zeile^.last:=nil;
              zeile^.next:=nil;
              zeile^.line[1]:=chr(9);
         end;

procedure write_poi(zeile:linpoi);
         var lauf:byte;
             zeic:char;
         begin
              lauf:=1;
              repeat
                   zeic:=zeile^.line[lauf];
                   write(zeic);
                   lauf:=succ(lauf);
              until ((lauf>80) or (zeic=chr(0)));
              clreol;
         end;

function conv_line(zeile:linpoi):lintyp;
         var lauf:byte;
             help:lintyp;
             zeic:char;
         begin
              help:='';
              lauf:=1;
              zeic:=zeile^.line[lauf];
```

```
            while ((lauf<81) and (zeic<>chr(9))) do
            begin
                help:=concat(help,zeic);
                lauf:=succ(lauf);
                zeic:=zeile^.lineÄlaufÜ;
            end;
            conv_line:=help;
        end;

procedure conv_poi(strings:lintyp;var zeile:linpoi);
        var lauf:byte;
        begin
            for lauf:=1 to 80 do zeile^.lineÄlaufÜ:=chr(0);
            for lauf:=1 to length(strings) do
                zeile^.lineÄlaufÜ:=stringsÄlaufÜ;
        end;

procedure init_poi(load:boolean);
        begin
            mark(firstline);
            release(firstline);
            init_line(firstline);
            if load then
            begin
                reset(fil);
                acline:=firstline;
                repeat
                    readln(fil,line);
                    init_line(lastline);
                    lastline^.last:=acline;
                    acline^.next:=lastline;
                    conv_poi(line,lastline);
                    acline:=lastline;
                until eof(fil);
                acline:=firstline^.next;
                firstline:=acline;
                firstline^.last:=nil;
            end else
            begin
                acline:=firstline;
                lastline:=firstline;
            end;
        end;

procedure save_text;
        var savel:linpoi;
        begin
            rewrite(workfil);
            savel:=firstline;
            repeat
                writeln(workfil,conv_line(savel));
                savel:=savel^.next;
            until savel=nil;
        end;

procedure print_status;
        begin
            gotoxy(1,24);
            write('          ');
            gotoxy(1,24);
            if insert then write(' Insert') else write(' Overwrite');
        end;

procedure ins_poi;
        var help:linpoi;
        begin
            init_line(help);
            help^.next:=acline;
            help^.last:=acline^.last;
            acline^.last:=help;
            help^.last^.next:=help;
            acline:=help;
        end;

procedure del_poi;
        begin
            if acline<>firstline then
            begin
                acline:=acline^.last;
                acline^.next^.next^.last:=acline;
                acline^.next:=acline^.next^.next;
                if acline^.next<>nil then acline:=acline^.next
                                     else scrow:=pred(scrow);
                if scrow<1 then scrow:=1;
            end else
            begin
                acline:=acline^.next;
                acline^.last:=nil;
                firstline:=acline;
            end;
        end;

function start_col:byte;
        var lauf:byte;
        begin
            lauf:=1;
            while (acline^.lineÄlaufÜ=chr(9)) do lauf:=succ(lauf);
            start_col:=lauf;
        end;

procedure goto_pos(loc:byte);
        var lauf:byte;
        begin
            loc:=pred(loc);
            gotoxy(1,scrow);
            if loc<>0 then for lauf:=1 to loc do write(acline^.lineÄlaufÜ);
        end;

function headline(poi:linpoi):linpoi;
        var help:linpoi;
            lauf:byte;
        begin
            lauf:=scrow;
            help:=acline;
            while ((lauf<>1) and (help<>firstline)) do
            begin
                lauf:=pred(lauf);
                help:=help^.last;
            end;
            headline:=help;
```

