mc - CP/M-COMPUTER

EPF

Eprom-Floppy für den mc-Computer

Graf Elektronik Systeme GmbH

ges

EPROM-FLOPPY mit Hardware-Boot 10.4.1986

1. Einführung

1.1 Wozu dient die Baugruppe 'EPROM-Floppy' ?

Selbst bei Verwendung einer RAM-Floppy müssen die Programme und Daten zunächst mal von Floppy-Disk geladen werden. Auch jeder Warmstart erfordert einen Floppyzugriff mit entsprechenden Wartezeiten. Daher liegt es nahe, oft benötigte Programme wie Editor, Compiler, Assembler und Utilities in Form von Firmware bereitzuhalten. Hinzu kommt, daß EPROMs heutzutage in Kapazitäts- und Preisregionen vorgerückt sind, die einen Einsatz in einer EPROM-Floppy ermöglichen. Hier sind besonders die neuen, billigen Einmaleproms zu erwähnen, die keine Löschfenster mehr besitzen. Besonders in Verbindung mit einer RAM-Floppy und unter Ausnutzung des DMA-Controllers ist ein Arbeiten fast ohne I/O-Wartezeiten möglich, so wird z.B. der Macroassembler M80 in 70ms in den Arbeitsspeicher geladen und eine Kopie der gesamten 320k EPROM-Floppy in die RAM-Floppy erfolgt in ca. 7 Sekunden. Die hier vorgestellte ECB-Karte ist in erster Linie für den mc-CP/M-Computer gedacht, kann aber auch in anderen ECB-Systemen eingesetzt werden.

1.2 Wie setzt man die EPROM-Floppy ein?

Nach Austausch des Monitor-EPROMs erfolgt auf den I-Befehl die Auswahlmeldung 'EPROM = 1 , FLO2-Boot = 2, Winchester = 3, FLO1-Boot = 4. Wählt man die Möglichkeiten 2, 3 oder 4, funktioniert der Computer in gewohnter Weise, mit der Ausnahme, daß als Laufwerk C nun die EPROM-Floppy zur Verfügung steht. Um die Vorteile der EPROM-Floppy jedoch voll auszunutzen, bootet man normalerweise von der EPROM-Floppy durch Eingabe einer 1. Jetzt bildet die EPROM-Floppy das Laufwerk A. Die Floppy-Laufwerke melden sich unter C und D.

Mit Hilfe der EPROM-Floppy läßt sich ein Computersystem ohne Floppy Laufwerke realisieren, falls nur Festprogrammanwendung geplant ist oder der Rechner als Workstation im Rechnerverbund eingesetzt wird, wobei viele Rechner auf eine gemeinsame große Winchesterplatte Zugriff haben. In den meisten Fällen jedoch wird man wenigstens ein Floppy-Laufwerk anschließen, um das zu bearbeitende Programm in die RAM-Floppy zu laden. Anschließend erfolgt die Bearbeitung mit Hilfe der in der EPROM-Floppy gespeicherten Dienstprogramme unter voller Ausnutzung der Transfergeschwindigkeit.

In dieser Anleitung wird genau beschrieben, wie man das Betriebssystem CP/M und Programme seiner Wahl resident in der EPROM-Floppy ablegt.

Wichtiger Hinweis : CP/M 2.2 darf nur von lizensierten Benutzern auf Eproms gebracht werden!

2. Schaltungsbeschreibung

Im Anhang A ist die Schaltung der EPROM-Floppy abgebildet. Fünf Zähler vom Typ 74LS161 generieren die Zugriffsadresse und wählen den entsprechenden EPROM-Baustein aus. Dazu werden die Zähler über den Datenbus mit einer Adresse geladen, die die Startadresse des zu übertragenden Datenblocks darstellt. Dieser Ladevorgang für die Zähler ist für jeden zusammenhängenden Datenblock nur einmal nötig, alle folgenden Lese-, oder (im Falle des RAMs) Schreibzugriffe takten die Zähler um eine Adresse weiter. Die Adressenzuordnung der Zähler sowie des DMA-Controllers geschieht in bekannter Weise mit einem 4 Bit-Vergleicher und einem Dekoder vom Typ 74LS138. Tabelle 1 zeigt die Zuordnung von Adressen und Zählern.

X ≙ DIL-Schalterstellung

X0	:	EPROM-Adressen 12-18, Track-Anwahl
X2	:	EPROM-Adressen 4-11, Sektor-Anwahl
X6	:	EPROM-Adressen 0- 3, Bytes in Sektor
X5	:	EPROM-Daten
X8	:	DMA-Controller Adresse

Tabelle 1. Adreßbelegung der Platine

Die eigentliche Auswahl der EPROMs ist über je einen Dekoder vom Typ 74159 und 74156 realisiert. Diese Dekoder werden über die Adresse X0 aktiviert und teilen den gesamten Datenbereich in 16KByte große Blöcke auf. Je nachdem welche EPROM-Typen (27128 oder 27256) in welcher Reihenfolge bestückt werden, müssen entsprechende Brücken eingelötet werden. EPROMs vom Typ 27256 belegen zwei 16kByte Blöcke. Pin 27 ist dazu auf der Platine bereits mit der Adresse A14 verbunden. Bei Verwendung von EPROMs vom Typ 27128 muß diese Verbindung aufgetrennt und Pin 27 auf +5V gelegt werden. Werden keine EPROMs vom Typ 27256 verwendet, dann müssen auch die Brücken unter IC 27 aufgetrennt und entsprechend Tabelle 2b,c verdrahtet werden. Tabelle 2 zeigt die Bedeutung der Brücken, die Lage geht aus der folgenden Abbildung (Abb. 1) hervor. Ein Beispiel für eine gemischte Bestückung der Platine ist ebenfalls in Tabelle 2 zu finden. Sämtliche JMP (Brücken) sind mit Drahtbrücke einzustellen.

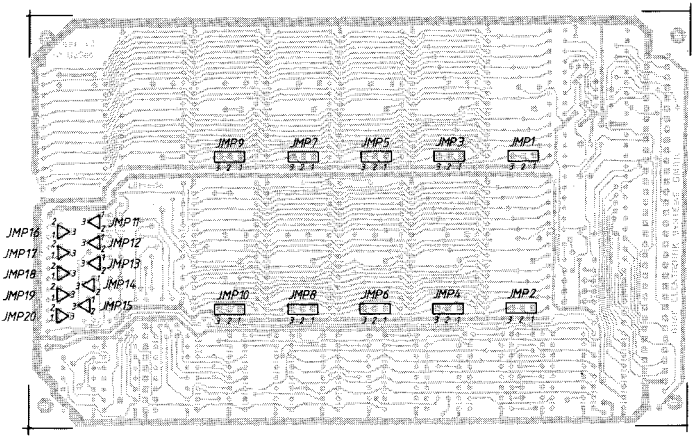


Abb. 1: Lage der JMP von der Lötseite der Leiterplatte gesehen.

1) Brücken unter den EPROMs:

Steckplatz	default 27256	27128
JMP1 (unter J4)	3 <-> 2	2 <-> 1
JMP2 (unter J5)	2 <-> 3	2 <-> 1
JMP3 (unter J8)	2 <-> 3	2 <-> 1
JMP4 (unter J9)	2 <-> 3	2 <-> 1
JMP5 (unter J12)	2 <-> 3	2 <-> 1
JMP6 (unter J13)	2 <-> 3	2 <-> 1
JMP7 (unter J15)	2 <-> 3	2 <-> 1
JMP8 (unter J16)	2 <-> 3	2 <-> 1
JMP9 (unter J19)	2 <-> 3	2 <-> 1
JMP10(unter J20)	2 <-> 3	2 <-> 1

2) Brücken unter IC 27:

a) Ausschließliche Verwendung von EPROMs vom Typ 27256:
(Defaulteinstellung)

für J13	JMP 20	1 <-> 2 <-> 3
für J8	JMP 19	1 <-> 2 <-> 3
für J9	JMP 18	1 <-> 2 <-> 3
für J4	JMP 17	1 <-> 2 <-> 3
für J5	JMP 16	1 <-> 2 <-> 3
für J16	JMP 11	1 <-> 2 <-> 3
für J15	JMP 12	1 <-> 2 <-> 3
für J12	JMP 13	1 <-> 2 <-> 3
für J20	JMP 14	1 <-> 2 <-> 3
für J19	JMP 15	1 <-> 2 <-> 3

b) Ausschließliche Verwendung von EPROMs vom Typ 27128:

für J13	JMP20/1 <-> JMP20/3
für J8	JMP20/2 <-> JMP19/3
für J9	JMP19/1 <-> JMP18/3
für J4	JMP19/2 <-> JMP17/3
für J5	JMP18/1 <-> JMP16/3
für J16	JMP18/2 <-> JMP11/3
für J15	JMP17/1 <-> JMP12/3
für J12	JMP17/2 <-> JMP13/3
für J20	JMP16/1 <-> JMP14/3
für J19	JMP16/2 <-> JMP15/3

c) Gemischte Bestückung:
Beispiel

J13 = 27128	:	JMP20/1 <-> JMP20/3
J8 = 27256	:	JMP20/2 <-> JMP19/1 <-> JMP19/3
J9 = 27256	:	JMP19/2 <-> JMP18/1 <-> JMP18/3
J4 = 27128	:	JMP18/2 <-> JMP17/3

Tabelle 2 : Brücken auf der Platine

Um die Stromaufnahme der Platine so gering wie möglich zu halten, werden alle EPROMs im stand-by-Modus betrieben, bis sie tatsächlich angesprochen werden. Dies bedingt bei 6 MHz-Betrieb jedoch EPROMs mit einer Zugriffszeit von 250ns. Ein weiterer Dekoder vom Typ 74159 unterteilt den untersten 16k Bereich in 1KByte große Teilbereiche. An Hand der Memory-Map

(Bild 2) erkennt man die Lage von CP/M und des Directories. Um Files im RAM ablegen zu können, muß auch ein Teil des Direktories im RAM liegen. Dieser RAM-Directory Teil muß dem EPROM-Directory vorangestellt werden, da CP/M den ersten freien Directory Platz verwendet und die Anzahl der EPROM-Directory Einträge variabel ist. Daraus erklärt sich die auf den ersten Blick etwas merkwürdige Dekodierung von RAM und System-EPROM sowie die Adressenumsetzung für das System-EPROM mit Hilfe der 4 Gatter. Diese Umsetzung bildet das EPROM-Directory, das auf Grund der Dekodierung eigentlich außerhalb des EPROM-Bereiches liegt, wieder in das EPROM ab.

KByte	Adresse	Baustein	Kommentar
1- 6	00000h-015FFh	System-EPROM	CCP, BDOS
7	01600h-01BFFh	System-EPROM	BIOS
8	01C00h-01FFFh	/	nicht belegt
9	02000h-023FFh	RAM	32 entries Directory
10	02400h-027FFh	System-EPROM	" im EPROM ab Adr.1C00h
11- 16	02800h-03FFFh	RAM	RAM z.B. für 'submit'
17- 32	04000h-07FFFh	Programm-EPROM	27128
33- 48	08000h-0BFFFh	Programm-EPROM	27128 >oder 1 mal 27256
49- 64	0C000h-0FFFFh	Programm-EPROM	27128
65- 80	10000h-13FFFh	Programm-EPROM	27128 >oder 1 mal 27256
81- 96	14000h-17FFFh	Programm-EPROM	27128
97-112	18000h-1BFFFh	Programm-EPROM	27128 >oder 1 mal 27256
113-128	1C000h-1FFFFh	Programm-EPROM	27128
129-144	20000h-23FFFh	Programm-EPROM	27128 >oder 1 mal 27256
145-160	24000h-27FFFh	Programm-EPROM	27128
161-176	28000h-2BFFFh	Programm-EPROM	27128 >oder 1 mal 27256
177-192	2C000h-2FFFFh	Programm-EPROM	27128
193-208	30000h-33FFFh	Programm-EPROM	27128 >oder 1 mal 27256
209-224	34000h-37FFFh	Programm-EPROM	27128
225-240	38000h-3BFFFh	Programm-EPROM	27128 >oder 1 mal 27256
241-256	3C000h-3FFFFh	Programm-EPROM	27128
257-272	40000h-43FFFh	Programm-EPROM	27128 >oder 1 mal 27256
273-288	44000h-47FFFh	Programm-EPROM	27128
289-304	48000h-4BFFFh	Programm-EPROM	27128 >oder 1 mal 27256
305-320	4C000h-4FFFFh	Programm-EPROM	27128
321-336	50000h-53FFFh	Programm-EPROM	27128 >oder 1 mal 27256

Bild 2 : Memory - Map der Baugruppe

Der DMA-Controller wird über Adresse X8 angesprochen. Erinnert sei an das Daisy-Chain-Prinzip der Interrupt-Priorisierung, in welches der DMA-Controller auch einbezogen werden muß (Leitungen IEI und IEO). Der DMA-Baustein wird leider nur als A-Version für max. 4MHz angeboten. Unser Baustein (SGS) arbeitete jedoch auch tadellos bei 6MHz, evtl. sollte man verschiedene Fabrikate ausprobieren. Der Refresh des Hauptspeicher ist bei DMA-Betrieb durch die sektorweise Datenanforderung durch das BDOS immer gewährleistet. Will man keinen DMA-Betrieb, wird dieser Baustein in der Platine nicht bestückt; ein entsprechendes BIOS ohne DMA ist im Anhang H zu finden.

3. Aufbauanleitung

3.1 CMOS-Warnung

CMOS-Bausteine sind hochempfindlich gegen elektrostatische Aufladung! Bewahren oder transportieren Sie CMOS-Bausteine nur auf dem leitenden Schaumstoff! Alle Pins müssen kurzgeschlossen sein.

Tip: Fassen Sie an ein geerdetes Teil (z.B. Heizung, Wasserleitung oder an den Schutzkontakt der Steckdose, bevor Sie einen Baustein berühren.

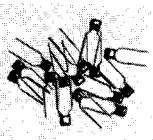
Bitte beachten Sie hierzu auch den Artikel "Schutzmaßnahmen für MOS-Schaltungen" in unserer Zeitschrift LOOP3.

