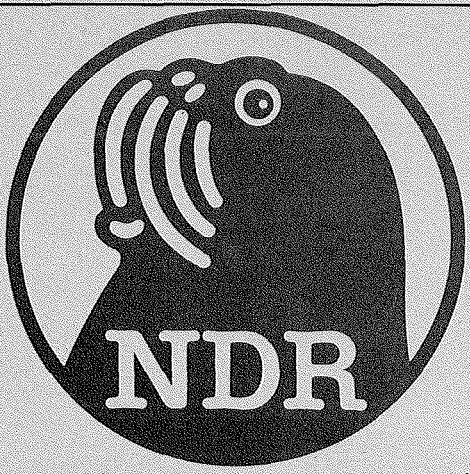
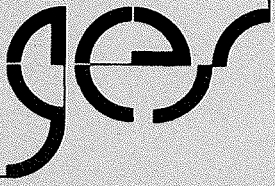


Handbuch

Einsteigerpaket

Graf Elektronik Systeme GmbH



Inhaltsverzeichnis

Herzlichen Glückwunsch

Hier liegt es endlich vor Ihnen, Ihr Einsteigerpaket. Da Sie wahrscheinlich darauf brennen, mit Ihrem System zu arbeiten, fassen wir uns so kurz wie möglich. Bevor Sie aber wild drauf los programmieren, sollten Sie doch die nächsten Seiten lesen.

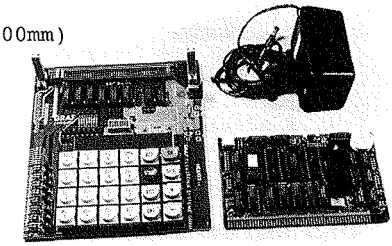
Wir packen aus

Verfahren Sie hierbei bitte nach dem Motto:

"Vertrauen ist gut, Kontrolle ist besser!"

Prüfen Sie, ob folgende Teile vorliegen:

- HEXIO2 (Baugruppe mit Tasten 180mm*200mm)
- SBC3 (Steckbaugruppe 160mm*100mm)
- Steckernetzteil
- 2 Führungsschienen
- 2 Schrauben M3*8
- 2 Muttern M3
- 4 Christiani-Kurse "Mikroelektronik Einführung mit dem NDR-Computer"
- 1 Handbuch (in dem Sie gerade lesen)



Was es mit den einzelnen Baugruppen auf sich hat, erfahren Sie im nächsten Abschnitt.

Beschreibung der Bauelemente

HEXIO2

Die HEXIO2 ist die sog. Ein/Ausgabe- (englisch:Input/Output:I/O) Baugruppe Ihres Systems. Eine I/O-Baugruppe ist ein Gerät, mit dem Sie mit Ihrem Computer "sprechen" können und er mit Ihnen.

SBC3

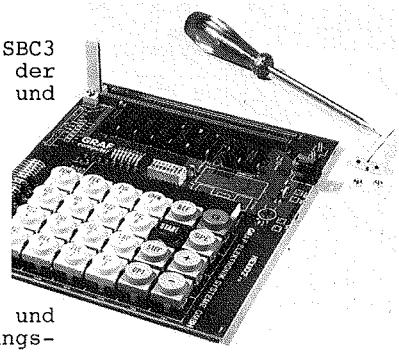
Die SBC3 ist das Herz des Systems. Sie stellt den eigentlichen Computer dar. SBC steht für "Single-Board-Computer"; das bedeutet "Einplatinencomputer".

Steckernetzteil

Das Steckernetzteil ist die Spannungsversorgung des Computers. Es wird in eine Steckdose gesteckt und erzeugt dann eine Spannung, die Sie für den Betrieb Ihres Computers brauchen. Wenn es Sie interessiert: es sind +5,0 Volt Gleichspannung.
Bitte verwenden Sie nur das mitgelieferte Steckernetzteil. Bei der Verwendung eines Anderen besteht die Gefahr, daß Sie mit Ihrem Computer Rauchzeichen geben!

Führungsschienen

Die Führungsschienen sollen die SBC3 (den eigentlichen Computer) auf der Ein/Ausgabeeinhiet (HEXIO2) halten und mechanisch stabilisieren.



Und jetzt der Zusammenbau

Noch nichts einstecken: Bitte Geduld!

Nehmen Sie die HEXIO2-Baugruppe und schrauben Sie die beiden Führungsschienen an den dafür vorgesehenen Stellen an. Diese Stellen sind auf der Leiterplatte genauso gekennzeichnet, wie auf dem nebenstehenden Bild. Die benötigten Schrauben liegen dem System bei.

In die Führungsschienen, die Sie gerade angeschraubt haben, stecken Sie jetzt die SBC3 ein (**Bauteile vorne**). Durch die beiden Schienen können Sie dabei eigentlich nichts falsch machen. Achten Sie aber trotzdem darauf, daß jeder Stift der Stiftleiste der SBC3 in eine Buchse der Steckerleiste der HEXIO2 paßt. Sollte dies nicht der Fall sein, kontrollieren Sie bitte, ob einer der Stifte vielleicht verbogen oder geknickt ist. Gegebenenfalls müßten Sie ihn ge-

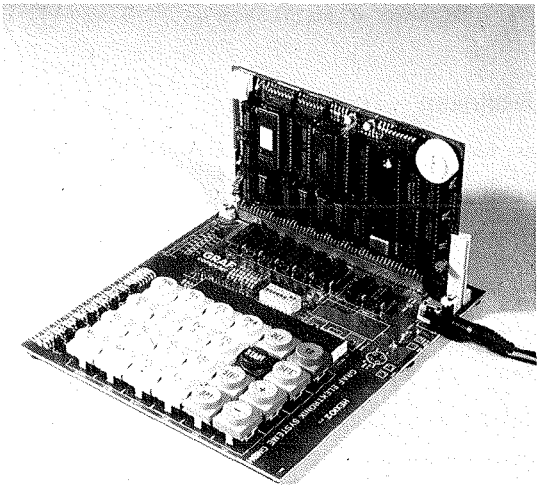
rade und parallel zu den anderen biegen. Jetzt müssen Sie nur noch für die nötige Spannung am System sorgen. Diese erzeugen Sie mit dem Steckernetzteil. Stecken Sie den kleinen Stecker wie auf dem folgenden Bild gezeigt, in die Kleinspannungsbuchse. Das Netzteil noch nicht in die Steckdose stecken!

Vor dem Einstecken: Platz bestimmen

Bevor Sie jetzt einstecken, sollten Sie erst einen geeigneten Platz für Ihren Computer finden. Ein geeigneter Platz sollte folgende Bedingungen erfüllen:

- trocken
- staubfrei
- keine zu heißen Unterlagen
- keine direkte Sonnenbestrahlung
- keine elektrisch leitenden Teile unter oder über dem System

Ideal wäre ein Tisch, auf dem Ihr System stehen bleiben könnte. Sollten Sie längere Zeit nicht damit arbeiten, wäre es sinnvoll Ihr System in einem Schrank zu deponieren, wo es vor Staub und äußeren Einwirkungen geschützt ist.



Nun Endlich: Einstecken

Achtung:

Verwenden Sie nur das mitgelieferte Steckernetzteil!

Bevor Sie jedoch das Steckernetzteil in eine Steckdose stecken, überprüfen Sie bitte nochmals Ihren Aufbau. Das fertig aufgebaute Gerät sollte genauso aussehen wie das System auf dem nebenstehenden Photo.

HALLO-1.1

Mit dem Einstecken des Steckernetzteils in die Steckdose, ist Ihr Computer eingeschaltet. Auf der Anzeige sollte nun eine Begrüßung Ihres Rechners an Sie stehen:

HALLO-1.1

Ist dies nicht der Fall drücken Sie bitte den RESET-Taster, den Sie links oben auf der SBC3 finden. Reagiert der Computer jetzt immer noch nicht, stecken Sie ihn aus und überprüfen Sie den Aufbau nocheinmal. Durch Druck auf den "RESET"-Taster kommen Sie immer wieder zum Anfangszustand. Es kann also nichts passieren!

Der erste Test: Es muß blinken

Mit Ihrem System haben Sie auch ein EPROM mitgeliefert bekommen, auf dem sich einige kleinere Programme befinden. Diese können Sie mit einigen einfachen Befehlen starten. Als erstes starten Sie einmal das Programm "**Blinklicht**". Die Vorgehensweise ist hierbei, wie folgt:

1. Drücken Sie nach dem Einschalten des Systems die Taste **START**. Auf der Anzeige Ihres Systems erscheint jetzt ein Vorschlag für eine mögliche Startadresse Ihres Programms (siehe Rand).

2. Da das Blinklichtprogramm aber auf einem anderen Speicherplatz abgelegt ist, müssen Sie diesen Platz dem Computer mitteilen. Ansonsten wählt er den angezeigten Platz als Anfangsadresse aus. Geben Sie jetzt nacheinander die Zahlen 1 3 B 0 ein. Dies ist die Startadresse des Blinklicht-Programms.

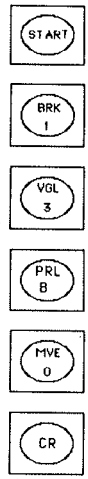
4. Jetzt müssen Sie dem Rechner nur noch sagen, daß die eben eingegebene Adresse als Startadresse verwendet werden soll. Hierzu drücken Sie die **CR**-Taste.

Sobald Sie diese Befehle eingegeben haben, fangen die Leuchtdioden auf der HEXIO2 zu blinken an.

Nun müssen die Leuchdioden auf der HEXIO2 in gleichmäßigen Zeitabständen blinken.

HALLO-1.1

Zum Test
eingeben:



Jetzt aber: Lehrbriefe korrigieren

Wenn Ihnen das Blinken inzwischen zuviel geworden ist (Abschalten durch "RESET-Taster"), sollten Sie jetzt daran gehen die mitgelieferten Christiani Kurse zu korrigieren. Dies ist deshalb nötig, da diese Kurse noch für die alte Version des Einsteigerpaketes geschrieben sind, wir Ihnen aber die aktuellste Version bieten wollen.

Auf den nächsten Seiten sind einige Blätter abgedruckt, die Sie statt der Seiten in den beiliegenden Christiani Kursen abheften sollten. Es ist sowieso empfehlenswert, sich nach dem Durcharbeiten der vier Hefte die einzelnen Abschnitte nach Themen geordnet in einem Ordner abzulegen. Auf diese Weise erhalten Sie ein wertvolles Nachschlagewerk das Ihnen bei vielen Fragen die beim Arbeiten mit Ihrem System auftauchen, helfen kann. Bei den auszutauschenden Seiten handelt es sich um folgende:

Lehrbrief 1:

Seite	B5
Seite	B6
Seite	B7
Seite	B8
Seite	B17
Seite	B18

Lehrbreif 3:

Seite	B88
Seite	B89
Seite	B92
Seite	B93
Seite	B94
Seite	B95
Seite	B96
Seite	B100

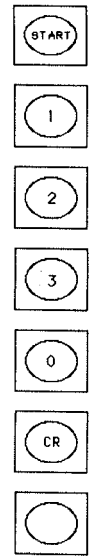
Lehrbrief 4:

Seite	A130
Seite	C21
Seite	D14

Die oben angesprochenen Blätter können Sie einfach gegen die im Kurs vorhandenen Seiten austauschen. Sie finden die Seiten im Anhang A an diese Beschreibung.

Zum Abschluß ein Spiel: Reaktionszeit

Genauso wie Sie vorher schon das Programm "Blinklicht" aufgerufen haben, können Sie jetzt auch das Programm "**Reaktionszeit**" aufrufen. Die Vorgehensweise ist hierbei genauso wie oben beschrieben (**START** drücken, Startadresse **1 2 3 0** eintippen und **CR** drücken).
Sie müssen statt der Adresse 1 3 B 0 von vorhin die Adresse **1 2 3 0** eingeben.

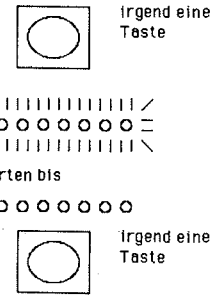


Wenn Sie das Programm starten, leuchten zunächst alle acht Leuchtdioden auf. Nach einer bestimmten Zeit, erlöschen die LED`s. Wenn Sie jetzt eine Taste drücken, mißt der Computer die Zeit, die Sie benötigt haben, um auf das Erlöschen der Leuchtdioden zu reagieren.

Die Zeit wird in Minuten/ Sekunden/ Zehntel und Hundertstel Sekunden angezeigt. Durch Druck auf eine beliebige Taste wird sie wieder angehalten.

Machen Sie 'mal einen Wettkampf:
00 00 15 ist schon ganz gut!

Natürlich können Sie diese Uhr auch als Stoppuhr verwenden!



Anschließend finden Sie eine Beschreibung und die Listings der Programme, die bereits fest in Ihrem System installiert sind.

Nichts geht mehr?

Für den Fall, daß sich Ihr System "aufhängen" sollte, gibt es eine recht einfache, aber sehr wirkungsvolle Methode, um den Normalzustand wieder herzustellen: Ein Druck auf den RESET-Taster!

Jetzt aber Christiani-Kurse durcharbeiten

Legen Sie jetzt dieses Handbuch beiseite und nehmen Sie die mitgelieferten Christiani-Kurse zur Hand. Arbeiten Sie die Kurse in aller Ruhe durch. Ein schnelles Überfliegen der einzelnen Abschnitte bringt nichts, weil Sie sich nichts einprägen können.

Los geht`s mit dem Lernen

Auch beim Programmieren ist es wie bei anderen Tätigkeiten, wenn man es mit Freunden machen kann macht es doppelt soviel Spaß als wenn man es alleine tut. Es gibt bestimmt auch in Ihrer Bekanntschaft Leute, die sich für die Computerei interessieren. Außerdem ist es empfehlenswert, nicht die ganzen Kurse innerhalb von einem Tag durchzuarbeiten, da Sie nach einer bestimmten Zeit nicht mehr aufnahmefähig für den Lehrstoff sind. Lassen Sie sich ruhig Zeit; dann verstehen Sie den Inhalt des Lehrganges umso besser.

Abschließend noch einige Tips, damit Sie möglichst viel Freude mit Ihrem Computer haben:

- Schreiben Sie sich Ihre Programme immer auf, das wird Ihnen am Anfang zwar als sehr lästig erscheinen, aber spätestens wenn Sie das erste Programm geschrieben haben, das sich selbst zerstört, sind Sie vom Sinn dieser Maßnahme überzeugt.

- Die SBC3 verfügt über eine Akkupufferung die Ihre Programme auch nach dem Ausstecken des Systems im Speicher hält.

- Gewöhnen Sie sich an, Ihre Programme immer zu kontrollieren, um einer Selbstzerstörung des Programms vorzubeugen.

Genauere Einweisungen in die Programmierung in Z80-Maschinensprache geben Ihnen die beiliegenden "Christiani"-Kurse.

Nicht vergessen: Garantiekarte einsenden

Vergessen Sie bitte nicht Ihre ausgefüllte Garantiekarte an uns einzusenden. Nur wenn Sie das getan haben können wir eine Garantie für Ihr System übernehmen.

