MZ 700/800

ANWENDERMAGAZIN

Sharp MZ 700/800 Anwenderclub

Nr.:7/8 1987/88

Software zu Clubpreisen

Listings

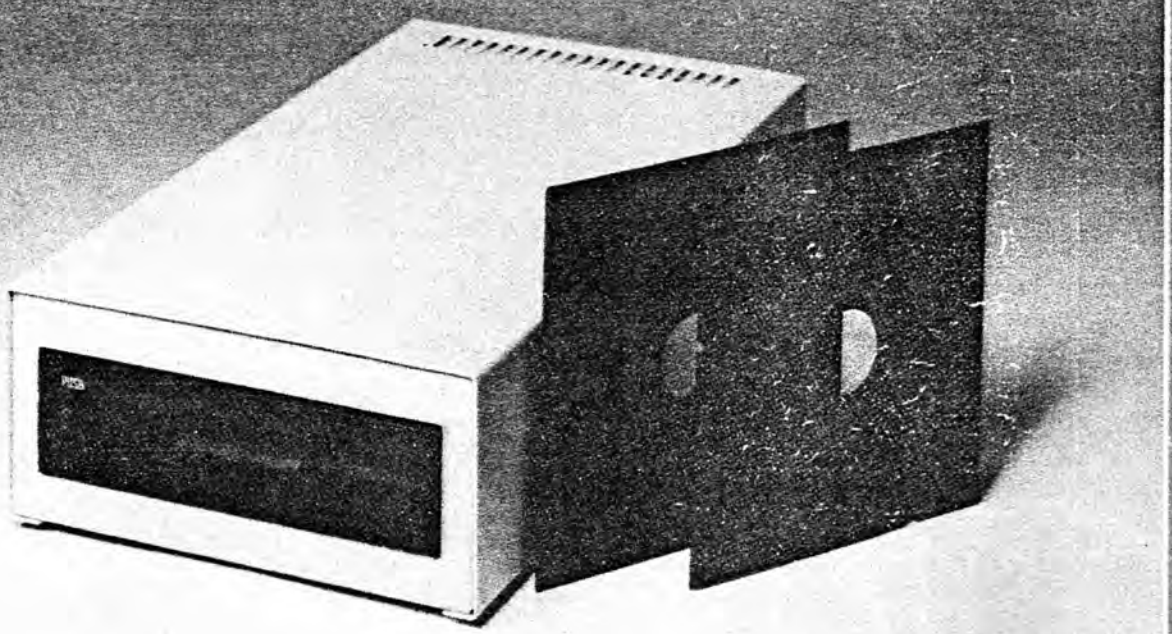
Tips und Tricks
MZ 700-800

POWER-DRIVE.

Die neue Floppy Station!
The new Floppy System!

NEU!
Jetzt auch 3.5" Laufwerke super preisgünstig.

SFD 800 - 320 KB
5 1/4" FLOPPY STATION FÜR DEN SHARP MZ 800
5 1/4" INCH FLOPPY SYSTEM FOR THE SHARP MZ 800



K & P

KERSTEN & PARTNER
DATENSYSTEME GMBH

Peliserkerstr.86 5100 AACHEN 0241/ 533108

Verehrter MZ 700/800 Freund!

Wir freuen uns, Ihnen die siebte Ausgabe des Anwendermagazins vorstellen zu dürfen. Zu Beginn daher einige Anmerkungen.
-Zuerst möchten wir die vielen neuen Mitglieder begrüßen, die seit dem Massenverkauf des MZ 800 unserem Club beigetreten sind. Viele werden sich vielleicht über das Magazin freuen, ohne zu wissen wie es entsteht. Wir möchten daher darauf hinweisen, daß wir im wesentlichen nur Beiträge von Mitliedern, die uns mit der obligatorischen Erlaubnis zur Veröffentlichung zugesandt wurden, zusammenstellen und drucken lassen. Es muß sich bei den Beiträgen keineswegs um professionelle Programme oder Hardwarebeschreibungen handeln. Wir würden uns über eine etwas regere Aktivität sehr freuen. Als Anregung: Es wurde z.B. ein Beitrag über Modems gewünscht (Platine, Plan, Stückliste, Software).
-Wir stellen auch eine rege Nachfrage nach der Clubdiskette fest, die in nächster Zeit als vorläufige Version erscheinen wird. Wir haben zwar schon einige Programme, es könnte aber nicht schaden, wenn das Angebot noch größer würde (s. auch #5,#6).
-Aus aktuellem Anlaß möchten wir darauf hinweisen, daß im Moment QD's (2.8") schwer zu bekommen sind. In Anbetracht der Speicherkapazität dieser Disks sollte man sich den Neukauf eines QD-Laufwerkes genau überlegen.
-An dieser Stelle vielen Dank an Herrn Ehm, der uns sein "Maschinensprache- Handbuch" zum Abdruck zur Verfügung gestellt hat. Die Fortsetzung folgt im nächsten Anwendermagazin.
-Als letztes: es gibt neue Preise bei der Hard- und Software. (Wir haben darauf keinen Einfluß, da unsere EK Preise gleich unseren VK Preisen sind.)

Mit freundlichen Grüßen

Ihre Redaktion

==========

Herausgeber: MZ 700/800 Anwenderclub
c/o Germar Nikol
Peliserkerstr. 86
D-5100 Aachen

Tel.: 0241/533105 nur Mo. 17.00-19.00

==========

Maschinensprache-Handbuch

MZ-800

	Inhalt	Seite
1.	Der MZ-800 und seine Maschinensprache	3
1.1	Einführung	3
1.2	Daten und Befehle Datenformate, Register, Adressierungsarten	4
1.3	Befehlsspezifikationen. Ergebnisanzeigen, Laden und Speichern, Strichrechnung, Vergleichen, Boolsche Algebra, Bitverarbeitung, Schieben, Springen, Schalleuregister Blocktransfer, Externbefehle	7
2.	Entwicklung von Assemblerprogrammen	17
2.1	Maschinensprache	17
2.2	Assembliervorgang	19
2.3	Testhilfe und Debugger	21
3.	Periphere Bausteine im MZ-800	23
3.1	Parallel-Interface 8255 (PPI). Initialisieren des PPI, Tastatur, Kassette	23
3.2	Timerbaustein 8253. Initialisieren, Uhr, Alarme, Tonerzeugung.	30
3.3	Z80A-PIO. Initialisieren des PIO, Drucker rücksetzen, Zeichen ausgeben	34
3.4	Floppy-Disk-Controller 2791 Register, Befehle/Status ,Sektoren lesen/schreiben	37
3.5	RAM-Erweiterung MZ-1R18	43
3.6	Grafik-Erweiterung MZ-1R25	44
4.	Speicher-Management im MZ-800	48
4.1	Speicherumschaltung	48
4.2	Bildspeicher-Steuerung Betriebsarten, Scrollen, Palette, Lese/Schreib-Formatregister, Hintergrundfarbe	48 48
4.3	Zusammenfassung MZ-800 Portadressen	57
5.	Maschinensprache-Unterprogrammbibliothek	58
5.1	16-Bit Wort-Arithmetik	58
5.2	24-Bit Fließpunkt-Arithmetik	59
5.3	Wissenschaftliche Funktionen	62
5.4	Konvertieren von Zahlen	67

Anhang 1 Assemblerprotokoll der Programmbibliothek.
Anhang 2 Assemblerprotokoll des Z80-Befehlssatzes.

Literaturhinweise

Rodnay Zaks
Programmierung des Z80
1981, Sybex-Verlag GmbH Dusseldorf ISBN 3-88745-006-X

James W. Coffron
Z80 Anwendungen
1984, Sybex-Verlag GmbH Dusseldorf ISBN 3-88745-037-X

The TTL Data Book
1984, Texas Instruments ISBN 3-880078-017-4

MZ-700 Service Manual
1983, Sharp Corporation Yamatokiriyama, Japan

MZ-800 Service Manual
1984, Sharp Corporation Yamatokiriyama, Japan

MZ-800 Technical Reference Manual
1985, Sharp Corporation Osako, Japan

Vollrath Dirksen, Wolfgang Fiehig
Einarbeitung in die Betriebssysteme CP/M und OR6 B, um ein Diskettenlaufwerk softwaremäßig anzusteuern.
1983, Diplomarbeit ,FH Wilhelmshaven

Ulrich Ehm
MZ-700/800 Maschinensprache-Handbuch
1985, Fischl GmbH, Berlin ISBN 3-924 327-07-06

1. Der MZ-700/800 und seine Maschinensprache

1.1 Einführung

Die Vielfalt der heute eingesetzten Mikroprozessoren mit ihren verschiedenen Befehlssätzen ist etwa vergleichbar mit der Entwicklung trivialer Bauelemente wie Schrauben und Muttern, die viele Jahre gebraucht haben, um sie untereinander weltweit einigermaßen kompatibel zu machen. Damit soll gesagt sein, daß auch für geübte Assembler-Programmierer die Einarbeitung in ein neues Konzept zeitraubender sein kann als z.B. das Umsteigen in höheren Sprachen wie BASIC auf PASCAL. Wissenschaftliche Berechnungen in Maschinenssprache sind auf Mikrocomputern nur möglich, wenn eine Fließpunktarithmetik im Betriebssystem implementiert ist, denn bis auf wenige Ausnahmen können Mikrocomputer heute noch keine Fließpunktrechnungen ausführen.

Beim Umgang mit höheren Programmiersprachen wird die Flexibilität eines Computersystems aber vorwiegend nicht durch das Hardware-Konzept, sondern vielmehr durch die Intelligenz des eingesetzten Interpreters oder Compilers bestimmt. Programmieren in Maschinensprache findet seine Grenzen alleine in der Hardware-Konfiguration. Daher lassen sich viele Probleme z.B. der Meß-Steuer-Regeltechnik oder auch Textverarbeitung oft effektiver in Maschinensprache gestalten. Bei Echtzeit-Datenverarbeitung, d.h. der Computer kommuniziert mit einem Prozeß, ist ein Maschinensprache-Programm dann unumgänglich, wenn das eingesetzte höhere Programmsystem zeitlich dem Prozeß nachläuft. Die Leistung von Maschinensprache ist grundsätzlich höher als die aller anderen Sprachen, die gleiche Arbeit wird in kürzerer Zeit verrichtet, denn auch ein Compiler wird im allgemeinen Befehlssequenzen nicht so optimal absetzten wie ein versierter Maschinensprache-Programmierer.
Der zur Zeit noch sehr häufig eingesetzte Mikroprozessor ist der Z80 von ZILOG, der auch im MZ eingebaut und voll kompatibel ist mit dem nun über 12 Jahre alten Modell 8080 von Intel.
Der Z80 besitzt Bytestruktur, d.h. die die kleinste adressierbare Einheit ist ein Byte. Solch eine Struktur ist für kleinere Aufgaben der Meß-Steuer-Regeltechnik sowie Textverarbeitung ausreichend. Aufgaben der Steuerungstechnik im allgemeinen ausreichend.

Mikroprozessoren realisieren die Logik zum Lesen, Dekodieren und Ausführen von Maschinenbefehlen auf einem Chip. Die gleiche Logik brauchte vor etwa 20 Jahren größenordnungsmäßig die 1000-fache Leistung, den 100000-fachen Raum und war etwa 2000-fach teurer.

1.2 Daten und Befehle

Jede Zentraleinheit ZE (Central-Prozessor-Unit CPU), egal ob im Großrechner oder Taschencomputer, besteht aus drei wesentlichen Bestandteilen.

Steuerwerk
Rechenwerk
Kanalwerk

Das Steuerwerk ist gewissermaßen das Nervensystem, welches den zu verarbeitenden Datenfluß dirigiert.
Bestandteile des Steuerwerkes sind

Taktsystem
Befehlszähler BZ (Programm Counter PC)
Befehlsdekodierregister BR (Instruction Register IR)

Das Taktsystem teilt den Anlagenteilen der ZE die notwendigen Zustände zum fristgerechten Ablauf zu. Je schneller das Taktsystem arbeitet, desto zügiger werden die Daten verarbeitet. Die erreichbare Grenzgeschwindigkeit wird durch die verwendete Technologie und den Aufbau des Computers bestimmt.
Der Z80 ist mit den Grenzfrequenzen
Z80 bis 2 MHz
Z80A bis 4 MHz
Z80B bis 8 MHz
erhältlich. Im MZ-700/800 ist ein Z80A eingebaut, der mit 3.5469 MHz betrieben wird.
Ab c.a. 6 MHz wird der Aufbau kritisch, nur wenige Personalcomputer-Hersteller beherrschen diesen noch zuverlässig. Außerdem steigt der Preis der Bauteile beträchtlich an.
Bei Großrechnern sind Taktfrequenzen bis 24 MHz üblich und Standart 64-Bit-Parallelverarbeitung, aber die Geschwindigkeit hat auch ihren Preis.
Ein Programm besteht immer aus Befehlen und Daten. Allgemein kann man sagen, in einem Befehl ist verschlüsselt, was mit den Daten geschehen soll. Der Befehlszähler BZ enthält immer die Adresse des nächsten auszuführenden Befehles, der zu Beginn einer Befehlsausführung aus dem durch BZ fixierten Speicherplatz in das Befehlsdekodierregister BR gelesen wird.
Je nach Länge des Befehles sind beim Z80 bis zu vier Befehl-Lese-Zustände (Teich-Zyklen) notwendig. Nach jedem Befehl-Lese-Zustand wird BZ inkrementiert, d.h. um Eins erhöht., so daß BZ nach Lesen des ganzen Befehles auf den nächsten im Speicher stehenden Befehl zeigt (BZ ist ein Zeigerregister).
Im BR wird der im Binärformat vorliegende Befehl dekodiert und anschließend ausgeführt.
Ein Befehl besteht im allgemeinen aus Operator und Operanden. Z80-Befehle können keinen, einen oder zwei Operanden haben. Im Operator ist die Befehlsart wie Laden, Arithmetik u.s.w. verschlüsselt und wie der Operand zu interpretieren ist.
Operanden können Daten, Registernamen oder Speicherplatzadressen sein. Daten können wiederum Zeichen, Zahlen, Meßwerte, Adressen oder allgemeine Binärmuster (Masken) sein.

Datenformate

Der Z80 kennt Byte- und Wortverarbeitung. Für Zahlen wird die bei der Assemblierung übliche Notation verwendet:

```
Byte: n =              10     Dezimalzahl
      n =             0AH     Sedezimalzahl
      n =       00001010B     Binärzahl
Wort:nn =              10     Dezimalzahl
     nn =            000AH     Sedezimalzahl
     nn =0000000000001010B     Binärzahl
```

Die Bitstellen einer Zahl werden nach ihrer Wertigkeit bezeichnet.

```
Byte: D7  ... D0
Wort: D15 ... D0
```

D7 b.z.w D15 sind das MSB =Most Significant Bit
D0 ist das LSB =Least Significant Bit

Das MSB hat bei arithmetischen Operationen die Bedeutung eines Vorzeichen-Bit.

MSB =0 positive Zahl (die Zahl 0 heißt auch positiv)
MSB =1 negative Zahl

Mit den übrigen Binärstellen, also den Betragsstellen, wird bei positiven Zahlen der Absolutbetrag der Zahl entsprechend der dualen Wertigkeit, bei negativen Zahlen das Zweierkomplement der Zahl dargestellt. Eine so definierte Zahl umfaßt den Bereich

```
Byte:          10000000B ... n ...          01000000B
                    -128  ... n                 +127
Wort: 1000000000000000B ... nn ... 0111111111111111B
                  -32768  ... nn ...            +32767
```

Die Bildung des Zweierkomplementes herüber und hinüber geschieht am einfachsten durch folgende Regel:

Lasse die niederwertigen Bitstellen einschließlich der ersten Eins von rechts unverändert.
Invertiere alle links davon stehenden Bitstellen.

Beispiele im Byte-Format:

```
+64  =01000000B          +1  =00000001B
-64  =11000000B          -1  =11111111B
```

Die jeweilige negative Kleinstzahl

```
Byte:   -128 =         10000000B =  80H
Wort  -32768 = 1000000000000000B =8000H
```

ist nicht komplementierbar, d.h. sie besitzt keinen entsprechenden positiven Wert.