Stückliste EPF Ausgabe 2
13.10.86 Bischof

1	Original GES-Platine mit Lötstoplack r2			
2	60731	74159	IC23, 27	4-Bit Binärdekoder
2	60077	74LS02	IC28, 10	4 NOR mit je 2 Eingängen
2	60079	74LS04	IC25, 7	6 Inverter
1	60138	74LS85	IC3	4 Bit Vergleicher
1	60094	74LS138	IC6	3-Bit Binärdekoder
1	60732	74LS156	IC24	2 2-Bit Binärdekoder
5	60100	74LS161	IC11,14,17,18,21	prog. 4 Bit Binärzähler
1	60115	74LS245	IC2	8 Bit Bus-Transceiver
1	10357	HM6264	IC22	statischer RAM 8k
1	60735	Z80 A DMA	IC1	Z80 DMA-Controller
1			VG	64polige Messerleiste
1	60294	DIP4	DIP	4-poliger DIL-Schalter
1	60733	8 * 3,9k	RN1	Netzwerkwiderstand 8x3,9k Ohm
15	60239	100 nF	C2...C16	Kondensatoren 100nF
19	60640	3,9k	R1...R19	Widerstände 3,9k Ohm
1	60270	47 uF	C1	Kondensator 47uF
12	60190	SO28		28polige Sockel
1	60193	SO40		40poliger Sockel
2	60188	SO24		24polige Sockel
1	60187	SO20		20poliger Sockel
8	60185	SO16		16polige Sockel
4	60183	SO14		14polige Sockel
1		Handbuch EPROM-Floppy		

Bauteile der Baugruppe EPF

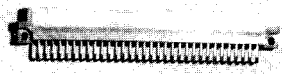
15 Kondensatoren 100 nF



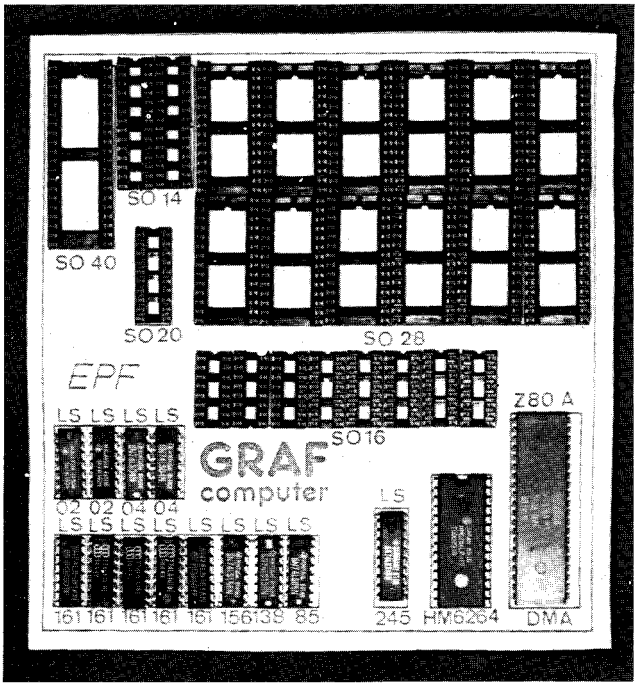
1 Kondensator 47 uF

1 4-poliger DIL-Schalter

1 64-polige Messerleiste



1 Netzwerkwiderstand 8*3,9 kOhm



3.3 Aufbau Schritt für Schritt

Achtung: Wie aus dem Bestückungsplan (Anhang B) hervorgeht, liegen einige Widerstände und Kondensatoren unter den ICs. Daher müssen Fassungen mit entsprechenden Aussparungen im Boden verwendet werden.

Auf einer Seite der Leiterplatte steht der Hinweis "löts" (Lötseite); auf dieser Seite wird ausschließlich gelötet. Die Bauteile sind nur auf der anderen Seite aufzustecken, der Bestückungsseite.

Vor dem Einlöten der Sockel sollten die unter den IC's liegenden Bauteile (siehe Bestückungsplan) eingesetzt und geprüft werden, ob sich die Sockel problemlos einsetzen lassen. Dies sind die Widerstände R1 bis R5 und R10 bis R19, sowie die Keramikkondensatoren C1 bis C14. Ist dies nicht der Fall (z. B. bei Billigsockeln), müssen die Sockel entsprechend zurechtgefeilt werden.

Nun wird die Leiterplatte mit den IC-Sockeln bestückt. Dabei muß darauf geachtet werden, daß die Sockel richtig aufgesteckt werden. Im Bestückungsplan sind die Richtungen mit einer Kerbe gekennzeichnet. Sie muß mit der Richtung der Kerbe in der Fassung übereinstimmen. Außerdem ist die Lage der Fassungen auch auf der Bestückungsseite der Platine durch den Aufdruck (falls vorhanden) sehr deutlich zu erkennen.

Es sollten alle Fassungen auf einmal aufgesteckt werden und zum Verlöten umgedreht werden; dabei ist es hilfreich, wenn man beim Umdrehen die Fassungen mit einem Stück Karton auf die Platine drückt. So wird erreicht, daß die Fassungen alle eben und gerade liegen. Beim Löten sollten wiederum nur zwei Pins jeder Fassung (möglichst diagonal) verlötet werden. So können anschließend schräg liegende Fassungen noch problemlos korrigiert werden. Bevor die restlichen Pins verlötet werden, sollte noch auf die Bestückungsseite geschaut werden, ob die Fassungen richtig liegen und die Richtungen der Fassungen stimmen.

Der Kondensator C1 ist gepolt und darf auf keinen Fall falsch herum eingelötet werden. Der Pluspol ist mit einem "+" und evtl. einem schwarzen Strich gekennzeichnet. Im Bestückungsplan ist der Pluspol ebenfalls mit einem "+" gekennzeichnet.

Die Kondensatoren C2-C16 sind ungepolt und können ohne auf die Polung zu achten eingelötet werden.

Die restlichen Widerstände R6 bis R9 werden stehend eingelötet. Farbcode der Widerstände: orange - weiß - rot.

Der Netzwerkwiderstand hat einen gemeinsamen Anschluß, der mit einem Punkt gekennzeichnet ist. Der Punkt am Bauelement muß mit dem Punkt am auf dem Bestückungsdruck übereinstimmen.

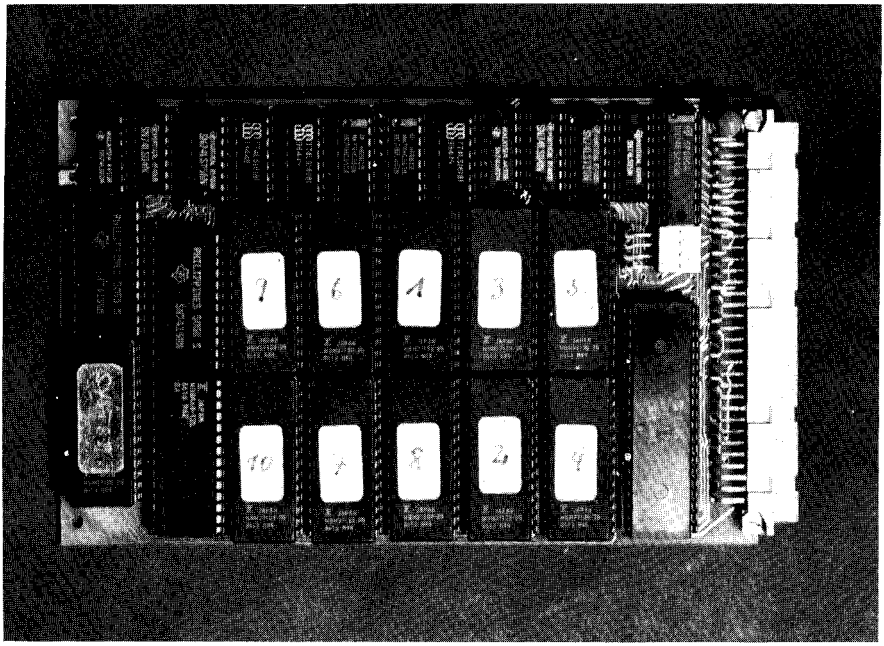
Der DIL-Schalter soll so eingesetzt werden wie auf dem Bestückungsdruck eingezeichnet. Falls Sie ihn anders herum einlöten erfüllt er zwar dieselbe Funktion aber die Ziffern

auf dem DIL-Schalter sind vertauscht.

Bevor man die Karte nun weiter bestückt, sollte man die Tests aus Kapitel 4.1 durchführen.

Sind die Test nach Kapitel 4.1 erfolgreich verlaufen, bestückt man alle TTL-ICs, das CMOS-RAM und den DMA-Controller. Die so bestückte Karte wird nun, nachdem man mit dem DIP-Schalter einen freien I/O-Adreßbereich ausgewählt hat (unsere Software - d.h. Monitor und BIOS - verwendet die Adressen 00 - 08), auf den Bus gesteckt und mit Hilfe des Monitors weiter getestet. Wie das gemacht wird ist in Kapitel 4.2 beschrieben.

Hat bis hierher alles geklappt, sind nur noch die EPROMs mit der entsprechenden Software zu bestücken. Alle dafür nötigen Schritte werden in Kapitel 5 gezeigt.



4. Testanleitung

4.1 Erste Prüfung ohne ICs

Die Platine ist bis jetzt erst mit den Sockeln und mit den passiven Bauelementen bestückt. Mit diesem Aufbau wird der erste Test durchgeführt.
Zu diesem Test muß die Baugruppe in den Bus gesteckt werden. Achten Sie beim Einstecken darauf, daß Sie die Baugruppe richtig herum einsetzen. Ein falsches Einstecken, z.B. um ein Pin zu weit rechts kann zu Kurzschlüssen führen und kann Bauelemente zerstören.

Nach dem Einstecken der Leiterplatte muß der Rechner weiter problemlos funktionieren. Falls nein - weiter mit Kapitel 6.

Messen Sie zunächst, ob an allen IC-Sockeln die Versorgungsspannung von +5V anliegt. Dabei liegt bei Standard-TTL-Bausteinen jeweils am letzen Pin einer Fassung (z.B. bei 14-poligen an Pin 14) die Versorgungsspannung von +5V. 0V bzw. Masse liegt jeweils auf dem letzten Pin der ersten Reihe (bei 14-poligen auf Pin 7, bei 16-poligen auf Pin 8, bei 20-poligen auf Pin 10).

Achtung: Bei Speicher-IC's oder anderen (nicht TTL-) Bauelementen kann die Versorgungsspannung an anderen Pins liegen!

Liegt die Versorgungsspannung +5V und 0V (Masse) an den richtigen Pins an, dann kann die Karte, wie in Kapitel 3 beschrieben, mit allen TTL-ICs, dem CMOS-RAM und dem DMA-Controller bestückt werden. Dabei muß auf die Richtung der ICs geachtet werden. Die Markierung auf dem IC muß mit der Kerbe in der Fassung übereinstimmen.
Die Karte wird nun, nachdem man mit dem DIP-Schalter einen freien I/O-Adreßbereich ausgewählt hat (unsere Software - d.h. Monitor und BIOS - verwendet die Adressen 00 - 08), auf den Bus gesteckt und mit Hilfe des Monitors (s.u.) weiter getestet. Wie das gemacht wird steht in Kapitel 4.2.

4.2 Test der EPROM-Floppy mit Hilfe des Monitors

Auch nach dem Bestücken der TTL-ICs, des RAMs und des DMA-Controllers sollte sich der Rechner nach dem Einschalten der Versorgungsspannung wie bisher melden. Ist das nicht der Fall, überprüft man die Richtung des Bustreibers 74LS245, Pin 1 muß Low-Pegel haben, oder man entfernt den DMA-Controller noch einmal um den Fehler zu lokalisieren. Hilfestellung bei der Fehlersuche ist in Kapitel 6 zu finden.

Ist soweit alles klar, kann man mit den Monitorbefehlen an Hand von Tabelle 3 Schreib- und Lesezugriffe in's RAM durchführen. Dabei werden außer dem CMOS-RAM selber, die Dekodierung, die Zähler und der Bustreiber getestet:

```
 1) Auf RAM-Bereich positionieren
Q006 00          : Low Byte der Adressen
Q002 00          : Sektor-selekt
Q000 02          : Track-select(2 tracks offset)
```

```
 2) Werte ins RAM einschreiben
Q005 AA
Q005 55
Q005 FF
Q005 00......

 3) Wieder  neu positionieren und oben eingeschriebene  Werte
    auslesen
Q006 00          : Low Byte der Adressen
Q002 00          : Sektor-selekt
Q000 02          : Track-select
QI05             : Auslesen hier AA
QI05                             55
QI05                             FF
QI05....                         00   usw.
```

Tabelle 3 : Test mit Hilfe des Monitor

Mit Hilfe eines Debugger-Programms, z.B. DDT oder ZSID, und des Testprogramms aus Bild 5 kann nun der DMA-Controller getestet werden: Nachdem man das Testprogramm eingegeben hat legt man beliebige 128 Bytes, die man per DMA-Zugriff in das 6264 RAM transferieren möchte, ab der Bufferadresse 1000H im Speicher ab. Die Variablen IOT und IOS sind so eingestellt, daß ein Bereich innerhalb des RAMs angesprochen wird.

* Dann startet man mit 'G100,10D' auf Adresse 100H und legt einen Breakpoint auf Adresse 10DH. Beobachtet man mit einem Oszilloskop gleichzeitig die Busleitung 11a (BSREQ), sollte dort ein Impuls aufgetreten sein, der DMA-Aktivität verrät.
* Nun wird der Buffer überschrieben.
* Die Leseoperation 'G11C,129 stellt den alten Bufferinhalt wieder her.