Empfehlenswertes Zubehör

Bestnr.:	Beschreibung:	Preis
10834	GES-Katalog	10,--
10286	Monitor-Listing	35,--
10061	LOOP-Abonnement	20,--
10096	18-polige Buchsenleiste (3 erforderlich)	3,80 pro Stück

Vorteile der Zeitschrift LOOP

Für ein Abonnement der Zeitschrift "LOOP" sprechen folgende Punkte:

- mit der Zeitschrift "LOOP" haben Sie ein wertvolles Nachschlagewerk für alles, was den NDR-Computer betrifft

- wenn Sie die LOOP abonniert haben sind Sie nicht alleine, da Sie dort Verbindungen mit anderen Computerbauern aufnehmen können

- Sie werden durch die LOOP auch immer auf dem laufenden gehalten, was Software für Ihr System angeht

- die LOOP ist immer aktuell, da sie alle zwei Monate neu erscheint

Viel Spaß beim Programmieren

Einige Programme sind schon fest 'drin. Falls Sie während des Durcharbeitens noch einige Programme ansehen wollen: Wir haben einige für Sie vorbereitet.

Sie können diese Programme zuerst 'mal starten und mit ihnen spielen. Wir wollen aber mehr:

Ihnen beibringen wie man "Programmiert"

Deshalb haben wir die Quellistings dieser Programme sowie umfangreiche Erklärungen beigefügt.

Hier noch die Programme mit Startadressen in einer kurzen Tabelle:

Adresse	Programmname
1 0 0 0 H	Lotto
1 0 5 0 H	Datum
1 1 5 0 H	Taste
1 1 7 0 H	Taste2
1 1 9 0 H	Lauf1
1 1 E 0 H	Lauf2
1 2 0 0 H	Läuf3
1 2 3 0 H	Reaktionszeittest
1 3 0 0 H	Würfel
1 3 6 0 H	Sägezahnfunktion
1 3 B 0 H	Blinklicht

Lottozahlen

Wenn Sie allwöchentlich das Problem haben, sich für Ihren Lotto-Tip zu entscheiden, dann kann Ihnen das Programm LOTTO helfen. Dieses Programm ermittelt mit Hilfe eines sogenannten Zufallszahlengenerators eine mögliche Tipfolge. Es kann natürlich nicht das richtige Ergebnis vorhersagen! Erwarten Sie also nicht sofort "Sechs Richtige" !

Wie kann der der Mikroprozessor "Zufallszahlen" erzeugen, er kann doch keinen Würfel werfen!? In der Tat ist der Begriff "Zufall" im Zusammenhang mit Computern umstritten. Schauen wir uns den erwähnten Würfel einmal genauer an: Wenn Sie alle Umstände genau kennen würden, Gewicht, genaue Form und Oberfläche des Würfels, evtl. Luftbewegungen, Richtung, Stärke und Drehimpuls des Wurfes etc., könnten Sie dann nicht das Ergebnis des Würfelns vorausberechnen? Genauso ist es bei computererzeugten Zufallszahlen. Wie fast alles in und um einen Computer ließen sich diese Zahlen vorhersagen, nur wäre das praktisch viel zu kompliziert und bei der Methode, die wir verwenden wollen, praktisch unmöglich.

Also noch einmal: Wie kann der Mikroprozessor Quasi-Zufallszahlen erzeugen?

Der Mikroprozessor Z80 bietet hier eine recht einfache Möglichkeit. Es gibt einen Befehl ld a,r (lade a mit dem Inhalt von r), wobei a der Akkumulator ist und r das "Refresh"-Register. Refresh heißt Auffrischen und bezeichnet bei speziellen Speicherbausteinen (dynamische RAMs) einen Vorgang, der regelmäßig wiederholt werden muß, um nicht den Speicherinhalt zu verlieren. Im Refresh-Register r des Z80 steht eine Zahl zwischen 0 und 127, die bei diesen Speicherbausteinen angibt, welcher Teil des Speichers als nächstes aufgefrischt werden muß. Da so ein Refresh sehr häufig durchgeführt werden muß, wird das Register r sehr schnell weitergezählt, so daß jedesmal, wenn wir einen Befehl ld a,r verwenden, im Akku eine andere Zahl ankommt.

Das Programm LOTTO benutzt genau diese Methode. Das Unterprogramm random dient hier dazu, eine Zufallszahl zwischen 1 und 38 zu erzeugen. Dazu wird zunächst der Akku mit dem Wert des Refresh Registers geladen, damit haben wir eine Zufallszahl zwischen 0 und 127. Nun wird von dieser Zahl solange 38 abgezogen, bis sie kleiner als 38 ist. Damit haben wir eine Zahl zwischen 0 und 37. Jetzt wird noch eine Eins zum Akkuinhalt addiert, das nennt man inkrementieren (englisch mit "c" statt "k", deshalb "inc"). Wir wollen die Zahl aber im Registerpaar HL haben, um sie einfach mit Hilfe des HEXMON-Unterprogrammes PrtDez (Print Dezimal) ausgeben zu können. Also laden wir das Register l mit dem Inhalt des Akkus und setzen das Register h auf Null. Soweit, so gut!
-Aber... Damit das LOTTO- Programm brauchbar wird, ist entprechend den Regeln die Einschränkung erforderlich, daß jede der Zahlenkugeln nur einmal gezogen werden darf. Eigentlich muß sich der Computer nur merken, welche Zahlen er bereits gezogen hat. Dafür gibt es mehrere Möglichkeiten, er könnte zum Beispiel eine Liste führen, in der alle Kugeln stehen, die noch nicht gezogen wurden. Oder er benutzt eine Tabelle, in der für jede Kugel steht, ob sie schon gezogen wurde. Nach diesem lezteren Verfahren arbeitet das Programm LOTTO.

Da für diese Methode bereits zu Beginn des Programmablaufes Vorbereitungen getroffen werden müssen, kehren wir also zurück an den Anfang des Programmes und erklären alles schön der Reihe nach (wie es sich schließlich für eine anständige Dokumentation gehört). Beim Start des Programms muß zunächst die oben erwähnte Tabelle erstellt werden. Das geschieht in der Schleife clrLoop. Zunächst wird aber die Anzahl der Kugeln in das Register b geladen, das wieder als Schleifenzähler dient. In das Registerpaar HL kommt die letzte Adresse, die noch zur Tabelle gehört, denn in clrLoop wird die Tabelle von hinten nach vorne erzeugt. Dazu wird in den Speicherplatz, dessen Adresse in HL steht, der Inhalt von b geschrieben. Dazu dient der Befehl ld (hl),b. HL enthält hier eine Adresse, man nennt das dann Zeiger (oder neudeutsch "pointer"), HL zeigt auf die Stelle in der Tabelle, an die die nächste Zahl kommt. Da die Tabelle von hohen zu niedrigen Adressen erstellt wird, muß HL nun um eins vermindert (dekrementiert) werden, das übernimmt der Befehl dec hl. Danach sorgt der djnz Befehl wieder dafür, daß die Schleife noch einmal durchlaufen wird, außer wenn b bereits den Wert Null erreicht hat. Anschließend steht an der Adresse store+1 eine 1, bei store+2 steht eine 2, bei store+10 steht eine 10 (0A sedezimal) u.s.w. Später soll überall dort, wo eine Kugel gezogen wurde eine Null stehen, damit das Unterprogramm random erkennen kann, ob die Zufallszahl, die es ermittelt hat noch vorhanden ist, oder ob eine andere ermittelt werden muß.

Nach dem die Tabelle eingerichtet ist folgt das Hauptprogramm. In einer Scheife, die maximal 7 mal durchlaufen wird, ruft das Hauptprogramm zunächst das Unterprogramm PRINT auf, das den Text "Lotto" zur Anzeige bereitstellt. Danach wird im Unterprogramm random die Zufallszahl generiert und auf Zulässigkeit hin überprüft. Wie wird's gemacht?

Dazu wird die Anfangsadresse der Tabelle, also die Adresse store, in das Registerpaar DE geladen (DE zeigt auf die Tabelle, HL ist der Index in diese Tabelle, also die laufende Nummer des Eintrags). Addiert man nun HL und DE, so erhält man die Adresse des Bytes, in dem eigentlich die Zufallszahl stehen müßte, das geschieht mit dem Befehl add hl,de.

HL zeigt jetzt also auf das fragliche Byte in der Tabelle. Dann vergleicht das Programm dieses Byte (also das Byte, dessen Adresse in HL steht) mit dem Akku (in dem immer noch die Zufallszahl steht): cp (hl). Ist der Unterschied Null, dann war das fragliche Byte dasselbe wie die Zufallszahl im Akku, das bedeutet, daß diese Kugel noch nicht gezogen wurde. Andernfalls, also wenn der Unterschied nicht Null ist, wurde sie schon einmal gezogen und der Befehl jr nz,random (Jump Relative if Not Zero) springt noch einmal an den Anfang des Unterprogramms, um eine neue Zahl zu erzeugen. (Das ist nicht unbedingt geschickt, aber da das Refresh-Register sich auch für den Mikroprozessor zu schnell ändert, kommt beim nächsten Versuch wahrscheinlich eine andere Zahl heraus. Praktisch merkt man nachher nicht, daß das Programm für einige Zahlen länger braucht als für andere.) Wenn nun (evtl. nach mehreren Versuchen) eine Zahl gezogen wurde, die noch nicht vergeben ist, muß diese Zahl jetzt als benutzt markiert werden. Dazu dient der Befehl ld (hl),0. Er lädt das Byte, dessen Adresse in HL steht mit dem Wert 0. Da das Hauptprogramm als Ergebnis des Unterprogramms eine Zahl im Registerpaar HL erwartet, nicht die Adresse, die jetzt noch dort steht, muß von HL wieder die Anfangsadresse der Tabelle, die immer noch in DE steht, abgezogen werden, dann kann die Rückkehr ins Hauptprogramm erfolgen. Leider gibt es den Befehl sub hl,de beim Z80 nicht, man darf hier aber auch sbc hl,de (SuBtract with Carry) verwenden, da an dieser Stelle im Programm kein Übertrag (Carry) vorliegen kann.
Abschliessend wird ins' Hauptprogramm zurückgesprungen (mit dem Befehl ret). Nach dem Befehl call HoleTaste folgt jetzt ein cp 57H. Warum das?
Nun, das Unterprogramm HoleTaste des HEXMON wartet bis eine Taste gedrückt wird und legt dann den Code dieser Taste im Akkumulator ab. Der Befehl cp (compare) vergleicht diesen Wert mit dem Code der CR Taste (57 sedezimal). Wurde die CR-Taste gedrückt, dann ist der Unterschied zum Vergleichswert Null. Deshalb springt der Befehl jr nz,ENDE (Jump Relative if Not Zero) zur Adresse ENDE wenn eine andere Taste als CR gedrückt wurde. Man kann dieses Programm also durch drücken einer beliebigen Taste abbrechen oder mit CR weitermachen.

Starten Sie dieses Programm mit dem START-Befehl, so erscheint die Schrift "Lotto" und die erste gezogene Zahl. Jetzt können Sie sich diese Zahl notieren und CR drücken um die nächste Zahl zu bekommen, oder Sie drücken irgendeine andere Taste, dann erlischt die Anzeige und nach einem kurzem Moment erscheint das HALLO-1.1. Das passiert auch, wenn Sie alle sieben Zahlen gezogen haben.

Lassen Sie sich jetzt einmal alle Zahlen anzeigen, das heißt Sie starten das Programm und drücken immer wieder CR bis das HALLO-1.1 erscheint. Nun können Sie sich die vom Programm angelegte Tabelle des Programms "angucken". Drücken Sie dazu die Taste SPE und geben als Adresse den sedezimalen Wert von store (=80D9 sedezimal) an. Dies ist der Eintrag für die Kugel Null, da es aber keine Null gibt, steht hier nichts sinnvolles. Also drücken Sie + oder CR um zum nächsten Speicherplatz zu kommen. Dort erscheint nun 01 oder 00, je nachdem ob die Zahl 1 gezogen wurde oder nicht. Schreiben Sie diesen und alle folgenden Speicherplätze auf, so erhalten Sie zum Beispiel die Folge:

01 02 03 04 05 06 07 00 00 0A 0B 0C 00
0E 0F 10 11 12 13 14 15 16 17 18 19 00
1B 1C 1D 1E 1F 00 00 22 00 24 25 26 und
hier ist die Tabelle zu Ende.
Da überall dort, wo eine Null steht eine Zahl gezogen wurde, muß das Programm folgende Zahlen ermittelt haben (sedezimal): 08 09 0D 1A 20 21 23.
Tatsächlich hatte es diese Zahlen in folgender Reihenfolge ausgegeben:
9 26 33 35 32 8 13 (dezimal).