Bei Stückzahl-Rechnungen (z.B. Adressrechnungen) gibt es nur positive Zahlen. Dann ist der Zahlenbereich:

```
Byte:0 ...   255 =          00000000B ...          11111111B
Wort:0 ... 65536 = 0000000000000000B ... 1111111111111111B
```

Wird ein Wort als Speicherplatz-Adresse interpretiert, so läßt sich mit einem Wort jeder Platz des 64k-Speicherraumes fixieren.

Register

Im schon erwähnten Rechenwerk erfolgt die eigentliche Datenverarbeitung, nachdem der auszuführende Befehl vom Speicher in das BR gelesen und dekodiert wurde.

Der Z80 besitzt ein vollwertiges Datenregister, den byteorientierten Akkumulator A.

Der Akkumulator kann:

- Laden aus Registern und Speicherplätzen
- Speichern
- Strichrechnung, Inkrementieren, Dekrementieren
- Boolsche Algebra
- Einzelbit-Verarbeitung
- Schieben und Rollieren des Akkuinhaltes rechts oder links um Eins.

Die byteorientierten Datenregister B C D E H L können:

- Laden aus Registern
- Inkrementieren, Dekrementieren
- Einzelbit-Verarbeitung
- Schieben und Rollieren des Registerinhaltes rechts oder links um Eins.

Die Datenregister B C D E H L sind als Registerpaare BC DE HL zusammen mit den Indexregistern IX IY an den Z80-Adreßbus angeschlossen, so daß im Wortformat aus dem Speicher geladen oder dorthin gespeichert werden kann.

In den Registerpaaren HL IX IY ist außerdem Strichrechnung möglich.

Wie sind nun die Operanden eines Befehles zu interpretieren ?

Der Z80 kennt vier Adressierungsarten.

Unmittelbare Adressierung eines Datums

Das Datum ist der Operand selbst.

Beispiele:

```
LD A,n      Lade Akkumulator A mit Konstante n
LD HL,nn    Lade Registerpaar HL mit Konstante nn
            Das höherwertige Byte steht in H
            Das niederwertige Byte steht in L.
```

LD ist der Operator, A, HL sowie die Zahlen n, nn sind die Operanden.

Der Operator sagt aus, was mit den Operanden geschehen soll.

Quelle ist eine Konstante, Senke ist ein Register.

Das unmittelbare Laden eines Speicherplatzes ist beim Z80 nicht möglich.

Direkte Adressierung eines Datums

Im Operandenteil steht die Speicherplatz-Adresse eines Datums. Das Datum selbst kann Variable sein, wenn eine RAM-Adresse vorliegt.

Beispiele:

```
LD A,(adr)    Lade Akkumulator A mit Inhalt des Speicher-
              platzes adr
LD (adr),A    Speichere Inhalt des Akkumulators A nach
              Speicherplatz adr
LD HL,(adr)   lade Registerpaar HL mit Wort aus adr.
LD (adr),HL   Speichere Inhalt des Wortr-Registerpaares HL
              nach adr
```

Worte belegen zwei Speicherplätze. Es ist wichtig, zu wissen, daß der Z80 immer das niederwertige Byte zuerst speichert oder lädt.

Indirekte Adressierung eines Datums

In einem Registerpaar steht die Speicherplatz-Adresse, eines Datums im Byte-Format. Solch ein Registerpaar wird dann auch Zeiger genannt.

Beispiele:

```
LD A,(HL)     Lade Akkumulator A mit dem Inhalt des Speicher-
              platzes, dessen Adresse im Zeiger-Registerpaar
              HL steht.
LD (HL),A     Speichere den Inhalt des Akkumulators A in den
              den Speicherplatz, dessen Adresse im Zeiger-
              registerpaar HL steht.
```

Indirekt-indizierte Adressierung eines Datums

Die 16-Bit Idexregister IX, IY ermöglichen relative Adressierung zum Registerinhalt.
Beispiele:

LD A,(IX+d)	Lade Akkumulator A mit dem Inhalt des Speicherplatzes, dessen Adresse sich aus dem Inhalt des Zeigerregisters IX additiv mit der vorzeichenbehafteten Distanz -128 ... d +127 berechnet.
LD (IX+d),A	Speichere Inhalt des Akkumulators A in den Speicherplatz, wie eben beschrieben.
LD (IX+d),n	Speichere Konstante n in den Speicherplatz, wie eben beschrieben.

Leider ist bei dieser nicht sehr glücklichen Syntax gemäß ZILOG eine Unterscheidung zwischen direkter und indirekter Adressierung nicht ohne weiteres ersichtlich. Bei Anfängern führt dieser Umstand bisweilen zu Programmierfehlern.

1.3 Befehlsspezifikationen

Die hier aufgezeigten Beispiele sollen dem Einsteiger einige grundlegende Kenntnisse der Z80-Befehlsstruktur vermitteln.
Einen vollständigen assemblierten Befehlssatz als Arbeitsgrundlage enthält Anhang 2.

Ergebnisanzeigen

Von besonderer Wichtigkeit für den dynamischen Ablauf von Befehlsfolgen sind die Ergebnisanzeigen von Strichrechnungen, Vergleichsbefehlen, Boolscher Algebra, Bitbefehlen und Schiebeoperationen. Diese Ergebnisanzeigen werden im Programmzustandsregister F gespeichert.
Wichtige Ergebnisanzeigen sind:

Art der Anzeige	Zustand	Ergebnis	Kennzeichen
Vorzeichen	S =0 S =1	Plus oder Null Minus	P M
Null-Anzeige	Z =0 Z =1	nicht Null Null	NZ Z
Überlauf	V =0 V =1	kein Überlauf Überlauf	PO PE
Übertrag	C =0 C =1	kein Übertrag Übertrag	NC C

Ein Überlauf V=1 kann bei arithmetischen Operationen wie ADC SBC auftreten, nämlich wenn der Zahlenbereich des Zweierkomplementes erreicht oder überschritten wird.
Beispiel:

```
 +64   01000000
 +64   01000000
-----  --------
-128   10000000
```

Physikalisch wird im Z80 das Übertragsbit zum Vorzeichenbit mit dem externen Übertrag C verglichen. Bei Ungleichheit wird das Überlaufbit gesetzt. Dann hat ein unerlaubter Vorzeichenwechsel stattgefunden.
Bei der Operation

```
 +64   01000000
 -64   11000000
-----  ---------
   0  100000000
```

ist das Übertragsbit zum Vorzeichenbit zwar auch gesetzt, aber ebenfalls der externe Übertrag, so daß hier das Überlaufbit nicht gesetzt wird, es tritt kein unerlaubter Vorzeichenwechsel auf.

Bei Boolscher Algebra AND OR XOR hat der Überlauf die Bedeutung einer Quersummenergänzung über das Rechenergebnis.
P =V =1 Quersumme ist ungerade, Parität P =1
P =V =0 Quersumme ist gerade, Parität P =0

Laden und Speichern

wurde schon in Kap. 1.1.2 angesprochen. Es ist wichtig zu wissen, daß diese Befehle nicht das Programmzustandsregister beeinflussen. Soll z.B. der Inhalt eines Speicherplatzes auf Null untersucht werden, muß nach dem Laden eine anzeigensetzende Operation folgen, die den Wert selbst nicht verändert.

```
LD A,(adr)        Lade A mit Inhalt von adr
OR A              Odere A mit A
```

oder im Wortformat

```
LD HL,(adr)       Lade HL mit Inhalt ab adr
LD A,H            Lade A mit H
OR L              Odere A mit L
```

Arithmetische Operationen

Es sind Strichrechnungen im Byte- und Wortformat möglich. Jedoch kann nicht mit Speicherplätzen direkt operiert werden.
Die Ergebnisse von arithmetischen Operationen ADD ADC SUB SBC im Byteformat beinflussen die Vorzeichen- und Null-Anzeige, den Übertrag sowie den Überlauf des Programmzustandsregisters F.
Beispiele für Strichrechnungen im Byteformat:

ADD A,n	Addiere zu A die Konstante n
SUB A,r	Subtrahiere von A den Inhalt des Registers r = A B C D E H L
ADC A,(HL)	Addiere zu A den Inhalt des Speicherplatzes plätzes, dessen Adresse im Registerpaar HL steht, und zusätzlich den Übertrag C der der letzten anzeigensetzenden Operation
SBC A,(IX+d)	Subtrahiere von A den Inhalt des Speicherplatzes, dessen Adresse sich aus dem Inhalt des Registerpaares IX additiv mit der Dis-Distanz d berechnet, und zusätzlich den Übertrag der letzten anzeigensetzenden Operation

Weiterhin ist Strichrechnung im Wortformat möglich.
Beispiele:

ADD HL,ss	Addiere zum Inhalt des Registerpaares HL den Inhalt des Registerpaares ss =BC DE HL oder SP
ADD IX,pp	Addiere zum Inhalt des Registerpaares IX den Inhalt des Registerpaares pp =BC DE IX oder SP
SBC IY,rr	Subtrahiere vom Inhalt des Registerpaares IY den Inhalt des Registerpaares rr =BC DE IY oder SP, und zusätzlich den Übertrag der letzten anzeigensetzenden Operation.

Um zwei ab adr1 und adr2 stehende Worte zu addieren, ist etwa folgende Befehlsfolge abzugeben:

```
LD HL,(adr1)      Lade HL mit Inhalt ab adr1
LD BC,(adr2)      Lade BC mit Inhalt ab adr2
ADD HL,BC         Addiere zum Inhalt von HL den Inhalt von BC
```

Zu beachten ist, daß der Z80 grundsätzlich das niederwertige Byte zuerst lädt oder speichert. Ein Wort im Speicher steht also immer verkehrt herum.

Sollen 2 Speicherworte, deren Adressen in adr1 und adr2 stehen, addiert werden, ist etwa folgende Befehlsfolge abzugeben.

```
LD IX,(adr1)      Lade Adresse des 1.Summanden nach IX
LD IY,(adr2)      Lade Adresse des 2.Summanden nach IY
LD L,(IX+0)       Lade niederwertiges Byte 1.Summanden nach L
LD H,(IX+1)       Lade hoherwertiges Byte 1.Summanden  nach H
LD C,(IY+0)       Lade niederwertiges Byte 2.Summanden nach C
LD B,(IY+1)       Lade hoherwertiges Byte 2.Summanden  nach B
ADD HL,BC           Addiere HL mit BC
```

Hier liegt eine indirekte Speicheraddition vor. Bei Großrechnern ist hierzu in der Regel nur ein Befehl notwendig.
Die folgende Befehlsfolge simuliert eine 32-Bit Addition.
In adr1, adr2 seien die beiden Summanden gemäß Z1L0S-konfiguration abgelegt (tiefstes Byte zuerst).

```
LD IX,adr1        Lade Adresse des 1.Summanden nach IX
LD IY,adr2        Lade Adresse des 2.Summanden nach IY
LD A,(IX+0)
ADD A,(IY+0)
LD E,A            In E steht tiefstes Ergebnisbyte.
LD A,(IX+1)
ADC A,(IY+1)      Addieren mit Übertrag
LD D,A            In D steht zweites Ergebnisbyte
LD A,(IX+2)
ADC A,(IY+2)
LD C,A            In C steht drittes Ergebnisbyte
LD A,(IX+3)
ADC A,(IY+3
LD B,A            In B steht höchstes Ergebnisbyte
```

Das duale Ergebnis ist in den Registerpaaren BC DE abgelegt. Der Befehl ADC addiert gegenüber ADD zusätzlich das Übertragsbit der letzten Rechnung. Zu beachten ist, daß Laden und Speichern nicht die Ergebnisanzeigen beinflussen.
Die Inhalte von Registern oder Speicherplätzen können um Eins erhöht oder erniedrigt werden.
Beispiele:

```
INC r             Erhöhe Register r = A B C D E H L um Eins.
                  Das Ergebnis beeinflußt die Vorzeichen-
                  und Null-Anzeige sowie den Überlauf.
DEC (HL)          Erniedrige den Inhalt des Speicherplatzes,
                  dessen Adresse im Registerpaar HL steht, um
                  Eins.
                  Das Ergebnis beeinflußt die Vorzeichen- und
                  Null -anzeige sowie den Überlauf.
```

Wortorientiertes Erhöhen oder Erniedrigen setzt keine Ergebnisanzeigen:

```
DEC HL            Erniedrige Registerpaar HL um Eins
INC IY            Erhöhe Registerpaar IY um Eins
```


Vergleichsbefehle

sind nur im Akkumulator A moglich. Es wird subtrahiert, jedoch kein Ergebnis abgelegt. Wohl aber werden die Vorzeichen- und Nullanzeige sowie der Überlauf und Übertrag vom Ergebnis beeinflußt.
Beispiele:

```
CP n              Vergleiche Akkumulator A mit Konstante n
CP r              Vergleiche Akkumulator A mit Register
                  r =A B C D E H L
CP (IX+d)         Vergleiche Akkumulator A mit Inhalt des
                  Speicherplatzes, dessen Adresse sich
                  dem Inhalt des Registerpaares IX und
                  der Distanz d additiv berechnet
```

Boolsche Algebra

ist nur im Akkumulator A moglich. Es wird die jeweilige Bitstelle des Akkumulators mit der entsprechenden Bitstelle des Operanden verknupft, also 8 logische Verknupfungen mit je 2 Eingangen durchgefuhrt.
Das Ergebnis beeinflußt die Vorzeichen- und Null-Anzeige sowie die Parität P/V (siehe Ergebnisanzeigen).
Es ist wichtig zu wissen, daß logische Operationen grundsatzlich den Überlrag loschen.
Beispiele:

```
AND n             'Unde' Akkumulator A mit Konstante n
                  n =0 loscht den Akku, weil UND-Bedingung
                  nie erfullt
OR (HL)           'Odere' Akkumulator A mit dem Inhalt des
                  Speicherplatzes, dessen Adresse im
                  Registerpaar HL steht.
XOR r             'Exklusiv-Oder' des Akkumulators A mit
                  einem Register r =A B C D E H L
                  r =A löscht A, weil jede Bitstelle mit
                  sich selbst verglichen wird.
```

Einzelbit-Verarbeitung

Es konnen einzelne Bit in Registern oder Speicherplatzen gesetzt, geloscht oder getestet werden.
Bit setzen oder loschen beeinflußt keine Anzeigen.
Das Ergebnis von Testbefehlen beeinflußt die Null-Anzeige.
Die Bitnummer b (0...7) bezeichnet die duale Wertigkeit der Bitstelle innerhalb eines Byte. Die Bitnummer selbst ist leider immer eine Konstante, d.h. sie steht unmittelbar in der Befehlsfolge und kann kaum lediglich durch eine andere Befehlsfolge verändert werden. Dies freilich nur, wenn die erste Befehlsfolge im ROM abgelegt ist.
Beispiele:

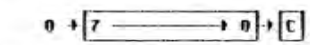
```
SET b,r           Setze Bit b im Register r =A B C D E H L
RES b,(HL)        Losche Bit b des Speicherplatzes, dessen
                  Adresse im Registerpaar HL steht
BIT b,(IY+d)      Teste Bit b des Speicherplatzes, dessen
                  Adresse sich aus dem Inhalt des Register-
                  paares IY und der Distanz d additiv
                  berechnet
```

Schieben

Ein Register oder eine Speicherzelle wird zusammen mit dem Übertrag als Schieberegister aufgefaßt.
Schieben nach links um Eins heißt Multiplizieren mit Faktor 2.
Schieben nach rechts um Eins heißt Dividieren durch Faktor 2.
Die Vorzeichen- und Null-Anzeige sowie der Übertrag und Parität P werden vom Ergebnis beeinflußt.
Beispiele:

SLA r — Schiebe Registerinhalt r =A B C D E H L links um Eins. Das MSB wandert in den Übertrag. Von rechts rückt eine Null in das LSB

SRL (HL) — Schiebe den Inhalt des Speicherplatzes, dessen Adresse im Registerpaar HL steht, rechts um Eins. Das LSB wandert in den Übertrag. Von links rückt eine Null in das MSB.
Das Datum wird als Binärmuster aufgefaßt.