Dieser Test sollte zunächst mit 4MHz durchgeführt werden, da nicht alle DMA-Controller mit einer Taktfrequenz von 6MHz arbeiten. Eine weitere Fehlerquelle kann eine falsche I/O-Adresseneinstellung sein. Ist die Karte soweit fehlerfrei, kann mit der Programmierung der EPROMs begonnen werden.

```
0000'                   ASEG
                        ORG     100H
                        .Z80
1000            BUFF    EQU     1000H       ;DMA-BUFFERADRESSE
0006            LOSEPR  EQU     006H
0002            HISEPR  EQU     002H
0000            TRAEPR  EQU     000H
0005            EPRFL   EQU     005H
                        ;
                        ;EPROM - FLOPPY
                        ;
0100 CD 013A    REPROM: CALL    EPROAD      ;Zähler setzen
0103 06 0E              LD      B,14D       ;Tabellenlaenge
0105 0E 08              LD      C,08H       ;DMA-Controller Port
0107 21 010E            LD      HL,DMARD    ;Befehlstabellenanfang
010A ED B3              OTIR                ;Controller programmieren
010C AF                 XOR     A
010D C9                 RET
                        ;  Befehle für DMA-Controller 'LESEN'
010E C3         DMARD:  DB      0C3H        ;WR6 RESET
010F 79                 DB      079H        ;WR0 B->A
0110 1000               DW      BUFF        ;DMA-BUFFERADRESSE
0112 7F                 DB      07FH        ;BLOCKLAENGE - 1
0113 00                 DB      0
0114 14                 DB      014H        ;WR1: A-MEMORY,INC ADRESS
0115 28                 DB      028H        ;WR2: B-I/O CONST ADRESS
0116 C5                 DB      0C5H        ;WR4: BURST-MODE
0117 05                 DB      005H        ;WR4: I/O-PORT ADRESS
0118 82                 DB      082H        ;WR5: STOP AM BLOCKENDE
0119 CF                 DB      0CFH        ;BEFEHL 'LADEN'
011A B3                 DB      0B3H        ;FORCE READY
011B 87                 DB      087H        ;ENABLE DMA
                        ;
011C CD 013A    WEPROM: CALL    EPROAD      ;Zähler setzen
011F 06 10              LD      B,16D       ;Tabellenlänge
0121 0E 08              LD      C,08H       ;DMA-Controller Port
0123 21 012A            LD      HL,DMAWR    ;Befehlstabellenanfang
0126 ED B3              OTIR                ;Controller programmieren
0128 AF                 XOR     A
0129 C9                 RET
                        ;  Befehle für DMA-Controller 'SCHREIBEN'
012A C3         DMAWR:  DB      0C3H        ;WR6 RESET
012B 79                 DB      079H        ;WR0 B->A
012C 1000               DW      BUFF        ;DMA-BUFFERADRESSE
012E 7F                 DB      07FH        ;BLOCKLAENGE - 1
012F 00                 DB      0
0130 14                 DB      014H        ;WR1: A-MEMORY,INC ADRESS
0131 28                 DB      028H        ;WR2: B-I/O CONST ADRESS
0132 C5                 DB      0C5H        ;WR4: BURST-MODE
0133 05                 DB      005H        ;WR4: I/O-PORT ADRESS
0134 82                 DB      082H        ;WR5: STOP AM BLOCKENDE
0135 CF                 DB      0CFH        ;BEFEHL 'LADEN'
0136 05                 DB      005H        ;WR0: A->B
0137 CF                 DB      0CFH        ;BEFEHL LADEN
0138 B3                 DB      0B3H        ;FORCE READY
0139 87                 DB      087H        ;ENABLE DMA
                        ;
013A 3E 00      EPROAD: LD      A,0          ; Sektor 0..31a128 byte
013C D3 06              OUT     (LOSEPR),A ; Bits 0 bis 3
013E 3A 0151            LD      A,(IOS)     ; 128 byte = 7 bit
```

```
0141 CB 27              SLA   A           ; nur noch 5  bits hier
0143 CB 27              SLA   A           ; 3 bit nach oben
0145 CB 27              SLA   A
0147 D3 02              OUT   (HISEPR),A  ; Bits 4 bis 11
0149 3A 014F            LD    A,(IOT)     ; Track 0...(DSK/2-1)
014C D3 00              OUT   (TRAEPR),A  ; Bits 12 bis 18
014E C9                 RET
014F 0004       IOT:    DW    4
0151 0001       IOS:    DW    1
                        END
```

Bild 5 : Testprogramm für den DMA-Controller

5. Software

5.1 Monitoränderung

Um von der EPROM-Floppy booten zu können ist natürlich eine kleine Änderung im Monitor nötig. Dazu dient das Programmsegment in Bild 6. Eventuell müssen die Adressen an die jeweilige DIP-Schalterstellung angepaßt werden.

```
        ASEG
        ORG 0F7CCH
        ;************************************
        ;*          MONITOR - PATCH         *
        ;*                                  *
        ;* BOOT-Programm fuer EPROM - Floppy *
        ;* Ersetzt den Code im neuen Monitor *
        ;* ab Adresse F7CCH                 *
        ;*                MICHAEL GILGE     *
        ;************************************
        ;
FLPMSG:
        DEFB 0DH,0AH
        DEFM "EPROM-Boot = 1, FLO2-Boot = 2, "
        DEFM "Winchester = 3, FLO1-Boot = 4 : "
        FLPL    EQU $-FLPMSG
        ;
BOOT1:
        CALL    INIFLP          ;Vektoren etc definieren
        LD      HL,FLPMSG
        LD      B,FLPL          ;Meldung ausgeben
        CALL    TOM             ;dann Antwort abwarten
        CALL    CI
        AND     7FH             ;ohne Paritaet
        LD      C,A
        CALL    CO
        CP      "I"             ;I-Vektor fuer Benutzerbefehl
        JP      Z,IEXEC
        CP      "1"             ;EPROM-Boot
        JP      Z,EPRBOO
        CP      "3"             ;Winchester Boot
        JP      Z,WIEX
        CP      "4"             ;Boot mit FLO1
        JP      NZ,FLO2BO
        LD      A,1
        LD      (FLOCO),A       ; 1 = FLO1
        JP      BOOTX

FLO2BO: CP      '2'             ;Boot mit FLO2
        JP      NZ,NOBOOT
        XOR     A
        LD      (FLOCO),A       ; 0 = FLO2
        JP      BOOTX

NOBOOT: JP      ERROR

BOOTX:                          ; automatisch Format bestimmen
        LD      C,00000001B     ; immer von Laufwerk A  Seite 0
        CALL    GETFLOP         ; Formatierung bestimmen
        JP      C,ERROR         ; bei Fehler zurueck ins Menue
        LD      HL,80H          ; bei einfacher Dichte nöach 80h
```

```
        BIT     4,C             ; =0, dann dd, sonst sd
        JR      NZ,EXEC1        ; sd
        LD      HL,0FC00H       ; sonst 1K Block ggf. dorthin.
EXEC1:  PUSH    HL              ; rev (1.5) 1.1
        LD      B,1             ; lesen
        LD      D,0             ; track 0
        LD      E,1             ; sektor 1
        CALL    FLOPPY
        POP     HL              ; Sprungziel zurueck
        JP      NZ,ERROR
        JP      (HL)            ; Starten sonst
        ;
WIDAT:  DEFB    0,0,0,0         ; Track=0 Sektor=0

WIEX:   LD      HL,80H          ; Boot 128 Bytes auf Adresse 0
        LD      DE,WIDAT
        LD      B,1             ; lesen
        CALL    HDSYS           ; start
        JP      NZ,ERROR        ; Fehler beim Boot
        JP      80H             ; und ausfuehren

        ;**************************************
        ;*                                    *
        ;* BOOT-Programm fuer EPROM - Floppy  *
        ;*                                    *
        ;**************************************
        ;
        LOSEPR  EQU     006H    ;EPROM-FLOPPY INSIDE SEKTOR
        HISEPR  EQU     002H    ;             SEKTOR-SELECT
        TRAEPR  EQU     000H    ;             TRACK-SELECT
        EPRFL   EQU     005H    ;             DATEN
EPRBOO:
        LD      A,0             ; FESTLEGUNG DES FLOPPYCONTR.
        LD      (FLOC0),A       ; 1 = FLO1 , 0 = FLOSASI
        LD      HL,0D400H       ; CP/M - BASE
        LD      D,38H/2         ; # SEKT. a 256BYTE=1C00 BYTE
        LD      A,0             ; =28 STUECK  CCP, BDOS, BIOS
        OUT     (LOSEPR),A      ; ZAEHLER SETZEN
        OUT     (HISEPR),A      ; BEGINN BEI 0000H
        OUT     (TRAEPR),A
        LD      C,EPRFL         ; ADRESSE PORT ZUM LESEN
EPRSEC: LD      B,0             ; EIN SEKTOR
        INIR
        DEC     D
        JP      NZ,EPRSEC
        JP      0EA00H          ; START BIOS COLDBOOT
        ;
```

Bild 6. Neuer "Boot-Teil" im Monitor für Floppy-, Winchester- und EPROM-Boot. Diesen Programmteil im Monitorsourcefile anstelle der ursprünglichen Boot-Routinen eingeben.

Nach Einsetzen des neuen Monitor-EPROMs auf der CPU-Platine und Anwahl von 'I2' sollte der Floppy-Boot wie bisher funktionieren.

Nach dem im nächsten Abschnitt beschriebenen Programmieren des System-EPROMs, kann dann auch mit 'I1' von der EPROM-Floppy gebootet werden.

5.2 CP/M-System im EPROM

Im System-EPROM werden neben CCP/BDOS und BIOS auch das Directory der EPROM-Floppy abgelegt. Zuerst sollte man CCP und BDOS als COM-File auf der Diskette ablegen. Dies geschieht mit Monitorbefehlen und anschließendem 'SAVE' wie in Tabelle 5 gezeigt oder mit Hilfe von 'SYSGEN':

1. Schritt: CP/M booten
2. Schritt: Reset des Computers, CP/M bleibt im Speicher
3. Schritt: mit Monitor CCP und BDOS nach 100h verschieben mit ' MD400,E9FF,100 '
4. Schritt: CP/M erneut booten
5. Schritt: auf Diskette schreiben mit ' SAVE 22 SYSTEM.COM '

Tabelle 5 : Ablegen von CCP und BDOS auf Diskette

Nun wird das neue BIOS (Anhang F) eingegeben, assembliert und gelinkt oder an Hand des Hex-Dumps eingetippt. Will man keinen DMA-Controller verwenden, müssen die unten abgebildeten Routinen aus Bild 8 anstelle der entsprechenden aus Anhang F verwendet werden; außerdem entfallen dann die Tabelle 'INITAB' (Adresse EB2FH bis EB36H) und der Code von EBF5H bis EBFDH (DMA-Controller Ini).Der Warm-Boot bleibt in beiden Fällen gleich, da er mit Blockeingabe Befehlen realisiert ist.

```
;
;EPROM - FLOPPY
;
REPROM: CALL    EPROAD      ;ZÄHLER SETZEN
        LD      B,128       ;EIN SEKTOR
        LD      C,EPRFL     ;ADRESSE DER EPROM-FLOPPY
        LD      HL,(IOD)    ;ZIEL-ADRESSE
        INIR                ;BLOCK-EINGABE
        XOR     A
        RET
;
WEPROM: CALL    EPROAD      ;ZÄHLER SETZEN
        LD      B,128       ;EIN SEKTOR
        LD      C,EPRFL     ;ADRESSE DER EPROM-FLOPPY
        LD      HL,(IOD)    ;QUELLEN-ADRESSE
        OTIR                ;BLOCK-AUSGABE
        XOR     A
        RET
;
;RAM - FLOPPY
;
RDRFL:  CALL    RFLADR      ;ZÄHLER SETZEN
        LD      B,128       ;KOPIEREN VON 128 BYTE
        LD      C,0B7H      ;ADRESSE DER RAM-FLOPPY
        LD      HL,(IOD)    ;ZIEL-ADRESSE
        INIR                ;BLOCK EINGABE
        XOR     A
        RET
WRRFL:  CALL    RFLADR      ;ZÄHLER SETZEN
        LD      B,128       ;KOPIEREN VON 128 BYTE
        LD      C,0B7H      ;ADRESSE DER RAM-FLOPPY
        LD      HL,(IOD)    ;QUELLEN-ADRESSE
        OTIR                ;BLOCK EINGABE
```

```
        XOR     A
        RET
```

Bild 8 : Eprom-Floppy- und Ram-Floppy-Routinen ohne DMA-Betrieb, Änderungen gegenüber Bild 7

Um beim Booten von Floppy nicht auf die Programme in der Eprom-Floppy verzichten zu müssen, ist im Anhang G ein BIOS-Dump zur Ablage auf Diskette abgedruckt. Dieses BIOS bedient die EPROM-Floppy als Laufwerk C, und die RAM-Floppy als Laufwerk D. Tabelle 6 stellt die Laufwerkszuordnungen beim Booten von Diskette bzw. von der EPROM-Floppy zusammen.

Laufwerk:	BIOS nach Bild 7	BIOS nach Bild 9
A>	Eprom-Floppy, max 328kByte + 7kByte für CP/M-System DMA-Control.oder Block-I/O	5-1/4 Zoll-Laufwerk 780 kBytes,4 track offset 1024bytes/sector,160tracks
B>	RAM-Floppy, 1MByte Zugriff auf Files mittels DMA-Control.oder Block-I/O	^ dto.
C>	5-1/4 Zoll-Laufwerk 780 kBytes,4 track offset 1024bytes/sector,160tracks	Eprom-Floppy, max 328kByte + 7kByte für CP/M-System DMA-Contr. oder Block-I/O
D>	^ dto.	RAM-Floppy, 1MByte Zugriff auf Files mittels DMA-Contr. oder Block-I/O
E>	8 Zoll Laufwerk,single density,single side, IBM-Format	
F>	Winchester Laufwerk	

Tabelle 6 : Übersicht über Laufwerkszuordnungen

Unabhängig von der verwendeten BIOS-Version muß auf Adresse EAA7H im Dump des BIOS aus Anhang G der Wert der Variablen 'SIZE' angepaßt werden (jetzt :00A3) oder der Wert muß im Macroaufruf DISKDEF entsprechend eingesetzt werden. In Abhängigkeit von der Anzahl und den verwendeten EPROM-Typen, kann man Tabelle 7 den Wert der Variablen 'SIZE' entnehmen. Dieser Wert entspricht der Kapazität der EPROM-Floppy als Vielfaches der Blockgrösse von 2KByte.
Beispiel: Man verwendet 5 x 27256 und 1 x 27128, dies ergibt für SIZE den Wert 92 entsprechend 184 KByte. Dieser Wert enthält schon den Platz für das Directory von 2KByte und 6KByte RAM.