Vielleicht benutzen Sie den hier vorgestellten Zufallszahlengenerator einmal als Grundlage für ein selbstentworfenes, kleines Würfelspiel ?

```
                        ; Programm  LOTTO-1     1.0
                        ;
                        ; Dies Programm ermittelt per Zuffalszahlengenerator die Lottozahlen für
                        ; die nächste Ziehung, leider arbeitet der Algorithmus noch nicht fehlerfrei,
                        ; so daß die "Sechs Richtigen" nicht garantiert werden können.
                        .Z80
                        Ziehungen  EQU  6+1     ; Wieviel Kugeln werden gezogen
                        Kugeln     EQU  38    ; aus wieviel Kugeln wird gezogen

                        HoleTaste  EQU   000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        Print      EQU   0015H  ; Kopiert einen Text in die Anzeige
                        PrtDez     EQU   0021H  ; Druckt eine Dezimalzahl
                        Anzeige    EQU   8000H  ; Adresse des Anzeige-Speichers

                        store      EQU  80FFH-Kugeln  ; Freier RAM-Speicher für eine Tabelle der Zahlen

                                   ORG 1000H

                        LOTTO:

                        ; Zunächst wird eine Tabelle angelegt, in der steht, welche Zahlen noch nicht
                        ; gezogen wurden, es werden praktisch alle Kugeln in die Ziehungsmaschine
                        ; gefüllt. Diese Tabelle beginnt bei der Adresse store, sie muß im RAM stehen.

1000  06 26                        ld b,Kugeln     ; wieviel Kugeln gibt es
1002  21 FF 80                     ld hl,store+Kugeln   ; letztes Byte der Tabelle für noch vorhandene
                        clrLoop:                        ; Zahlen, die hier angelegt wird
1005  70                           ld (hl),b       ; Zahl eintragen
1006  2B                           dec hl          ; Adresse herunterzählen
1007  10 FC                        djnz clrLoop    ; für jede Kugel wiederholen

                        ; Jetzt werden die Ziehungen durchgeführt.

1009  06 07                        ld b,Ziehungen  ; dazu muß die Anzahl in Register b stehen
                        loop:
100B  C5                           push bc         ; Zähler für die Wiederholung retten
100C  21 43 10                     ld hl,text      ; Adresse des Textes in das Registerpaar HL laden
100F  CD 15 00                     call Print      ; Unterprogramm zur Textausgabe aufrufen
1012  CD 29 10                     call random     ; Unterprogramm zum würfeln einer Zahl
1015  DD 21 07 80                  ld ix,Anzeige+7 ; Hier kommt die Zahl hin
1019  CD 21 00                     call PrtDez     ; Gewürfelte Zahl ausgeben
101C  CD 0C 00                     call HoleTaste  ; Auf Tastendruck warten
                                                   ; Hier steht im Akku der Tastencode
101F  FE 57                        cp 57H          ; Taste CR gedrückt ?
1021  20 03                        jr  nz,ENDE     ; wenn nicht Programm beenden
1023  C1                           pop bc          ; Schleifenzähler wiederherstellen
1024  10 E5                        djnz loop       ; b herunterzählen und springen falls nicht null
                        ENDE:
1026  C3 00 00                     jp  0000H       ; Zurück in den HEXMON springen,
                                                   ; nach kurzer Zeit erscheint wieder das HALLO-1.1

                        ; Soweit das Hauptprogramm,
                        ; jetzt kommt nur noch das Unterprogramm zum Ziehen einer Zahl.

                        random:                    ; Würfel eine Zahl von 1 bis Kugeln
1029  ED 5F                        ld a,r          ; lade den Refresh-Wert in den Akku, dieser Wert ist
                                                   ; zufällig genug
102B  FE 26             modulo:  cp Kugeln         ; Ist der Wert kleiner als Kugeln ?
102D  38 04                        jr c,endrandom  ; wenn ja fertig
102F  D6 26                        sub Kugeln      ; sonst Kugeln vom Wert abziehen
1031  18 F8                        jr modulo       ; und noch einmal testen
                        endrandom:
1033  3C                           inc a           ; der Wert liegt zwischen 0 und Kugeln-1
```

```
                                        ; er soll aber zwischen 1 und Kugeln liegen, also
                                        ; wird er um eins erhöht (inkrementiert)
1034  6F                        ld l,a      ; das Ergebnis soll im Registerpaar HL stehen
1035  26 00                     ld h,00h    ; das obere Byte ist null
1037  11 D9 80                  ld de,store ; Anfangsadresse der Tabelle in Registerpaar de laden
103A  19                        add hl,de   ; Adresse in der Tabelle berechnen
103B  BE                        cp (hl)     ; Vergleich die Zahl in der Tabelle mit der Zahl im
                                        ; Akkumulator, dort steht immer noch die gezogene
                                        ; Kugel, in der Tabelle steht entweder Null, oder
                                        ; auch die gezogene Kugel
103C  20 EB                     jr  nz,random ; standen verschiedene Zahlen in Akku und Tabelle,
                                        ; dann wurde diese Zahl schon einmal gezogen.

                        ; Jetzt wurde eine neue Zahl gezogen

103E  36 00                     ld (hl),0   ; Diese Zahl löschen, damit sie nicht noch einmal
                                        ; gezogen werden kann
1040  ED 52                     sbc hl,de   ; Adresse der Tabelle wieder abziehen, damit die
                                        ; gezogene Zahl in HL steht
1042  C9                        ret         ; zurück ins Hauptprogramm


                        text:
1043  C7 A3 87 87 A3 FF FF FF   DB  0C7h, 0a3h, 87h, 87h, 0a3h, 0ffh, 0ffh, 0ffh
                                ; hier steht  L o t t o
                        END
```

DATUM

Wissen Sie eigentlich schon, auf welchen Wochentag der Jahrtausendwechsel, also Sylvester 1999 fallen wird? Ich kann es Ihnen verraten: der 31.12.1999 wird ein Freitag sein. Um das herauszufinden, habe ich keinen "Tausendjährigen Kalender" oder ähnlich umständliche Sachen benötigt, mir hat ein Einsteigerpaket gereicht. Mit dem richtigen Programm kann das nämlich für beliebige Daten den Wochentag berechnen.

Das ist möglich, da unser Kalender, der sogenannte Gregorianische Kalender, den Papst Gregor der XIII. 1582 einführte, sehr regelmäßig ist. Bis 1582 galt der Julianische Kalender, in dem immer wieder Änderungen vorgenommen werden mußten, da die Sonne und der Mond sich nicht nach diesem Kalender richten wollten. Seit 1582 werden diese Änderungen durch die Schaltjahre vermieden. Diese werden nach einfachen Regeln ermittelt:

- Alle durch vier teilbaren Jahre sind Schaltjahre (zum Beispiel 1992 / 4 = 498, also ist 1992 ein Schaltjahr).

- Alle durch 100 teilbaren Jahre sind kein Schaltjahre, obwohl sie durch vier teilbar sind (Zum Beispiel das Jahr 1900).

- Alle durch 400 teilbaren Jahre sind Schaltjahr, obwohl sie durch 100 teilbar sind. (Zum Beispiel das Jahr 2000)

Damit wird der Kalender recht einfach berechenbar. Man berechnet folgende Funktion:
x:= (Tag + (2*Monat)+(((3*Monat)+3) DIV 5) + Jahr + (Jahr DIV 4) - (Jahr DIV 100) + (Jahr DIV 400) + 2) MOD 7

Dann ist x eine Zahl zwischen 0 und 6, wobei 0 Samstag bedeutet, 1 Sonntag u.s.w. Leider gilt diese Formel nicht für Januar und Februar, diese Monate werden als 13. bzw. 14. Monat des Vorjahres gerechnet.

Dabei sind DIV und MOD Operatoren für die Division mit Rest. Ein Beispiel: 13 geteilt durch 5 gibt 2 Rest 3. Das schreibt man in den meisten höheren Programmiersprachen so: 13 DIV 5 = 2

und 13 MOD 5 = 3, also DIV liefert das Ergebnis der Division (ganzzahlig) und MOD den Rest.

DATUM ist ein Programm, das nach der obigen Formel den Wochentag für beliebige Daten nach 1582 berechnen kann. Starten Sie es einmal! Es erscheint die leere Anzeige mit einer Null ganz rechts. Geben Sie nun den Tag ein, z.B. 13 und drücken dann CR. Wieder erscheint nur eine Null. Jetzt können Sie den Monat eingeben, z.B. 4 und die Eingabe mit CR abschließen. Schon wieder steht eine Null einsam in der Anzeige, jetzt fehlt noch die Eingabe des Jahres, z.B. 1965 und auch diese Eingabe wird mit CR beendet. Sofort erscheint ganz links die Schrift di, die Abkürzung für Dienstag. Und tatsächlich war der 13.4.1965 ein Dienstag. Jetzt haben Sie zwei Möglichkeiten: Drücken Sie noch einmal CR so können Sie einen weiteren Tag berechnen, drücken Sie irgendeine andere Taste landen Sie wieder im HEXMON.

Schauen wir uns das Programm mal näher an:

Es gliedert sich in drei große Teile und ein Unterprogramm. Recht einfach ist die Eingabe, da dazu nur Unterprogramme des HEXMON aufgerufen werden. Etwas schwieriger ist die Ausgabe, denn dort muß aus einer Zahl zwischen 0 und 6, die im Registerpaar BC steht, ein Text gemacht werden. Dazu wird einfach zu dieser Zahl die Startadresse einer entsprechenden Tabelle, die TagTabelle heißt, addiert. An dieser Adresse steht nun das erste Zeichen, das einfach in den Anzeigespeicher kopiert wird. Sieben Byte weiter in der Tabelle steht das zweite Zeichen, da es sieben Tage gibt. Also wird zu der Adresse noch einmal sieben addiert und das an dieser Adresse stehende Zeichen in die zweite Stelle des Anzeigespeichers kopiert.

Das eigentlich komplizierte ist aber die Berechnung der Formel. Hier muß mit 16-Bit Werten gerechnet werden, da z.B. die Jahreszahl nicht in einem Byte dargestellt werden kann. Deshalb wird das Programm etwas länger. Ausserdem gibt es keinen Z80 Befehl zum dividieren oder um den Rest zu berechnen. Dafür mußte ein Unterprogramm her, das dividiere heißt. Da hier dieses Unterprogramm nicht sehr oft aufgerufen wird, muß es auch nicht

besonders schnell sein. Es wird die "natürliche Methode" benutzt, solange die zu teilende Zahl größer oder gleich der Zahl, durch die geteilt wird ist, wird diese einfach von der anderen abgezogen. Nun muß nur noch mitgezählt werden, wie oft dies gelungen ist. Anders formuliert: Der Rechner zählt, wie oft er die Zahl, durch die geteilt wird, von der anderen abziehen kann. In unserm Beispiel vom Anfang: 13 durch 5 ist ? 13 ist größer als 5, also kann 5 von 13 abgezogen werden, das gibt 8. 8 ist größer als 5, also kann 5 von 8 abgezogen werden, das gibt 3. 3 ist kleiner als 5, also sind wir fertig. Wir konnten zweimal abziehen und es sind drei übriggeblieben, also 13 DIV 5 ist 2 und 13 MOD 5 ist 3. Auch in diesem Unterprogramm macht es sich unangenehm bemerkbar, daß wir mit 16-Bit Zahlen rechnen müssen, denn der Z80 hat keinen Vergleichsbefehl für solche Zahlen. Man muß also die einzelnen Bytes der beiden Zahlen miteinander vergleichen. Ist das höherwertige Byte der ersten Zahl kleiner als das der zweiten, dann muß die ganze erste Zahl kleiner sein, wir sind fertig. Sind die beiden höherwertigen Byte ungleich, dann ist also das erste größer, und damit ist die ganze erste Zahl größer. Es bleibt noch der Fall, daß die höherwertigen Byte gleich sind, dann müßen wir die niederwertigen Byte vergleichen. Können wir nicht weiter subtrahieren, dann muß das Ergebnis noch in die gewünschten Register gebracht werden. Hier fällt auf, daß der Z80 auch nicht alle denkbaren 16-Bit Transportbefehle kennt, deshalb wird hier ein Umweg über den Stack gemacht. Die Befehlsfolge push hl pop bc bewirkt dasselbe wie ld bc,hl.

Vielleicht an dieser Stelle mal eine Erklärung zum Stack:

Der Stack von dem hier die Rede ist, ist ein Top-Down-Stack (zu deutsch: Von-Oben-Nach-Unten-Stapel). Das bedeutet, daß die ersten Eintragungen in den oberen Regionen des Stapels abgelegt werden und man sich mit jeder weiteren Eintragung immer weiter nach unten bewegt. Diese Vorgehensweise erscheint uns zunächst etwas befremdend, wenn wir von der Vorstellung eines Bücherstapels ausgehen, denn dieser wächst ja von unten nach oben. Vielleicht hilft Ihnen auch folgendender Vergleich zum Verständnis des Top-Down-Stapels: Man