```
        end;

procedure print_screen(top:linpoi);
        var lauf:byte;
            help:linpoi;
        begin
             lauf:=1;
             help:=top;
             clrscr;
             repeat
                  gotoxy(1,lauf);
                  write_poi(help);
                  help:=help^.next;
                  lauf:=succ(lauf);
             until ((help=nil) or (lauf=24));
             print_status;
        end;

procedure ins_char(zeile:linpoi;loc:byte);
        var lauf:byte;
        begin
             for lauf:=79 downto loc do
                 zeile^.line^[succ(lauf)]:=zeile^.line^[lauf];
             zeile^.line^[loc]:=' ';
        end;

procedure del_char(zeile:linpoi;loc:byte);
        var lauf:byte;
        begin
             for lauf:=succ(loc) to 80 do
                 zeile^.line^[pred(lauf)]:=zeile^.line^[lauf];
             zeile^.line^[80]:=chr(0);
        end;

procedure new_line;
        var help:linpoi;
        begin
             scrow:=succ(scrow);
             if scrow>23 then
             begin
                  gotoxy(1,24);
                  writeln;
                  scrow:=23;
                  print_status;
             end;
             if acline^.next=nil then
             begin
                  init_line(help);
                  help^.last:=acline;
                  acline^.next:=help;
                  acline:=help;
                  lastline:=help;
             end else acline:=acline^.next;
             gotoxy(1,scrow);
             write_poi(acline);
             goto_pos(start_col);
             col:=start_col;
        end;

procedure old_line;
        begin
             if acline^.last<>nil then
             begin
                  scrow:=pred(scrow);
                  acline:=acline^.last;
                  if scrow<1 then
                  begin
                       scrow:=1;
                       print_screen(acline);
                  end
                  else begin
                       gotoxy(1,scrow);
                       write_poi(acline);
                  end;
                  goto_pos(start_col);
                  col:=start_col;
             end;
        end;

procedure init_pro;
        var oldprg:boolean;
        begin
             clrscr;
             writeln;
             writeln(' Full Screen EDitor for use with M80 (ta) assembler');
             writeln(' FSED V 1.0 copyright 1988, by Lars Hanke');
             writeln;
             writeln(' FSED is a product of M&R - Software, Asgard');
             writeln;
             insert:=false;
             scrow:=1;
             col:=1;
             new(firstline);
             if paramcount>0 then line:=paramstr(1)
             else begin
                  writeln;
                  write('* ');
                  readln(line);
             end;
             line:=adapt(line);
             ok:=(line<>'');
             writeln;
             old:=false;
             if ok then
             begin
                  init_files(line,oldprg);
                  writeln(' *** process ',line);
                  if oldprg then
                  begin
                       writeln;
                       write('Load? ');
                       repeat read(kbd,ans); ans:=upcase(ans); until ans in ['Y','N'];
                       old:=(ans='Y');
                  end;
                  init_poi(old);
                  finish:=false;
             end else
             begin
                  finish:=true;
                  writeln(' *** Invalid filename *** ');
```

Full Screen EDitor unter CP/M

```
            end;
        end;

procedure next_char;
        begin
            col:=succ(col);
            if acline^.lineA[col]=chr(2) then acline^.lineA[col]:=' ';
        end;
*** End of: fsed.inc

procedure escapes;
        var mode:char;
        begin
            read(kbd,mode);
            mode:=upcase(mode);
            case mode of
                'X': finish:=true;
                'E': begin
                        save_text;
                        finish:=true;
                     end;
                'O': begin
                        init_pos(old);
                        scrow:=1;
                        col:=start_col;
                        print_screen(acline);
                     end;
                'S': save_text;
                'R': begin
                        print_screen(headline(acline));
                        goto_pos(col);
                     end;
            end;
        end;

procedure Q_Controls;
        var com:char;
            lauf:byte;
        begin
            read(kbd,com);
            com:=upcase(com);
            case com of
                'I': begin
                        while ((col>1) and (acline^.lineA[col]<>chr(9))) do
                              col:=pred(col);
                        ins_char(acline,col);
                        gotoxy(1,scrow);
                        write_pos(acline);
                        goto_pos(col);
                     end;
                'R': begin
                        acline:=firstline;
                        print_screen(acline);
                        scrow:=1;
                        col:=start_col;
                        goto_pos(col);
                     end;
                'C': begin
                        acline:=lastline;
                        print_screen(headline(acline));
                        col:=start_col;
                        goto_pos(col);
                     end;
                'Y': begin
                        for lauf:=col to 80 do acline^.lineA[lauf]:=chr(0);
                        clreol;
                     end;
                'S': begin
                        col:=start_col;
                        goto_pos(col);
                     end;
                'D': begin
                        while ((col<80) and (acline^.lineA[col]<>chr(0))) do
                              col:=succ(col);
                        goto_pos(col);
                     end;
            end;
        end;