Variable SIZE für DISKDEF im BIOS

Anzahl 32k EPROMs:	0	1	2	3	4	5	6	7	8	9	10
Anzahl 16k EPROMs: 0	4	20	36	52	68	84	100	116	132	148	164
1	12	28	44	60	76	92	108	124	140	156	
2	20	36	52	68	84	100	116	132	148		
3	28	44	60	76	92	108	124	140			
4	36	52	68	84	100	116	132				
5	44	60	76	92	108	124					
6	52	68	84	100	116						
7	60	76	92	108							
8	68	84	100								
9	76	92									
10	84										

Tabelle 7. Speichergrösse für das BIOS

Nun muß noch das Directory als File auf der Diskette angelegt werden. Hier gibt es zwei Möglichkeiten:

1) Man gibt an Hand von Bild 10 das Directory von Hand ein; dabei dient Bild 10 nur als Anhaltspunkt für die Form des Directory, schließlich will man ja seine eigenen 'Lieblingsprogramme' in die EPROM-Floppy schießen. Die ersten 3 Blocks sind dabei für den RAM-Bereich reserviert (File 'DUMMY').

```
1C00 E544554D 4D592020 2046494C 00000030    .DUMMY   FIL...0
1C10 01020300 00000000 00000000 00000000    ................
1C20 00444920 20202020 20C34F4D 00000018    .DI      .OM....
1C30 04050000 00000000 00000000 00000000    ................
1C40 00504950 20202020 20C34F4D 0000003A    .PIP     .OM...:
1C50 06070809 00000000 00000000 00000000    ................
1C60 00535441 54202020 20C34F4D 00000029    .STAT    .OM...)
1C70 0A0B0C00 00000000 00000000 00000000    ................
1C80 00535542 4D495420 20C34F4D 0000000A    .SUBMIT  .OM....
1C90 0D000000 00000000 00000000 00000000    ................
1CA0 00535745 45503330 20C34F4D 0100004F    .SWEEP30 .OM...O
1CB0 0E0F1011 12131415 16171819 1A000000    ................
1CC0 005A5349 44202020 20C34F4D 00000050    .ZSID    .OM...P
1CD0 1B1C1D1E 1F000000 00000000 00000000    ................
1CE0 00444454 5A202020 20C34F4D 0000004F    .DDTZ    .OM...O
1CF0 20212223 24000000 00000000 00000000     !"#$...........
1D00 00444553 504F4F4C 20C34F4D 00000014    .DESPOOL .OM....
1D10 25260000 00000000 00000000 00000000    %&..............
1D20 004C3830 20202020 20C34F4D 0000003B    .L80     .OM...;
1D30 2728292A 00000000 00000000 00000000    '()*............
1D40 004D3830 20202020 20C34F4D 0100000E    .M80     .OM....
1D50 2B2C2D2E 2F303132 33000000 00000000    +,-./0123.......
1D60 004D4241 53494320 20C34F4D 0100003B    .MBASIC  .OM...;
1D70 34353637 38393A3B 3C3D3E3F 00000000    456789:;<=>?....
1D80 00575320 20202020 20C34F4D 0100000A    .WS      .OM....
1D90 40414243 44454647 48000000 00000000    @ABCDEFGH.......
1DA0 0057534D 53475320 20CF5652 0100006B    .WSMSGS  .VR...k
1DB0 494A4B4C 4D4E4F50 51525354 55565700    IJKLMNOPQRSTUVW.
1DC0 0057534F 564C5931 20CF5652 01000080    .WSOVLY1 .VR....
1DD0 58595A5B 5C5D5E5F 60616263 64656667    XYZ[\]^_`abcdefg
1DE0 0057534F 564C5931 20CF5652 02000006    .WSOVLY1 .VR....
```

```
1DF0 68000000 00000000 00000000 00000000    h...............
1E00 00463830 20202020 20C34F4D 01000055    .F80      .OM...U
1E10 696A6B6C 6D6E6F70 71727374 75760000    ijklmnopqrstuv..
1E20 00464F52 4C494220 20D2454C 0100004F    .FORLIB  .EL...O
1E30 7778797A 7B7C7D7E 7F808182 83000000    wxyzäöüß........
1E40 00544C49 53542020 20C34F4D 00000077    .TLIST   .OM...w
1E50 84858687 88898A8B 00000000 00000000    ................
1E60 00545552 424F2020 20C34F4D 0100006E    .TURBO   .OM...n
1E70 8C8D8E8F 90919293 94959697 98999A00    ................
1E80 00545552 424F2020 20CD5347 0000000C    .TURBO   .SG....
1E90 9B000000 00000000 00000000 00000000    ................
1EA0 00545552 424F2020 20CF5652 00000008    .TURBO   .VR....
1EB0 9C000000 00000000 00000000 00000000    ................
1EC0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1ED0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1EE0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1EF0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F00 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F10 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F20 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F30 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F40 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F50 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F60 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F70 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F80 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1F90 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FA0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FB0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FC0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FD0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FE0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
1FF0 E5E5E5E5 E5E5E5E5 E5E5E5E5 E5E5E5E5    ................
```

Bild 10 : Sample Direktory der Eprom-Floppy
im System-Eprom im letztem 1kByte Bereich
d.h. ab Adresse 1C00h

2) Eine anderer Weg besteht darin, alle Files, die in die EPROM-Floppy sollen auf einer neuen Diskette zusammenzustellen, z.B. auf Laufwerk B. Das Programm aus Bild 11 liest nun das Directory dieses Laufwerks. Beachten muß man allerdings, daß zuerst ein 6KByte Dummy-File auf der Diskette angelegt wird (siehe unter 1) und daß die Blockanzahl von Laufwerk B kleiner als 255 ist. Anderenfalls werden im Directory 4 Bytes pro Blockzuweisung statt nur 2 Bytes verwendet (die EPROM Floppy bleibt auf jeden Fall unter 255 Blocks). Ausweg falls >255 Blöcke/disk : Vorübergehende Änderung der Diskparameter für Laufwerk B.

```
        ;************************************
        ; PROGRAMM BRINGT DAS DIRECTORY DER *
        ; DISKETTE IM LAUFWERK B NACH 100H  *
        ; BIS 4FFH ENTSPR. 1K DIRECTORY     *
        ;************************************
        ;
        .Z80
CONOUT  EQU     0F009H
        ;
        LD      HL,100H         ; NACH 100 LADEN
        LD      D,2             ; DIREKTORY AUF TRACK 2
```

```
        LD      E,1             ; 1 SEKTOR a 1024 Bytes
RDSEC:  PUSH    HL
        PUSH    DE
        PUSH    BC
        LD      C,11010010B     ;LAUFWERK 1 DOUBLE DENSE MINI
        LD      B,1             ;READ
        CALL    0F027H
        POP     BC
        POP     DE
        POP     HL

        LD      HL,MSG
        CALL    PRMSG
        JP      0

PRMSG:  LD      A,(HL)
        OR      A
        RET     Z
        PUSH    HL
        LD      C,A
        CALL    CONOUT
        POP     HL
        INC     HL
        JP      PRMSG
        ;
MSG:    DB      0DH,0AH
        DB      "LOADING DIRECTORY OF DRIVE B TO 100h - 04FFh"
        DB      0DH,0AH
        DB      "SAVE IT WITH 'SAVE 4 DIR.DAT' "
        DB      0
        ;
        END
```

```
0500 21 00 01 16 02 1E 01 E5 D5 C5 0E D2 06 01 CD 27 !..............'
0510 F0 C1 D1 E1 21 2A 05 CD 1D 05 C3 00 00 7E B7 C8 ....!*.......ß..
0520 E5 4F CD 09 F0 E1 23 C3 1D 05 0D 0A 4C 4F 41 44 .O....#.....LOAD
0530 49 4E 47 20 44 49 52 45 43 54 4F 52 59 20 4F 46 ING DIRECTORY OF
0540 20 44 52 49 56 45 20 42 20 54 4F 20 31 30 30 68  DRIVE B TO 100h
0550 20 2D 20 30 34 46 46 68 0D 0A 53 41 56 45 20 49  - 04FFh..SAVE I
0560 54 20 57 49 54 48 20 27 53 41 56 45 20 34 20 44 T WITH 'SAVE 4 D
0570 49 52 2E 44 41 54 27 20 00 04 00 00 00 00 00 00 IR.DAT' ........
```

Bild 11 : Programm zum Zugriff auf das directory einer Floppy
(nur 1K da EPROM-Floppy nur 32 Einträge im EPROM hat)

Endlich ist es soweit. CCP, BDOS , BIOS und das Directory werden mit dem Debugger zusammengestellt und in ein EPROM vom Typ 2764 programmiert.

Es kann auch ein EPROM vom Typ 27128 verwendet werden, dann muß allerdings noch Pin 26 = A13 entsprechend auf Low- oder High-Pegel (default A13=low durch Platinenlayout) gelegt werden. So können zum Beispiel zwei verschiedene Betriebssystemvarianten (mit/ohne DMA) durch Umlöten von Pin 27 am System-EPROM ausgewählt werden.

Nun muß es möglich sein, von der EPROM-Floppy aus zu booten. Mit dem 'DIR'-Befehl sollten alle in das Directory eingegebenen Programmnamen erscheinen, das Programm STAT muß mindestens noch 6kByte Platz ausweisen. Ein weiterer Test ist das

Kopieren eines kleinen (≤ 6KByte) Files in den RAM-Bereich der EPROM-Floppy und die anschließende Überprüfung, ob der Name des Files richtig in das Directory eingetragen wurde und das File selber richtig abgespeichert wurde.

Die folgende Tabelle faßt die Schritte, die zur Inbetriebnahme der Karte notwendig sind noch einmal zusammen:

1) Monitor ändern, assemblieren, linken (oder patchen) und in neues Eprom programmieren
2) Neues oder geändertes BIOS assemblieren, linken und auf Diskette sichern
3) CCP und BDOS mit Befehlen nach Tabelle 5 auf Diskette ablegen
4) Directory editieren oder nach Zusammenstellung der gewünschten Files mit Programm nach Bild 11 lesen
5) Mit Debugger CCP, BDOS, BIOS und das Directory im Speicher zusammenstellen und auf Diskette sichern
6) Dieses File in ein 8kByte Eprom programmieren

7) Eprom auf die Platine setzen, Ram 6264 bestücken, Monitor austauschen , Reset des Rechners
8) Mit I-Befehl und nachfolgender '1' aus Eprom-Floppy booten, 'dir'-Befehl sollte alle Directoryeinträge erscheinen lassen
9) Mit Debugger 16kByte oder 32kByte (je nach Epromtyp) Programmcode in der Reihenfolge des Directory durch laden und verschieben zusamenstellen
10) Eprom programmieren und nach reset testen ob Programme vorhanden und lauffähig sind
11) Schritte 9) und 10) wiederholen bis alle Programme als Firmware vorliegen

5.3 Aus Software wird Firmware

Ist soweit alles zufriedenstellend verlaufen, kann mit der Fleißarbeit begonnen werden, die gewünschten Programme in EPROMs abzulegen. Dazu stellt man die Programme hintereinander mit dem Debugger an Hand des Direktories zusammen. Es ist zu beachten, daß die kleinste Einheit 1 Block d.h. 2KByte ist.

Beispiel:

```
Programm A, 12K : 1000H - 3FFFH
Programm B,  2K : 4000H - 47FFH    > 1x 27256
Programm C, 30K : 4800  - 9000
                  H       H
```

Der Rest des Programmes C wird zu Beginn des folgenden EPROMs programmiert. Sollten Sie kein Programmiergerät für EPROMs der Typen 2764/27128/27256 besitzen, aber ein Programmiergerät für die Typen 2516/2532 Ihr Eigen nennen, ermöglichen Ihnen die Schaltungen im Anhang E diese EPROMs dennoch zu programmieren. Allerdings dauert der ganze Vorgang dann etwas länger, da kein Schnellprogrammieralgorithmus verwendet werden kann. Die Schaltungen programmieren die EPROMs in 8 Bereichen zu je 2 bzw. 4KByte.

6. Fehlersuchanleitung

Sollte Ihre Baugruppe bei den in Kapitel 5 beschriebenen Tests nicht funktionieren, so heißt es jetzt systematisch auf Fehlersuche zu gehen.

Wir wollen Ihnen nun ein paar Vorschläge machen, wie eine systematische Fehlersuche mit und ohne Oszilloskop vor sich gehen kann:

6.1 Mögliche Fehler und ihre Behebung

6.1.1 Sind die bisher verwendeten Baugruppen in Ordnung? (Funktionierte das System ohne die Baugruppe)

6.1.2 Sind die Jumper richtig gesteckt?

6.1.3 Machen Sie zuerst eine Sichtprobe. Können Sie irgendwo auf der Platine unsaubere Lötstellen (zuviel Lötzinn, manchmal zieht das Lötzinn Fäden) erkennen, die eventuell einen Kurzschluß verursachen könnten? Dann müssen sie diese Lötstellen nachlöten und die unzulässige Verbindung beseitigen.

6.1.4 Haben Sie auch alle ICs richtig herum und am richtigen Platz eingesteckt? (Vergleiche mit Bestückungsplan)

6.1.5 Sind alle gepolten Bauteile (Elkos, Dioden, usw.) richtig herum eingelötet?

6.1.6 Haben sie auch keine Lötstelle vergessen zu löten? (sehen sie lieber noch einmal nach)

6.1.7 Sehen Sie irgendwo "kalte" Lötstellen? Kalte Lötstellen erkennt man daran, daß sie nicht glänzen, sie sind im Vergleich mit richtig gelöteten Lötstellen trübe.

6.1.8 Haben Sie auch nicht zu heiß gelötet? Wenn der Lötkolben zu heiß eingestellt ist und (oder) Sie zu lange auf der Lötstelle bleiben, dann kann es passieren, daß sich die Leiterbahnen von der Platine lösen und Unterbrechungen bilden. Ferner kann es auch passieren, daß Durchkontaktierungen unterbrochen werden, oder daß Bauteile durch zu heißes Löten zerstört werden.