bezeichnet ihn mitunter auch als "Kellerstapel", weil die deutsche Übersetzung "Von-Oben-Nach-Unten-Stapel" doch etwas unbeholfen wirkt, und zum anderen der Stapel bei fortlaufender Beschickung von einer fiktiven Decke aus in den "Keller" herabwächst. Beiden Stapelarten ist jedoch gemeinsam, daß zuerst alle darüberliegenden (beim Bücherstapel) bzw. alle darunterliegenden (beim Top- Down- Stapel) Ablagen entfernt werden müssen, um an ein bestimmtes Objekt zu gelangen. In beiden Fällen nennt man das auch LIFO-Prinzip (Last In - First Out), da das letzabgelegte (Last In) Objekt in jedem Falle als Erstes (First Out) wieder vom Stapel genommen werden muß, um an eventuell höherliegende Objekte (denken Sie an den von der Decke aus herabwachsenden Stapel) gelangen zu können. Genauso arbeitet der Z80 mit seinem Stapel. Auf diesem Stapel liegen (natürlich) nur Zahlen. Diese Zahlen kommen aus den Registern des Prozessors mit dem Befehl "push..." und werden nachher auch wieder in Register gebracht mit dem Befehl: "pop...". Der Befehl push ix legt also die Zahl, die im Register IX steht auf den Stapel. Nun kann der Prozessor also nur noch an diese Zahl heran, da sie ganz "unten" im Stapel liegt. Bekommt er nun den Befehl pop hl, dann ist es ihm egal, aus welchem der Register die zu holende Zahl in den Stapel geschoben wurde, er nimmt sich lediglich die zuletzt abgelegte Zahl des Stapels und gibt sie in das HL-Register. So ist es also möglich, Daten von Registern in andere zu bringen, für die es keinen Transportbefehl (ld) gibt.

```
                        ; Programm  DATUM     1.0
                        ;
                        ; Dieses Programm berechnet zu einem Datum den Wochentag

                        .Z80

                        Tag         EQU  80F0H  ; Speicher für den Tag
                        Monat       EQU  80F1H  ; Speicher für den Monat
                        Jahr        EQU  80F2H  ; Speicher für das Jahr (2 Byte)
                        x           EQU  80F4H  ; Hilfsvariable

                        HoleTaste   EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        GetDez      EQU  002AH  ; Liest eine Dezimalzahl in HL ein
                        Clear       EQU  0033H  ; Löscht die Anzeige
                        Anzeige     EQU  8000H  ; Adresse des Anzeige-Speichers

                                    ORG  1050H

                        DATUM:

                        ; Zunächst ein Datum einlesen

1050  DD 21 07 80                   ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1054  21 00 00                      ld hl,0         ; Default Wert in HL
1057  CD 2A 00                      call GetDez     ; Dezimalzahl einlesen
105A  22 F0 80                      ld (Tag),hl     ; Ergebnis speichern, das zweite Byte wird wieder
                                                    ; überschrieben
105D  DD 21 07 80                   ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1061  21 00 00                      ld hl,0         ; Default Wert in HL
1064  CD 2A 00                      call GetDez     ; Dezimalzahl einlesen
1067  22 F1 80                      ld (Monat),hl   ; Ergebnis speichern, das zweite Byte wird wieder
                                                    ; überschrieben
106A  DD 21 07 80                   ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
106E  21 00 00                      ld hl,0         ; Default Wert in HL
1071  CD 2A 00                      call GetDez     ; Dezimalzahl einlesen
1074  22 F2 80                      ld (Jahr),hl    ; Ergebnis speichern, diesmal sind zwei Byte gültig
1077  CD 33 00                      call Clear      ; Anzeige löschen

                        ; Dann daraus den Wochentag berechnen

107A  3A F1 80                      ld a,(Monat)    ; Monat in den Akku laden
107D  FE 03                         cp 3            ; ist es Januar oder Februar ?
107F  30 09                         jr nc,MonOK     ; nein, Korrektur überspringen

                           ; Korrektur für Januar und Februar, zur Berechnung der Schalttage

 081  C6 0C                         add a,12        ; Monat um 12 erhöhen
1083  32 F1 80                      ld (Monat),a    ; und wieder speichern
1086  21 F2 80                      ld hl,Jahr      ; Adresse von Jahr in HL laden
1089  35                            dec (hl)        ; um eins erniedrigen, zum Ausgleich, da Monat um
                                                    ; 12 erhöht wurde
                        MonOK:
108A  2A F0 80                      ld hl,(Tag)     ; den Tag in l laden, dabei kommt der Monat nach h
108D  26 00                         ld h,0          ; h ist Null, HL enthält jetzt den Tag
108F  ED 5B F1 80                   ld de,(Monat)   ; in E steht der Monat
1093  16 00                         ld d,0          ; D ist Null, also steht in DE der Monat
1095  19                           add hl,de       ; Monat zu HL addieren
1096  19                           add hl,de       ; noch einmal addieren
1097  22 F4 80                      ld (x),hl       ; und Ergebnis Speichern
109A  62                            ld h,d          ; Monat nach HL laden
109B  6B                            ld l,e
109C  19                           add hl,de       ; Monat zu HL addieren
109D  19                           add hl,de       ; noch einmal addieren, jetzt steht 3*Monat in HL
```

```
109E  11 03 00                ld de,3
10A1  19                      add hl,de       ; 3 dazu addieren
10A2  11 05 00                ld de,5         ; In DE steht 5
10A5  CD 1C 11                call dividieren ; Unterprogramm zum teilen, in HL steht jetzt
                                              ; ((3*Monat)+3)/5
10A8  ED 5B F4 80             ld de,(x)       ; Zwischenwert nach DE bringen
10AC  19                      add hl,de       ; zum bisherigen Zwischenwert addieren
10AD  ED 5B F2 80             ld de,(Jahr)    ; Jahr nach DE bringen
10B1  19                      add hl,de       ; und auch addieren
10B2  23                      inc hl          ; plus zwei,
10B3  23                      inc hl          ; also zweimal inkrementieren
10B4  22 F4 80                ld (x),hl       ; und wieder speichern
10B7  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10BA  11 04 00                ld de,4         ; durch 4 teilen,
10BD  CD 1C 11                call dividieren ;  gibt die Zahl der Schaltjahre
10C0  ED 5B F4 80             ld de,(x)       ; Zwischenwert holen
10C4  19                      add hl,de       ; zum addieren
10C5  22 F4 80                ld (x),hl       ; und wieder speichern
10C8  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10CB  11 64 00                ld de,100       ; durch 100 teilen,
10CE  CD 1C 11                call dividieren ;  gibt die Zahl der ausgefallenen Schaltjahre
10D1  54                      ld d,h          ; nach DE kopieren
10D2  5D                      ld e,l
10D3  2A F4 80                ld hl,(x)       ; Zwischenwert laden
10D6  ED 52                   sbc hl,de       ; ausgefallene Schaltjahre abziehen
10D8  22 F4 80                ld (x),hl       ; und wieder speichern
10DB  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10DE  11 90 01                ld de,400       ; durch 400 teilen,
10E1  CD 1C 11                call dividieren ;  gibt die Zahl der Schaltjahre, die nicht ausfielen
10E4  ED 5B F4 80             ld de,(x)       ; Zwischenwert laden
10E8  19                      add hl,de       ; zum addieren
10E9  11 07 00                ld de,7         ; durch 7 Teilen,
10EC  CD 1C 11                call dividieren ; der Rest ist die Nummer des Wochentags

                      ; Nun das Ergebnis ausgeben
                      ; In BC steht eine Zahl zwischen 0 und 6, dabei steht 0 für Samstag, 1 für
                      ; Sonntag, .... und 6 für Freitag

10EF  60                      ld h,b          ; Ergebnis nach HL bringen
10F0  69                      ld l,c          ; da dort gerechnet werden kann
10F1  11 0E 11                ld de,TagTabelle
10F4  19                      add hl,de       ; Adrsse des ersten Buchstabens in der Tabelle
                                              ; berechnen, steht in HL
10F5  7E                      ld a,(hl)       ; Buchstaben holen
10F6  32 00 80                ld (Anzeige),a  ; und in die Anzeige schreiben
10F9  11 07 00                ld de,7         ; jetzt Adresse des zweiten Buchstabens berechnen,
10FC  19                      add hl,de       ; der steht sieben Byte weiter
10FD  7E                      ld a,(hl)       ; zweiten Buchstaben holen
10FE  32 01 80                ld (Anzeige+1),a ; auch in die Anzeige schreiben
                      Fertig:
1101  CD 0C 00                call HoleTaste  ; Auf Tastendruck warten, damit die Anzeige gelesen
                                              ; werden kann
1104  FE 57                   cp 57h          ; Taste  CR  gedrückt ?
1106  20 03                   jr nz,ENDE      ; nein ? Dann Programm beenden.
1108  C3 50 10                jp DATUM        ; sonst noch einmal starten
110B  C3 00 00          ENDE: jp 0            ; zum HEXMON-Start springen


110E  92 92 AA A1 AA A1 8E    TagTabelle: DB    92h, 92h, 0aah, 0a1h, 0aah, 0a1h, 8eh
                              ; das heißt       S   S    M    d    M    d    F
1115  88 A3 A3 CF CF A3 AF              DB     88h, 0a3h, 0a3h, 0cfh, 0cfh, 0a3h, 0afh
                              ; das heißt       A    o    o    I    I    o    r
```

```
                        dividieren:

                        ; Einfaches Unterprogramm zum Dividieren mit Rest
                        ; Für große Zahlen, die durch kleine geteilt werden, ist es nicht besonders
                        ; schnell, es gibt bessere Methoden.
                        ;
                        ; Eine Zahl steht in HL, sie wird durch DE geteilt.
                        ; Das Ergebnis steht wieder in HL, der Rest in BC

111C  F5                        push af         ; Akku und Flags auf den Stack retten
111D  DD E5                     push ix         ; ix auch retten
111F  DD 21 00 00               ld ix,0         ; In IX wird die Anzahl der Schleifendurchläufe
                                                ; mitgezählt
1123  7C                DivLoop: ld a,h         ; Oberes Byte des Divisors in A
1124  BA                        cp d            ; mit dem oberen Byte des Dividenten vergleichen
1125  38 0C                     jr c,DivEnde    ; Wenn kleiner muß gar nicht weiter getestet werden
1127  20 04                     jr nz,Div       ; HL ist größer als DE, also weitermachen
1129  7D                        ld a,l          ; Unteres Byte in A
112A  BB                        cp e            ; mit dem unteren Byte des Dividenten vergleichen
112B  38 06                     jr c,DivEnde    ; HL ist kleiner als DE, also fertig.
112D  DD 23             Div:    inc ix          ; Mitzählen
112F  ED 52                     sbc hl,de       ; einmal abziehen
1131  18 F0                     jr DivLoop      ; und wiederholen
                        DivEnde:
1133  E5                       push hl         ; Rest nach bc bringen
1134  C1                       pop bc
1135  DD E5                    push ix         ; Zahl der Schleifendurchläufe ist das Ergebnis
1137  E1                       pop hl
1138  DD E1                    pop ix          ; Register wieder herstellen
113A  F1                       pop af
113B  C9                       ret             ; und zurück zum Hauptprogramm

                        END
```

TASTE

Mit dem Programm TASTE können Sie auf einfache Art und Weise sehr schnell den Code einer Taste ermitteln. Wenn Sie zum Beispiel wissen wollen, welchen Code die Taste START erzeugt, starten Sie das Programm TASTE, darauf erscheint:

t A S t E

auf der Anzeige. Drücken Sie nun die START-Taste, so erscheint der Tastencode der START-Taste, nämlich 4b (sedezimal). Nun können Sie weitere Tastencode ermitteln, drücken Sie zum Beispiel CR, so erscheint 57, das ist der sedezimale Code der Taste CR. Vielleicht fällt Ihnen schon eine Regelmäßigkeit auf, probieren Sie doch mal BEF und SPE, dann bekommen Sie 47 und 5b. Schauen Sie sich nun die Anordnung der Tasten an und versuchen einmal zu erraten, welchen Code die 0- oder MVE-Taste hat! Um wieder einen HEXMON-Befehl eingeben zu können, müßen Sie die RESET Taste drücken, dann erscheint wieder das HALLO-1.1.

Wie macht das Programm das ?

Im Programm werden viele Unterprogramme des HEXMON benutzt:

HoleTaste

fragt solange die Tastatur ab, bis eine Taste gedrückt wurde. Während des Wartens ist die Anzeige sichtbar.

Print

kopiert einen Text (der hier Tabelle genannt wird) in die Anzeige. Dieser Text besteht aus acht Anzeigecoden, die man dem HEXMON-Handbuch entnehmen kann.

PrtAc

bedeutet print accumulator, es wandelt den Inhalt des Registers A (Akkumulator) in eine zweistellige Sedezimalzahl um und kopiert die Anzeigecode dieser Zahl an die Adresse, die in den Registern IX angegeben wird. Hier wird als Adresse der Speicherbereich für die Anzeige angegeben, so daß die Sedezimalzahl ganz links in der Anzeige erscheint. Macht man aus dieser Adresse Anzeige+6, bzw. aus dem Befehl 21 06 80 statt 21 00 80,

und läßt man den Befehl call Clear weg (oder ersetzt ihn durch NOP-Befehle, CD 33 00 wird zu 00 00 00), dann bleibt die Schrift tAStE in der Anzeige stehen und der Tastencode erscheint rechts daneben.

Clear

löscht den Speicher für die Anzeige. Außerdem wird noch die Adresse Anzeige (8000 sedezimal) verwendet, dort speichert HEXMON die Anzeige. (Das ist notwendig, da aus technischen Gründen (Multiplexbetrieb) immer nur eine der acht Anzeigeeinheiten arbeitet. Um nun eine 8-stellige Anzeige zu bekommen, muß HEXMON regelmäßig alle Anzeigen nacheinander einschalten. Dafür braucht er aber wiederum die Information, was angezeigt werden soll. Diese Information steht im Speicher von 8000 bis 8007 sedezimal. Drücken Sie die RESET-Taste, dann erscheint evtl. eine der Anzeigen heller, während alle anderen dunkel werden. Das liegt daran, das HEXMON nun nicht mehr alle Anzeigen nacheinander aktiviert, so daß nur die zuletzt eingeschaltete leuchtet.)

Das eigentliche Programm ist dann recht einfach: Mit Hilfe des Unterprogrammes Print wird der Text tAStE in die Anzeige geschrieben, dann wird mit Hilfe des Unterprogrammes HoleTaste ein Tastedruck erwartet. Der Tastencode dieser Taste steht anschließend im Akkumulator. Nun wird dieser Code mit Hilfe des Unterprogramms PrtAc in die Anzeige geschrieben, nachdem vorher mit dem Unterprogramm Clear die Anzeige gelöscht wurde. Das war's eigentlich. Um weitere Tastencode ermitteln zu können, folgt noch ein Sprung (jump) zur Stelle loop (Schleife), wo dann wieder auf einen Tastendruck gewartet wird.

Verwendete Befehle:

call		ruft ein Unterprogramm auf.
ld	load	lädt ein Register oder ein Registerpaar (z.B. HL) mit einem Wert, um Beispiel der Adresse "Tabelle"
jr	jump relative	springt zu einer anderen Stelle im Programm, wobei die Stelle nicht als absolute (sedezimal vierstellige) Adresse angegeben wird, sondern als

relative Adresse (auch Displacement oder Offset was soviel wie "Versatz" bedeutet). Das Displacement darf jedoch nur zwei Sedezimalstellen (entsprechend 1Byte) groß sein. Das hat zur Folge, daß ein Vorwärtssprung über maximal 127 Speicherzellen und ein Rückwärtssprung über maximal 128 Speicherstellen hinweg erfolgen kann. Bei der Interpretation des Displacements wird das Höchstwertigste der acht Bits als Vorzeichenbit aufgefaßt und die sieben niederwertigeren Bits ergeben nur noch einen Wertebereich von 127 (=2^6+ 2^5+ 2^4+ 2^3+ 2^2+ 2^1+ 1). Dieser Wert wird zur Adresse im Register PC (dem Befehlszähler) addiert. Dort stand nämlich bisher die Adresse, an der der jr- Befehl im Programm steht. Das Ziel des Sprunges bezieht sich also auf die Adresse des Sprungbefehls, daher der Name relative Adresse.

```
                        ;  Programm TASTE         1.0
                        ;
                        ;  Kleines Utility, ermöglicht das ermitteln von Tastencodes
                        ;  und zeigt, wie man HEXMON Unterprogramme nutzen kann

                        .Z80

                        HoleTaste  EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        Print      EQU  0015H  ; Kopiert einen Text in die Anzeige
                        PrtAc      EQU  0018H  ; Schreibt den Akumulator-Inhalt Sedezimal in die Anzeige
                        Clear      EQU  0033H  ; Löscht die Anzeige
                        Anzeige    EQU  8000H  ; Adresse des Anzeige-Speichers

                                    ORG  01150H

1150  21 65 11             TASTE:  ld hl,tabelle   ; Adresse des Textes in Register HL
1153  CD 15 00                     call Print      ; Text ausgeben
1156  CD 0C 00             loop:   call HoleTaste  ; Auf Tastendruck warten, Tastencode steht im Acum.
1159  CD 33 00                     call Clear      ; Anzeigefeld löschen
115C  DD 21 00 80                   ld ix,Anzeige+0 ; Position der Ausgabe in der Anzeige ganz links
1160  CD 18 00                     call PrtAc      ; Tastencode sedezimal ausgeben
1163  18 F1                        jr loop         ; wiederholen

                        tabelle:
1165  87 88 92 87 86 FF FF FF        db 87H, 88H, 92H, 87H, 86H, OFFH, OFFH, OFFH
                        ; hier steht    t A S t E

                        end
```

```
                        ; Programm TASTE-2        1.0
                        ;
                        ; Kleines Utility, ermöglicht das ermitteln von Tastencodes
                        ; und zeigt, wie man HEXMON Unterprogramme nutzen kann
                        ; Es ist fast identisch mit dem Programm TASTE.
                        ; Nur die Anzeige ist anders: der Text  tAStE bleibt ständig sichtbar
                        ; und die Ausgabe der Tastenkodes erfolgt rechts daneben.

                        .Z80

                        HoleTaste  EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        Print      EQU  0015H  ; Kopiert einen Text in die Anzeige
                        PrtAc      EQU  0018H  ; Schreibt den Akumulator-Inhalt Sedezimal in die Anzeige
                        Anzeige    EQU  8000H  ; Adresse des Anzeige-Speichers

                                   ORG  01170H

1170  21 82 11            TASTE2: ld hl,tabelle   ; Adresse des Textes in Register HL
1173  CD 15 00                    call Print      ; Text ausgeben
1176  CD 0C 00            loop:   call HoleTaste  ; Auf Tastendruck warten, Tastencode steht im Acum.
1179  DD 21 06 80                 ld ix,Anzeige+6 ; Position der Ausgabe in der Anzeige ganz links
117D  CD 18 00                    call PrtAc      ; Tastencode sedezimal ausgeben
1180  18 F4                       jr loop         ; wiederholen

                        tabelle:
1182  87 88 92 87 86 FF FF FF        db 87H, 88H, 92H, 87H, 86H, 0FFH, 0FFH, 0FFH
                        ; hier steht    t A S t E

                        end
```

Lauflicht

Das Einsteigerpaket besitzt neben den Siebensegment-Anzeigen auch noch acht LEDs, die normalerweise alle leuchten. Gerade diese LEDs sind aber besonders einfach anzusteuern, da sie direkt mit dem IO-Port (Input/Output - Ein/Ausgabe) mit der Nummer 2 angeschlossen sind. Um zum Beispiel alle LEDs auszuschalten kann man den HEXMON Befehl IOS (Input/Output setzen) benutzen:

Wenn HEXMON sich mit "HALLO-1.1" oder "--------" oder "-bef-" meldet, drückt man die Taste 8/IOS. Daraufhin erscheint 00, damit ist die Nummer des IO-Ports gemeint. Sie wollen Port 2 setzen, also tippen Sie 2 CR, und erhalten die Anzeige 02 00. Nun können Sie den Wert, den Sie ausgeben wollen, angeben. Geben Sie einmal F F CR ein: die LEDs erlöschen. Tippen Sie nun 0 0 CR so leuchten alle wieder auf.