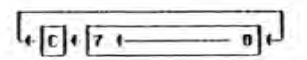


SRA (IX+d) — Schiebe den Inhalt des Speicherplatzes, dessen Adresse sich aus dem Inhalt des Registerpaares IX additiv mit der Distanz d berechnet,rechts arithmetisch um Eins. Das LSB wandert in den Übertrag.
Von links rückt eine Null in das MSB,wenn die Zahl positiv, und eine Eins, wenn die Zahl negativ war.
Das Datum wird als vorzeichenbehaftete Zahl im Zweierkomplement aufgefaßt.

Rotieren

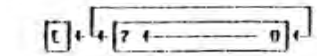
Ein Register oder eine Speicherzelle wird zusammen mit dem Übertrag als Ring ausgefaßt.
Beispiele

RL r — Rotiere Registerinhalt r =A B C D E H L links durch den Übertrag um Eins

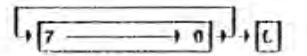


RR (HL) — Rotiere den Inhalt des Speicherplatzes, dessen Adresse im Registerpaar HL steht, rechts durch den Übertrag um Eins.

RLC r — Rotiere Registerinhalt R = A B C D E H L links um Eins.
Das MSB gelangt außerdem in den Übertrag.



RRC (IX+d) — Rotiere den Inhalt des Speicherplatzes, dessen Adresse sich aus dem Inhalt des Registerpaares IX additiv mit der Distanz d berechnet, rechts um Eins.
Das LSB gelangt außerdem in den Übertrag.

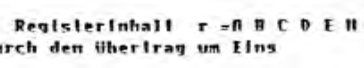


Im Register B möge eine Zahl n=0...7 stehen. Mit einer Befehlsfolge soll im Register A das n.te Bit gesetzt sein.

```
     INC B        B =B+1
     XOR A        A =0
     SCF          Übertrag setzen
JUMP:RLA          A rotieren durch Übertrag
     DJNZ JUMP    B =B-1 und Sprung, wenn B ungleich Null
```

Schieben von Registerpaaren

Schieben von Registerpaaren ist nicht möglich. Mit dem kleinen Trick
ADD HL,HL
ADD IX,IX
ADD IY,IY
können diese Registerpaare aber doch bequem links um Eins geschoben werden.
Die Befehlsfolge
SLA D
RL E
hätte die Wirkung wie
ADD DE,DE
,welche aber nicht implementiert ist.
Rechtsschieben geht nur byteweise.
SRL H
RR L
Eine der insgesamt 200 Linien des Video-RAM (VRAM) im MZ-800 besteht bei 640 Punkten aus 80 Speicherplätzen mit 8Bx8 Punkten. Der Bildmöge als x,y-Koordinatensystem aufgefaßt werden (siehe auch Kap. 4.1)



Das Registerpaar HL enthalte die y-Koordinate, das DE-Register die x-Koordinate. Eine Befehlsfolge soll den gewählten Bildpunkt setzen oder löschen.

```
     ADD HL,HL      HL= 2 y
     ADD HL,HL      HL= 4 y
     ADD HL,HL      HL= 8 y
     ADD HL,HL      HL= 16 y
     LD B,H
     LD C,L         BC= 16 y
     ADD HL,HL      HL= 32 y
     ADD HL,HL      HL= 64 y
     ADD HL,BC      HL= 80 y
     LD BC,8000H    BC= VRAM-Anfangsadresse
     ADD HL,BC      HL= absolute Linienanfangsadresse
     LD A,E
```

```
      AND 00000111B  A =Bit. Nr.
      LD B,3
SPR1: SRL D
      RR E                x/2  x/4  x/8
      DJNZ SPR1           B= B-1  und Sprung, wenn B ungleich Null
      ADD HL,DE           HL= absolute VRAM-Adresse
      LD B,A
      INC B               B= Bit.Nr. +1
      XOR A               A =0
      SCF                 Übertrag setzen
SPR2: RLA                 A= n.tes Bit generieren
      DJNZ SPR2           B =B-1  und Sprung, wenn B ungleich Null
      OUT (E0H),A         VRAM einschalten
adr : OR (HL)             A mit Information in VRAM mischen
      LD (HL),A           und rückspeichern
      OUT (E1H),A         VRAM ausschalten
Soll das adressierte Bit gelöscht werden, so muß man ab adr  abgeben
adr : CPL                 Ausblendmaske generieren
      AND (HL)            n.tes Bit ausblenden
      LD (HL),A           und Rest rückspeichern
      OUT (E1H),A         VRAM ausschalten
```

Sprungbefehle

Springen bedeutet Umladen des Befehlszählers BZ durch Befehle. Aus den im Anzeigenregister F gespeicherten Ergebnisanzeigen können auch Sprungverzweigungen realisiert werden.

```
LD HL,adr       Lade HL mit adr
BIT 3,(HL)      Nullanzeige Z= 1, wenn Bit D3 in Zelle adr =0
JP Z,WEIT       BZ= WEIT, wenn D3 =0
```

Von Konditionen abhängige Sprünge sind bedingt.
Die Kondition in F wird bei Abarbeiten des Befehles nicht verändert.
Um Speicherplatz zu sparen, kann man auch relative Sprünge verwenden.

```
adr: JR Z,e     BZ= adr+e, wenn Z=1
```

Es werden nur 2 statt 3 Byte wie bei absoluten Sprüngen benötigt.
Der Z80 addiert zum Momentanwert des BZ= adr+2 die relative ein Byte lange, vorzeichenbehaftete Distanz e-2.
Im Sedezimalkode des Befehles muß daher e-2 abgesetzt sein. Es ist darauf zu achten, daß der erlaubte Bereich

```
-126 ... e ... +129
```

eingehalten wird.
Diese etwas komplizierte Überprüfung löst ein Assembler mit einer einfachen Befehlsfolge. HL enthalte die Distanz e.

```
      DEC HL              e-1
      DEC HL              e-2
      ADD HL,HL           2 (e-2)
      LD A,H
      OR A
      JR Z,RICHTIG        wenn HL=   0...254      e=   0...127
      INC A
      JR Z,RICHTIG        wenn HL=-256... -2      e=-126... -1
FALSCH:
```

Mitunter läßt sich der dekrementierende Sprung

```
      DJNZ e          B= B-1 und Sprung wenn B ungleich Null
```

wie unter der Rubrik 'Rollieren' verwendet, vorteilhaft einsetzen.

Stapelzeiger

Der Z80 besitzt neben den erwähnten Registerpaaren noch ein spezielles Zeigerregister SP (Stapelzeiger), das immer auf ein Speicherwort zeigt und bei Start einer Befehlsfolge geladen werden kann.

```
LD SP,adr       Lade Stapelzeiger mit adr,unmittelbar
```

Über SP können nach dem Prizip "wer zuletzt kommt, geht zuerst" die Inhalte von Registerpaaren gespeichert und rückgeladen werden.

```
PUSH rr         Schalte Zeiger SP ein Wort tiefer und speich-
                ere den Inhalt von rr in das durch SP fixier-
                te Speicherwort
```

Nach Befehlsende zeigt SP auf das gespeicherte Wort. Der Speicherbereich (Stapel) wird nach unten hin aufgefüllt. SP ist also zu Beginn einer Befehlsfolge mit der Endadresse des Stapels zu laden.
Rückladen erfolgt mit

```
POP rr          Lade das durch SP fixierte Speicherwort nach
                rr und schalte Zeiger SP um ein Wort höher.
```

Nach Befehlsende zeigt SP auf das nächste noch nicht zurückgeholte Wort. Über SP ist also indirekte Speicherwort-Adressierung möglich.
Der Z80 kennt keine Befehle etwa der Art

```
LD IX,BC
```

Mit der Befehlsfolge

```
PUSH IX
POP BC
```

ist ein Umladen aber möglich. Der Inhalt von IX steht jetzt auch in BC.
Das F-Register ist zusammen mit A als Registerpaar AF definiert.
Nach der Befehlsfolge

```
PUSH AF
POP BC
LD A,C
```

kann mit dem Bild von F manipuliert und rückgeladen werden.

```
LD C,A
PUSH BC
POP AF
```

Der Übertrag läßt sich auch direkt beeinflussen

```
SCF             Setze Übertrag
CCF             Komplementiere Übertrag
```

Unterprogrammtechnik

Innerhalb eines Programmes wird möglicherweise öfter eine 32-Bit-Addition gebraucht. Dann empfiehlt sich die Unterprogrammtechnik. Versorgungsparameter dieses Unterprogrammes sind die Summandenadressen in IX und IY.

```
Aufrufende Befehlsfolge          Unterprogramm
     LD IX,adr1             adr: PUSH AF
     LD IY,adr2                  LD A,(IX+0)
     CALL adr                    ADD A,(IY+0)
fadr:.                           .
     .                           .
     .                           POP AF
                                 RET
```

Es tauchen die Befehle CALL, RET auf. Der Unterprogrammaufruf CALL adr rettet den Inhalt des BZ, also die Fortsetzadresse fadr des aufrufenden Programmes über SP in den Stapel und lädt anschließend BZ mit adr unmittelbar, d.h. der nächste Befehl wird aus adr gelesen.
Das Unterprogramm muß mit RET (Return) abgeschlossen werden. RET wirkt wie POP BZ. BZ wird also mit der Fortsetzadresse des aufrufenden Programmes rückgeladen, der nächste Befehl wird aus fadr gelesen. CALL adr kann beliebig oft in einem anderen Programm vorkommen. Es ist darauf zu achten, daß die Zahl der PUSH und POP innerhalb eines Unterprogrammes gleich ist, weil sonst BZ falsch

rückgeladen wird. CALL und RET können auch als bedingte Sprünge deklariert sein, z B.
CALL Z,adr Unterprogrammsprung, wenn Nullanzeige gesetzt.
RET NC Rücksprung, wenn kein Übertrag

Registertausch

Eine weitere Einrichtung des Z80 sind seine Schattenregister A' B' C' D' E' F' H' L'. Mit den Tauschbefehlen
EX AF,AF' Tausche AF mit Schatten A'F'
EXX Tausche BC DE HL mit Schatten B'C' D'E' H'L'
können die Inhalte der Registerpaare ausgetauscht werden. Allerdings sind Schattenregister keine Rechenregister und nur als Hintergrundspeicher zu verstehen.
Mit dem Befehl
EX DE,HL Tausche HL mit DE
kann die Information vertauscht werden. Um den Inhalt des Registers DE um Eins links zu schieben, kann man schreiben
EX DE,HL
ADD HL,HL
EX DE,HL

Blocktransfer

Zum Verschieben von Speicherblöcken dienen die Blocktransferbefehle. Sie verlangen Versorgungsparameter in den Registerpaaren
HL ... Quelle DE ... Senke BC ... Länge in Byte
Die Befehlsfolge
LD HL,anf1 Anfangsadresse der Quelle
LD DE,anf2 Anfangsadresse der Senke
LD BC,nn
LDIR
speichert nn Bytes ab adr1 nach adr2. HL und DE fungieren als aufwärtszählende Zeiger, BC als abwärtszählender Stückzähler.
LDIR läßt sich immer verwenden, wenn die Speicherbereiche nicht überlappen. Bei Überlappung ist zu unterscheiden, ob die Adresse der Quelle höher als die der Senke ist. Ist dies der Fall, kann obige Befehlsfolge beibehalten werden.
Ansonsten muß man
LD HL,end1 Endadresse der Quelle
LD DE,end2 Endadresse der Senke
LD BC,nn
LDDR

abgehen. HL und DE fungieren jetzt als abwärtszählende Zeiger.
Die Befehlsfolge
LD HL,43FFH
LD DE,4400H
LD BC,400H
LDDR
verschiebt einen 1-kByte langen Speicherblock 4000H-43FFH nach 4001H-4400H, also um ein Byte höher.
Die Befehlsfolge
LD HL,4001H
LD DE,4000H
LD BC,400H
LDIR
bringt ihn an die alte Stelle zurück.
Die Blockladebefehle lassen sich vorteilhaft zum Einfügen und Löschen von Zeichen oder Zeilen in Textverarbeitungssystemen verwenden.

Externbefehle

Der Informationsaustausch zu peripheren Geräten erfolgt durch Ein-Ausgabe -befehle.
IN A,(PORT)
Die ein Byte lange Portadresse PORT wird auf die untere Hälfte des Adreßbusses gelegt und adressiert über einen außerhalb des Z80 zu installierenden Adreßdekoder eine von insgesamt 256 möglichen externen Einheiten.
Die externe Einheit schaltet daraufhin seine Dateninformation auf den Datenbuß des Z80, der sie in das Register A übernimmt.
OUT (PORT),A
legt den Inhalt des Registers A auf den Datenbuß und die Portadresse PORT auf die untere Hälfte des Adreßbusses.
Ferner ist indirekte Portadressierung über das Register C möglich. C enthält die Portadresse.
IN r,(C)
OUT (C),r
r = A B C D E H L
Die indirekten Portbefehle legen außerdem den Inhalt des Registers B auf die obere Hälfte des Adreßbusses.
Ferner ist blockweiser Verkehr möglich.
OTIR
schaltet den Inhalt der Speicherzelle, deren Adresse im Registerpaar HL steht auf den Datenbuß und den Inhalt des Registerpaares BC als Externadresse auf den Adreßbus.
Anschließend wird HL inkrementiert und B dekrementiert.
Es ist darauf zu achten, daß B erst nach dem Inkrementieren auf den Adreßbus gelangt.
Der Befehl wird wiederholt, bis B =0.
Bei
OUTI
unterbleibt die Befehlswiederholung.
Die Nullanzeige Z =1 ist gesetzt, wenn B =0 nach Befehlsablauf.
OTDR
verhält sich wie OTIR, jedoch wird HL dekrementiert.
OUTD
verhält sich wie OUTI, jedoch wird HL dekrementiert.
INIR
schaltet den Inhalt des Registerpaares BC als Externadresse auf den Adreßbus und liest den Datenbus in die Speicherzelle, deren Adresse im Registerpaar HL steht.
Anschließend wird HL inkrementiert und B dekrementiert.
Es ist darauf zu achten, daß B erst nach dem Inkrementieren auf den Adreßbus gelangt.
Der Befehl wird wiederholt bis B =0.
Bei
INI
unterbleibt die Befehlswiederholung, sonst wie INIR.
Die Nullanzeige Z =1 ist gesetzt, wenn B =0 nach Befehlsablauf.
INDR
verhält sich wie INIR, jedoch wird HL dekrementiert.
IND
unterbleibt die Befehlswiederholung, sonst wie INDR.
In Kapitel 3 werden im Zusammenhang mit der Konfiguration des des MZ-700/800 zahlreiche Beispiele mit Extern-Befehlsfolgen aufgezeigt.