6.1.9 Nehmen Sie alle ICs aus ihren Fassungen. Nehmen Sie sich die Layouts zur Hand und kontrollieren Sie alle Leiterbahnen, mit einem Durchgangsprüfer oder mit einem Ohmmeter auf Durchgang. Bereits kontrollierte Leiterbahnen können Sie, der Übersicht wegen, auf dem Layout mit Bleistift durchstreichen oder mit Farbstiften nachziehen.

6.1.10 Prüfen sie die Versorgungsspannung mit einem Digital-Voltmeter (am Bus +5V, nicht am Netzgerät, da am Kabel bei starker Belastung bis zu 0.5V abfallen können). Toleranzen von +- 5% also von 4,75V bis 5,25V sind erlaubt. Falls die Spannung zu gering ist, prüfen Sie, ob die Verbindung vom Netzteil zum Bus mit ausreichend dickem (mind. 2 mm Quadrat) Kabel erfolgt ist. Gege-

benenfalls müssen Sie Ihr Netzteil nachregeln. Vorsicht: Nie über 5,1V nachregeln, da sich auf einigen Platinen 5,1V Zenerdioden befinden, die ab 5,1V durchschalten, was entweder zum Zusammenbruch Ihrer Versorgungsspannung führt oder die Zenerdiode bis zu Ihrer Zerstörung erhitzt. Übrigens: Wir empfehlen 5,05V.

Wenn Sie alle Leiterbahnen kontrolliert haben und nichts gefunden haben, dann ist die Wahrscheinlichkeit groß, daß ein Bauteil defekt ist.

Wenn Sie einen Prüfstift oder ein Oszilloskop haben, dann können Sie jetzt überprüfen, ob an den jeweiligen Ausgängen die richtigen Signale anliegen. Welche Signale wo anliegen müssen, können Sie aus der Schaltungsbeschreibung, aus dem Schaltplan und Ihren eigenen Überlegungen entnehmen.

Falls Sie keine Meßgeräte haben, dann müssen Sie alle Bauteile systematisch austauschen, bis Sie das Defekte gefunden haben. Verwenden Sie dazu eventuell eine zweite Baugruppe (die eines Freundes oder eines Bekannten).

Sollten Sie gar nicht zurande kommen, hilft Ihnen unser Pauschal-Reparatur-Service, dessen Bedingungen Sie der Preisliste entnehmen können.

7. Diverses

7.1 Verbesserungsmöglichkeiten/Erweiterungen

Zu erwähnen bleibt noch, daß der DMA-Controller natürlich systemweit verwendet werden kann, also auch z.B. für Floppyzugriffe. Dies erfordert jedoch einige Änderungen im Monitor. Insgesamt gesehen ist die EPROM-Floppy eine echte Alternative zu 7 bis 8 mal teureren Festplattenlaufwerken und diesen in Punkto Geschwindigkeit weit überlegen, immerhin fördert der DMA-Controller die Daten mit 1MByte/s. Aber auch ohne DMA ist die Arbeitsgeschwindigkeit atemberaubend und wird nur durch die Bildschirmausgabe und durch die echte Rechenzeit des Prozessors begrenzt.

7.2 Ausblick

Korrekturen für dieses Handbuch werden in der Zeitschrift LOOP bekanntgegeben. Man sollte dann die fehlerhaften Stellen von Hand korrigieren.

7.3 Kritik

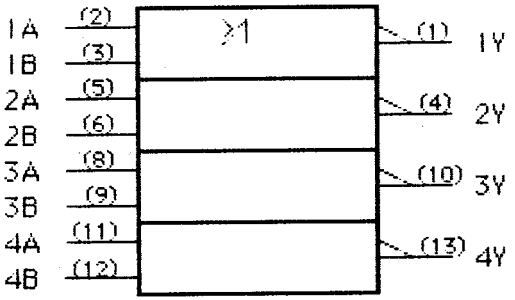
Bitte senden Sie uns die ausgefüllte Kritikkarte, die dem Bausatz beiliegt, zurück. Sie helfen uns, unsere Produkte und unseren Service noch besser zu gestalten. Für Fehlermeldungen und Verbesserungen, die dieses Handbuch betreffen, sind wir immer dankbar!

8. Unterlagen zu den verwendeteten ICs

8.1 Datenblätter TTL-ICs

74LS02

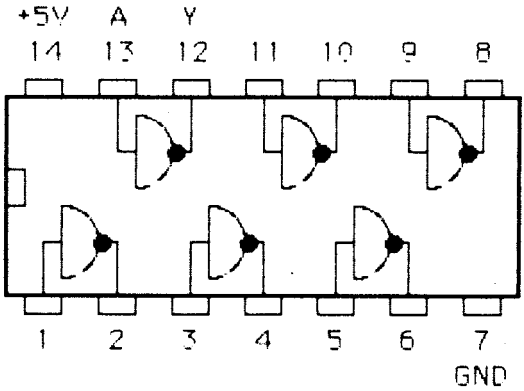
4 NOR mit je 2 Eingängen



Typ. Impulsverzögerungszeit:	10 ns
Typ. Versorgungsstrom:	6 mA
Positive Logik:	ja

74LS04

6 Inverter



Logiktabelle:

A	Y
0	1
1	0

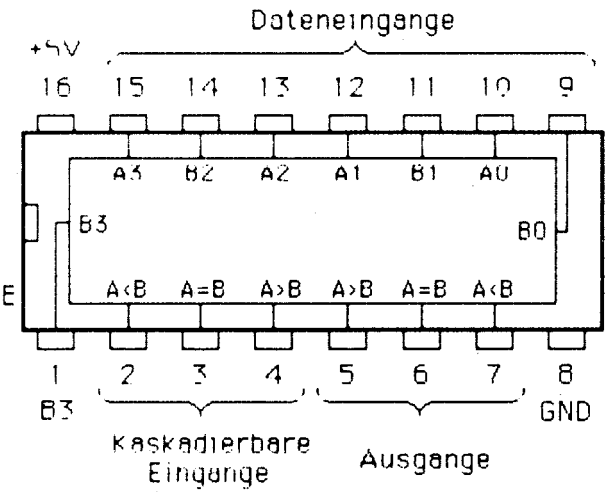
Typ. Impuls-Verzögerungszeit: 10 ns

Typ. Versorgungsstrom: 4 mA

positive Logik:
Y = A

74LS85

4-Bit Vergleicher



Typ Vergleichszeit für 4 Bit

24 ns

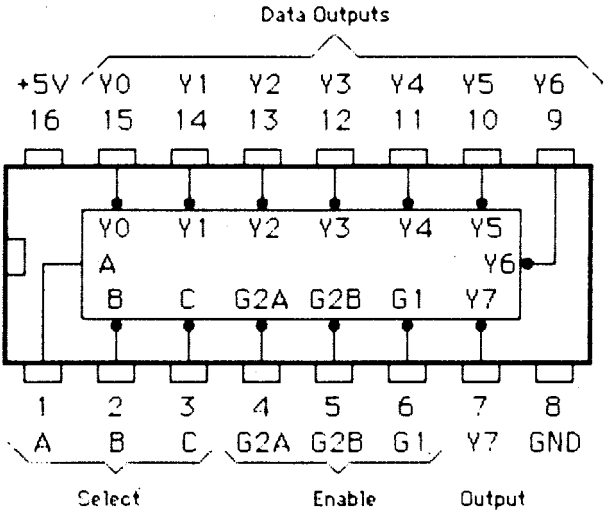
Typ Leistungsaufnahme

52 mW

Dateneingänge				Kaskadierbare Eingänge			Ausgänge		
A3, B3	A2, B2	A1, B1	A0, B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=A3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=A1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

74LS138

3-Bit Binärdekoder/Demultiplexer (3 zu 8)



Logiktabelle:

Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

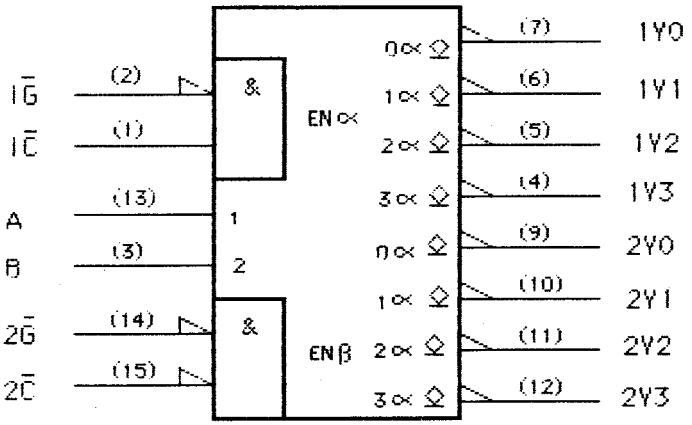
Positive Logik *G2 = G2A + G2B

Typ. Impulsverzögerungszeit 22 ns

Typ. Versorgungsstrom 7 mA

74LS156

2 2-Bit Binärdekoder/Demultiplexer



Logiktabelle:

Inputs				Outputs			
Select		Strobe	Data				
B	A	G	C	Y0	Y1	Y2	Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

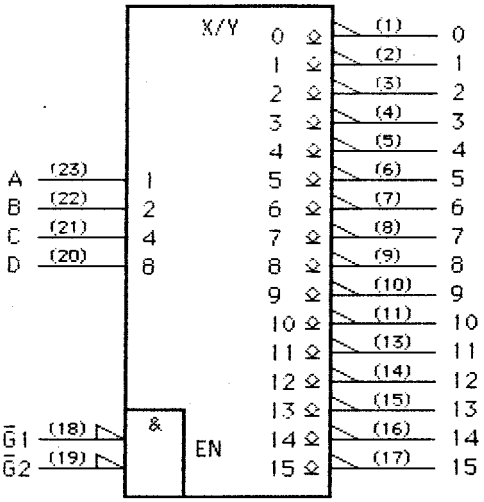
Typ. Impulsverzögerungszeit: 32 ns

Typ. Versorgungsstrom: 6 mA

Positive Logik: ja

74159

4 Bit Binärdekoder



Wahrheitstabelle:

Inputs						Outputs
G1	G2	D	C	B	A	ON*
L	L	L	L	L	L	0
L	L	L	L	L	H	1
L	L	L	L	H	L	2
L	L	L	L	H	H	3
L	L	L	H	L	L	4
L	L	L	H	L	H	5
L	L	L	H	H	L	6
L	L	L	H	H	H	7
L	L	H	L	L	L	8
L	L	H	L	L	H	9
L	L	H	L	H	L	10
L	L	H	L	H	H	11
L	L	H	H	L	L	12
L	L	H	H	L	H	13
L	L	H	H	H	L	14
L	L	H	H	H	H	15
L	H	X	X	X	X	None
H	L	X	X	X	X	None
H	H	X	X	X	X	None

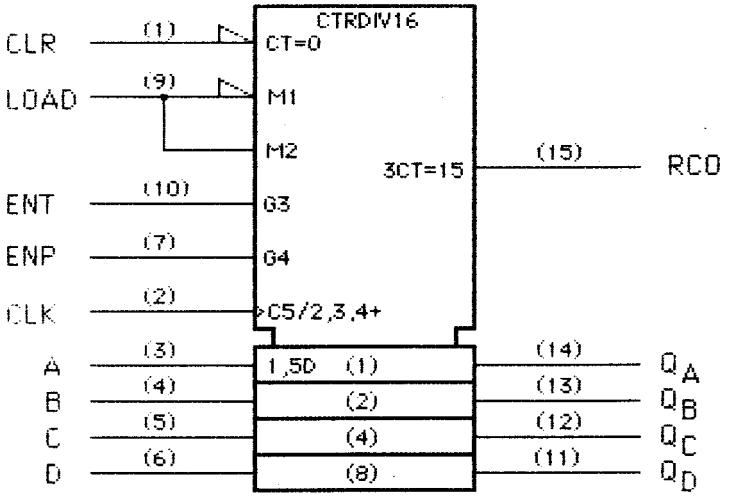
Typ. Impulsverzögerungszeit: 24 ns

Typ. Versorgungsstrom: 34 mA

Positive Logik: ja

74LS161

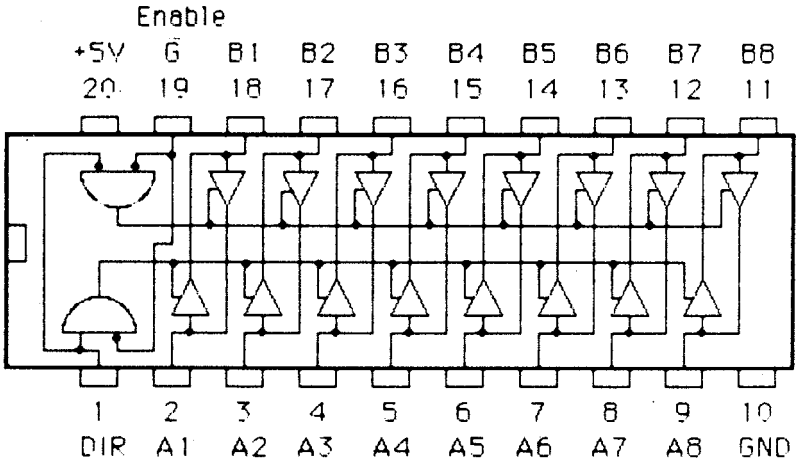
programierbarer 4-Bit Binärzähler



Typ. Versorgungsstrom: ca. 18 mA

74LS245

8-fach Bus-Transceiver mit 3-state Ausgangen



Function Table

ENABLE $\bar{G}$	DIRECTION CONTROL DIR	OPERATION
L	L	B data to A bus
L	H	A data to B bus
H	x	Isolation

Typ. Impuls-Verzögerungszeit: 20 ns

Typ. Versorgungsstrom: 75 mA

8.2 Datenblatt Z80A DMA

Z8410 Z80® DMA Direct Memory Access Controller

Zilog

Product Specification

April 1985

FEATURES

- Transfers, searches, and search/transfers in Byte-at-a-Time, Burst, or Continuous modes. Cycle length and edge timing can be programmed to match the speed of any port.
- Dual port addresses (source and destination) generated for memory-to-I/O, memory-to-memory, or I/O-to-I/O operations. Addresses may be fixed or automatically incremented/decremented.
- Next-operation loading without disturbing current operations via buffered starting-address registers. An entire previous sequence can be repeated automatically.
- Extensive programmability of functions. CPU can read complete channel status.
- Standard Z80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic. Sophisticated, internally modifiable interrupt vectoring.
- Direct interfacing to system buses without external logic.