procedure editor;
        var com:char;
            lauf2,lauf,code:byte;
        begin
            read(kbd,com);
            code:=ord(com);
            case code of
                27: escapes;
                 5: old_line;
                24: new_line;
                13: begin
                        new_line;
                        if insert then
                        begin
                            ins_pos;
                            print_screen(headline(acline));
                            goto_pos(start_col);
                        end;
                    end;
                 4: begin
                        write(acline^.lineA[col]);
                        next_char;
                    end;
                19: begin
                        gotoxy(1,scrow);
                        col:=pred(col);
                        if col<1 then col:=1
                        else goto_pos(col);
                    end;
                17: q_controls;
                22: begin
                        if insert then insert:=false else insert:=true;
                        print_status;
                        goto_pos(col);
                    end;
                 7: begin
                        del_char(acline,col);
                        gotoxy(1,scrow);
                        write_pos(acline);
                        goto_pos(col);
```

```
    end;
25: begin
        del_pos;
        print_screen(headline(acline));
        col:=start_col;
        goto_pos(start_col);
    end;
14: begin
        ins_pos;
        print_screen(headline(acline));
        col:=start_col;
        goto_pos(start_col);
    end;
 3: begin
        lauf:=1;
        while ((acline^.next<>nil) and (lauf<24)) do
        begin
            lauf:=succ(lauf);
            acline:=acline^.next;
        end;
        print_screen(headline(acline));
        col:=start_col;
        goto_pos(col);
    end;
18: begin
        lauf:=1;
        while ((acline^.last<>nil) and (lauf<24)) do
        begin
            lauf:=succ(lauf);
            acline:=acline^.last;
        end;
        if acline=firstline then scrow:=1;
        print_screen(headline(acline));
        col:=start_col;
        goto_pos(col);
    end;
 1: begin
        lauf:=pred(col);
        while ((acline^.lineÄlaufÜ in Ä'Ø'..'9','A'..'Z','a'..'z'Ü)
              and (lauf>1)) do lauf:=pred(lauf);
        if lauf<1 then lauf:=1;
        col:=lauf;
        goto_pos(col);
    end;
 6: begin
        lauf:=succ(col);
        while ((acline^.lineÄlaufÜ in Ä'Ø'..'9','A'..'Z','a'..'z'Ü)
              and (lauf<80)) do lauf:=succ(lauf);
        lauf:=lauf mod 80;
        col:=lauf;
        goto_pos(col);
    end;
20: begin
        lauf:=succ(col);
        while ((acline^.lineÄlaufÜ in Ä'Ø'..'9','A'..'Z','a'..'z'Ü)
              and (lauf<80)) do lauf:=succ(lauf);
        lauf:=lauf mod 80;
        for lauf2:=lauf to 80 do
        begin
            acline^.lineÄlauf2-lauf+colÜ:=acline^.lineÄlauf2Ü;
                acline^.lineÄlauf2Ü:=chr(0);
            end;
            gotoxy(1,scrow);
            write_poi(acline);
            goto_pos(col);
        end;
     9: begin
            if insert then ins_char(acline,col);
            acline^.lineÄcolÜ:=chr(9);
            next_char;
            gotoxy(1,scrow);
            write_poi(acline);
            goto_pos(col);
        end;
  32..125: begin
            if insert then ins_char(acline,col);
            acline^.lineÄcolÜ:=com;
            next_char;
            gotoxy(1,scrow);
            write_poi(acline);
            goto_pos(col);
        end;
  end; (* of case *)
end;

procedure edit;
      begin
          while not(finish) do editor;
      end;