2. Entwicklung von Assemblerprogrammen

2.1 Maschinensprache

Die in Kap.1 verwendete Notation der Operatoren nennt man Mnemonic's, d.h. die binäre Verschlusselung des Operators wird einem mnemotechnischen Ausdruck wie LD, ADD, JP usw. gleichgesetzt. So ist die Befehlsfolge

```
LD A,5
ADD A,(HL)
```

für den Menschen sicherlich verständlicher als die binäre

```
00111110B  000001010B
10000110B
```

oder sedezimale

```
3E05H
86H
```

Verschlusselung. Aufgabe eines Assemblers ist es, mnemotechnische Ausdrucke in den Maschinenkode, also in den Binarkode der ZE umzuwandeln. Wahrend der Binarkode einer bestimmten ZE eindeutig zugeordnet sein muß, sind die Mnemonic's bei der Projektierung des Assemblers frei wahlbar. Die Intel-8080-Mnemonic's lauten für dieselben Befehle vollig anders und sind manchmal ein bißchen verwirrend. Allerdings hat auch die ZILOG-Z80-Notation ihre Tücken. Der Einsteiger erkennt oft nicht, ob Laden oder Speichern vorliegt oder welche Adressierungsart gemeint ist. Viele Befehlsgruppen haben ein festes Binarraster, z.B.

```
01xxxyyy   xxx =r   yyy =r'   lade r mit r'
```

Die beim 8080 nicht ausgenutzten Befehlskodes

```
CBH
DDH
EDH
FDH
```

wurden beim Z80 für Befehlsexpansion ausgenutzt.
DDH leitet eine IX-Operation ein
FDH leitet eine IY-Operation ein
Insgesamt ist aber kein systematischer Zusammenhang von Befehl und Dualkode erkennbar. Die Befehlsdekodierung vor der Befehlsausführung geschieht intern wohl über Dekodierhilfsmuster. Diese Tatsache erschwert die Entwicklung eines Assemblers. Bei Großrechnern ist dieser Zusammenhang im allgemeinen einfach und klar gegliedert.
Im Maschinenkode muß dann programmiert werden, wenn Befehlsfolgen mit einem BASIC-Interpreter generiert werden sollen. Das ist ein sehr umstandliches und fehlerfreundliches Verfahren und schreckt manchen BASIC-Programmierer ab, sich überhaupt mit Maschinensprache zu beschaftigen. BASIC sollte, wenn schon nicht symbolische Marken, doch Mnemonic's kennen. Allerdings wurde diese Einrichtung je nach Komfort 2-3 kByte zusatzlich erfordern.
Die sedezimale oder gar binäre Erstellung von Befehlsfolgen ist zeitraubend, unhandlich und auch nach viel Übung noch fehleranfällig. Wenn ein Programmierer wahrend der Testphase Befehle oder Daten hinzufugt, verschieben sich Datenadressen, Sprungziele und Befehlsoperanden, so daß die gesamte Befehlsfolge neu überarbeitet werden muß. Trotzdem sollte jeder, der sich mit seinem Computer ernsthaft beschaftigen will, diese Methode bis zu einem gewissen Grade üben, um auf diese Weise die Funktionen seines Gerates besser zu verstehen.
Abschnitt A4 des Bedienerhandbuches enthält den Z80-Befehlsvorrat im Binarkode. Bei der Erstellung eines Maschinensprache-Programmes kann etwa so vorgegangen werden.
Beispiel: Eine Zahl nn im HL-Register soll mit der Zahl 10 multipliziert werden.

a. Zunachst wird etwa in der Mitte des Blattes die mnemotechnische Befehlsfolge mit erklarendem Kommentar hingeschrieben

```
ML10:ADD HL,HL      HL =2 nn
     LD B,H
     LD C,L         BC =2 nn
     ADD HL,HL      HL =4 nn
     ADD HL,HL      HL =8 nn
     ADD HL,BC      HL =(8+2) nn =10 nn
```

b. Suchen des zugehorigen Binärkodes in einer Befehlsliste wie z.B. im Anhang 2 und Eintragen in die Befehlsfolge.

```
LD B,H   ...44H
LD C,L   ...4DH
ADD HL,HL...29H
ADD HL,BC ; 09H
```

c. Der Adreßpegel soll z.B. bei 4000H beginnen. Der Sedezimalkode wird eingetragen und der laufende Adreßpegel jeweils um die Bytelange ikrementiert.

```
4000H:29      ML10:ADD HL,HL      HL =2 nn
4001H:44           LD B,H
4002H:4D           LD C,L         BC =2 nn
4003H:29           ADD HL,HL      HL =4 nn
4004H:29           ADD HL,HL      HL =8 nn
4005H:09           ADD HL,BC      HL =(8+2) nn =10 nn
```

d. Mit der Monitoranweisung Mhhhh kann der Sedezimalkode in den MZ eingegeben werden.

Solch ein Multiplikationsprogramm ist wichtiger Bestandteil eines Konvertierprogrammes Dezimal...Dual-zahl. Die Ziffern werden einzeln über das Monitor-Unterprogramm CALL ?KEY abgerufen. Das erste gefundene Zeichen ungleich Dezimalziffer beendet den Konvertiervorgang. Die Dualzahl (0...65535) wird im HL-Register als Ergebnisparameter des CALL DZDU abgelegt.

```
DZDU:LD HL,0
DZD1:CALL 0903H          CALL ?KEY
     SUB 30H
     RET C               wenn < 30H
     CP 10
     RET NC              wenn > 39H
     ADD HL,HL           Allwert 2
     LD B,H
     LD C,L
     ADD HL,HL           Allwert 4
     ADD HL,HL           Allwert 8
     ADD HL,BC           Allwert 10
     LD B,0
     LD C,A              BCD-Ziffer
     ADD HL,BC           +(10 Allwert)
     JR DZD1
```

Eine Überprufung auf Überlauf erfolgt nicht. Zahlen > 65335 werden falsch konvertiert. Gleichfalls werden führende Blank nicht überlesen oder kein Vorzeichen erkannt, es können nur Betragszahlen konvertiert werden. Der Leser möge nach Erkennen des logischen Zusammenhangs eine "Hand-Assemblierung" in den Sedezimalkode vornehmen wie soeben gezeigt. Zu beachten ist die unterschiedliche Bytelange der Befehle. Im "1.Durchlauf" wird nach Festlegung des Adreßpegel-Anfanges der Befehlskode und der laufende Adreßpegel eingetragen. Dabei werden die symbolischen Adressen (Marken) DZDU,DZD1 ihren absoluten Pegeln gleichgesetzt und im "2.Durchlauf" im Befehlsoperanden eingetragen. Ebnso wird der umgekehrte Fall oft benotigt, namlich eine Dualzahl als Dezimalzahl anzuzeigen. Die zu konvertierende Dualzahl steht im HL-Register als Versorgungsparameter des CALL DUDZ.
Das Monitor-Unterprogramm CALL PRNT zeigt ein ASCII-Zeichen am Bildschirm an und schaltet die Koordinaten um eine Stelle weiter.

```
DUDZ:LD BC,2710H        mm =10000
     CALL DUD1
     LD BC,03E8H        mm = 1000
     CALL DUD1
     LD BC,0064H        mm =  100
     CALL DUD1
     LD BC,000AH        mm =   10
     CALL DUD1
     LD BC,0001H        mm =    1
DUD1:LD A,2FH
DUD2:INC A              A =30H +n
     OR A
     SBC HL,BC          nn -mm
     JR NC,DUD2
     ADD HL,BC
     JP 0017H           CALL PRNT
```

führende Nullen werden nicht unterdrückt. Auch hier möge der Leser nach Erkennen des logischen Aufbaus, evtl. mit zusätzlichem Kommentar versehen, "Assembler-Spielen", indem Adreßpegel und Sedezimalkode in zwei Durchläufen erzeugt werden.
Komfortable Konvertierprogramme enthält Anhang I

2.2 Assembliervorgang

Die in Kap. 1.2.1 gezeigte "Hand-Assemblierung" in den Maschinenkode erfolgt mit einem Assembler automatisch. Ein Assembler ist ein Compiler, der aus mnemotechnischen Ausdrücken den speziellen Binärkode einer ZE erzeugt und im Speicher ablegt.
Der Assembliervorgang kann auch auf einem fremden System stattfinden, das mit einer anderen ZE ausgestattet ist. Solche Assembler heißen Cross-Assembler. Wie eine höhere Sprache muß ein Assembler einen Editor enthalten, um die mnemotechnischen Ausdrücke und Datenformate zu gestalten. Im einfachsten Fall erzeugt eine Assemblerzeile einen Maschinenbefehl oder ein Maschinendatum. Ein Maschinendatum kann ein Byte, ein Wort, eine Adresse für Adreßrechnungen, ein Text oder allgemeines Binärmuster (Maske) sein.
Die in den Textpuffer (Editorpuffer) eingeschriebenen Assemblerzeilen bilden den Quellkode. Mit Hilfe des Editors werden Assemblerzeilen gelöscht, korrigiert und umgespeichert oder neue Zeilen eingefügt. Im "Lochkarten-Zeitalter (EDV-Steinzeit)" geschah dies durch Umsortieren. Da Korrekturen auf einer Karte nicht möglich sind, mußte in diesem Fall die Karte neu erstellt werden. Dies war eine sehr zeitraubende und kostspielige Methode. Der Leser möge erkennen, wie sehr die Automation die Methoden der Automation verändert.
Eine Assemblerzeile, wie sie in den Editorpuffer eingeschrieben wird, besteht im allgemeinen aus einer Marke (symbolische Adresse, Label), einem Operator, Operanden und Kommentaren, die formatlos abgelegt werden, um Speicherplatz zu sparen. Endezeichen einer Zeile ist CR.

```
ESEL:LD HL,AFFE;Lade HL mit AFFE,unmittelbar
Marke:Operator  1.Operand,2.Operand    ;Kommentar
------ --------  --------- ---------    ----------
ESEL :LD        HL        ,AFFE        ;Lade HL mit AFFE ,unmittelbar
```

Beim Assemblieren werden die Ausdrücke formatiert. Da einzelne Terme entfallen können, müssen diese von links nach rechts eindeutig fixierbar sein.

a. Nachgestellter Doppelpunkt kennzeichnet Marke
b. Trat keine Marke auf, muß ein Operator folgen
c. Nachgestelltes Space nach Operator listet Operanden folgen
d. Nachgestelltes Komma listet 2.Operanden folgen
e. Nachgestelltes Semikolon oder CR nach Operator kennzeichnet einen operandenlosen Befehl
f. Semikolon leitet begleitenden Kommentar ein

g. Beginnt eine Zeile mit Semikolon, so ist dies eine reine Kommentarzeile

Diese Fixierung ist seit altersher bei Assemblern üblich.
Ist der Quellkode erzeugt, erfolgt der erste Durchlauf. Der Assembler stellt die Bytelänge von Befehlen und Daten fest und inkrementiert entsprechend den laufenden Adreßpegel. Kann ein Operator nicht identifiziert werden, muß dies als Fehler angezeigt werden. Wird eine Marke gefunden, so erfolgt Eintragung zusammen mit dem laufenden Adreßpegel in das Adreßbuch. Beim Eintrag wird überprüft, ob eine gleichlautende Marke bereits eingetragen ist. Dies darf nicht der Fall sein und führt zur Fehlermeldung.
Wohl dürfen aber gleiche absolute Adressen verschiedene Marken haben. Traten während des ersten Durchlaufes formale Syntaxfehler wie soeben beschrieben auf, müssen diese mit Hilfe des Editors erst beseitigt werden, bevor der zweite Durchlauf gestartet werden kann. Einige Assembler gestatten direktes Editieren während des Durchlaufes.
Die wesentliche Aufgabe des ersten Durchlaufes ist die Zuordnung der Marken zu deren absoluten Adressen.
Im zweiten Durchlauf wird der eigentliche Maschinenkode (Objektkode) im Programmbuch erzeugt. Marken werden untersucht und deren Binärwert im Operandenteil des Befehles oder im Datum eingetragen. Nicht im Adreßbuch eingetragene Operandenmarken führen zu Fehlermeldungen, dann muß neu editiert werden.
Ein Assembler kann wie jeder andere Compiler nur formale Fehler (Syntaxfehler) erkennen, dies sind im wesentlichen nicht interpretierbare Befehle oder fehlende oder doppelte Marken.
Zum Studium des gesamten Befehlsvorrates des Z80 sei das Nachschlagewerk "Die Programmierung des Z80" von Rodnay Zaks (ISBN 3-88745-006X) empfohlen.
Bestandteil eines Programmes sind neben Befehlsfolgen auch die Daten. Beispiele für Datendefinitionen sind die Operatoren DEFB, DEFW, DEFM oder DEFS.

```
DEFB n          definiert n als Byte, absolut oder symbolisch
DEFW nn         definiert nn als Wort, absolut oder symbolisch
DEFM "ACHTUNG!" definiert "..." als Text
DEFS n          Adreßpegel um n Byte weiterschalten
```

An dieser Stelle sei der Einsteiger noch einmal daran erinnert, daß Daten niemals zwischen Befehlen stehen dürfen, weil jede ZE diese als Befehle interpretieren würde. Vor Beginn eines Datenfeldes muß immer ein unbedingter Sprungbefehl stehen und Sprungziele dürfen nie Datenadressen sein.
Fehler etwa der Art

```
5000H:CD0020            CALL 2000H
5003H:414348545354E4721 DEFM "ACHTUNG!"
5008H:C30050            JP 5000H
```

werden immer wieder gemacht. Der Text würde folgende unsinnige Befehlsfolge erzeugen

```
5000H:CD0020            CALL 2000H
5003H:41                LD B,C          A
5004H:43                LD B,E          C
5005H:48                LD C,B          H
5006H:54                LD D,H          T
5007H:55                LD D,L          U
5008H:4E                LD C,(HL)       N
5009H:47                LD B,A          G
500AH:21E300            LD HL,00E3H     !
500DH:40                LD B,B
```

Nachdem der Text als Befehle abgearbeitet wurde, ist hier BZ asynchron ,der Sprungbefehl JP 4000H wird falsch interpretiert.
Neben den Datenoperatoren werden zur Steuerung des Assemblervorganges Steueroperatoren benötigt, welche keinen Binärkode absetzen.

```
    ORG nn    Adreßpegel-Anfang des Objektkodes bei nn
ADR:INT       Nachfolgender Befehl oder Datum hat Marke ADR
    SKP       Formularvorschub für Assembler-Protokoll
ADR:EQU nn    Der symbolischen Adresse ADR wird Wert nn zugewiesen
    END       Programm-Ende
```

2.3 Testhilfe und Debugger

Selbst von erfahrenen Assembler-Programmierern kann man nicht erwarten, daß komplexe Maschinensprache-Programme auf Anhieb fehlerfrei arbeiten.
Der Programmablauf läßt sich aber mit Hilfe eines Testhilfeprogrammes (Debugger) überwachen. Ein Disassembler hat die Aufgabe, einen im Arbeitsspeicher stehenden Maschinenkode in Mnemonic-Befehlsfolgen oder Datenformate rückzuwandeln. Wie ein Maschinenkode zu interpretieren ist, als Befehlsfolgen, Bytes, Worte oder Texte, muß dem Disassembler mitgeteilt werden.
Ein Disassembler kann vorteilhaft zur Darstellung unbekannter Maschinensprache-Programme eingesetzt werden. Als erstes wird der zu untersuchende Speicherbereich auf Texte abgesucht, alsdann auf Befehlsfolgen. Bereiche mit sich widersprechenden Befehlsfolgen können dann nur noch Datenbytes oder Datenworte sein.
Adressenparameter von Befehlen können selbstverständlich nur als Zahlenwerte dargestellt werden. Nach Erkennen der Programmlogik kann schließlich hintennach ein Quellkode mit symbolischenn Marken geschrieben werden. Allerdings erfordert dies Verfahren einige Übung und Erfahrung, besonders wenn Datenfelder und Befehlsfolgen dauernd wechseln. Weil der Z80 keine feste Befehlslänge besitzt, können je nach Such-Anfangsadresse völlig andere Befehlsfolgen entstehen. Nach einigen unsinnigen Befehlen synchronisiert sich erstaunlicherweise jedoch die Disassemblierung.
Mit Hilfe eines Debuggers kann man Marken in kritische Befehlsfolgen, wie z.B. vor bedingte Sprünge setzen. Nach Aktivieren des auszutestenden Maschinensprache-Programmes stoppt die Befehlsausführung bei Auflaufen auf solch eine Marke, nachdem sämtliche Registerinhalte in Speicherzellen gerettet wurden. Anschließend wird der Bedienteil des Debuggers und des Disassemblers aktiviert, mit welchem die Registerinhalte und deren Schalten in allen möglichen Formaten angezeigt und auch verändert werden können. Auf diese Weise sind logische Programmfehler erkennbar. Müssen Befehle zwecks Korrektur in eine Befehlsfolge eingefügt werden, kann dies entweder mit Hilfe eines "Balkons" oder durch erneutes Editieren und Assemblieren geschehen.
Bei der Balkontechnik überschreibt man die Befehlsfolge mit einem unbedingten Sprung oder Unterprogrammaufruf auf freie Speicherbereiche, in welche die einzufügenden Befehle mit Hilfe des Debuggers eingetragen und mit einem Rücksprung abgeschlossen werden.
Beispiel:
In die Befehlsfolge

```
4000H:ADD HL,HL
4001H:LD B,H
4002H:LD C,L
4003H:ADD HL,HL
4004H:ADD HL,BC
```

soll zwischen 4003H und 4004H der Befehl ADD HL,HL zusätzlich eingefügt werden. Der Speicherbereich ab 5000H sei frei. Weil CALL nn drei Bytes absetzt, muß die Eingabe bei 4001H beginnen:

```
4000H:ADD HL,HL          5000H:LD B,H
4001H:CALL 5000H         5001H:LD C,L
4004H:ADD HL,BC          5002H:ADD HL,HL
                         5003H:ADD HL,HL
                         5004H:RET
```


Betriebssysteme für Maschinensprache-Entwicklungen für die SHARP-Serie MZ-80A, MZ-80B, MZ-700 und MZ-800 können zu einem günstigen Preis von DM 30,- (Kassettenversion) und DM 40,- (Floppy-Version) vom Verfasser dieses Handbuches bezogen werden. Assembler ,Disassembler sowie Debugger sind gleichzeitig verfügbar, so daß sich in diesem Fall die umständliche Balkontechnik erübrigt. Bei diesen Systemen erfolgt der Eintrag einer Testmarke in eine Befehlsfolge durch Überschreiben des Befehles mit einem Unterprogrammaufruf RST 7, welcher ein Byte lang ist. Erreicht BZ diese Marke, erfolgt Einsprung in den Debugger. Der Inhalt von SP wird gerettet und die Register in einem internen Hilfsstapel gepuffert und der überschriebene Befehl generiert. Die Registerinhalte können dann als Byte, Wort Befehl oder Zeichen angezeigt werden. Bei Programmfortsetzung werden die Register sowie SP generiert.