GENERAL DESCRIPTION

The Z80 DMA (Direct Memory Access) is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features that optimize transfer speed and control with little or no external logic in systems using an 8- or 16-bit data bus and a 16-bit address bus.

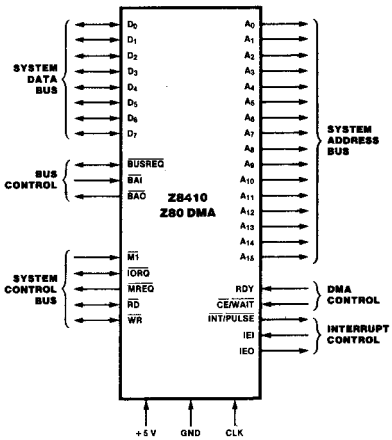


Figure 1. Pin Functions

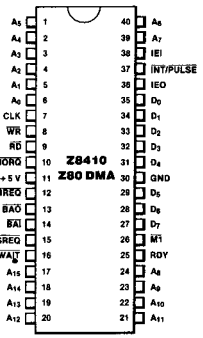


Figure 2. 40-pin Dual-In-Line Package (DIP), Pin Assignments

Transfers can be done between any two ports (source and destination), including memory-to-I/O, memory-to-memory, and I/O-to-I/O. Dual port addresses are automatically generated for each transaction and may be either fixed or incrementing/decrementing. In addition, bit-maskable byte searches can be performed either concurrently with transfers or as an operation in itself.

The Z80 DMA contains direct interfacing to, and independent control of, system buses, as well as sophisticated bus and interrupt controls. Many programmable features, including variable cycle timing and auto-restart, minimize CPU software overhead. They are especially useful in adapting this special-purpose transfer processor to a broad variety of memory, I/O and CPU environments.

The Z80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin, plastic or ceramic DIP. It uses a single +5V power supply and the standard Z80 Family single-phase clock.

FUNCTIONAL DESCRIPTION

Classes of Operation. The Z80 DMA has three basic classes of operation:

- Transfers of data between two ports (memory or I/O peripheral)
- Searches for a particular 8-bit maskable byte at a single port in memory or an I/O peripheral
- Combined transfers with simultaneous search between two ports

Figure 4 illustrates the basic functions served by these classes of operation.

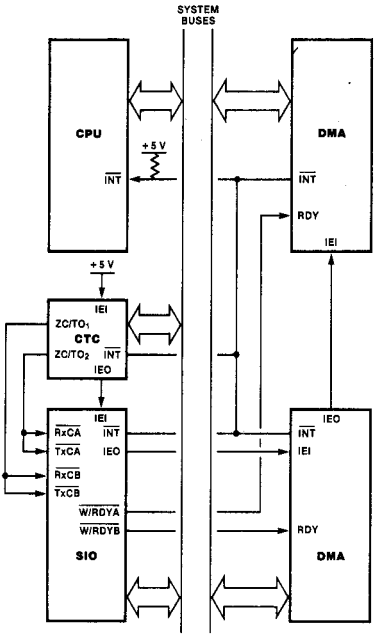


Figure 3. Typical Z80 Environment

During a transfer, the DMA assumes control of the system address and data buses. Byte by byte, data is read from one addressable port and written to the other addressable port. The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data may be written from one peripheral to another, from one area of main memory to another, or from a peripheral to main memory and vice versa.

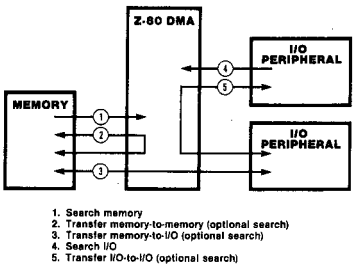
During a search-only operation, data is read from the source port and compared byte by byte with a DMA-internal register containing a programmable match byte. This match byte may optionally be masked so that only certain bits within the match byte are compared. Search rates up to 1.25M bytes per second can be obtained with the 2.5 MHz Z80 DMA or 2M bytes per second with the 4 MHz Z80A DMA.

In combined searches and transfers, data is transferred between two ports while simultaneously searching for a bit-maskable byte match.

Data transfers or searches can be programmed to stop, or interrupt, under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

Modes of Operation. The Z80 DMA can be programmed to operate in one of three transfer and/or search modes:

- *Byte-at-a-Time:* data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.



1. Search memory
2. Transfer memory-to-memory (optional search)
3. Transfer memory-to-I/O (optional search)
4. Search I/O
5. Transfer I/O-to-I/O (optional search)

Figure 4. Basic Functions of the Z80 DMA

- *Burst:* data operations continue until a port's Ready line to the DMA goes inactive. The DMA then stops and releases the system buses after completing its current byte operation.
- *Continuous:* data operations continue until the end of the programmed block of data is reached before the system buses are released. If a port's Ready line goes inactive before this occurs, the DMA simply pauses until the Ready line comes active again.

In all modes, once a byte of data is read into the DMA, the operation on the byte will be completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Due to the DMA's high-speed buffered method of reading data, operations on one byte are not completed until the next byte is read in. This means that total transfer or search block lengths must be two or more bytes, and that block lengths programmed into the DMA must be one byte less than the desired block length (count is N-1 where N is the block length).

Commands and Status. The Z80 DMA has several writable control registers and readable status registers available to the CPU. Control bytes can be written to the DMA whenever the DMA is not controlling the system buses, but the act of writing a control byte to the DMA disables the DMA until it is again enabled by a specific command. Status bytes can also be read at any such time, but writing the Read Status Byte command or the Initiate Read Sequence command disables the DMA.

Control bytes to the DMA include those which effect immediate command actions such as enable, disable, reset, load starting-address buffers, continue, clear counters, and clear status bits. In addition, many mode-setting control bytes can be written, including mode and class of operation, port configuration, starting addresses, block length, address counting rule, match and match-mask byte, interrupt conditions, interrupt vector, status-affects-vector condition, pulse counting, auto restart, Ready-line and Wait-line rules, and read mask.

Readable status registers include a general status byte reflecting Ready-line, end-of-block, byte-match, and interrupt conditions, as well as 2-byte registers for the current byte count, Port A address, and Port B address.

Variable Cycle. The Z80 DMA has the unique feature of programmable operation-cycle length. This is valuable in tailoring the DMA to the particular requirements of other system components (fast or slow) and maximizes the data-transfer rate. It also eliminates external logic for signal conditioning.

There are two aspects to the variable cycle feature. First, the entire read and write cycles (periods) associated with the source and destination ports can be independently programmed as 2, 3, or 4 T-cycles long (more if Wait cycles are used), thereby increasing or decreasing the speed with which all DMA signals change (Figure 5).

Second, the four signals in each port specifically associated with transfers of data (I/O Request, Memory Request, Read and Write) can each have its active trailing edge terminated one-half T-cycle early. This adds a further dimension of flexibility and speed, allowing such things as shorter-than-normal Read or Write signals that go inactive before data starts to change.

Address Generation. Two 16-bit addresses are generated by the Z80 DMA for every transfer operation, one address for the source port and another for the destination port. Each address can be either variable or fixed. Variable addresses can increment or decrement from the programmed starting address. The fixed-address capability eliminates the need for separate enabling wires to I/O ports.

Port addresses are multiplexed onto the system address bus, depending on whether the DMA is reading the source port or writing to the destination port. Two readable address counters (2 bytes each) keep the current address of each port.

Auto Restart. The starting addresses of either port can be reloaded automatically at the end of a block. This option is selected by the Auto Restart control bit. The byte counter is cleared when the addresses are reloaded.

The Auto Restart feature relieves the CPU of software overhead for repetitive operations such as CRT refresh and many others. Moreover, when the CPU has access to the buses during byte-at-a-time or burst transfers, different starting addresses can be written into buffer registers during transfers, causing the Auto Restart to begin at a new location.

Interrupts. The Z80 DMA can be programmed to interrupt the CPU on four conditions:

- Interrupt on Ready (before requesting bus)
- Interrupt on Match
- Interrupt on End of Block

Any of these interrupts cause an interrupt-pending status bit to be set, and each of them can optionally alter the DMA's interrupt vector. Due to the buffered constraint mentioned under "Modes of Operation," interrupts on Match at End of Block are caused by matches to the byte just prior to the last byte in the block.

The DMA shares the Z80 Family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z80 CPU environment, the DMA passes

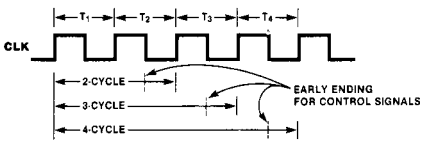


Figure 5. Variable Cycle Length

its internally modifiable 8-bit interrupt vector to the CPU, which adds an additional eight bits to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. In this process, CPU control is transferred directly to the interrupt routine, so that the next instruction executed after an interrupt acknowledge is the first instruction of the interrupt routine itself.

Pulse Generation. External devices can keep track of how many bytes have been transferred by using the DMA's pulse output, which provides a signal at 256-byte intervals. The interval sequence may be offset at the beginning by 1 to 255 bytes.

The Interrupt line outputs the pulse signal in a manner that prevents misinterpretation by the CPU as an interrupt request, since it only appears when the Bus Request and Bus Acknowledge lines are both active.

PIN DESCRIPTION

A_0-A_{15}. *System Address Bus* (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

$\overline{BAI}$. *Bus Acknowledge In* (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the $\overline{BAI}$ pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their $\overline{BAI}$ connected to the $\overline{BAO}$ of a higher-priority DMA.

$\overline{BAO}$. *Bus Acknowledge Out* (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system buses. $\overline{BAI}$ and $\overline{BAO}$ form a daisy chain for multiple-DMA priority resolution over bus control.

$\overline{BUSREQ}$. *Bus Request* (bidirectional, active Low, open drain). As an output, it sends requests for control of the system address bus, data bus, and control bus to the CPU. As an input when multiple DMAs are strung together in a priority daisy chain via $\overline{BAI}$ and $\overline{BAO}$, it senses when another DMA has requested the buses and causes this DMA to refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a buffer between it and the CPU because it is unidirectional into the CPU. A pull-up resistor is connected to this pin.

$\overline{CE}$/$\overline{WAIT}$. *Chip Enable and Wait* (input, active Low). Normally this functions only as a $\overline{CE}$ line, but it can also be programmed to serve a $\overline{WAIT}$ function. As a $\overline{CE}$ line from the CPU, it becomes active when $\overline{WR}$ or $\overline{RD}$ and $\overline{IORQ}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control, command bytes from the CPU to the DMA, or status bytes from the DMA to the CPU. As a $\overline{WAIT}$ line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. *System Clock* (input). Standard Z80 single-phase clock at 2.5 MHz (Z80 DMA) or 4.0 MHz (Z80A DMA). For 2.5 MHz clocks, a TTL gate with pullup resistor may be adequate to meet the timing and voltage level specification. For 4.0 MHz clocks, use a clock driver with an active pullup to meet the V_{IH} specification and rise-time requirements. In all cases there should be a resistive pullup to the power supply of 10K ohms (max) to ensure proper power when the DMA is reset.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.

IEI. *Interrupt Enable In* (input, active High). This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$/PULSE. *Interrupt Request* (output, active Low, open drain). While the CPU is the bus master, this output requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{IORQ}$ output Low during an $\overline{M1}$ cycle. It is typically connected to the $\overline{INT}$ pin of the CPU with a pullup resistor and tied to all other $\overline{INT}$ pins in the system. This pin can also be used to generate periodic pulses to an external device when the DMA is bus master (i.e., the CPU's $\overline{BUSREQ}$ and $\overline{BUSACK}$ lines are both Low and the CPU cannot see interrupts). While the DMA is the bus master, this output can be programmed to pulse each time 256 transfers have occurred.

$\overline{IORQ}$. *Input/Output Request* (bidirectional, active Low, 3-state). As an input, this indicates that the lower half of the address bus holds a valid I/O port address for transfer of control or status bytes from, or to, the CPU, respectively; this DMA is the addressed port if its $\overline{CE}$ pin and its $\overline{WR}$ or $\overline{RD}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the lower half of the address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{IORQ}$ and $\overline{M1}$ are both active simultaneously, an interrupt acknowledge is indicated.

$\overline{M1}$. *Machine Cycle One* (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI, ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{M1}$ is active as each opcode byte is

fetched. An interrupt acknowledge is indicated when both $\overline{M1}$ and $\overline{IORQ}$ are active.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. After the DMA has taken control of the system buses, it indicates a DMA transfer request from, or to, memory.

$\overline{RD}$. *Read* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.

RDY. *Ready* (input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (Byte, Burst, or Continuous), the RDY line indirectly controls DMA activity by causing the $\overline{BUSREQ}$ line to go Low or High.

$\overline{WR}$. Write (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled write to a memory or I/O port address.

INTERNAL STRUCTURE

The internal structure of the Z80 DMA includes driver and receiver circuitry for interfacing with an 8-bit system data bus, a 16-bit system address bus, and system control lines (Figure 6). In a Z80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{CE}/\overline{WAIT}$ line.

The DMA's internal data bus interfaces with the system data bus and services all internal logic and registers. Addresses generated from this logic for Ports A and B (source and destination) of the DMA's single transfer channel are multiplexed onto the system address bus.

Specialized logic circuits in the DMA are dedicated to the various functions of external bus interfacing, internal bus control, byte matching, byte counting, periodic pulse generation, CPU interrupts, bus requests, and address generation. A set of twenty-one writable control registers and seven readable status registers provides the means by which the CPU governs and monitors the activities of these logic circuits. All registers are eight bits wide, with double-byte information stored in adjacent registers. The two address counters (two bytes each) for Ports A and B are buffered by the two starting addresses.

The 21 writable control registers are organized into seven base-register groups, most of which have multiple registers. The base registers in each writable group contain both control/command bits and pointer bits that can be set to address other registers within the group. The seven readable status registers have no analogous second-level registers.