begin
    init_pro;
    clrscr;
    print_screen(acline);
    gotoxy(1,scrow);
    col:=start_col;
    goto_pos(col);
    edit;
    close(workfil);
    close(fil);
    clrscr;
    writeln(' FSED disactivated    MGR, Asgard 1988');
end.
### End of: FSED.pas
```


MagaZin 700/800 - 5+6/88

MZ-800 als Timecode-Computer

In einer Zeit, in der sich viele vom MZ-800 abwenden, weil sie meinen, er wäre nicht mehr gut genug, freut man sich besonders, wenn jemand professionelles Zubehör zum MZ-800 entwickelt.

Stephan Scharf, Inhaber eines professionellen Videostudios, hat für den MZ-800 ein Eprom entwickelt, das den MZ-800 in einen professionellen Timecode-Generator verwandelt.

Was hat es mit dem Timecode auf sich?

Vor ca. dreißig Jahren hat die Firma Ampex den ersten Videorekorder entwickelt. Die Begeisterung war groß, denn Video hat gegenüber dem herkömmlichen Film den Vorteil niedrigerer Kosten und größerer Flexibilität.

Doch nach dem ersten Jubel kamen auch viele Fragen auf. Wie sollte man die Videoaufnahmen so nachbearbeiten, daß von der Vielzahl der gemachten Aufnahmen letztlich ein endgültiger Film entstand?

Das ist einer der Vorteile eines „richtigen" Films: man zählt einfach die Filmtransportlöcher und kann exakt festlegen, wo der Film geschnitten werden soll. Das Videoband hat jedoch keine Transportlöcher.

1963 schuf man deshalb „elektronische Transportlöcher"; man nahm auf der Audiospur Impulse auf, die von einem elektronischen Zähler ausgewertet werden konnten. Dropouts (schlechte Videobandstellen) und der schlechte Band-Kopf-Kontakt beim Umspulen machten das Verfahren aber zu ungenau. Außerdem mußte der Zähler erst einmal justiert werden. Dazu mußte man das Band immer an den Bandanfang zurückspulen, um das Zählwerk auf Null zu justieren.

Wichtig ist natürlich auch, daß man das Band mit unterschiedlicher Geschwindigkeit laufen lassen kann und trotzdem eine exakte Zählung erhält.

Im Jahre 1967 löste man alle Probleme durch Einführung des Timecode Verfahrens. Dabei wird dem Aufnahmeband (ob Video oder Audio ist egal) der Timecode aufgespielt. Er zählt nicht nur die exakte Zeit sondern auch die Anzahl der Bilder und legt sie codiert ab. So kann man jedes Bild oder jede Bandstelle immer wieder ganz exakt anfahren. Bei diesem Verfahren ist es auch nicht mehr nötig, zunächst an den Bandanfang zu fahren.

1972 wurde das Verfahren genormt. Je nach Norm gibt es verschiedene Timecodes, denn beim Film gibt es 24 Bilder (frames) pro Sekunde, beim PAL-Video 25 (EBU-Norm; ***E**uropian **B**roadcasting **U**nion*) und beim amerikanischen NTSC 30 (SMPTE-Norm; ***S**ociety of **M**otion **P**icture and **T**elevision **E**ngineers*).

Beim MZ-800 wird das eingebaute Monitor-Eprom gegen das neue Eprom ausgetauscht, wenn der Computer nur noch als Timecode-Computer Verwendung finden soll oder man baut es umschaltbar ein.

Beim Systemstart ist das Programm sofort bereit. Alle Daten werden sehr groß auf dem Monitor abgebildet (Display), so daß man auch aus größerer Entfernung alle Daten gut lesen kann.

Das Programm hat folgende Funktionen:

1. Generator
- Generierung des EBU-80Bit-Timecodes 25 Frames/Sec.