An den SHARP-User Club !

Tips für den Basic Interpreter 5Z-008:
(QD-Basic)

1.
Die Blockgröße bei sequentiellen Dateien ist festgelegt auf 1024 Bytes.
Dadurch wird das Laden und Speichern von größeren BSD-Dateien sehr langwierig und belastet das Laufwerk unnötig.Im Interpreter befindet sich die Blockgröße in den Bytes 38B3 und 38B4.
Die Anfangswerte nach dem Laden sind:
  38B3=0 und 38B4=4
Das entspricht hexadezimal 0400 oder dezimal 1024 Byte!
Diese Adressen können nun geändert werden.
Beispiel:Am Anfang eines Programms steht
  POKE &38B4,20
Damit wird die Blockgröße auf 2000(HEX) Bytes erhöht.
Die Blockgröße darf den freien Speicherplatz nicht überschreiten.
Ein Programm,das eine Datei mit veränderter Blockgröße abspeichert,kann auch nur eine solche Datei lesen.
Kleinere Blockgrößen in den obigen Adressen führen zu 'File not found error'.

2.
Die folgende Routine ermöglicht es, einzelne Programmzeilen aus einem Basicprogramm abzuspeichern.

Eingabe von:
```
  WOPEN#1,"Dateiname"
  LIST#1,100-200
  CLOSE#1
```

Damit werden die Zeilen 100-200 als BSD Datei abgespeichert.
Zum Laden muß man eingeben:
```
  LOAD"Dateiname",A  oder
  MERGE"Dateiname",A
```

3.
Abfrage des Joystick-Ports
Der Basicbefehl JOY(n) liefert bei der Verbindung von HBLNK und 1..4 folgende Werte:
 JOY(0) bis JOY(3) jeweils 255
Beliebige Kombinationen sind möglich.

Ferner ist es möglich,Masse und 1..4 zu verbinden.
Die Abfrage ist jetzt mit PEEK(&E008) möglich.
Es muß nur der gültige Bereich mit X=PEEK(&E008)AND 30 ausgeblendet werden.
X liefert dann für jede der 16 verschiedenen Anschlußkombinationen verschiedene Zahlen von 0 bis 28 in Zweierschritten.

Anschlußbelegung, Ansicht von Hinten:

Ansicht von hinten

```
 Masse 1 2 H +5V        Masse 3 4 H +5V
     * * * * *              * * * * *
```

Übrigends,die Idee mit der Clubdiskette ist sehr gut!
Vergeßt dabei die QD-Besitzer nicht!

Zum Schluß noch eine Frage:
Ich habe eine BASIC Compiler ohne Anleitung billig erworben.
Vielleicht hat irgendjemand eine Kopie für mich.
Er meldet sich nach dem Laden so:
** Basic-Interpreter V1.2 **
Copyright (C) 1983 by F.S$'soft

Nach dem Compileraufruf mit 'COM':
--- Basic Compiler V1.2 ---

Hans-Peter R

****

* PIO Selbstgebaut für den MZ-800 *

****

Oliver Spatscheck, Ghagäckerweg 45, 7170 Schwäb. Hall, 0791/41925

Liebe Sharp Computerfreunde,

seit einiger Zeit schon benötige ich einen Parallelport für meinen Sharp MZ-800 da mir die 8 Bit am Joystickport und die 8 Bit am Druckerport nicht mehr ausreichen. Da ich aber Schüler bin und mir die teuren Fertiggeräte nicht leisten kann, plante ich mir folgenden Erweiterungsport, an den ich bis zu 8 Parallelports anschließen kann.

Der Erweiterungsanschluß macht nicht mehr, als 8mal je 4 Adressen zu decodieren (man benötigt immer 4 Adressen für einen 8255) d.h. er stellt an den 8 Erweiterungsanschlüssen die 8 Datenleitungen,das RESET Signal, GND, 5V, ein CS Signal, ein WR Signal, ein RD Signal und die Adressleitungen 1 und 2 zur Verfügung. Das hat zusätzlich noch den Vorteil, daß man an den 8 Erweiterungsports fast jede, geringfügig veränderte Pherepherieschaltung anschließen kann (Uhr in Vorbereitung), da alle Erweiterungs-IC's diese Signale benötigen.

Der 8255, den ich als PIO-Schaltung angeschlossen habe, wurde dikrekt an den Erweiterungsport angeschlossen. Da ich die Ausgänge nicht gepuffert habe, sollte man keine großen Ströme direkt entnehmen. Die 3 Ports sind je auf einen 13-poligen Stecker geführt. Die 5V habe ich nicht weitergeführt, da man sonst nur in Versuchung gerät, sie zu verwenden, was eine Überlastung des Netzteils zur Folge haben könnte.

Aufbau Erweiterungsport:
Die IC's in dem Erweiterungsport können direkt eingelötet werden (wer im Löten nicht geübt ist, sollte Sockel verwenden). Bevor man aber diese einlötet, sollte man die Stecker und die Brücken einlöten. Die Brücken sind durch Zahlen gekennzeichnet, d.h. immer die gleichen Zahlen müßen verbunden werden.

Aufbau des PIO:
Für den 8255 sollte ein Sockel verwendet werden und der 8255 sollte erst nach allen Lötarbeiten eingesetzt werden.

Anschluß an den Computer:
Ich habe mir dazu eine passendes Platinenstück aus doppelseitig

weiter nächste Seite

beschichteter Platine hergestellt, an welchem ich oberflächlich Kabel angelötet habe. Das Platinenstück kann dann in einen der beiden Slots am Computer gesteckt werden. Ich habe es in den Unteren gesteckt, da ich eine Floppy besitze.

WICHTIG:
Bei allen Verbindungen ist darauf zu achten, daß der richtige Pol mit dem richtigen Pol verbunden wird. Dazu muß man auf die Bezeichnung achten, da die Pole nicht in der richtigen Reihenfolge nebeneinander liegen.

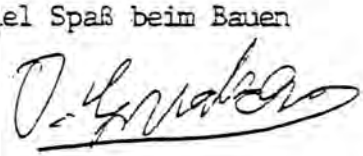
Am Erweiterungsport habe ich an den Aus- und Eingängen Stecker verwendet. Am Parallelport habe ich die Eingänge direkt auf die Paltine gelötet.

Auf die Programmierung des 8255 möchte ich hier nicht näher eingehen. In diesem Zusammenhang verweise ich auf das Buch Z80 Anwendungen von James W.Cofforn, aus dem Sybex Verlag, welches alle Perepheriebausteine zum Z80 Mikroprozessor ausführlich und leichtverständlich erklärt.

Ich möchte auch noch darauf hinweisen, daß ich keinerlei Haftung für Schäden die durch diese Schaltung entstehen übernehme.

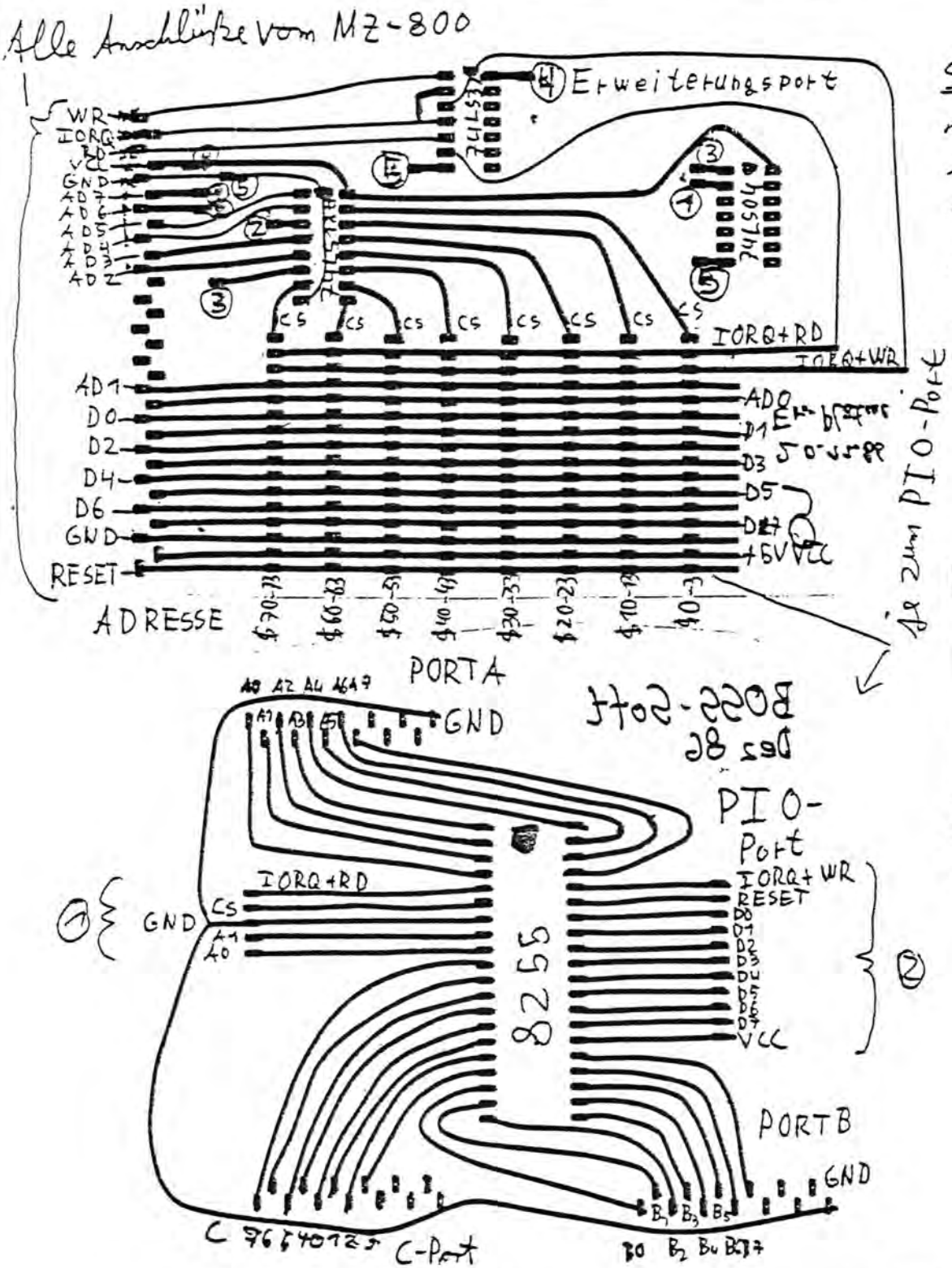
Für Clubmitglieder die nicht die Möglichkeit haben, Platinen herzustellen erkläre ich mich bereit gegen einen Unkostenanteil von 30,00DM pro Platinenset in Form eines V-Schecks, die Platinen herzustellen.

Viel Spaß beim Bauen



p.s. Wer noch offene Fragen hat soll mich einfach anrufen.

BESTÜCKUNGSSEITE

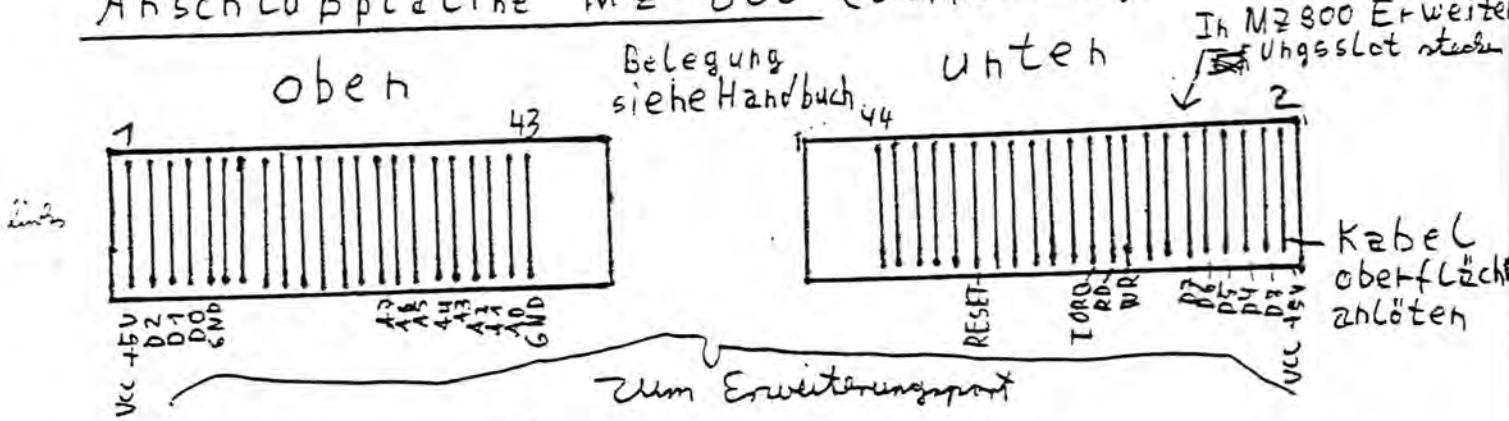


Stückliste

- 1 x 74LS138
- 1 x 74LS32
- 1 x 74LS04
- 1 x Steckverbindung 31pol DIN41617
- 1 x Buchsenleiste ↑ 31pol passend
- 8 x Stiftleiste 16pol
- 8 x Buchsenleiste passend
- 1 x 8255
- 1 x IC-Sockel 40pol
- 3 x Steckverbind. 13pol DIN41617
- 3 x Buchsenverbind passend
- Kabel

①+② zu einem Erweiterungsport

Anschlußplatine MZ-800 (doppelseitig)



STICHWORT: *Centro*

HARDWARE: Computer : Sharp MZ-800
Drucker : Panasonic KX-P1090
Interface : von Kersten & Partner
DIP-Schalter : Alle auf ON

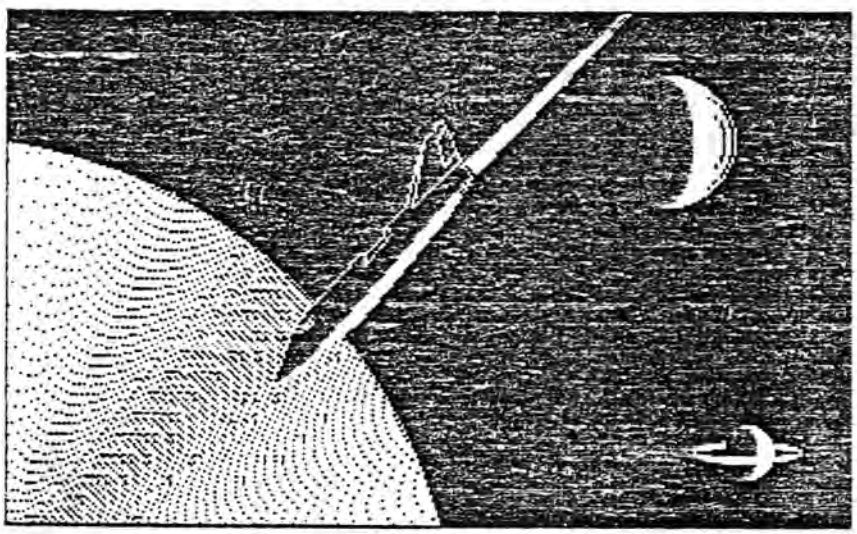
Mit der oben genannten Hardwarekonfiguration läuft mein Drucker ohne Probleme. Interessant mag sein, daß der DIP-Schalter für die Centronics-betriebsart auf ON steht, was heißt, daß die computereigene Centronics-schnittstelle überbrückt ist. Der Drucker wird also wie ein MZ Drucker angesteuert (Siehe hierzu auch Bedienungshandbuch Seite 7-3).
Als Ausgleich hierfür ist zwischen den Drucker und den Computer ein Interface geschaltet. Das Interface sieht aus wie ein kleines, schwarzes Kästchen mit einem Schalter. Dieses Interface übernimmt die Arbeit des Centronicsinterfaces des Computers. Der Schalter hat die Funktion zwischem dem Centronics-Code und dem Sharp-Code hin- und her zu schalten.
Eine kleine Frage ergibt sich natürlich hieraus: Wieso umständlich, wenn's auch einfach geht. Man könnte doch auch das Centronics Interface des Computers benutzen. Die Antwort lautet ganz einfach: Is nich! Einige meiner Freunde und ich haben schon alles mögliche ausprobiert, um den Drucker direkt an den Computer anzuschliessen. Es hat nie funktioniert. In dem Centronics-Interface ist der Wurm drin. Die einzige Abhilfe hat jetzt dieses kleine extra Interface gebracht. Der Drucker läuft jetzt ohne Schwierigkeiten.
Dieses Interface bietet jetzt noch einen Vorteil: Da man mit dem Schalter leicht zwischen Sharp-Code und Centronics-Code umschalten kann, ist es auch recht einfach Graphicen auszudrucken. Für Text verwendet man die Centronics-Position des Schalters, für Graphicen den Sharp-Code. Für Graphicen braucht der Code nicht umgewandelt zu werden. Logische Schlußfolgerung: man kann immer acht Punkte zusammenfassen, um die dann weiter an den Drucker zu geben (für nähere Erläuterung siehe Listing). Allerdings darf man dann nicht vergessen wieder zurück zum Text-Modus zu schalten, sonst sehen Listings und Briefe recht merkwürdig aus.