Wird auf diesen Port eine Zahl ausgegeben, so läßt jedes 0-Bit eine LED leuchten, ein 1-Bit dagegen schaltet sie aus. Probieren Sie einmal aus was passiert, wenn Sie nacheinander die Zahlen (sedezimal) FE, FD, FB, F7, EF, DF, BF und 7F auf Port 2 ausgeben! Nacheinander leuchten alle acht LEDs auf, wenn Sie schnell genug tippen könnten, würde der Eindruck eines Laufenden Punktes entstehen. Was liegt nun näher, als diese "Tippaufgabe" einem Programm zu übergeben? Dazu müssen Sie nur wissen, wie ein Programm einen Wert an einen Port ausgeben kann. Das geht recht einfach mit zwei Befehlen: ld a,0FFh (3E FF) und out (02H),a (D3 02) schaltet zum Beispiel alle LEDs aus.

Nun ein erstes Programm: Sie geben acht mal hintereinander diese vier Byte ein (mit dem HEXMON Befehel SPEeichern) und ersetzen dabei jedesmal die FF durch die weiter oben angegebene Folge FE, FD, FB, F7, EF, DF, BF und 7F. Also zum Beispiel an Adresse 8300 (sedezimal) 3E FE D3 02 3E FD D3 02 3E FB D3 02 3E F7 D3 02 3E EF D3 02 3E DF D3 02 3E BF D3 02 3E 7F D3 02 eigeben. Dann drücken Sie BEF und anschließend STEP. Geben Sie die erste Adresse an (im Beispiel 8300) und drücken CR. Jetzt müßte 8300 3E auf der Anzeige stehen. Wenn Sie jetzt mehrmals hintereinander CR drücken, wandert ein Lichtpunkt über die LEDs. Wenn die

LED ganz links leuchtet müssen Sie aufhören, denn unser Programm geht hier nicht weiter. Das können Sie aber leicht ändern: Fügen Sie noch zwei Byte an das Programm an, 18 und DE (im Beispiel auf Adresse 8320 und 8321); das bedeutet jr 8300 - jump relative to 8300, das Programm fängt wieder von vorne an. Probieren Sie das Programm nocheinmal mit dem STEP Befehl aus! Diesmal haben Sie schon ein komplettes Lauflicht. Wenn das Programm fehlerfrei arbeitet, können Sie daran gehen, den Programmablauf im Prozessortakt stattfinden zu lassen. Drücken Sie dazu BEF, START, geben die Adresse (im Beispiel 8300) ein und drücken CR. Nanu? Alle LEDs leuchten halbhell, irgendwas scheint nicht zu funktionieren! Drücken Sie erstmal RESET, damit das HALLO-1.1 wieder erscheint. Wahrscheinlich wissen Sie schon, wohin der Hase läuft: Das Programm ist zu schnell, unser Auge kann nicht folgen. Dann machen wir es doch langsamer! Nach jedem Schritt soll das Programm einen kleinen Moment warten. Warten ist etwas, das der Mikroprozessor eigentlich gar nicht kann, wir müssen ihn irgendend was tun lassen, er kann zum Beispiel zählen. Das können Sie in einem Unterprogramm programmieren, das dann nach jedem Schritt mit dem CALL Befehl aufgerufen wird. In diesem Unterprogramm lassen Sie dem Mikroprozessor zum Beispiel von B000 (sedezimal) bis FFFF (sedezimal) zählen und dann noch einmal von FFFF bis 0000 (sedezimal). Dabei tritt ein Übertrag auf (FFFF + 0001 ist ja eigentlich 10000). Diesen Übertrag (Carry) können Sie zum Abbruch des Wartens benutzen. Ihr Unterprogramm muß dann nur noch mit Hilfe des RET (Return) Befehls ins Hauptprogramm zurückkehren. Setzen Sie das alles zusammen, so erhalten Sie das Programm LAUF-1.

Das ist aber nur ein erster Ansatz, wir haben zwar ein funktionsfähiges Lauflicht, das Programm ist aber recht lang und etwas umständlich. Denn im Befehlssatz des Mikroprozessors Z80 gibt es eine Gruppe von Befehlen, die uns hier sehr helfen können: Die Rotations- und Schiebebefehle. Natürlich dienen diese Befehle nicht eigentlich dazu Lauflichter zu programmieren, man kann mit diesen Befehlen zum Beispiel in bestimmten Fällen Zahlen multiplizieren und dividieren.

Um die Wirkungsweise dieser Befehle zu verstehen, schauen wir uns zum Beispiel den RLCA Befehl an. RLCA bedeutet Rotiere Links Circulär den Akkumulator. Gemeint ist, daß Irgendetwas linksherum im Kreis bewegt wird, und daß dieses Etwas im Akkumulator steht. Im Akkummulator stehen doch Zahlen, wie kann man Zahlen denn im Kreis bewegen ? Schauen Sie sich folgendes Beispiel mit vierstelligen Dezimalzahlen an: 1234 -> 2341 -> 3412 -> 4123 -> 1234 uund so weiter. Das ist also mit "Rotiere Circulär" gemeint. Im Akkumulator des Mikroprozessors stehen aber keine Dezimal- sondern Binärzahlen. Wie sieht es damit aus ?

Auch dazu ein Beispiel:
0001 -> 0010 -> 0100 -> 1000 -> 0001
u.s.w.,
oder wie beim Z80 mit achtstelligen Binärzahlen:
00000001 -> 00000010 -> 00000100 ->
00001000 -> 00010000 -> 00100000 ->
01000000 -> 10000000 -> 00000001
u.s.w.

Wenn wir jetzt als erste Zahl 11111110 nehmen, dann entsteht genau die Folge, die wir oben für unser Lauflicht benutzt haben. Nach diesem Verfahren ist das Programm LAUF-2 geschrieben, nur wird hier als Startwert 11101110 verwendet, so daß immer zwei LEDs gleichzeitig leuchten. Dieser Wert wird zunächst in den Akkumulator geladen, dann folgt die Hauptschleife des Programms: Ausgeben des Akkumulators an die LEDs, warten, Akku zirkulär rotieren und das ganze wiederholen. Zum warten wird wieder das Unterprogramm wait benutzt, das Sie schon aus LAUF-1 kennen.

Dieses neue Verfahren läßt sich nun sehr leicht abwandeln. Sie können zum Beispiel 00000001 als Startwert nehmen (01 sedezimal), dann leuchten alle LEDs bis auf eine. Oder Sie nehmen den Wert 55 sedezimal, das ist 01010101 binär, dann wechseln sich jeweils eine leuchtende und eine dunkle LED ab.

Wenn Sie aber den Wert 6D sedezimal (01101101 binär) nehmen, sieht das Lauflicht etwas seltsam aus, das liegt daran, das Sie drei helle LEDs nicht gleichmäßig über 8-Bit verteilen können. Wir brauchen also ein neuntes Bit. Dieses neunte Bit kann man im Carry-Flag speichern. Das Carry-Flag ist ein Flag

(eine Flagge, entweder gesetzt oder nicht gesetzt, halbmast gibt es hier nicht), das einen Übertrag anzeigt (engl.: carry - Übertrag).

Dieses Carry-Flag wird von dem Befehl RLA, Rotiere den Akkumulator links herum, benutzt. Bei diesem Befehl wandert das Bit 7 des Akkumulators (das Bit ganz links) in das Carry, während das Carry in Bit 0 (das Bit ganz rechts) wandert:

	I	Carry	I	Akkuinhalt
1.	I	1	I	01101101
2.	I	0	I	11011011
3.	I	1	I	10110110
4.	I	1	I	01101101
5.	I	0	I	11011011
6.	I	1	I	10110110
7.	I	1	I	01101101
8.	I	0	I	11011011
9.	I	1	I	10110110

Das Carry kann nicht mit an die LEDs ausgegeben werden, da nur acht da sind. Trotzdem sieht das Lauflicht mit Programm LAUF-3 besser aus, als LAUF-2 mit dem Anfangsert 01101101 binär. Vergleicht man LAUF-2 mit LAUF-3, dann sieht man, das sich nicht nur der Rotierbefehl geändert hat. Beim Programmstart muß das Carry gesetzt werden, das macht der Befehl SCF - Set Carry Flag. Außerdem verändert das Unterprogramm wait das Carry-Flag, denn es zählt bis ein Übertrag auftritt. Wir wollen aber nach Aufruf des Unterprogramms unser Original-Carry wieder verwenden. Deshalb muß der Inhalt des Flag-Registers am Anfang des Unterprogrammes gerettet werden, das übernimmt der Befehl push AF (Schiebe Akku und Flags auf den Stack). Am Ende des Unterprogramms, vor dem Return zum Hauptprogramm, wird der alte Zustand wieder hergestellt mit dem Befehl pop AF (Nehme Akku und Flags vom Stack). Diese Befehle arbeiten mit einem besonderem Speicherbereich, dem Stack (Stapel). Wo dieser Speicherbereich anfängt steht im Register SP, dem Stackpointer (Zeiger auf den Stapel). Mit push Befehlen kann man Register des Z80 auf den Stack schieben, von dem man sie später mit pop Befehlen wieder herunterholt. Auf diese Art konnten wir den Inhalt des Akkumulators und des Carry-Flags zwischenspeichern, so daß das Hauptprogramm gar nicht merkt, daß diese Register im Unterprogramm benutzt wurden.

```
                        ;   Programm  LAUF-1     1.0
                        ;
                        ;   Einfaches Lauflicht.

                        .Z80
                                ORG 01190H

                        LAUF1:
1190  3E FE                      ld a,11111110b  ; Akku laden,
1192  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die erste LED leuchtet
1194  CD CA 11                   call wait
1197  3E FD                      ld a,11111101b  ; Akku laden,
1199  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die zweite LED leuchtet
119B  CD CA 11                   call wait
119E  3E FB                      ld a,11111011b  ; Akku laden,
11A0  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die dritte LED leuchtet
11A2  CD CA 11                   call wait
11A5  3E F7                      ld a,11110111b  ; Akku laden,
11A7  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die vierte LED leuchtet
11A9  CD CA 11                   call wait
11AC  3E EF                      ld a,11101111b  ; Akku laden,
11AE  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die fünfte LED leuchtet
11B0  CD CA 11                   call wait
11B3  3E DF                      ld a,11011111b  ; Akku laden,
11B5  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die sechste LED leuchtet
11B7  CD CA 11                   call wait
11BA  3E BF                      ld a,10111111b  ; Akku laden,
11BC  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die siebte LED leuchtet
11BE  CD CA 11                   call wait
11C1  3E 7F                      ld a,01111111b  ; Akku laden,
11C3  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben, die achte LED leuchtet
11C5  CD CA 11                   call wait
11C8  18 C6                      jr LAUF1

11CA  21 00 B0          wait:    ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
11CD  11 01 00                   ld de,0001H     ; oder hier
                        Wloop:
11D0  19                         add hl,de       ; HL um einen hochzählen (in DE steht 1)
11D1  30 FD                      jr nc,Wloop     ; Wiederholen, bis Überlauf (Carry),
                                                 ; also springen, wenn kein Carry (no carry - nc)
11D3  C9                         ret             ; zurück zum Hauptprogramm
                        end
```

```
                        ;  Programm  LAUF-2     1.0
                        ;
                        ;  Einfaches Lauflicht, aber geschickter programmiert als LAUF-1

                        .Z80
                               ORG 011E0H

                        LAUF2:
11E0  3E EE                    ld a,11101110b  ; Akku laden
                        LauLoop:
11E2  D3 02                    out (02H),a     ; Akku an die LEDs ausgeben, zwei LEDs leuchten
11E4  CD EA 11                 call wait2      ; warten
11E7  07                       rlca            ; Rotiere den Akku circulär nach links
11E8  18 F8                    jr LauLoop      ; Wiederholen

11EA  21 00 B0          wait2: ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
11ED  11 01 00                 ld de,0001H     ; oder hier
                        W2loop:
11F0  19                       add hl,de       ; HL um einen hochzählen (in DE steht 1)
11F1  30 FD                    jr nc,W2loop    ; Wiederholen, bis Überlauf (Carry),
                                               ; also springen, wenn kein Carry (no carry - nc)
11F3  C9                       ret             ; zurück zum Hauptprogramm
                        end
```

```
                        ;   Programm  LAUF-3     1.0
                        ;
                        ;   Lauflicht mit neun Takten

                        .Z80

                                ORG 01200H

                        LAUF3:
1200  3E 6D                      ld a,01101101b  ; Akku laden, drei LEDs leuchten
1202  37                        scf             ; Set Carry Flag, da Carry mitrotiert wird
                        LauLoop:
1203  D3 02                      out (02H),a     ; Akku an die LEDs ausgeben
1205  CD 0B 12                   call wait3      ; warten
1208  17                        rla             ; Rotiere den Akku nach links durch das Carry
1209  18 F8                     jr LauLoop      ; Wiederholen

120B  F5                  wait3:  push af       ; rette Akku und Flags auf den Stack
120C  21 00 B0                    ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
120F  11 01 00                    ld de,0001H     ; oder hier
                          W3loop:
1212  19                          add hl,de       ; HL um einen hochzählen (in DE steht 1)
1213  30 FD                       jr nc,W3loop    ; Wiederholen, bis Überlauf (Carry),
                                                  ; also springen, wenn kein Carry (no carry - nc)
1215  F1                        pop af          ; Akku und Carry wieder in Originalzustand versetzen
1216  C9                        ret             ; zurück zum Hauptprogramm
                        end
```

REAKTIONSZEIT

Mit dem Programm Reaktionszeit können Sie feststellen, wie gut Ihre Reaktion auf ein Signal ist. In diesem Fall, müssen Sie auf ein Erlöschen der Leuchtdioden reagieren.

Das Programm wartet nach dem Aufruf zunächst auf eine Taste. Wenn Sie nun eine beliebige Taste drücken, beginnt das Programm zu laufen. Zunächst leuchten die acht Leuchtdioden auf der HEXIO2. Die Dauer diese Leuchtens ist bei jedem Durchlauf verschieden. Dies wird erreicht durch das Laden eines Zeitfaktors, mit einem Wert aus dem Refresh-Register. Das Refresh-Register ist ein Zähler, der die Speicher des Rechners nachläd. Das Nachladen der Register erfolgt in sehr kurzen Zeitabständen, da der Rechner sonst die Daten wieder vergißt.