- wahlfreies Setzen von Zeit und Usergruppen
- manuelles Setzen des Genlock-Bits bei Fremdsynchronisierung
- Phasenbit ein- und ausschaltbar
- ASCII-Code Bit setzbar

2. Reader
- Lesen aller 80Bit-Timecodenormen:
  24 Frames/Sec.
  25 Frames/Sec. EBU
  30 Frames/Sec. SMPTE
  29,97 Frames/Sec. SMPTE
  29,97 F/Sec. SMPTE Drop Frame
- Lesegeschwindigkeiten ca. 1/20 bis 3fach vor- und rückwärts

3. Regenerator
Alle eingelesenen Timecodenormen sowie auch Fremdcodes werden regeneriert.

4. Event-Interface
- 2 Schaltausgänge können mit je 256 Schaltzeiten belegt werden. Jede Schaltzeit ist wahlfrei mit fünf Impulsformen belegbar:
  a) nicht aktiv
  b) Impuls 0,1 Sec.
  c) Impuls 0,1 Sec. moduliert mit 1000 Hertz
  d) High-Status
  e) Low-Status
- Speicherung der Schaltzeichen auf Tonträger

5. Display
Überdimensional große Darstellung der Timecodedaten auf einem handelsüblichen Datenmonitor oder Fernsehgerät. Selbst aus größerer Entfernung zum Bildschirm sind die Daten einwandfrei ablesbar.

Das Eprom wird mit einer ausführlichen Anleitung und allen nötigen Kabeln geliefert. Es kostet DM 398,-- Der MZ-800 eignet sich hier ideal, weil er zuletzt sehr preiswert zu erhalten war, äußerst robust ist (kann tagelang eingeschaltet bleiben) und mit dem eingebauten Kassettenrekorder die Schaltzeiten ideal speichern kann.

Für Interessierte haben wir eine Videokassette (VHS) vorbereitet, die Sie gegen DM 20,-- bestellen können. ■

Kleinanzeigen

Um Ihnen zu helfen, wenn Sie etwas suchen, kaufen oder verkaufen wollen, bieten wir Ihnen den Kleinanzeigenmarkt. Für bis zu fünf Zeilen zahlen Sie nur DM 10,--. Händler zahlen nur DM 20,--. Wenn Sie Gewerbetreibender sind, beachten Sie bitte, daß das aus Ihrer Kleinanzeige aus wettbewerbsrechtlichen Gründen deutlich hervorgehen muß. Jede Zeile darf bis zu 27 Anschläge haben. Bitte beachten Sie, daß Kleinanzeigen nur bei Vorkasse berücksichtigt werden können.

Verkaufe Quick Disk MZ-1F11und 45 Quick-Disk VB DM 400,--. Mit Super-Software S. Calgo, Textwriter, MZ-Cad, Nakamoto, 45 QD DM 300,--. ☎ 09826/578

Verkaufe MZ-800 Quick-Disk/ Disketten, ser. Schnittst. (MIDI/RS232) Assembl., Pascal, Serviceunterl., Handbücher. VB DM 380,--. Günther Jäger, ☎ 0208/685050

MZ-800 mit sehr viel Fachliteratur und Text-, Schach- und Karteiprogrammen abzugeben von
G. Fleck, Dachsweg 68, 5000 Köln 40, ☎ 0221/50 39 55

Wir sind eine Gruppe MZ-User und suchen **Kontakt** zu anderen. Bitte wenden Sie sich an Dietmar Richter, Fr.-Liszt-Str. 7, DDR-7010 Leipzig

---

Suche Software auf 3,5" Disketten für K&P-Floppy. Suche Disketten-Doktor und Assembler. Volker Neurath, Am Nordhang 87, 5620 Velbert

---

MZ-700, Plotter, Kassette, QD, 80-Zeichenkarte, I/O-Bus, 2 5,25" LW, Zenith 12" in Bernstein, mehrere Handbücher. Software dBase II, Multiplan, Wordstar, Basic, Pascal, Schach, CP/M usw. inkl. MZ-800 ohne I/O-Bus.
VB DM 1000,-.
☎ 030/70000931 (tagsüber)

Verkaufe MZ-800, Monitor 12", 10 Disketten leer, QD, RAM-Datei Akku gepuffert. Preis VB. R. Benner, 5248 Wissen 1 ☎ 02742/4463 nach 17.00 Uhr

---