Dieser Text wurde mit dem Drucker gedruckt. Für Graphicbeispiele siehe weiter unten.

Das Folgende Programm macht ein Hard-Copy von einem Bildschirm mit der Auflösung 320 * 199 Punkte. Da es in Basic ist, ist es leider nicht das schnellste, aber es soll ja auch mehr als Demo fungieren.
Natürlich ist das folgende Programm für den MZ-800.

```
64000 REM -- Zeilenvorschub auf 8 Punkte begrenzen:
64010 PRINT/PCHR$(27,51,24);
64020 REM -- Gegebenfalls ELITE - Schriftart definieren, um 1-1 Kopie zu
             erhalten:
64030 PRINT/PCHR$(27,80,0);
64040 REM -- Acht Punkte von Bildschirm holen und nachsehen welche Punkte an
             oder aus. Acht Punkte zu einer Zahl zwischen 0 und 255 zusammen-
             fassen. Funktioniert wie PATTERN-Befehl (siehe auch Bedienungs-
64050 REM -- handbuch Seite 6-74),nur das hier die Daten an den Druckerkopf
             gegeben werden:
64060 FORY=0TO192STEP8:GR=0:GR$="":FORX=0TO160:FORZ=0TO7
64070 IFPOINT(X,Y+7-Z)THENGR=GR+2^Z
64080 NEXT:GR$=GR$+CHR$(GR):GR=0:NEXT:PRINT/PCHR$(27,75,LEN(GR$),0)+GR$;
```

```
64090 GR=0:GR$=""
64100 REM -- Nochmal, da String sonst zu lang wird:
64110 FORX=161TO319:FORZ=0TO7
64120 IFPOINT(X,Y+7-Z)THENGR=GR+2^Z
64130 NEXT:GR$=GR$+CHR$(GR):GR=0:NEXT:PRINT/PCHR$(27,75,LEN(GR$),0)+GR$:GR$=""
64140 REM -- Wiederhole bis ganzer Bildschirm kopiert. Dann Druckerreset
          fuer normalen Papiervorschub.
64150 NEXT:PRINT/PCHR$(27,64);
```



==========

MZ 700/800 Anwenderclub
c/o Germar NIKOL
Rathausstr. 3a
D-5100 Aachen

Name:.....
Vorname:.....

Erlaubnis:

Hiermit erteile ich dem MZ-Anwenderclub die Erlaubnis zur Verbreitung des im folgenden näher bezeichneten Programmes an MZ 700/800 Anwenderclubmitglieder. Kurzbeschreibung:

.....

.....

.....

.....

.....

Mir ist bekannt, daß die Programme MZ 700/800 Anwenderclub Mitgliedern unentgeltlich zum eigenen Gebrauch oder Nutzen zur Verfügung gestellt werden.

Datum:..... Unterschrift:.....

==========

Hinweis: Programme, die ohne diese Erlaubnis eingesandt werden, können nicht berücksichtigt werden.

==========

SCHMITZ FRIEDRICH W.

CENTRONICSSCHNITTSTELLE MZ-800

An meinem Computer, ein MZ-800, habe ich einen Drucker STAR SG-10 angeschlossen. Verbunden sind die beiden Geraete mit einem Flachbandkabel 35-adrig, also keine besondere Centronicsschnittstelle. Der Computer ist mit den Dip-Schaltern auf Centronics umgeschaltet.

Ich persoenlich habe im Drucker ein auf dem MZ-800 angepasstes Eprom eingebaut (nimmt den Platz der frei definierbaren Zeichen ein). Es geht aber ohne dieses Eprom. Vor dem Ausdrucken eines Textes oder einer Datei in Basic braucht man nur den Drucker zu initialisieren durch INIT"LPT:M0,S2" oder wenn die RAM-Steckkarte vorhanden ist durch INIT"LPT:M1,S2". Beim STAR-Drucker dann noch ein: PRINT/P CHR$(27)"§" hinterherschicken, das ist bei alles.

In beiden Faellen nimmt der Drucker dann auch die kleinen Buchstaben. In keinen Falleist es moeglich, damit die Umlaute oder die SHARP-Graphik auszudrucken.

Beim Betriebssystem P-CP/M ist das Ganze relativ einfach. Damit der Drucker z.B. dBASEII Dateien ausdruckt muss mit dem Programm SETUP.COM der Printermode auf ASCII umgeschaltet werden und schon laeuft alles zur vollsten Zufriedenheit.

Dieses Textprogramm, mit dem dieses geschrieben wurde, ist direkt an den Drucker softwaremaessig angepasst, laeuft also ueber den MZ-800-code im SETUP.

Fordert man nicht eine umfangreiche Fehlerabfrage beim Drucker, so reicht die Abfrage der BUSSY-Leitung in jedem Fall aus.

Die gesamte angebotene Lösung ist natürlich nicht für Kleinbuchstaben und Sonderzeichen geeignet. Wenn aber jemand nicht gerade Text verarbeitet sondern 'Rechnen' läßt und Programme auflisten will, dann funktionierts prima.

Bernhard Beck

Kleinanzeigen — Kleinanzeigen — Kleinanzeigen

Suche für MZ 700/800: Diskettenlaufwerk für MZ 800 incl. Disk-Basic, Buchführungs- und Textverarbeitungsprogramm mit Handbuch sowie leistungsfähigen Drucker mit Einzelblatt, NLQ oder Speicherschreibmaschine mit Interface für MZ 800. Angebote an E. Hoppe, Gänseweide 1, 3308 Königslutter, Tel 05306/3056

MZ 800 Systemaufbau. Suche Soft- und Hardware, Floppy 5.25". Willi Delkers, Leipeltstr. 6c, 2102 Hamburg 93

Suche Kontakte zu Sharp MZ Anwendern im Raum Bünde/Herford/Lübbecke. Suche gebrauchten Drucker für MZ 731/821. G. Luda, Hellmannstr. 2, 498 Bünde 11, Tel 05223/60417

Verkaufe: Holzkötters MZ 800 Systemhandbuch für DM 30.- Denny Fliegner, Sonnenblick 34, 6430 Bad Hersfeld, Tel 06621/71650

Neuanfänger MZ 800, keine Kenntnisse, sammelt noch alle Informationen. Wer hat solche und kann mir schreiben? (Bezugsadresseen, Hard-, Software, etc.) Jürgen Stöhr, Sperlichstr. 64, 44 Münster ,Tel 0251/521073

Suche Hardcopy für OKI 20, Zeljko Kosanovic, Eschersheimer Landstr. 90, 6000 Frankfurt/Main 1

Suche Anschriften von MZ 800 Anwendern mit Akustik-Kopplern, Mailboxen. Jürgen Schmidt, Zum hohen Berg 2, 2151 Regesbostel

Wie kann ich eine QD mit 800er Controller (MZ-1E19) an den MZ 700 anschließen? Wie kann ich QD und 5.25" Floppy preisgünstig zusammen an den MZ 800 anschließen? Suche Kontakte zu MZ Anwendern im Raum Herford, A. Lücking, Mozartstr. 2, 4900 Herford

Suche Software, Listings + Manuals leihweise oder zum Kauf für Mz 80k & MZ 800. R. Fischer,Jakobsgarten 15, 6700 Ludwigshafen/Rh 27, Tel 0621/654690 (nach 17.00 Uhr)

****

* Verkaufe Bücher, ganz neu, frei Haus, Zahlung nach Lieferung.
** BASIC-80 UND CP/M, J.J. Purdum, Computer Persönlich. 20.--
*** TURBO PASCAL 1.xx bis 3.xx, G. Renner, Chip Verlag. 20.--
**** EINFÜHRUNG IN DIE PROGRAMMIERSPRACHE BASIC, Curran, Falken V. 10.--
* suche "CP/M (MZ 800) für 768 KB- Laufwerke Preis ? *
** suche "Hausverwaltung für 30 Wohnungen" CP/M Preis ? **
*** suche "Fibu" unter CP/M Preis ? ***
**** R. Hahn, PF. 123, 8704 Uffenheim, Tel. 09842/2816 ****

****

P C - F R A G E B O G E N

Tel.:
erreichbar ab : Uhr
Abs.:

Zurücksenden an :

Kersten & Partner GmbH

Peliserkerstr . 86

5100 AACHEN

R.Soetje

Badstr.19

7600 -Offenburg

Bitte geben Sie uns genaue Angaben über Ihren Rechner und be - sonders über Ihre Peripherie . Nur so können wir Ihnen mit - teilen , ob Ihre Komponenten in Ihrem evtl. neuen PC verwendet werden können . Vielen Dank !

Rechner :

Monitor :
Video Anschluß:(z.B.BAS,RGB)

Drucker :

Floppysysten :
Format : (z.B.3.5",5.25")

Bemerkungen :

Mein evtl.neuer PC sollte ein XT / AT (nicht gewünschtes strei - en) mit folgender Konfiguration sein :

C.G.A. KARTE / HERCULES-KARTE / E.G.A.- KARTE

FLOPPYLAUFWERKE : 1x360KB 2x360KB 1x1,2MB 2x1,2MB (f.AT)

SCHNITTSTELLEN : CENTRONICS RS 232 Sonstige :

FESTPLATTE : XT = 20 MB . 30 MB AT = 20 MB . 30 MB . 40 MB

MS-DOS wird benötigt JA / NEIN bereits vorhanden .

BEMERKUNGEN :

Ich möchte DM .- ausgeben . Bitte unterbreiten Sie mir ein Angebot unter Nutzung meiner vorhandenen Peripherie in einem PC .

Wenn Sie noch technische Fragen haben,rufen Sie Herrn Hirsch in der Zeit von 14.00 bis 17.00 unter 0241 - 533108 an .
oder

MONTAG von 15.00 bis 19.00 Uhr .

SONDERPREISLISTE FLOPPYSYSTEME SHARP MZ 700 - MZ 800

S F D 800

FLOPPYSYSTEM

Floppylaufwerk , 5.25" , 320 KB , Kontroller für MZ 800,

Anschlußkabel , deutsches Handbuch , Disc - Basic .

DM 475.- inkl. 14 %

S F D 700

FLOPPYSYSTEM

Floppylaufwerk , 5.25" , 320 KB , Kontroller für MZ 700

Anschlußkabel , deutsches Handbuch , Disc - Basic .

DM 500.- inkl. 14 %

Lieferzeit : min. 2 Wochen nach Auftragseingang .

Lieferung : per Nachnahme oder Vorkasse mit V - Scheck .

Bestellung : nur schriftlich mit diesem Formular !!!! .

Hiermit bestelle ich per Nachnahme / Vorkasse ./. 4% Skonto

____ Stück SFD 800 / SFD 700

Abs.:

Unterschrift : Datum :

Kersten & Partner Datensysteme GmbH Tel.: 0241-533108

Peliserkerstr. 86 5100 Aachen

Verantwortlich für den Impuls ist also die aufeinanderfolgende Ausgabe von 80h und 0 (XOR A setzt A zu Null!). Diese Reihenfolge wird vertauscht. Dazwischen liegt hier im speziellen Beispiel noch die Abfrage des Bussy-Bits (LD C,1; CALL BUSSY). Das kann jedoch von Fall zu Fall verschieden sein und wird nicht verändert.

Nun ist nur noch ein Punkt zu klären, der Port FEh ist nach dem Einschalten auf LOW gesetzt, muß also noch auf HIGH initialisiert werden. Das geschieht am Besten indem die entsprechende Routine einfach an die Software anhängt oder noch innerhalb unterbringt.

```
Einprung    : 3EFF   LD A,FFh
              D3FE   OUT (FEh),A
Rücksprung  : C3xxxx JP zurück
```

Wer diese Änderung nicht unterbringen kann hat aber noch eine 2. Möglichkeit: Zu Beginn des Druckvorganges, also wenn der Rechner Zeichen abschicken will, wechselt man am Drucker einfach 2 mal den OFFLINE/ONLINE- Modus. Das bewirkt, daß die BUSSY-Bedingung, an der das Programm zum Stehen gekommen ist, jetzt freigegeben wird und ein Zeichen ausgegeben werden kann. Danach ist der Port FEh im Ruhezustand dann wieder auf High, so daß alle folgenden Zeichen nun übermittelt werden können.

Anschlußbelegung des Verbindungskabels:

Sharp		Centro
1 RDP	---	1 STROBE
2 GND	---	19 GND
3 RD1	---	2 DATA 1
4 GND	---	20 GND
5 RD2	---	3 DATA 2
6 GND	---	21 GND
7 RD3	---	4 DATA 3
8 GND	---	22 GND
9 RD4	---	5 DATA 4
10 GND	---	23 GND
11 RD5	---	6 DATA 5
12 GND	---	24 GND
13 RD6	---	7 DATA 6
14 GND	---	25 GND
15 RD7	---	8 DATA 7
16 GND	---	26 GND
17 RD8	---	9 DATA 8
18 GND	---	27 GND
19 IRT	---	10 ACKNLG
20 GND	---	28 GND
21 RDA	---	11 BUSSY
.		.
.		.
.		.
.		.
.		.

Sämtliche Verbindungen sind herzustellen, entsprechend der Skizze. Diese Belegung ergibt sich aber automatisch, da die Stecker die Pins verschieden aneinanderreihen, sicherheitshalber nachprüfen!

Hier ist ein Lösungsvorschlag für den Direktanschluß eines Centronics- kompatiblen Druckers an den MZ-700.
Diese Lösung ist bewußt einfach gehalten, um jedem die nötige Anpassung zu ermöglichen, ganz gleich, was gerade genutzt wird (Assembler,Basic,....).
Es handelt sich hierbei um eine Softwareanpassung.

Zur Problemstellung:
Es gibt speziell 2 Änderungen zu bewältigen,
a) Strobe-Puls umkehren
b) Zeichensatz übersetzen.

Ich beschränke mich mit diesem Lösungsvorschlag nur auf Punkt 1, aus folgenden Gründen:
- sämtliche Großbuchstaben und Rechenzeichen können gedruckt werden, was für alle Programmlistings, Tabellen, Dateien oder Adressen ausreicht.
- nur ein einfaches Kabel stellt die Verbindung zwischen Drucker und Rechner her, somit eine schnelle und billige Lösung, die bei entsprechenden Steckern für Flachbandleitungen (Schneid- Klemm- Montage) ohne Löterbeiten und fehlersicher auszuführen ist. Ein Vertauschen von Leitungen ist praktisch ausgeschlossen.
- die Umkehrung des Strobe-Pulses vertauscht nur einige wenige Speicherinhalte, benötigt so keinen zusätzlichen Speicherplatz und ist damit in jeder Software anwendbar.