Nach einer bestimmten Zeit erlöschen die LED's. Jetzt beginnt die Zeitmessung. In der Anzeige sehen Sie nun drei Blöcke von Zahlen. Der ganz links gelegene Block zählt die Minuten (solange sollten Sie wohl nicht brauchen). Der mittlere Block zählt die Sekunden die vergehen, ehe Sie eine Taste drücken. Der rechte Block schließlich, zählt die Zehntel und die Hundertstel Sekunden..

Durch ein erneutes Drücken, auf irgendeine Taste startet das Programm von neuem.

```
                        .z80
                              TITLE   REAKTIONSTEST

                        HOLETASTE       EQU     000CH
                        CLEAR   EQU     0033H
                        ANZEIGE EQU     0009H
                        PRTAC   EQU     0018H
                        ANZFELD EQU     8000H
                        MINUTEN EQU     80FCH
                        SEKUNDEN EQU    80FDH
                        MILLISEKUNDEN EQU   80FEH

                              ORG     1230H

                        START:
1230  CD 33 00                CALL    CLEAR           ;Anzeige löschen

                        BEGINN:
1233  CD 0C 00                CALL    HOLETASTE       ;Tastatur abfragen

                        START1:
1236  3E 00                  LD      A,00000000B
1238  D3 02                  OUT     (02H),A         ;Akkuinhalt auf
                                                      ;LED's ausgeben
123A  32 FC 80               LD      (MINUTEN),A     ;Anfangswerte
123D  32 FD 80               LD      (SEKUNDEN),A    ;einstellen
1240  32 FE 80               LD      (MILLISEKUNDEN),A
1243  CD B3 12               CALL    WAIT

                        LED$AUS:
1246  3E FF                  LD      A,11111111B
1248  D3 02                  OUT     (02H),A         ;LED's ausschalten

124A  FD 21 FC 80            LD      IY,MINUTEN      ;Anfang der Zeit-
124E  FD 36 02 00            LD      (IY+2),0        ;messung

                        LOOP:
1252  06 63                  LD      B,063H          ;Anfang Warte-
                                                      ;schleife
                        MILLISEK:
1254  CD 7F 12               CALL    ZEITANZEIGE
1257  10 FB                  DJNZ    MILLISEK

1259  FD 7E 01               LD      A,(IY+1)        ;Sek.:=Sek.+1
125C  3C                     INC     A
125D  27                     DAA
125E  FD 77 01               LD      (IY+1),A

1261  FE 60                  CP      60H             ;Wenn Sek.=60
1263  20 14                  JR      NZ,MILLIO
1265  FD 7E 00               LD      A,(IY+0)        ; Min.:=Min.+1
1268  3C                     INC     A
1269  27                     DAA
126A  FD 77 00               LD      (IY+0),A

126D  FE 60                  CP      60H
126F  20 04                  JR      NZ,SEKO
1271  FD 36 00 00            LD      (IY+0),0        ;Minuten:=0

                        SEKO:
1275  FD 36 01 00            LD      (IY+1),0        ;Sekunden:=0
```

```
                        MILLIO:
1279  FD 36 02 00               LD      (IY+2),0          ;Millisek.:=0

127D  18 D3                     JR      LOOP              ;unendlich viele
                                                          ;Durchläufe
                        ZEITANZEIGE:
127F  C5                        PUSH    BC                ;Zähler einstellen
1280  06 03                     LD      B,03H

                        DISPLAY:
1282  DD 21 00 80               LD      IX,ANZFELD        ;Minuten anzeigen
1286  FD 7E 00                  LD      A,(IY+0)
1289  CD A8 12                  CALL    ANZEIGEN

128C  DD 23                     INC     IX
128E  FD 7E 01                  LD      A,(IY+1)          ;Sekunden anzeigen
1291  CD A8 12                  CALL    ANZEIGEN

1294  DD 23                     INC     IX
1296  FD 7E 02                  LD      A,(IY+2)          ;Millisek. anzeigen
1299  CD A8 12                  CALL    ANZEIGEN

129C  10 E4                     DJNZ    DISPLAY           ;Zähler dekrementieren
                                                          ;bis Zähler=0
129E  FD 7E 02                  LD      A,(IY+2)          ;Milli.:=Milli.+1
12A1  3C                        INC     A
12A2  27                        DAA
12A3  FD 77 02                  LD      (IY+2),A

12A6  C1                        POP     BC

12A7  C9                        RET

                        ANZEIGEN:
12A8  CD 18 00                  CALL    PRTAC             ;Akkuinhalt ausgeben

                        HOLL:
12AB  CD 09 00                  CALL    ANZEIGE           ;Anzeige aus Grund-
                                                          ;programm aufrufen
                                                          ;gleichzeigig Tastatur
                                                          ;abfragen
12AE  FE FF                     CP      0FFH              ;Taste gedrückt ?
12B0  38 14                     JR      C,STOP            ;Ja, Uhr anhalten

12B2  C9                        RET

                        WAIT:
12B3  ED 5F                     LD      A,R               ;Warteschleife
12B5  47                        LD      B,A               ;einstellen
                        W1:
12B6  21 00 F0                  LD      HL,0F000H
12B9  11 01 00                  LD      DE,0001H

                        W2:
12BC  CD C2 12                  CALL    WAIT1
12BF  10 F5                     DJNZ    W1                ;B dekrementieren
                                                          ;bis B=00h
12C1  C9                        RET

                        WAIT1:
12C2  19                        ADD     HL,DE             ;HL und DE addieren
12C3  30 FD                     JR      NC,WAIT1          ;bis HL=0FFFFH
12C5  C9                        RET
```

```
                        STOP:
12C6  D1                        POP     DE
12C7  CD D5 12                  CALL    SCHLEIFE        ;Warteschleife aufrufen

                        LOOP2:
12CA  CD 09 00                  CALL    ANZEIGE         ;Anzeigenroutine aufrufen

12CD  FE FF                     CP      OFFH            ;Vergleich ob Taste gedrückt
12CF  DA 33 12                  JP      C,BEGINN        ;wenn ja zum Anfang springen
12D2  18 F6                     JR      LOOP2

12D4  C9                        RET

                        SCHLEIFE:
12D5  21 01 00                  LD      HL,0001H        ;Warteschleife starten

                        SCHL1:
12D8  29                        ADD     HL,HL
12D9  30 FD                     JR      NC,SCHL1

12DB  C9                        RET

                        ENDE:
                                END
```

WÜRFEL

Bei diesem Programm handelt es sich um einen elektronischen Würfel. Es ist aber nicht so, daß der Computer irgendwelche Zufallszahlen berechnet, sondern er durchläuft eine Schleife. In dieser Schleife, schaltet er nacheinander alle LED's ein.

Da er nach jeder LED, die er eigeschaltet hat, in eine Warteschleife springt, in der er die Tastatur abfragt. Wenn eine Taste gedrückt wird, bleibt der Rechner an dieser Stelle stehen und zeigt solange das Würfelergebnis an.

```
                        TITLE   WÜRFEL    3

                holtaste        equ     000ch
                clear   equ     0033h
                tonum   equ     000fh
                anzeige equ     0009h

                        org     1300h

                start:
1300  CD 33 00              call    clear           ;Anzeige löschen

                auswahl:
1303  CD 0C 00              call    holtaste        ;Tastatur abfragen
1306  CD 0F 00              call    tonum           ;Tastencode bilden
1309  4F                    ld      c,a             ;und ins Register c laden

                aus:
130A  3E 0F                 ld      a,0fh           ;Akku mit 0fh laden
130C  B9                    cp      c               ;mit c vergleichen
130D  D4 13 13              call    nc,tab          ;wenn a=c dann Tabelle anspringen

1310  C3 03 13              jp      auswahl         ;sonst nocheinmal

                tab:
1313  3E FD                 ld      a,11111101b     ;Akku mit Wert laden
1315  D3 02                 out     (02h),a         ;ausgeben auf die LED's
1317  CD 3F 13              call    wait            ;Warteschleife aufrufen

131A  3E F9                 ld      a,11111001b     ;nächster Wert
131C  D3 02                 out     (02h),a
131E  CD 3F 13              call    wait

1321  3E F1                 ld      a,11110001b
1323  D3 02                 out     (02h),a
1325  CD 3F 13              call    wait

1328  3E E1                 ld      a,11100001b
132A  D3 02                 out     (02h),a
132C  CD 3F 13              call    wait

132F  3E C1                 ld      a,11000001b
1331  D3 02                 out     (02h),a
1333  CD 3F 13              call    wait

1336  3E 81                 ld      a,10000001b
1338  D3 02                 out     (02h),a
133A  CD 3F 13              call    wait

133D  18 D4                 jr      tab

                wait:
133F  21 FF FF              ld      hl,0ffffh       ;hl mit 0ffffh laden

                loop:
1342  29                    add     hl,hl           ;hl zu sich selbst addieren
1343  30 FD                 jr      nc,loop         ;bis hl voll (0ffffh),d.h.hier ein
                                                    ;Durchlauf
1345  CD 52 13              call    holetaste       ;Tastatur abfragen
1348  CD 0F 00              call    tonum           ;Tastencode bilden
134B  4F                    ld      c,a             ;in c-Register laden
134C  3E 01                 ld      a,01h           ;Akku mit 01h laden
134E  B9                    cp      c               ;mit c vergleichen
```

```
134F  30 B2                   jr      nc,auswahl          ;Vergleich o.k.zum Anfang springen

1351  C9                      ret

                        holetaste:
1352  06 08                   ld      b,8                 ;Tastatur 8 mal abfragen

                        holl1:
1354  CD 09 00                call    anzeige             ;wenn keine Taste gedrückt
1357  FE FF                   cp      0ffh                ;wurde wieder zu Hauptprogramm
1359  20 F7                   jr      nz,holetaste        ;springen
135B  10 F7                   djnz    holl1

135D  C9                      ret
```

SÄGEZAHN

Dieses Programm schaltet die LED's nacheinander ein. Wenn alle Leuchtdioden leuchten, werden sie im nächsten Schritt alle wieder gelöscht.

Auf diese Weise wird durch die Leuchtdioden ein Sägezahnfunktion simuliert, wie sie in der Technik recht häufig vorkommt.

Zur Funtionsweise des Programms:

Immer wenn der Akku mit einem neuen Wert geladen worden ist, gibt er diesen auf die LED's aus. Anschließend wird eine Warteschleife aufgerufen um den Wert eine bestimmte Zeit anzuzeigen.

Nach dieser Wartezeit wird ein neuer, incrementierter Wert in den Akku geladen. Nun werden Sie wieder ausgegeben und der Ablauf geht immer so weiter.

```
                        TITLE   SÄGEZAHN - FUNKTION

                clear   equ     0033h

                        org     1360h

                start:
1360 CD 33 00           call    clear           ;Anzeige löschen

                leucht:                         ;Hauptprogramm
1363 3E FF              ld      a,11111111b     ;Akku mit ffh laden
1365 D3 02              out     (02H),a         ;ausgeben auf die LED's
1367 CD A4 13           call    wait            ;Warteschleife aufrufen

136A 3E FE              ld      a,11111110b     ;eine Leuchtdiode nach der anderen
136C D3 02              out     (02h),a         ;einschalten und auf die LED's ausgeben
136E CD A4 13           call    wait            ;Warteschleife aufrufen

1371 3E FC              ld      a,11111100b
1373 D3 02              out     (02h),a
1375 CD A4 13           call    wait

1378 3E F8              ld      a,11111000b
137A D3 02              out     (02h),a
137C CD A4 13           call    wait

137F 3E F0              ld      a,11110000b
1381 D3 02              out     (02h),a
1383 CD A4 13           call    wait

1386 3E E0              ld      a,11100000b
1388 D3 02              out     (02h),a
138A CD A4 13           call    wait

138D 3E C0              ld      a,11000000b
138F D3 02              out     (02h),a
1391 CD A4 13           call    wait

1394 3E 80              ld      a,10000000b
1396 D3 02              out     (02h),a
1398 CD A4 13           call    wait

139B 3E 00              ld      a,00000000b
139D D3 02              out     (02h),a
139F CD A4 13           call    wait

13A2 18 BF              jr      leucht          ;unendlich oft wiederholen

                wait:
13A4 21 00 00           ld      hl,0000h        ;HL-Register mit 0000h laden
13A7 11 01 00           ld      de,0001h        ;DE-Register mit 0001h laden

                loop:
13AA 19                 add     hl,de           ;hl und de addieren
13AB 30 FD              jr      nc,loop         ;bis hl voll (0ffffh)

13AD C9                 ret                     ;dann zurück ins Hauptprogramm
```

BLINKLICHT

Mit diesem Programm wird ein Blinklicht realisiert, das über die Leuchtdioden angezeigt wird. Die LED's werden immer für eine bestimmte Zeit eingeschaltet und anschließend für die selbe Zeit wieder aus.

Die Funktionsweise des Programms ist recht einfach.

Der Akku wird zunächst mit dem Wert 1111 1111 b geladen. Dies entspricht einem Ausschalten der Leuchtdioden. Dann springt das Programm in eine Wrateschleife. Nach einer dort eingestellten Zeit, kehrt es wieder zurück ins Hauptprogramm. Ein neuer Wert, nämlich 0000 0000 b wird in den Akku geladen. Wieder ein Sprung in die Warteschleife und dann immer weiter so, bis der "RESET"-Taster gedrückt wird.

```
                        TITLE   BLINKLICHT

                clear   equ     0033h

                        org     13B0h

                start:
13B0  CD 33 00          call    clear           ;Anzeige löschen

              . blinken:
13B3  3E FF             ld      a,11111111b     ;LED's ausschalten
13B5  D3 02             out     (02h),a
13B7  CD C4 13          call    wait            ;Warteschleife aufrufen

13BA  3E 00             ld      a,00000000b     ;LED7S einschalten
13BC  D3 02             out     (02h),a
13BE  CD C4 13          call    wait            ;Warteschleife aufrufen

13C1  C3 B3 13          jp      blinken         ;unendlich oft wiederholen

                wait:
13C4  21 00 00          ld      hl,0000h        ;hl-Register mit 0000h laden
13C7  11 02 00          ld      de,0002h        ;de-Register mit 0001h laden

                loop:
13CA  19                add     hl,de           ;hl und de addieren
13CB  30 FD             jr      nc,loop         ;bis hl voll (0ffffh)

13CD  C9                ret
```

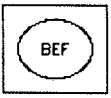
Was bedeuten die Tasten?

Um die nachfolgend beschriebenen Funktionen beobachten zu können, muß Ihr System sich im Befehlsmodus befinden, d.h. Sie müssen vor jedem neuen Test die BEF-Taste bzw. den RESET-Taster drücken.

Für die ersten Tests genügt es, wenn Sie die nachfolgend beschriebenen Tasten kennen.

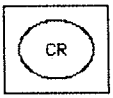
BEF

Bef ist die Abkürzung für Befehl. Mit dieser Taste kann man von den unterschiedlichen Eingabeformen aus, wieder zur Befehlseingabe gelangen. Ihr System befindet sich, nach einem Druck auf diese Taste im selben Zustand wie nach einem RESET. D.h. erwartet einen Befehl wie z.B. START oder STEP etc.