The registers are designated as follows, according to their base-register groups:

WR0-WR6—Write Register groups 0 through 6 (7 base registers plus 14 associated registers)

RR0-RR6—Read Registers 0 through 6

Writing to a register within a write-register group involves first writing to the base register, with the appropriate pointer bits set, then writing to one or more of the other registers within the group. All seven of the readable status registers are accessed sequentially according to a programmable mask contained in one of the writable registers. The section entitled Programming explains this in more detail.

A pipelining scheme is used for reading data in. The programmed block length is the number of bytes compared to the byte counter, which increments at the end of each cycle. In searches, data byte comparisons with the match byte are made during the read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

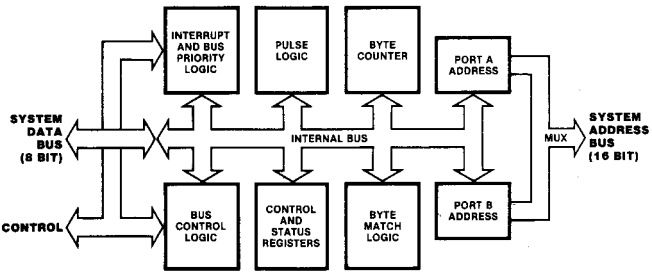


Figure 6. Block Diagram

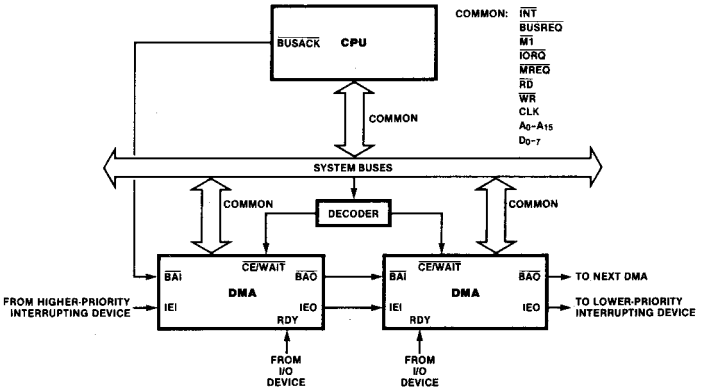


Figure 7. Multiple-DMA Interconnection to the Z80 CPU

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected (Zilog Microprocessor Applications Reference Book, Volume 1, # 00-2145-01, *The Z80 Family Program Interrupt Structure*). The system bus, however, may not be pre-empted. Any DMA that gains access to the system buses keeps them until it is finished.

Read Registers	
RR0	Status byte
RR1	Byte counter (low byte)
RR2	Byte counter (high byte)
RR3	Port A address counter (low byte)
RR4	Port A address counter (high byte)
RR5	Port B address counter (low byte)
RR6	Port B address counter (high byte)

Write Registers	
WR0	Base register byte Port A starting address (low byte) Port A starting address (high byte) Block length (low byte) Block length (high byte)
WR1	Base register byte Port A variable-timing byte
WR2	Base register byte Port B variable-timing byte
WR3	Base register byte Mask byte Match byte
WR4	Base register byte Port B starting address (low byte) Port B starting address (high byte) Interrupt control byte Pulse control byte Interrupt vector
WR5	Base register byte
WR6	Base register byte Read mask

PROGRAMMING

The Z80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests nor data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enable command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z80 CPU).

Reading. (Figure 8a) The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Writing. Control or command bytes are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

This is illustrated in Figure 8b. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)," then the next two bytes written to the DMA will be stored in that order in these two registers.

Fixed-Address Programming. A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address with LOAD command to WR6.
3. Declare Port A as a source in WR0.
4. Load Port A address with LOAD command to WR6.
5. Enable DMA in WR6.

Figure 9 illustrates a program to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050_H and the Port B peripheral fixed address is 05_H. Note that the data flow is 1001_H bytes---one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as the Z80 CPU's OTIR instruction.

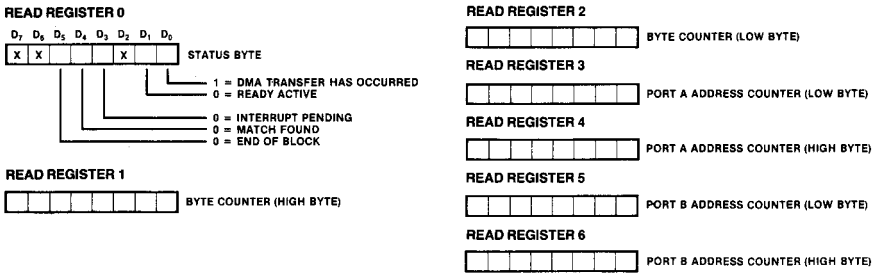


Figure 8a. Read Registers

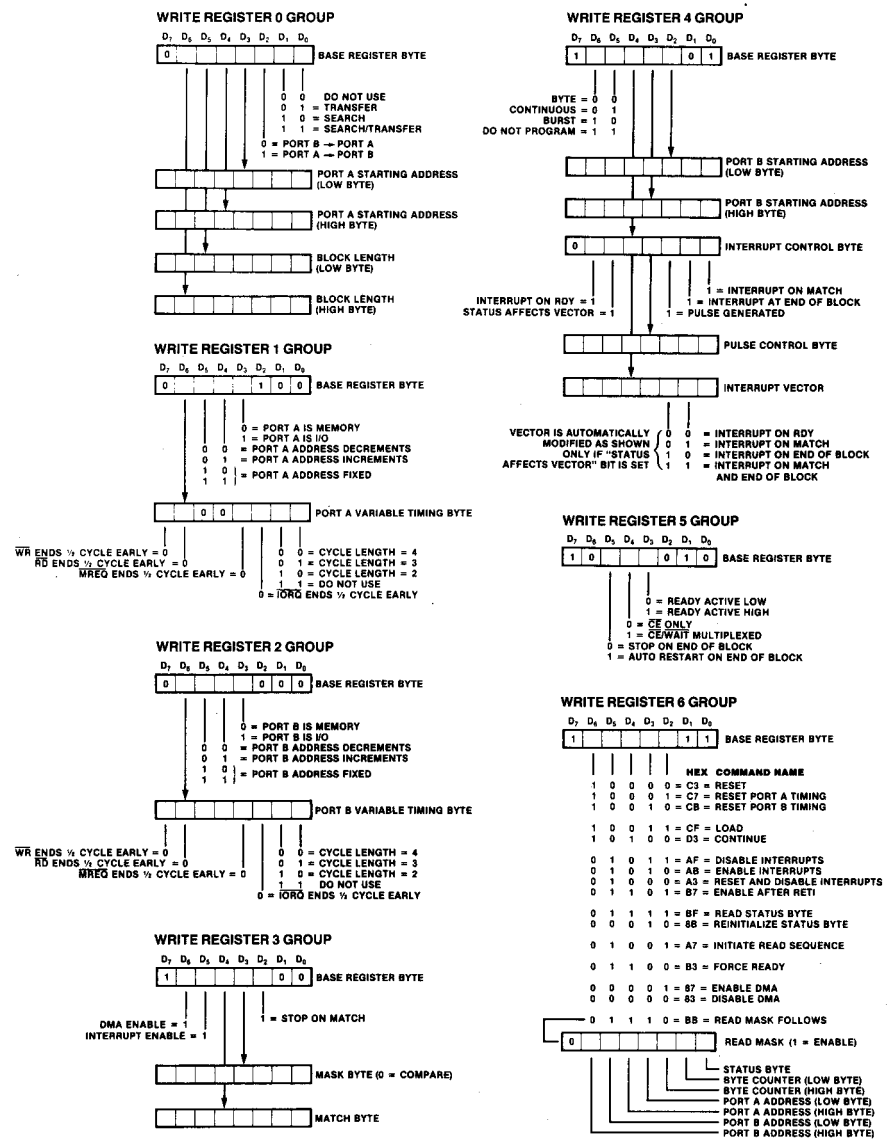
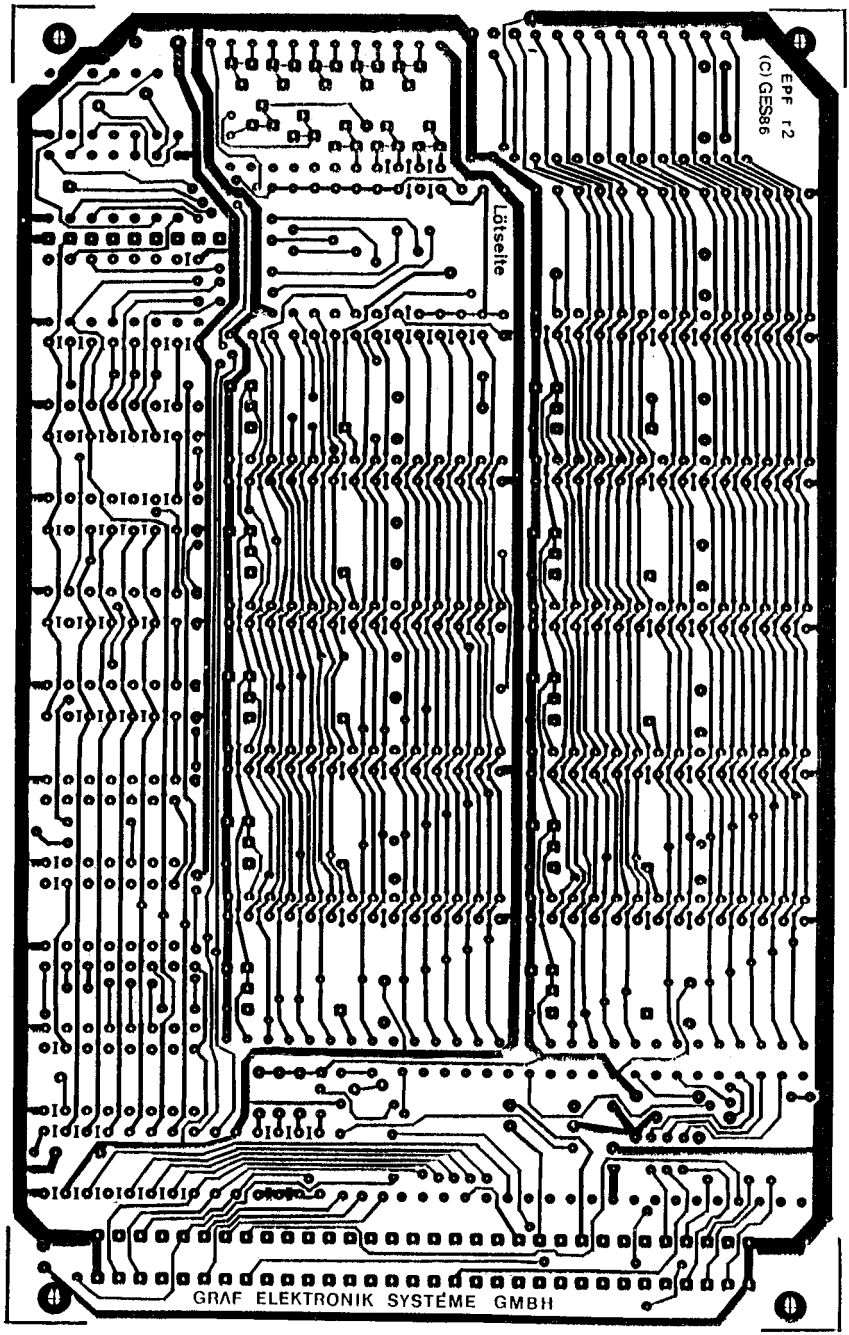


Figure 8b. Write Registers

Anhang C: Layout Lötseite



Comments	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HEX
WR0 sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B ⟶ A Temporary for Loading B Address*	0	1 Transfer, No Search	79
Port A address (lower)	0	1	0	1	0	0	0	0	50
Port A address (upper)	0	0	0	1	0	0	0	0	10
Block length (lower)	0	0	0	0	0	0	0	0	00
Block length (upper)	0	0	0	1	0	0	0	0	10
WR1 defines Port A as memory with fixed incrementing address.	0	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	1	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O	0	0	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Address Follows	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High.	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High	0	1	0	8A
WR6 loads Port B address and resets block counter.*	1	1	0	0	1	1	1	1	CF
WR0 sets Port A as source.*	0	0	0 No Address or Block Length Bytes	0	0	1 A ⟶ B	0	1 Transfer, No Search	05
WR6 loads Port A address and resets block counter.	1	1	0	0	1	1	1	1	CF
WR6 enables DMA to start operation.	1	0	0	0	0	1	1	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.
*These entries are necessary only in the case of a fixed destination address.