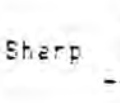
Allgemeine Vorgehensweise bei der Softwareänderung:

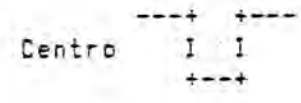
- Verlassen des jeweiligen Systems, notfalls mit RESET.
- Suche im Speicher nach der Zeichenkette die für das Stobe-Signal verantwortlich ist (siehe Listing unten), mittels eines einfachen Maschinenprogrammes oder dem Basic-Monitor.
- diese Speicherzellen umschreiben.
- evtl. Portinitialisierung anhängen
- Abspeichern der geänderten Software

Technische Daten und Informationen:

Die Portadresse FFh übermittelt die 8 Datenbits, FEh I/O Bussy und Strobe.
BUSSY ist das Signal des Druckers, welches seine Aufnahmebereitschaft signalisiert und kommt auf Leitung RDA = Steckerpin 21, Bit 0.
STROBE ist das Signal des Rechners, welches anliegende Daten als gültig abschickt, Leitung RDP = Steckerpin 1, Bit 7.

Umkehrung des STROBE-Pulses anhand des Beispieles 'MONITOR':






```
0198 3E80     LD A,80h                                  0198 AF       XOR A
019A D3FE     OUT (FEh),A                               0199 D3FE     OUT (FEh),A
019C 0E01     LD C,1          >>> so Ändern >>>         019B 0E01     LD C,1
019E CDB601   CALL BUSSY                                019D CDB601   CALL BUSSY
01A1 AF       XOR A                                     01A0 3E80     LD A,80h
01A2 D3FE     OUT (FEh),A                               01A2 D3FE     OUT (FEh),A
```

BASICPROGRAMMSCHUTZ FUER MZ-800 SHARP DISK-BASIC

Ich habe Ihren Artikel ueber das Schuetzen von Basicprogrammen in der Nr. 2 des Magazins gelesen.

Hier noch eine andere Loesung:

LIST-SCHUTZ EIN: POKE $6A97,$E4:POKE $6A9D,$00,$C9,$00

SAVE-SCHUTZ EIN: POKE $73C0,$CD,$64,$01

BYE -SCHUTZ EIN: POKE $6A59,$CD,$7A,$79

DIR -SCHUTZ EIN: POKE $7424,$CD,$F9,$CE

LIST-SCHUTZ AUS: POKE $6A97,$F4:POKE $6A9D,$3E,$01,$32

SAVE-SCHUTZ AUS: POKE $73C0,$00,$C9,$00

BYE -SCHUTZ AUS: POKE $6A59,$00,$C9,$00

DIR -SCHUTZ AUS: POKE $7424,$00,$C9,$00

Wie man diesen Schutz einsetzt bleibt jedem ueberlassen. Ich habe die besten Erfahrungen gemacht, in dem ich diesen Schutz in ein kleines Vorprogramm gesetzt habe, das das Hauptprogramm dann erst von Diskette oder Band in den Speicher zieht. Wenn man das Ganze ueber eine ON ERROR GOTO - Meldung mit anschliessender Loeschung laufen laesst, hat ein anderer schon seine liebe Not damit.

P-CPM AUTO-START

Die COM-Datei SUBMIT muss vorhanden sein.

Wenn vorhanden, dann eingeben:

PIP B:AUTO.SUB=CON:

Der Cursor springt dann unter diese Zeile und man kann das oder die Programme eingeben, die im AUTO-START beruecksichtigt werden sollen. Nicht verblueffen lassen, der Cursor springt, nachdem man das erste Programm abgeschickt hat, wieder an den Anfang zurueck. Das zweite Programm einfach ueber das erste hinwegschreiben. Anschliessend im SETUP unter AUTOEXECUTE mit SUBMIT AUTO.SUB einschreiben und auf Diskette speichern.

S H A R P – U S E R – C L U B

B O C H U M

An alle SHARP MZ-800 Besitzer Bochum, im Juli 1987

im R U H R G E B I E T

Betrifft :Sharp - User Club

Liebe SHARP MZ-800 Freunde .

Zum ersten Treffen eines SHARP MZ-800 USER-CLUB laden wir alle Interressenten recht herzlich am Samstag den 08.08.87 in die **Hotel Gaststätte Jägerhof** in 4630 Bochum 1, auf der Dorstenerstraße 89 ab 20 Uhr ein. Es soll sich bei der ersten Begegnung um ein zwangloses Treffen handeln. Der Zweck des Treffens ist das Kennenlernen und ein Gedankenaustausch, wie, wo und wann wir uns in Zukunft Treffen wollen, um unserem schönen Hobby noch mehr Interresse abgewinnen zu können. Es ist an der Zeit, daß im Herzen des Ruhrgebietes die Sharp-Freunde, die es ja, seit dem ein Warenhaus den SHARP MZ-800 sehr preißwert im Winterhalbjahr 86/87 angeboten hatte, zusammen kommen, um über eventuelle Probleme zu diskutieren. Wir stellen uns vor, daß bei diesen Treffen über mancherlei gesprochen wird, was ein einzelner in seinem stillen Kämmerchen auf sich allein gestellt, nicht lösen kann.

Kontaktadresse :Gerhard Schulte
Gustavstraße 10a
4630 Bochum 1
T.0234/512968

Karl Heinz Guschke
Dorstenerstraße 81
4630 Bochum 1
T.0234/513839

Mit freundlichen Grüßen

Gerhard Schulte

---------- MZ -Anwenderclub - Softwareservice ----------

MZ - Anwenderclub
c/o Germar NIKOL
Peliserkerstr 86
5100 AACHEN

Abs.:

Hiermit bestelle ich folgende Produkte :

Kassettensoftware MZ 700

...	Pascal Compiler T 700		DM 25.-
...	Assembler System "		DM 25.-
...	Fortran Compiler "		DM 25.-
...	Datenbank "		DM 25.-

Diskettensoftware MZ 700 , 5.25",320KB und 3.5" ,320KB . keine QD !!!!

...	Pascal Compiler D 700		DM 25.-
...	Fortran Compiler "		DM 25.-
...	Assembler System "		DM 25.-
...	Datenbank "		DM 25.-
...	Lager & Rechnung "		DM 35.-

Kassettensoftware MZ 800

...	Pascal Compiler T 800		DM 30.-
...	Assembler System "		DM 30.-
...	Fortran Compiler "		DM 30.-
...	Datenbank "		DM 30.-
...	Maschinensprache "		DM 25.-
...	Textwriter 80 Z. "	(nur Sharp Code)	DM 35.-
...	Function Plot "		DM 25.-
...	Hardcopy "		DM 25.-
...	Spiele "	(Preis für drei Spiele)	DM 25.-

Diskettensoftware MZ 800 für SFD 800 , MFD 800 , MZ 1F 19 , Keine QD !!

...	Pascal Compiler D 800		DM 30.-
...	Assembler System "		DM 30.-
...	Fortan Compiler "		DM 30.-
...	Datenbank "		DM 30.-
...	Maschinensprache "		DM 25.-
...	Textwriter 80 Z. "	(nur ASCII Code)	DM 35.-
...	Lager & Rechnung "		DM 40.-
...	MZ 700 Disc Basic für den MZ 800		DM 30.-

Bei obigen Produkten handelt es sich um Originalprodukte der Firma Kersten & Partner GmbH , die der Club zu Selbstkostenpreisen abgiebt . Der Versand erfolgt ausschließlich gegen Vorkasse durch überweisung auf das Pschk. Köln Nr.: 2635 43-502 G.NIKOL .
Porto und Verpackung werden pauschal mit DM 5.- berechnet .
Die Lieferzeit beträgt in der Regel ca.14 Tage .

Ich wünsche das/die Programm(e) auf 5.25" disc oder 3.5" disc .
Der Gesammtbetrag über DM ..... (inkl. 5.- Porto) wurde auf das Pschk. Köln überwiesen .

Datum : ..... Unterschrift : .....

Software für die Sharp Quick Disc ist nicht lieferbar !!!!!!!!!!!!!!!!

MZ - Anwenderclub Hardwareservice

MZ - Anwenderclub
c/o Germar NIKOL
Peliserkerstr.86
5100 AACHEN

Abs.:

Hiermit bestelle ich folgende Produkte :

... SFD 700 Floppysystem 5.25",320KB,D-Basic 700. DM 600.-

~~... MFD 700 Floppysystem 3.5" ,320KB,D-Basic 700. DM 450.-~~

... CE 700 Centronicsinterface MZ 700 DM 150.-

... CE 800 Centronicsinterface MZ 800 DM 150.-

... AD Adapterplatine für SFD/MFD 700 zum Anschluß an MZ 800 , 800 D - Basic . DM 45.-

... SFD 800 Floppysystem 5.25",320KB,D-Basic 800. DM 675.-

... FDC 700 Controller MZ 700 , D - Basic . DM 150.-

... FDC 800 Controller MZ 800 , D - Basic . DM 150.-

SONDERPOSTEN !!

... DFD 800 Doppelfloppysystem 2 x 5.25" , 2 x 320 KB , Controller,D-Basic,anschlußfertig für MZ 800. DM 850.-

~~... MFD 800 Floppysystem 3.5",320KB,Controller,D- Basic. DM 350.-~~

~~... 64 KB RAM v File (Sharp kompatibel) DM 150.-~~

... 80 ZK 80 Zeichenkarte 700 mit Basic . . (CP/M a.Anfrage) DM 55.-

... Video RAM MZ 800 Chip Satz DM 35.-

!! 03.09.1987 nur solange Vorrat reicht !!

Für Bastler :
Defekte oder teilbestückte Komponenten ohne Garantie .

... KG	Kunststoffgehäuse für 80-ZK oder PCG-Grafik	DM 10.-
... DFDG	Aludoppelfloppygehäuse für 2x5.25"	DM 30.-
... PCG 700	PCG-Grafik Platine , Anleitung , unbestückt.	DM 15.-

---

Die Produkte stammen aus Kersten & Partner GmbH Produktion und werden vom Club zu Selbstkostenpreisen abgegeben . (Garantie 6 Monate)

Der Versand erfolgt nur gegen Vorkasse auf das Pschk.Köln 2635 43-502 Germar NIKOL . Zuzügl. DM 5.- für Porto und Verpackung . oder V-scheck. Lieferzeit beträgt ca. 14 Tage nach Zahlungseingang .

Der Gesammtbetrag über DM .... (inkl. 5.- DM Porto) wurde auf das Pschk.Köln überwiesen .

Datum : .... Unterschrift : ....

~~Achtung Diskformate angeben : 5.25" / 3.5" / Tape / !!!!!!!!!!!!!!!!~~

Sharp MZ-700/800 - Software -

Sehr geehrter Kunde,

der Ihnen vorliegende Katalog beschreibt Ihnen in kurzer Form Software unseres Hauses für den Sharp MZ 800 und seinen Vorgänger Sharp MZ 700. Bei den für den MZ 800 angebotenen Programmen weisen wir ausdrücklich darauf hin, daß diese Programme speziell für den MZ 800 erstellt wurden. Es sind also keine Programme, die einfach in den MZ700 Modus umschalten wie es bei vielen im Handel befindlichen Programmen üblich ist.

Die meisten Programme stehen zur Zeit als Kassetten oder 5,25 und 3,5 Zoll Diskettenversion zur Verfügung.

Mit freundlichen Grüßen
Kersten & Partner
Datensysteme GmbH

Lager + Rechnung:

Programm zur Verwaltung von Lagerbeständen von 2000 Artikeln, (Umsatzstatistik, Angebot, Rechnungen, Lieferschein, Gutschriften). Da es menuegesteuert arbeitet, ist die Bedienung einfach. Lieferbar als Diskettenversion 5 1/4 und 3,5 Zoll für MZ 700+ MZ 800.

Textwriter

Das Programm „Textwriter" von K & P ist ein leistungsfähiges Hilfsmittel zur Bearbeitung eines Textes. Eine Schreibmaschine wird in jeglicher Weise ersetzt und durch die vielfältigen Möglichkeiten dieses Programmes in der Leistungsfähigkeit und im Schreibkomfort in jeder Weise übertroffen. Einschränkend muß gesagt werden, daß das Programm keinen Anspruch auf absolute Professionalität stellt. Die Eigenschaften der großen bekannten Textverarbeitungssysteme wie z.B. *WordStar werden nicht in allen Punkten erreicht - die bestehenden Einschränkungen fallen jedoch bei der praktischen Arbeit kaum ins Gewicht. Durch seine einfachere Handhabung dürfte „Textwriter" in einigen Punkten sogar angenehmen zu benutzen sein.

Textwriter arbeitet im 80 Zeichen Bildschirmmodus des MZ 800 und ist deshalb allen bekannten Textverarbeitungen, die für den MZ 700 erstellt wurden, überlegen. Das Programm verfügt über einen Helpmodus, der zu jeder Zeit aufgerufen werden kann. So kann sich auch der Anfänger schnell in das Programm einarbeiten. Der Helpmodus ist menuegesteuert. Textwriter bietet folgende Möglichkeiten:

Deutsche Schreibmaschinentastatur, Einfügen/Löschen einer Zeile, Zeilenweise/Seitenweise Scrollen, Cursorpositionierung, Statuszeile, Linken/Rechten Textrand setzen, Setzen/Löschen/Anspringen des Tabulators, Zeilenumbruch, Automatischer/Manueller Randausgleich, Suchen einer Textstelle, Druckerformat mit Kontrollcodes, Druckeranschluß durch K & P Centronicsinterface.

Textwriter belegt ca. 30-34 KB im Speicher.

„Textwriter" wird in zwei Versionen angeboten:

1. Diskettenversion MZ-800/D
2. Kassettenversion MZ-800/D

*Eingetragenes Markenzeichen der Firma Micropro-International

Pascal-System

Allgemeines:
Das K & P Pascal-System ist eine leistungsfähige Implementierung der Sprache „Pascal" auf Ihrem MZ-700/800.

Neben RECORD's und ARRAY's kann dieses System auch Zeigervariablen verarbeiten. Der Datentyp FILE wurde hingegen nicht in das System integriert. Ansonsten bringt das K & P Pascal-System optimale Voraussetzungen zur strukturierten Pascalprogrammierung.

Neben den Pascal-Standard-Funktionen und -Prozeduren bietet das System eine Reihe weiterer Funktionen/Prozeduren an, die die Möglichkeiten des MZ-800 bezüglich des Musik- und Farbgenerators ausnutzen. Um bei diesem System nicht ganz auf die Externe Speicherung von Daten verzichten zu müssen, können mit den zwei Prozeduren (SAVE-, LOADDATA), festgelegte Speicherbereiche auf Diskette beziehungsweise Kassette (je nach Version) abgelegt werden.

Das K & P Pascal-System ist in zwei Versionen erhältlich:

a) Kassetten-Version MZ700 + MZ800
   hier dient die Kassette als externes Speichermedium
b) Disketten-Version MZ700 + MZ800
   hier dient die Diskette als externes Speichermedium.

Die Vorteile der Disketten-Version liegen nicht nur bei den kurzen Lade- und Schreibzeiten, sondern auch beim integrierten FDOS (Disketten-Verarbeitungs-System), mit dessen Hilfe Arbeitsdisketten kopiert, Files von Disketten auf Kassette übertragen (und umgekehrt) werden können, etc.

KERSTEN & PARTNER
DATENSYSTEME GMBH
Wildbacher Mühle 83 · D-5100 Aachen · Tel. 0241/17 10 67-8

Aufbau des Pascal-Systems

Das „K & P Pascal-System" ist in drei Unterprogramme aufgeteilt:

a) der Editor
b) der Compiler
c) der Filer (Diskette)
   das CMT (Kassette)

a) Der Editor gestattet dem Benutzer, auf einfache Art und Weise Pascal-Quelltexte zu erstellen.
b) Der Compiler ermöglicht das Austesten der zuvor editierten Pascal Programme (Quelltexte) und das Erstellen von Objekt-Files (Maschinenprogramme), die letztlich das Endprodukt eines Übersetzungslaufs darstellen.
c) Der Filer ermöglicht dem Anwender den direkten Zugriff auf den externen Datenspeicher.