CR

Abkürzung für "Carriage Return" oder auf deutsch für Wagenrücklauf. Diese Funktion kennen Sie bestimmt von einer Schreibmaschine her. Mit dieser Taste werden sämtlichen Eingaben bestätigt. Egal ob Sie einen Befehl eingegeben haben oder ein beliebiges Byte eingetippt haben, müssen Sie die CR-Taste drücken damit Ihr Computer sich den eingegebenen Wert merkt bzw. die angezeigte Startadresse erkennt.



START

Wenn Sie diese Taste drücken, können Sie das an der angezeigten Stelle befindliche Programm starten. Es wird vom System aus immer die Adresse 8100 vorgeschlagen. Wenn Sie mit dieser Adresse einverstanden sind, drücken Sie jetzt die CR-Taste. Wenn nicht geben Sie mit dem Tastenfeld die Adresse ein, an der Ihr Programm sich befindet. Nach dem bestätigen der Adresse startet Ihr Programm.

Beispiel:

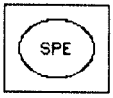
Starten des vorher bereits gestarteten Reaktionstestprogramms:

Eingabe	Anzeige	Bemerkung
BEF	- - - - - - - -	Befehlsmodus einstellen
START	A d r 8.1.0.0	Start
1	A d r 1.0.0.1	
2	A d r 0.0.1.2	Startadresse
3	A d r 0.1.2.3	eingeben
0	A d r 1.2.3.0	
CR		Startadresse bestätigen

Und schon läuft das Programm!

SPE

Abkürzung für Speicher. Nach dem Betätigen dieser Taste können Sie die Inhalte der Speicher anschauen und verändern. Auch hier wird zunächst die Adresse 8100 angezeigt. Wenn Sie damit einverstanden sind, quittieren Sie die Adresse mit der CR-Taste, wenn nicht geben Sie mit der Tastatur eine neue Adresse ein und bestätigen Sie diese mit der CR-Taste. Mit den Tasten +,-und CR kann man durch den Speicher rollen, d.h. man kann sich eine Adresse nach der anderen anschauen und bei Bedarf verändern.



STEP

Wenn Sie diese Taste gedrückt haben läuft das ausgewählte Programm in Einzelschritten ab. Die Funktion ist hauptsächlich zum Testen von Programmen gedacht. Wie bei der Funktion START wird auch hier eine Anfangsadresse angezeigt, im Normalfall 8100. Wenn Ihr Programm an dieser Stelle steht quittieren Sie die Adresse mit der CR-Taste, ansonsten geben Sie eine neue Adresse ein und bestätigen Sie diese mit **CR**. Jetzt haben Sie die Möglichkeit durch das Programm zu rollen, d.h. Sie können nach jedem Druck auf die **CR**-Taste, das erste Byte des vom Programm angesprungenen Befehls anschauen. Ferner haben Sie die Möglichkeit nach einem Druck auf die **BEF**-Taste und einem anschließenden Druck auf die **4 REG**-Taste die einzelnen Registerinhalte

anzuschauen. Wie Sie hierbei vorgehen müssen lesen Sie bitte bei der Beschreibung der **REG**-Taste nach.

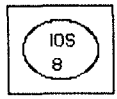
Auch hier ein Beispiel:

Eingabe	Anzeige	Bemerkung
BEF	- - - - - - - -	Befehlmodus einstellen
STEP	A d r 8.1.0.0	Einzelschrittfunktion
1	A d r 1.0.0.1	
3	A d r 0.0.1.3	Startadresse
B	A d r 0.1.3.B	eingeben
0	A d r 1.3.B.0	
CR		Startadresse bestätigen

Mit jedem neuen Druck auf die CR-Taste erscheint der nächste Befehl.

8 IOS

Abkürzung für IO-setzen oder zu deutsch Ein- oder Ausgabeadresse setzen. Mit dieser Funktion ist es möglich Werte auf bestimmte Portadressen auszugeben. D.h. man kann Bitkombinationen auf einer Anzeige ausgeben. Um die Funktion zu testen gehen Sie wie folgt vor. Drücken Sie die **8 IOS**-Taste. In der Anzeige steht jetzt **00** . Der Computer erwartet nun eine Portadresse auf der er die Bytes ausgeben soll. Geben Sie hier z.B. **02** ein und drücken Sie die **CR**-Taste. In der Anzeige erscheint jetzt **02 00**. Jetzt können Sie beliebige Bitkombinationen auf die Leuchdioden ausgeben und die eingegebenen Bytes erscheinen dort als leuchtende und dunkle LED's. Die Ausgabe erfolgt invertiert, d.h. die LED's die hell sind, sind von der Bitkombination her auf Null gelegt. Wenn Sie die Portadresse **00**oder **01** eingeben werden die Bytes auf der Anzeige angezeigt. Hierbei sehen Sie dann rechts nur die eingegebenen Werte.



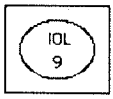
Hierzu ein Beispiel:

Eingabe	Anzeige	Bemerkung
8 IOS	0 0	Funktion anwählen
0	0 0	Portadresse anwählen
2	0 2	
CR	0 2 0 0	Portadresse bestätigen
A	0 2 0 A	Byte eingeben
A	0 2 A A	
CR		

Jetzt leuchtet jede zweite Leuchtdiode auf der HEXIO2

9 IOL

Abkürzung für IO-lesen oder auf deutsch Ein- und Ausgabeadressen lesen. Wenn Sie diese Taste drücken können Sie Bitkombinationen von einem Eingabeport der HEXIO2 einlesen. Gehen Sie wie folgt vor: Zunächst erscheint folgendes Bild in der Anzeige **00** . Wenn Sie jetzt **02** eingeben und CR drücken, können Sie die auf dem DIL-Schalter eingestellten Werte als Zahlenkombinationen anzeigen. Nachdem Sie die **CR**-Taste gedrückt haben, erscheint folgendes Bild **02 XX**. Wobei XX dem eingestellten Wert entspricht. Wenn Sie jetzt die **OPT**-Taste drücken erscheint der angezeigte Wert in binärer Darstellung. Nochmaliges drücken auf **OPT** und er erscheint wieder in hexadezimaler Form.



Und wieder ein Beispiel:

Eingabe	Anzeige	Bemerkung
9 IOL	0 0	Funktion anwählen
0	0 0	Eingabeport anwählen
2	0 2	
CR		Eingabeport bestätigen

Jetzt wird die, im Augenblick auf dem DIL-Schalter eingestellte Bitkombination angezeigt.

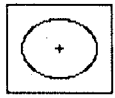
OPT	X X X X X X X X	binäre Ausgabe

Nun steht die eingestellte Bitkombination als Binärzahl in der Anzeige.

Die anschließend besprochenen Funktionen müssen Sie erst beherrschen, wenn Sie weitere Versuche anstellen wollen, oder wenn Sie selbst Programme schreiben wollen.

+

Diese Taste dient zum Anwählen der nächsten Adresse bzw. des nächsten Registers. Sie wird z. B. verwendet um die Speicherinhalte von mehreren Adressen anzusehen. Die Adresse wird jedesmal um eine Ziffer erhöht. In Verbindung mit der **SPE**-Taste kann mit dieser Funktion das komplette Programm Byte für Byte angeschaut werden.



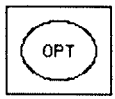
Beispiel:

Eingabe	Anzeige	Bemerkung
SPE	A d r 8.1.0.0.	Speicher anschauen
+	8 1 0 0 X X	Startadresse anwählen
+	8 1 0 1 X X	nächste Adresse anschauen

Auf diese Weise können Sie den kompletten Speicher Ihres Systems anschauen.

OPT

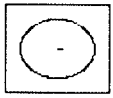
Abkürzung für Option. Dient dem Umschalten, z.B. von Zahlenbereichen oder von Darstellungsarten auf der Anzeige. In Verbindung mit der **UMW**-Funktion können Sie z.B. vom Heaxdezimalen Zahlensystem in das Dezimalsystem umzuwandeln und umgekehrt. Genauere Beschreibung der UMW-Funktion lesen Sie bitte dort nach.



Ein Beispiel hierfür, haben Sie bereits bei der Funktion **9 IOL** kennengelernt.

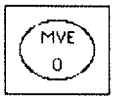
-

Diese Taste hat die selbe Funktion wie die +-Taste nur, daß damit die Adressen nicht erhöht, sondern decrementiert werden. Nach dem Anwählen der **SPE**-Taste können Sie rückwärts durch die Speicherinhalte rollen.



0 MVE

Abkürzung für move oder auf deutsch bewegen. Mit dieser Taste ist es möglich, einen kompletten Speicherbereich zu verschieben. Der Computer überprüft selbstständig, ob die zu verschiebenden Speicherbereiche sich überlappen. Auf diese Art wird garantiert, daß der Ganze zu übertragende Bereich verarbeitet wird. Um die Funktion auszutesten gehen Sie wie folgt vor: Drücken Sie die **0 MVE**-Taste, in der Anzeige steht jetzt **VON 0000**. Geben Sie die Anfangsadresse der zu verschiebenden Programm stelle ein und drücken Sie die **CR**-Taste. Jetzt steht in der Anzeige **bIS XXXX**. Beben Sie die Endadresse des zu verschiebenden Speicherbereiches ein und quittieren Sie diese. Jetzt steht auf der Anzeige **NAC XXXX**. Geben Sie hier die Adresse ein an die der Speicherbereich verschoben werden soll und quittieren Sie diese Adresse.

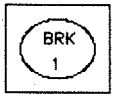


Beispiel:

Eingabe	Anzeige	Bemerkung
0 MVE	V O N 0 0 0 0	Funktion anwählen
8	V O N 0 0 0 8	Anfangsadresse eingeben
1	V O N 0 0 8 1	
0	V O N 0 8 1 0	
0	V O N 8 1 0 0	
CR	b I S 0 0 0 0	Anfangsadresse bestätigen
8	b I S 0 0 0 8	Endadresse festlegen
1	b I S 0 0 8 1	
F	b I S 0 8 1 F	
F	b I S 8 1 F F	
CR	N A C 0 0 0 0	Endadresse bestätigen
8	N A C 0 0 0 8	Zieladresse eingeben
2	N A C 0 0 8 2	
0	N A C 0 8 2 0	
0	N A C 8 2 0 0	
CR	- b E F -	Zieladresse bestätigen

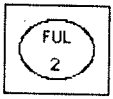
Damit haben Sie den kompletten Speicherbereich übertragen.

1 BRK



Abkürzung für "break" oder zu deutsch unterbrechen. Es lassen sich mit dieser Taste bis zu drei Unterbrechungen setzen, an denen das Programm dann automatisch unterbrochen wird. Zum Testen von Programmen ist das sehr nützlich. Vorgehensweise wie folgt: Nach dem Drücken der **1 BRK**-Taste erscheint auf der Anzeige **01 0000**. Hier können Sie den ersten **Break** setzen. Nach dem Bestätigen dieser Adresse können Sie noch zwei weitere Breaks auf die gleiche Art setzen. Beim Ablaufenlassen des Programms hält es zunächst an der von Ihnen als Breakadresse gesetzten Speicherstelle anhalten. Wenn Sie Ihr Programm dann von dieser Adresse neu starten bleibt es bei dem nächsten gesetzten Break stehen usw. Wenn Sie nur einen Break setzen wollen geben Sie nur für diesen eine Adresse ein, die anderen beiden können Sie dann durch einfaches Quittieren der angezeigten Adresse(**02 0000**) bzw. (**03 0000**) ungültig machen. Nach jedem Break können z.B. die Register angeschaut werden.

2 FUL



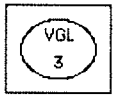
Abkürzung für füllen. Mit dieser Taste kann man einen Speicherbereich mit einem konstanten Wert auffüllen. Der Rechner überprüft automatisch ob alle angesprochenen Adressen auch richtig belegt wurden. Betätigen Sie die **2 FUL**-Taste. Es erscheint **VON 0000** auf der Anzeige. Geben Sie die Anfangsadresse des zu füllenden Bereiches an und quittieren Sie diese Adresse. Nun erscheint **bIS 0000** in der Anzeige. Geben Sie hier die Endadresse des zu füllenden Speicherbereiches ein (quittieren nicht vergessen). Jetzt müssen Sie nur noch einen Wert eingegeben mit dem der Speicherbereich gefüllt wird. In der Anzeige steht jetzt **dtA 00**. Nach der Eingabe erscheint wieder die **-bEF-** -Meldung.

Noch ein Beispiel:

Eingabe	Anzeige	Bemerkung
2 FUL	V O N 0 0 0 0	Funktion anwählen
8	V O N 0 0 0 8	Anfangsadresse eingeben
9	V O N 0 0 8 9	
0	V O N 0 8 9 0	
0	V O N 8 9 0 0	
CR	b I S 0 0 0 0	Anfangsadresse bestätigen
8	b I S 0 0 0 8	Endadresse eintippen
9	b I S 0 0 8 9	
F	b I S 0 8 9 F	
F	b I S 8 9 F F	
CR	d t A 0 0	Endadresse bestätigen
A	d t A 0 A	Datenwert eingeben
B	d t A A b	
CR	- b E F -	Datenwert bestätigen

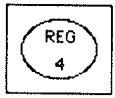
Jetzt steht im von Ihnen eingetippten Speicherbereich nur noch der Wert **AB**.

3 VGL



Abkürzung für Vergleich. Zwei Speicherbereiche werden miteinander verglichen. In der Anzeige erscheinen die selben Abfragen wie bei der **0 MVE**-Funktion. Sie müssen jetzt die zu vergleichenden Speicherbereiche eingeben. Treten irgendwelche Unterschiede beim Vergleich auf wird die Meldung **PRF XXXX** ausgegeben, wobei XXXX für die fehlerhafte Adresse steht. Ist der Vergleich in Ordnung erscheint die **-bEF-** ausgegeben.

4 REG



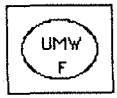
Abkürzung für Register. Nach dem Drücken auf diese Taste werden die Inhalte der Register des Computers angezeigt. Jetzt können die Registerinhalte angeschaut und verändert werden. In der Anzeige erscheint folgendes:
AF XXXX wobei die ersten beiden Zahlen den Akkuinhalt anzeigen und die rechten beiden den Inhalt des Flagregisters. Jetzt können Sie die die Inhalte der beiden Register verändern. Wenn Sie nur den Inhalt des Akkus verändern wollen, müssen Sie anschließend das Byte, das im

Flag stand, neu eingeben. Es können aber auch die anderen Register angeschaut bzw. verändert werden. Dazu müssen Sie die Tasten + oder CR drücken. Dann erscheinen nacheinander die anderen Register des Computers. Mit der - Taste gehen Sie in die andere Richtung. Es sind der Reihe nach:

AF XXXX
BC XXXX
DE XXXX
HL XXXX
SP XXXX
IX XXXX
IY XXXX
AF' XXXX
BC' XXXX
DE' XXXX
HL' XXXX
rI' XXXX

XXXX steht jeweils für den Inhalt des ausgewählten Registerpaares.