Ich suche **Kontakt** zu Besitzern von nachfolgend aufgeführten Artikeln zwecks Erfahrungsaustausch: CP/M-Basic- + Monitor-Listing auf Papier oder Source-Code-Diskette. Ich suche ferner die CP/M-Programme INST.SMT, UNIFORM + MULTIPLAN. Kontaktadresse: Edgar Lefgrün, Torneiweg 3, 2400 Lübeck
☎ 0451/36228

SOKO-BAN 1 ist die überragende neue Spielidee für den MZ-800. SOKO-BAN 1 ist ein Denk- und Grafikspiel. Bis Sie alle Bilder gelöst haben, werden Sie mindestens 1-2 Monate brauchen! Für nur DM 29,95 bestellen Sie es bei SD-Software, J. Sperlich, Kurzröderstr.5, D-6000 Frankfurt 50 ☎ 069/541841

Brandneue Software zu super Preisen (max. 10 DM) - Spiele u. Anwendungen - Infos gegen 1 DM bei TBE-Soft Behrendt, Hamsterweg 28, 4350 R'hausen

---

Wer kann mir WORD STAR so installieren, daß WORD STAR, beim Drucken eines Textes, die Steuercodes und ESC Sequenzen des EPSON LQ-850 auch ausführt? Theo Dame, Postfach 1431, 5760 Arnsberg 1

Wegen Systemaufgabe
700/800 Software, ca. 900 Programme, z. T. mit Beschreibung und Handbüchern, sehr günstig abzugeben. Verschiedene Hardware, neu und gebraucht. Kostenlose Liste anfordern. R. Hahn, Pf. 123, 8704 Uffenheim, ☎ 09842/2816

Restposten, supergünstig:
1x Assemblerkurs ASEM-4 für MZ-700 auf Diskette mit 2 Handbüchern (ca. 400 S.), Lösungsblättern und Referenzkarten) DM 50,--.
1x Programmieren mit ELAN, Lehrbuch, ca. 200 S., DM 10,--
1x Wirtschaft auf dem MZ-700. BASIC-Programme für den Anwender mit grafischer Darstellung, 220 S., DM 20,--
1x Die Geldmaschine, 250 Ideen für lukrative Nebenverdienste mit Ihrem Mikrocomputer. 230 S., DM 25,--
1x Dr. Logo Geschichtenbuch, deutsch, DM 10,--
1x Dr. Logo Storybook, englisch, DM 10,--
1x CHIP Special Computer-Programme, SHARP MZ-700/800, DM 10,--
1x MZ-800 Bedienerhandbuch, deutsch, DM 39,--
1x Informatik für kaufmännische Schulen, 200 S., mit Beispielen in BASIC für Commodore, leicht abzuwandeln, DM 10,--
1x Headline, Die Setzmaschine für CP/M. Diskette, ausführliche Anleitung, Utilities zur Erstellung eigener Schriften. Schrift kann gedreht, invertiert, bemustert etc. werden. Ausdruck auf Matrixdrucker. DM 120,--
MZ-Verlag, ☎ 04187/6533

Serielle Schnittstelle für den MZ-700/800

Die von mir bisher vorgestellten Datenübertragungsverfahren konnten alle ohne zusätzlichen Hardware-Aufwand angewendet werden. Will man mit den MZ-Computern anderweitig nach außen in Verbindung treten, so kommt man um eine Nachrüstung einer seriellen Schnittstelle RS232C nicht herum. In den letzten Heften sind einige Lösungen vorgeschlagen worden. Ob sie dem Schnittstellen-Standard V.24 entsprechen, um z.B. ein MODEM oder einen Akustikkoppler, einen seriellen Drucker oder ähnliches damit zu betreiben, kann ich nicht beurteilen. Der Schnittstellenstandard ist wichtig, weil CP/M und auch 800er-BASIC ihn benötigen.

Wegen des hohen Preises der damals auf dem Markt befindlichen seriellen Schnittstellen für den 800er hatte ich mich zum Selbstbau entschlossen. Da mir jemand eine Leiterplatine für die SIO-PIO-Karte des mc-CP/M-Computers geschenkt hatte, machte es keine Mühe, diese zu bestücken und anzuschließen, da alle geforderten Signalanschlüsse vom Sharp-I/O-Port angeboten werden.