Da das Pascal-System ca. 30 KByte benötigt, steht dem Anwender ein Arbeitsspreicher von ca. 34 KByte (für Quell- und Objektfiles) zur Verfügung.

Fortran Compiler:

Ist in der Lage sowohl Integer - als auch Realwerte zu verarbeiten. Die Kassetten-Versionen benötigen die gesamte 64-K-Byte. Davon belegt das System insgesamt 25-K-Byte des Speichers, also stehen dem Anwender 44-K-Byte zur Verfügung. Ebenso wie der Assembler beinhaltet der Fortran Compiler noch eine Maschinensprache und einen Single-Stepper. Durch die Menütechnik entfällt das Zwischenspeichern auf externe Speichermedien. Es werden keine weiteren Hilfsprogramme benötigt. Lieferbar als Kassetten- und Diskettenversion 5 1/4; 3,5 Zoll für MZ700 + MZ800.

Maschinensprache:

Hilfsprogramm zur Eingabe und zur Änderung von Programmen in Maschinensprache incl. Disassembler. Es lassen sich folgende Geräte ansprechen: Quick-Disc, Kassette, SFD 800 und RAM-Datei. Lieferbar als Kassetten- und Diskettenversion 5 1/4; 3,5 Zoll für MZ800.

Assembler-System

Dieses Programm ist eine leistungsfähige Implementierung der Assembler-Sprache. Der Name Assembler-System weist darauf hin, daß es sich nicht nur um einen Assembler handelt, sondern darüberhinaus noch eine Maschinensprache und einen „Singel-Stepper" (Trace-Modus) beinhaltet.

Durch die Menütechnik wird die Bedienung des Programms denkbar einfach gehalten. Sie können nach Belieben den Assembliermodus verlassen und z.B. zur Maschinensprache wechseln ohne daß hierdurch Ihr Assembliertext gelöscht wird.

Zeitraubendes Zwischenspeichern auf externe Speichermedien (insbesondere bei C-Version) entfällt also!

Das programm ist in sich abgeschlossen, d.h. Sie benötigen für den Assembliervorgang keine weiteren Hilfsprogramme (z.B. „Relocate-loader" oder „Symbolic Debugger"), wie Sie es vielleicht von anderen gleichnamigen Programmen her kennen! Durch diese Art der Anwendung und des Aufbaus wird die Erstellung eines lauffähigen Maschinenprogramms sehr beschleunigt.

Das Programm liegt z.Z. in drei Ausführungen vor, die an dieser Stelle zwecks Überblick genannt werden sollen:

a) Assembler-System 800/CX	für den MZ-800 mit CMT-Betrieb
b) Assembler-System 800/DX	für den MZ-800 mit Disk-Betrieb
c) Assembler-System 700/DX	für den MZ-700 mit Disk-Betrieb
d) Assembler-System 700/CX	für den MZ-700 mit CMT-Betrieb

„X" steht für die Versionsnummer

Sämtliche Versionen nutzen den vollen 64 K-Speicherbereich des MZ-700/800 aus. Für Anwender ergibt sich hieraus der große Textbereich von 44500 Bytes.

Aufbau des Programms:

Der Assembler ist in 5 Unterprogramme aufgeteilt.

a) Dis-Assembler
b) Assembler
c) Search-Adress
d) Hexmonitor
e) FDOS bzw. CMT

zu a) Mit dem Dis-Assembler können beliebige Maschinenprogramme disassembliert werden.

zu b) Der Assembler beinhaltet seinerseits einen „full screen-Editor" und natürlich den Assembler selbst.

zu c) Mit diesem Programmteil können Sie beliebige Adressen im gesamten Speicherbereich suchen.

zu d) Der Hexmonitor stellt eine kleine Maschinensprache dar und gestattet in der Hauptsache das Austesten und Verbessern noch nicht lauffähiger Maschinenprogramme, da solche Programme unter frei wählbaren Anfangsbedingungen (Register-Inhalte) gestartet werden können. Hierbei ist auch „Trace" und „Breakpoint"-Betrieb möglich!

zu e) Mit FDOS (Disk-Operating System) stehen Ihnen zahlreiche Kommandos zum Arbeiten mit Disketten zur Verfügung, die Sie in dieser Ausführung selten irgendwo finden!

Mit „CMT" können fertige Maschinenprogramme auf Band gespeichert werden.

Lieferbar als Kassetten- und Diskettenversion 5 1/4; 3,5 Zoll für MZ 700 + MZ 800

Business Basic:

Basic Interpreter der dem Anwender 30 KB zur Verfügung stellt, um eigene Programme zu schreiben. (Inhalt dieser Programme z.B. Rechnungen, Lagerhaltung usw.) Mit dem Business Basic lassen sich die SFD, MFD und die Kassette ansprechen. Die Quick-Disc wird nicht unterstützt. Lieferbar als Diskettenversion 5 1/4, 3,5 Zoll für MZ 800.

MZ 700 Disk Basic für den MZ 800

Das von den Floppysystemen MFD 700 (3,5 Zoll) und SFD 700 (5,25 Zoll) bekannte Basic wurde an den MZ 800 angepaßt. Mit diesem Basic sind Ihre S-Basic Programme auf dem MZ 800 lauffähig. Lieferbar als Diskettenversion 5,25 und 3,5 Zoll.

Disk Basic MZ 800

Das Disk Basic, MZ-B-800 ist 100% Sharp kompatibel. Es hat jedoch einige weitere Vorteile zu bieten. Zum Beispiel deutsche Umlaute von der Tastatur und die Möglichkeit die Hintergrundfarbe zu verändern. Die Speicherkapazität auf einer Diskette wurde von 280 KB auf insgesamt 320 KB erhöht. Bei Directory wird die Dateigröße in Bytes ausgegeben. Ferner werden alle Geräte von Disk-Basic aus angesprochen (z.B. QD, CMT, RAM etc.). Das Disk Basic belegt ca. 44 KB. (5,25 + 3,5 Zoll)

Datenbank:

Adresskarteien, Mitgliederlisten, Schallplattenverzeichnisse, Lagerbestände und Artikeldateien werden mit diesem Programm aufgebaut und verwaltet. 500 Adressen oder 1000 Artikel lassen sich so in einer Kartei speichern und bearbeiten. Alle Funktionen werden über zwei Bildschirmmenues angewählt. Lieferbar als Kassetten- und Diskettenversion 5 1/4; 3,5 Zoll für MZ 700 + MZ 800.

Kassettensoftware:

Funktion Plott:

Mit diesem Programm können Sie alle Funktionen einer Veränderlichen in einem cartesischen Koordinaten-System darstellen. Es können bis zu 3 Funktionen parallel gezeichnet werden. Die Darstellung geschieht auf dem Bildschirm, die ausgabe ist auf einem graphikfähigen Matrix-Drukker möglich. Das Bildschirm-Format, (40- oder 80-Zeichen) ist hierbei frei wählbar. Die eigentliche Berechnung erfolgt automatisch nach Eingabe der x/y Intervallgrenzen und der Funktionen. Die Kompatibilität zu folgenden Geräten ist gewährleistet:

Epson (+Kompatibel MX, FX, RX)	ITOH 8510, Serien NEC PC 80 XX	Centronics GLP Brother 1009 (ML + Basic)

Hardcopy:

Ausgabe des Bildschirminhalts auf einem graphikfähigen Matrix-Drukker, auch der 80-Zeichen Modus wird komplett auf dem Drucker ausgegeben. Die Kompatibilität zu folgenen Geräten ist gewährleistet:

Epson (+Kompatibel MX, FX, RX)	ITOH 8510, Serien NEC PC 80 XX	Centronics GLP Brother 1009 (ML + Basic)

K&P
KERSTEN & PARTNER
DATENSYSTEME GMBH
Wildbacher Mühle 83 · D-5100 Aachen · Tel. 0241/77 10 67-8

Der neue PC-EMULATOR von Kersten & Partner macht Ihren SCHNEIDER Home Computer zu einem leistungsstarken 16 bit Personal Computer. Der PC-EMULATOR ist disk- und hardwarekompatibel zum IBM PC und macht Ihren SCHNEIDER zu einem voll kompatiblen PC-System. Der PC-EMULATOR läuft unter MS-DOS und erlaubt damit die Benutzung des umfangreichen IBM kompatiblen Softwareangebots, welches bei den meisten Softwarehäusern erhältlich ist.

Der PC-EMULATOR wird in zwei verschiedenen Ausbaustufen geliefert, der PCE-I und der PCE-II. Beide verfügen über 512 KB Hauptspeicher. Der PCE-I beinhaltet ein 5.25" Floppylaufwerk mit 360 KB Speicherkapazität. Eine Nummer größer als der PCE-I ist der PCE-II. Er beinhaltet zwei 5.25" Floppylaufwerke mit insgesamt 720 KB Speicherkapazität. Beide Modelle der PCE-I als auch der PCE-II bieten die Möglichkeit einer späteren Erweiterung durch den PC kompatiblen Steckplatz. Jedes System wird mit einem ausführlichen Bedienerhandbuch und einem technischen Handbuch mit den verschiedenen Anschlußbelegungen ausgeliefert.

Bei anderen Zusatzgeräten ist es oft nötig, den Rechner zu öffnen, dies gilt jedoch nicht für den PC-EMULATOR. Der PC-EMULATOR wird einfach an den SCHNEIDER Systembus angeschlossen. So sind Sie in wenigen Minuten in der PC-Welt zuhause. Der Anschluß der vorhandenen Peripherie ist problemlos, da der Systembus wieder herausgeführt wird.

Technical Specifications:

- 8088 16 bit processor running at 5 Mhz
- 512 KB user memory
- Model PCE-I - Single, double sided, 40 track, disc drive 5.25", 360 KB.
- Model PCE-II - Twin, double sided, 40 track, disc drive 5.25", 720 KB
- One PC compatible hardware expansion bus.
- Integral stabilised power supply, 220 V, 50/60 Hz.
- Keybord text and graphics supplied by the AMSTRAD.
- Software compatibility allows Lotus 1-2-3 and all popular PC business programs to run without modification, subject to the constraints of the AMSTRAD keyboard and display

Technische Daten:

- 8088 Prozessor mit 5 Mhz
- 512 KB Hauptspeicher
- Modell PCE-I mit einem 5.25" Floppylaufwerk, 40 Track, 360 KB.
- Modell PCE-II mit zwei 5.25" Floppylaufwerken, 40 Track, 720 KB.
- PC kompatibler Steckplatz.
- Eingebautes Netzteil, 220 V, 50/60 Hz.
- Tastatur, Text- und Graphikbildschirm durch den SCHNEIDER.
- Softwarekompatibilität erlaubt die Benutzung von Lotus 1-2-3 und allen anderen PC kompatiblen Programmen im Rahmen der Möglichkeiten der SCHNEIDER Tastatur und des Bildschirms.

The new Kersten & Partner PC-EMULATOR will upgarde your AMSTRAD micro to a powerful 16 bit personal computer.
Disc and hardware compatible with the IBM PC, the PC-EMULATOR turns your AMSTRAD into a fully PC compatible system. It uses MS-DOS operating system allowing to use the massive range of IBM compatible buisiness software, programming aids, compilers and languages available from most software houses.

The PC-EMULATOR offers two levels of upgrade, the PCE-I and the PCE-II, both with 512 KB on board user memory as standard.
The PCE-I, contains a single, double sided 360 KB 5.25" disc drive and provides the low cost introduction to the PC-world for the Amstrad user.
A step up from the PCE-I is the PCE-II which offers twin, double sided 360 KB 5.25" disc drives for more data storage. Both the PCE-I and the PCE-II provide the possibility of further expansions using the PC compatible slot provided by the PC-EMULATOR. Each system comes with a user/technical manual and connecting leads.

Unlike other add-ons there is no need to open the AMSTRAD to make the connection. The compact and tidy PC-EMULATOR models simply plug in to the AMSTRAD system bus. Within minutes you can be in the PC-world. All existing peripherals can be further connected to the PC-EMULATOR.

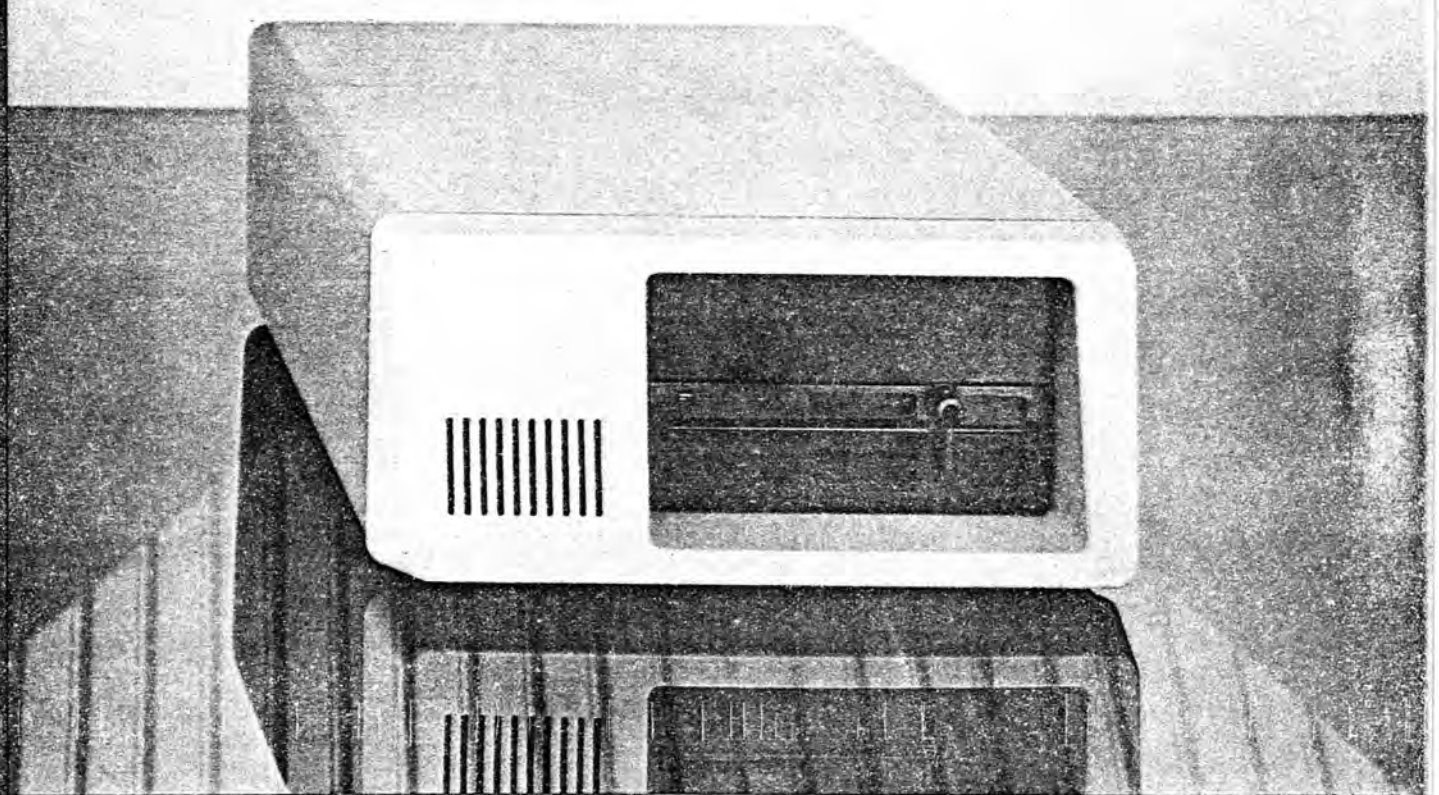
Der PC-EMULATOR ist für alle SCHNEIDER Typen lieferbar.
The PC-EMULATOR is available for all types of AMSTRAD micros.

ACHTUNG :
Nicht für Sharp-Rechner

KERSTEN & PARTNER
DATENSYSTEME GMBH

Händler/Dealer

PC EMULATOR



Der PC-EMULATOR. Eine Zusatzeinheit die Ihren SCHNEIDER zu einem IBM PC kompatiblen Rechner macht.
The PC-EMULATOR. The first IBM PC compatible upgarde for the AMSTRAD micros.