Figure 9. Sample DMA Program

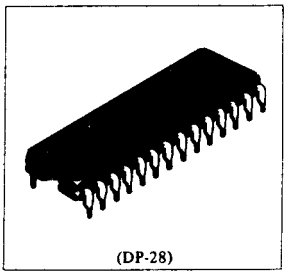
8.3 Datenblatt HM6264

HM6264P-10, HM6264P-12, HM6264P-15

8192-word x 8-bit High Speed Static CMOS RAM

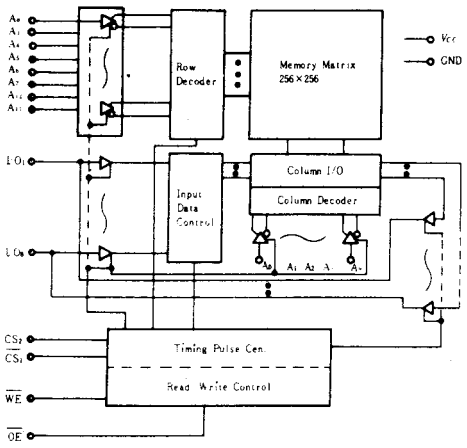
■ FEATURES

- Fast access Time 100ns/120ns/150ns (max.)
- Low Power Standby Standby: 0.1mW (typ.)
  Low Power Operation Operating: 200mW (typ.)
- Single +5V Supply
- Completely Static Memory. . . . . No clock or Timing Strobe Required
- Equal Access and Cycle Time
- Common Data Input and Output, Three State Output
- Directly TTL Compatible: All Input and Output
- Standard 28pin Package Configuration
- Pin Out Compatible with 64K EPROM HN482764

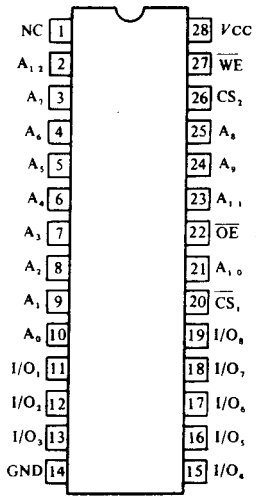


(DP-28)

■ BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit
Terminal Voltage *	V_T	−0.5 ** to +7.0	V
Power Dissipation	P_T	1.0	W
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature	T_{stg}	−55 to +125	°C
Storage Temperature (Under Bias)	T_{bias}	−10 to +85	°C

* With respect to GND. ** Pulse width 50ns: −3.0V

■ TRUTH TABLE

$\overline{WE}$	$\overline{CS_1}$	CS_2	$\overline{OE}$	Mode	I/O Pin	V_{CC} Current	Note
X	H	X	X	Not Selected (Power Down)	High Z	I_{SB}, I_{SB1}	
X	X	L	X		High Z	I_{SB}, I_{SB2}	
H	L	H	H	Output Disabled	High Z	I_{CC}, I_{CC1}	
H	L	H	L	Read	Dout	I_{CC}, I_{CC1}	
L	L	H	H	Write	Din	I_{CC}, I_{CC1}	Write Cycle (1)
L	L	H	L		Din	I_{CC}, I_{CC1}	Write Cycle (2)

X : H or L

9. Literatur

9.1 Die Zeitschrift LOOP

In unserer Zeitschrift LOOP wird regelmäßig über neue Produkte und Änderungen bzw. Verbesserungen berichtet. Es ist für Sie von großem Vorteil, LOOP zu abonnieren, denn dadurch ist sichergestellt, daß Sie auch immer über die neuesten Informationen verfügen.

Ein LOOP-ABO können Sie bei jeder Bestellung einfach mitbestellen.

Auch auf der Kritikkarte können Sie ein LOOP-Abo ganz einfach bestellen.

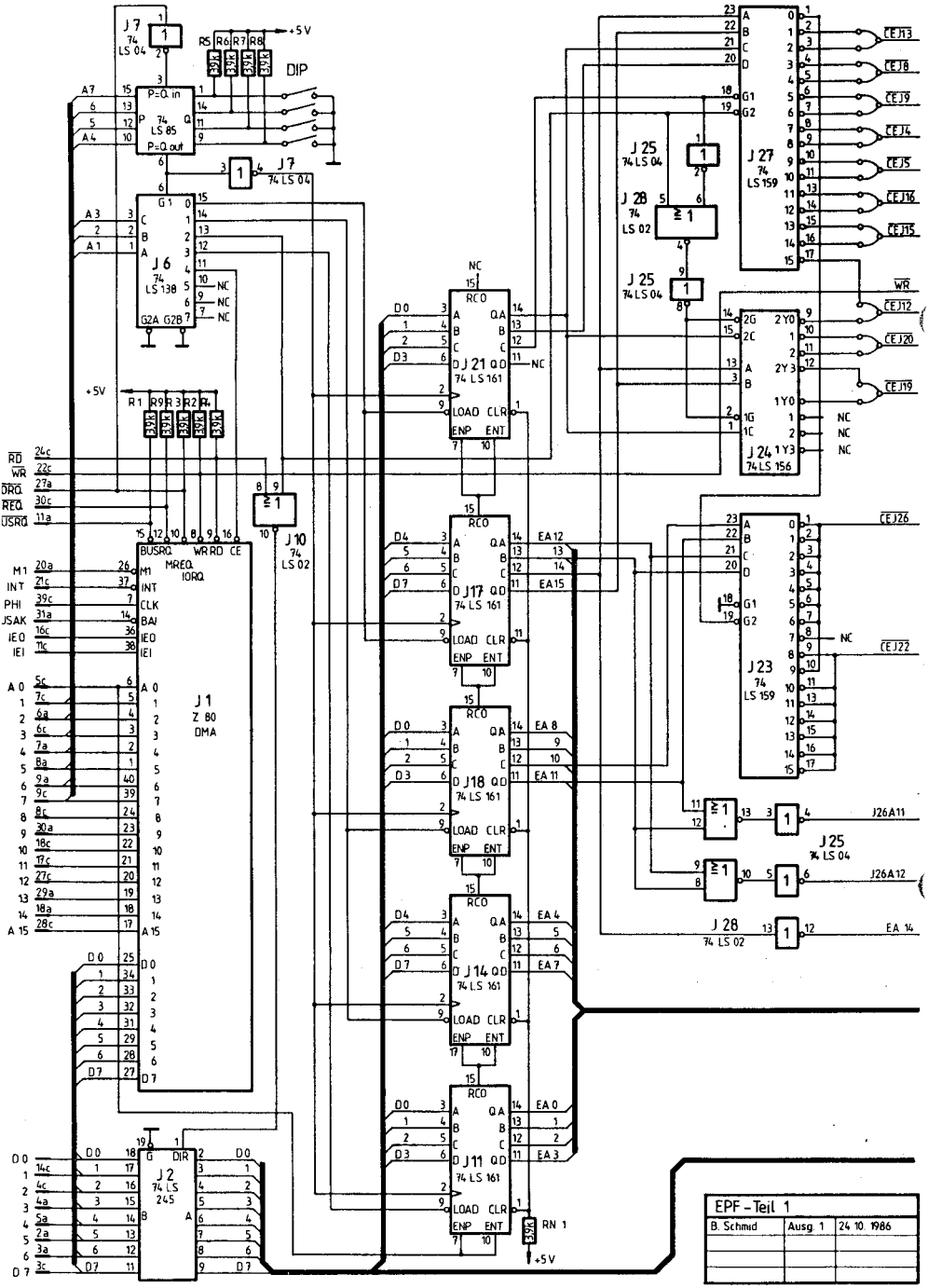
9.2 Literatur

/1/ Klein, Rolf-Dieter: Mini-Floppy-Anschluß, mc 2/1983,S.66
/2/ Das mc-CP/M-Sonderheft

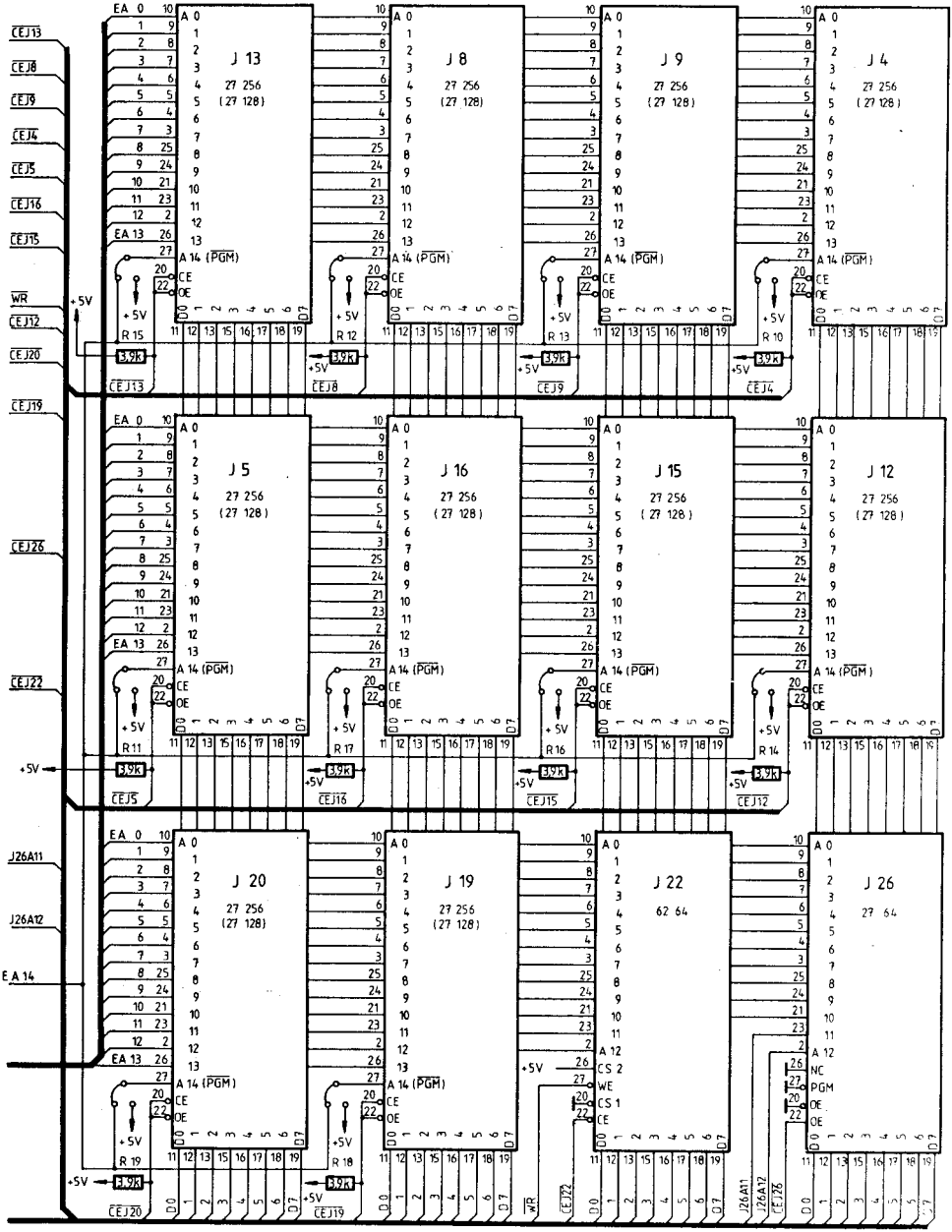
/3/ Brunner, Christian: Berechnung der Disk-Parameter-Tabellen für CP/M-Rechner, mc 7/1985, S.48
/4/ Sternberg,Günther: Eine RAM-Floppy für den mc-CP/M-Computer, mc 6/1985, S.86
/5/ CP/M Alteration Guide. Digital Research

/6/ Z80 DMA, Technical Manual, Zilog Inc.

/7/ Datenblätter der Firmen NEC und HITACHI für EPROMs bzw. RAMs der Typen 2764, 27128, 27256, 6264
/8/ Das mc-CP/M-Heft 7/86

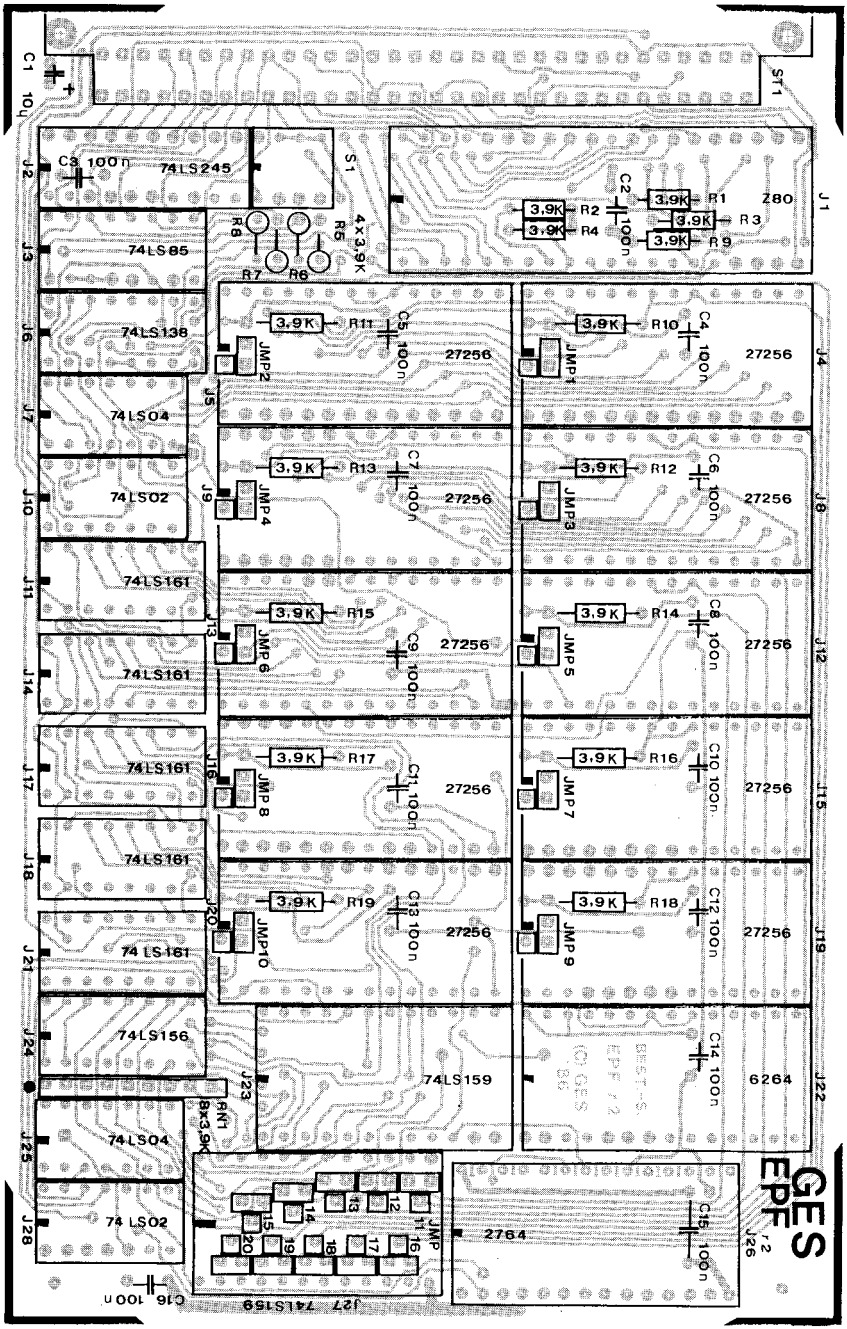


EPF - Teil 1		
B. Schmid	Ausg. 1	24. 10. 1986



EPF-Teil 2		
B. Schmid	Ausg. 1	24.10.1986

Anhang B: Bestückungsplan



Anhang D: Layout Bestückungsseite