Ein kleines Problem bereitete die 12V-Spannung. Man kann sie elegant mit einem MAX232 aus den vorhandenen 5V herstellen. Ich habe sie mit einem kleinen Platinentrafo und Gleichrichtung hergestellt. Das ist etwas billiger, allerdings auch größer. Da die mc-Platine jedoch ohnehin nicht mehr in den 800er hineinpaßte, spielte der Platzverbrauch keine Rolle.

Die mc-Platine liefert zwei serielle Ports, das paßte mir gut, weil man einen mit den DIP-Schaltern auf 300, den anderen auf 1200 Baud einstellen kann. Notfalls genügt auch einer. Der Parallel-Port hätte keine Vorteile gebracht; ich habe ihn auf der Platine nicht bestückt.

Ob man mit dem Selbstbau Geld sparen kann und ob fertige Schnittstellen noch lieferbar sind, weiß ich nicht. Auf jeden Fall kann man eine Menge lernen dabei, und es ist hochinteressant, an die Datenleitung mal einen kleinen Lautsprecher anzuschließen und die Datensignale zu hören, die man aussendet.

Die ersten Tests empfehle ich in 800er-BASIC durchzufhren, da der Interpreter wie gesagt in der Lage ist, mit dem INIT-Befehl die Schnittstelle zu initialisieren und mit den Dateibefehlen WOPEN#, PRINT# usw. zu bedienen.

Ein Platinenlayout kann ich aus urheberrechtlichen Gründen natürlich hier nicht abdrucken lassen. Die Zeitschrift 'mc' hilft da sicherlich weiter, das Projekt heißt 'SIO-PIO-Karte für den mc-CP/M-Computer'. Notfalls kann ich mit einer Ablichtung für den Eigengebrauch helfen. Wie gesagt, es ist unerheblich, welche serielle Schnittstelle man kauft oder selber baut, wenn sie nur dem Standard entspricht. Denn der ist Voraussetzung für das Betreiben der oben erwähnten Zusatzgeräte. Doch davon soll in einem der nächsten Beiträge berichtet werden.

Dirk Grube, Ulmenstr. 31, 2320 Plön
☎04522/8714

WINDOW-Technik 700 filelist

Da eine genaue Beschreibung zu den Programmen filelist und WINDOW-Technik 700 im MagaZin 700/800, Heft 3/88, fehlte, liefere ich sie hiermit nach:

WINDOW-Technik 700:

Das Programm ersetzt nicht eine perfekte Windowtechnik wie bei größeren Computern. Es dient lediglich dazu, den Inhalt des Bildschirms zu erhalten und gleichzeitig weitere Informationen geben zu können. Dabei wird kurzzeitig der Bildschirminhalt „zerstört" und nach einer Eingabe o. ä. wieder hergestellt. Das Fenster (WINDOW), in dem die Ein- und Ausgabe geschehen soll, ist mit den Variablen X1, Y1 und X2, Y2 definiert und wird mit dem CONSOLE-Befehl aufgerufen (Zeile 290 - 340). Ab Zeile 560 wird der alte Inhalt wieder hergestellt. Alles, was dazwischen steht, ist die Ein- und Ausgabe, die man selber gestalten kann.

filelist 700 (!):

Das Programm ist für den MZ-700 mit dem Disketten-BASIC von SHARP geschrieben und wird so wie es abgedruckt ist, auf dem 800er nicht laufen. Soweit ich weiß, müssen dazu alle Befehle, die mit Farbe zu tun haben, geändert werden.

Das Programm listet den Inhalt einer in BASIC erstellten Datei, wobei BSD und BRD Dateien zulässig sind. Auf Wunsch ist auch ein Ausdruck auf dem Drucker möglich. Sollten mehr als 999 Datensätze in einer Datei vorhanden sein, muß die USING-Anweisung in den Zeilen 390 und 510 entsprechend geändert werden.

Matthias Großmann,
Stauffenburg 10, 3370 Seesen 16