F UMW



Abkürzung für Umwandlung. Mit dieser Funktion können Sie dezimal vorgegebene Zahlen in hexadezimale Zahlen umrechnen. Das ist gerade beim Einstieg in die Programmierung recht hilfreich. Um Zahlensysteme ineinander umzuwandeln müssen Sie folgende Eingaben machen: Nach dem aktivieren der Funktion erscheint in der Anzeige 0000. Jetzt können Sie eine hexadezimale Zahl eingeben. Nach dem Druck auf die CR-Taste erscheint der dezimale Wert dieser Zahl. Nochmaliges Betätigen der CR-Taste bringt Sie zurück zur Eingabe einer neue Zahl. Wenn Sie aber die OPT-Taste drücken erscheint die Zahl auch in der dezimalen Darstellung. Nochmals OPT und es steht wieder die hexadezimale Darstellung in der Anzeige. Wenn nach der OPT Betätigung die CR-Taste gedrückt wird kann eine dezimale Zahl eingegeben werden. Sie können auch Dezimalzahlen mit Vorzeichen eingeben nur können Sie nach der Vorzeicheneingabe keine weiteren Ziffern Ihrer Zahl eintippen. Die Umwandlung erfolgt wie oben.

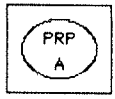
Auch hier noch ein Beispiel:

Eingabe	Anzeige	Bemerkung
F UMW	0 0 0 0	Funktion aufrufen
Eingabe einer beliebigen Zahl; wie vorher beschrieben		
CR	d X X X X X	Eingabe Bestätigung
CR	0 0 0 0	
neue Zahl eingeben		
OPT	d X X X X X	dezimal anzeigen
OPT	S X X X X	hexadezimal anzeigen
CR	0	umstellen auf dezimale Eingabe

Jetzt können Sie dezimale Zahlen eintippen und diese dann hexadezimale Zahlen umwandeln.

Die Funktionen die jetzt noch beschrieben werden, sind nur einzusetzen wenn Sie eine zusätzliche Steckerleiste in den zweiten Steckplatz einlöten. Dann könne Sie noch eine zusätzliche Baugruppe aufstecken. Das kann z.B.eine CAS-Baugruppe oder eine Prommer-Baugruppe sein.

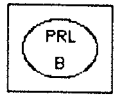
A PRP



Abkürzung für Eprom-Programmierung. Ein Eprom ist ein Bauteil, mit dem Sie Ihre Programme in den Festwertspeicher eines Rechners schreiben können. Mit dieser Taste ist es nun möglich Programme auf Eproms zu speichern. Ein Eprom (Erasable Programmable Read Only Memory) ist ein Baustein der frei programmierbar ist, aber nur noch mit einem speziellen UV-Löschgerät gelöscht werden kann. Diese Eproms können dann in einen Rechner eingesetzt werden und sind dann dort wie ein Festwertspeicher einsetzbar. Um ein EPROM zu programmieren, benötigen Sie die **Prommer**-Baugruppe. Näheres dazu finden Sie im **Prommer**-Handbuch. Es werden wie bei den **LAD** und **SPE** Funktionen die selben Eingaben verlangt. Ist der Programmiervorgang fehlerhaft erscheint auch hier die **PRF XXXX**-Meldung. Ist er aber richtig abgeschlossen erscheint die

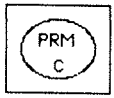
Meldung **burnEd** in der Anzeige, was soviel wie gebrannt bedeutet.

B PRL



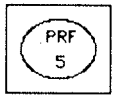
Abkürzung für Eprom-Lesen. Damit ist es möglich einen Eprom-Inhalt in einen Speicherbereich Ihres Systems zu übertragen. Das System verlangt nach dem Aufrufen der Funktion zunächst die Anfangsadresse des zu übertragenden Speicherbereiches, mit der Meldung **VON 0000**. Hier wird nomalerweise der Wert 0000 bestätigt. Jetzt verlangt das System die Größe des zu übertragenden Bereiches in dem es die Endadresse abfragt.(**bIS 07FF**) Wenn Sie den ganzen Speicherbereich des EPROMs übertragen wollen quittieren Sie mit der CR-Taste. Nun muß nur noch die Anfangsadresse im Speicher eingegeben werden (**NAC 8100**). Sind Sie mit der angezeigten Adresse 8100 einverstanden, drücken Sie die CR-Taste, ansonsten geben Sie einen anderen Wert ein. Tritt beim Laden ein Fehler auf z.B. kein freier Speicherbereich bei der angegebenen Adresse so erscheint die **PRF XXXX**-Meldung. XXXX ist die Adresse bei der der Fehler auftritt.

C PRM



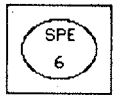
Abkürzung für Prommer. Diese Taste dient dazu, eine angeschlossene Prommerbaugruppe an Ihr System anzupassen. Ist eine Prommerbaugruppe angeschlossen und wird diese Taste gedückt, erscheint die Monoflop-Zeitdauer in Millisekunden auf der Anzeige. Diese müssen Sie auf **50 ms** einstellen. Die Einstellung erfolgt mit dem Trimmer, der auf der Prommer-Karte angebracht ist. Es ist nicht unbedingt nötig den Wert bis zur letzten Ziffer einzustellen.

5 PRF



Abkürzung für Prüfen. Daten werden vom Kasettenrekorder mit dem aktuellen Speicherinhalt verglichen. Hierfür benötigen Sie die **CAS**-Baugruppe, die Sie auf die HEXIO2 oder auf einen angeschlossenen Bus stecken müssen. Ist der Vergleich zwischen Speicherinhalt und Kassetteninhalt positiv wird die Meldung **St= XXXX** ausgegeben. Wenn irgendein Fehler auftritt wird erscheint **PRF XXXX** in der Anzeige. XXXX steht für die Adresse an der der Fehler auftrat. Eine weitere Fehlermöglichkeit ist ein Prüfsummenfehler der durch die Meldung **CHE** gemeldet wird. Ist der Vergleich fehlerlos, erscheint die **-bEF-**Meldung in der Anzeige.

6 SPE

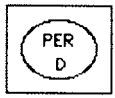


Abkürzung für Speichern. Die Programme des Arbeitsspeichers werden auf Kassette gespeichert. In der Anzeige erscheint zunächst eine **VON XXXX** Anfrage. Hier geben Sie die Anfangsadresse des zu übertragenden Speicherbereiches ein. Nach einem Druck auf die **CR**-Taste erscheint die Meldung **bIS XXXX** in der Anzeige. Hier geben Sie die Endadresse des zu übertragenden Bereiches ein. Bevor Sie die **CR**-Taste drücken müssen Sie aber den Kassettenrecorder einschalten. Der Übertrag wird mit einer Prüfsumme übertragen, so daß beim laden von Kassette eine zusätzliche Kontrolle gegeben ist.

7 LAD

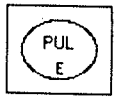
Abkürzung für laden. Mit dieser Funktion können Daten von einem Kassettenrekorder eingelesen werden. Die CAS-Baugruppe muß angeschlossen sein. Eine Kontrolle der übertragenen Daten wird genauso, wie bei der **6 SPE**-Funktion durchgeführt. Auch die Fehlermeldungen sind die selben. **PRF XXXX** für eine fehlerhafte Übertragung und **CHE** für einen Prüfsummenfehler. Bei einem richtigen Datentransfer erscheint in der Anzeige die Meldung **St= XXXX**, wobei XXXX die Startadresse des Programms ist. Ein weiterer Tastendruck und die **-bEF-** -Meldung erscheint. Jetzt kann normal weitergearbeitet werden. Genauere Angaben zur CAS-Baugruppe entnehmen Sie bitte dem entsprechenden Handbuch.

D PER



Abkürzung für Periodendauer. Mit dieser Funktion können Sie eine Periodendauer messen. Um die Funktion zu verwenden müssen Sie die CAS-Baugruppe einstecken und eine zusätzliche Leitung legen. Bit 7 des I/O-Ports 0 wird mit einer Leitung verbunden. Dies ist die Meßleitung. Diese Leitung wird mit Pin4 des 6850 auf der CAS-Baugruppe verbunden (Lötseite).Sind diese Veränderungen durchgeführt, erscheint die Periodendauer in ys auf der Anzeige. Jetzt können Sie mit dem Trl (Trimmer1) die Periodendauer auf ca. 833ys abgleichen. Siehe auch CAS-Handbuch.

E PUL



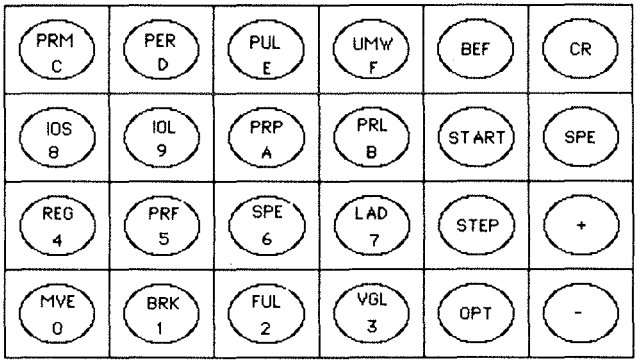
Abkürzung für Pulsdauer. Diese Taste hat eine ähnliche Funktion wie die vorher beschriebene Taste. Der Unterschied zur vorherigen Funktion besteht darin, daß hier die 0-Dauer des Signals gemessen wird. Dazu wird das vorher angebrachte Meßkabel mit dem Pin3 des 6850 auf der CAS-Baugruppe verbunden. Mit dem Trimmer 2 (Tr2) kann dann der Wert auf ca. 625ys eigestellt werden. Dazu muß die CAS-Baugruppe auf Aufnahme gesteckt werden und der Teststecker eigesteckt werden. Siehe auch CAS-Handbuch.

Jetzt nochmal in Kurzform

Taste	Bedeutung und Funktion
BEF	Befehlmodus einstellen
CR	Carriage Return, Bestätigung aller Eingaben
START	starten eines Programms
SPE	Speicherbereich anschauen, bzw. verändern
STEP	Einzelschrittfunktion, Programm kann in Einzelschritten ablaufen gelassen werden
+	nächste Stelle aufrufen, z.B. Speicherstellen oder Registerinhalte
OPT	arbeitet nur im Zusammenhang mit anderen Funktionen, dient dem Umschalten von Zahlenebenen
-	gleiche Funktion wie + , nur jeweils die vorherige Speicherstelle wird aufgerufen
0 MVE	Speicherbereiche verschieben, ein Speicherbereich wird in einen anderen verschoben
1 BRK	Breakpunkte setzen, Unterbrechungen für Programme können gesetzt werden, um z.B. Registerinhalte anzuschauen
2 FUL	Speicherbereiche mit einem frei wählbaren Wert füllen
3 VGL	zwei Speicherbereiche werden miteinander verglichen
4 REG	Registerinhalte der einzelnen Register anschauen mit + , - oder CR kann man alle Register nacheinander anschauen
5 PRF	prüft die Daten die auf Kassette geschrieben wurden, indem er sie mit dem entsprechenden Speicherbereich vergleicht, CAS-Baugruppe muß aufgesteckt werden
6 SPE	speichert Daten auf Kassette, bildet gleichzeitig eine Prüfsumme damit beim späteren Lesen eine Kontrolle der übertragenen Daten möglich ist CAS-Baugruppe muß aufgesteckt werden
7 LAD	Programme von Kassette laden, abgespeicherte Prüfsumme wird mit eingelesener verglichen CAS-Baugruppe muß aufgesteckt werden
8 IOS	Ausgabeport festlegen und Bytes auf diesen ausgeben

Taste	Bedeutung
9 IOL	Eingabeport festlegen und dort eingestellte Bytes auf der Anzeige ausgeben, mit OPT eine Ausgabe in binärer und hexadezimaler Form möglich
A PRP	Eproms programmieren, ein Speicherbereich kann auf ein Eprom programmiert werden, PROMMER-Baugruppe muß verwendet werden
B PRL	übertragen eines Eprominhalts in einen Speicherbereich, PROMMER-Baugruppe muß verwendet werden
C PRM	Funktion zum abgleichen der PROMMER-Baugruppe, auf der Anzeige erscheint ein Zahlenwert in ms, für PROMMER muß dieser Wert auf 50ms abgeglichen werden
D PER	Periodendauer einer anliegenden Frequenz wird gemessen, Meßleitung muß angeschlossen werden, dann wird kann die Periodendauer in ys gemessen werden, für die CAS-Baugruppe muß sie auf 833ys eingestellt werden
E PUL	Funktion mißt die Nulldauer eines Signals, für CAS-Baugruppe muß sie 624ys betragen
F UMW	hexadezimal in dezimal umwandeln und umgekehrt, mit OPT und CR kann umgeschaltet werden

TASTATUR HEXIO2



Stichwortverzeichnis

Byte:

Ein Byte ist die gebräuchliche Bezeichnung für eine Dateneinheit. Zusammengesetzt wird ein Byte, aus acht Bits. Ein Bit ist ein Wert der entweder 0, oder 1 sein kann.

Listing:

Die Liste der, in einem Programm verwendeten Befehle.

RAM:

Random-Access-Memory ist der Arbeitsspeicher Ihres Systems. Im RAM könne sie schreiben, d.h. selbst Daten und Programme eingeben und lesen. Er beginnt beim Einsteigerpaket auf 8100. Dies ist vom Grundprogramm her so festgelegt worden. Deshalb erscheint nach der Betätigung einer Funktionstaste immer die Anfangsadresse 8100.

ROM:

Read-Only-Memory ist der Festwertspeicher Ihres Systems. Er besteht aus dem Monitorprogramm und den anderen Programmen die in Ihrem System fest installiert sind. Das Monitorprogramm sorgt dafür, daß eine Verbindung "Mensch-Maschine" möglich ist.

Telefonservice 08 31- 62 11 jeden Mittwochabend bis 20.00 Uhr

---

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang
Telefon: 08 31 / 6 93 00

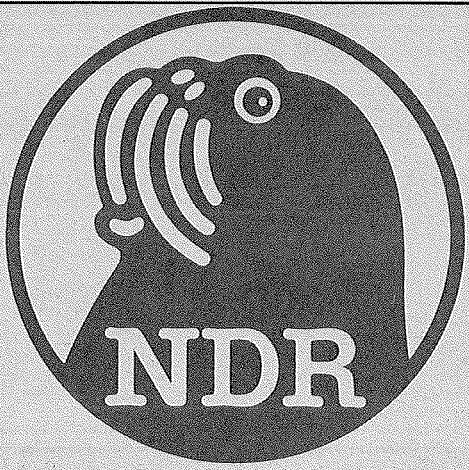
Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr

ges



HEXIO 2

Die hexadezimale Ein- und Ausgabebaugruppe

für den NDR-Computer

Graf Elektronik Systeme GmbH

INHALTSVERZEICHNIS Seite

1. Einführung

1.1 Zum NDR-Computer

Der NDR-Computer wird in der Fernsehserie "Computer Modular - Schritt für Schritt" aufgebaut, erklärt und in Betrieb genommen. Diese Serie wird vom Norddeutschen Rundfunk und vom Bayerischen Fernsehen ausgestrahlt. Es werden bald auch die Regionalsender anderer Bundesländer die Sendung in ihr Programm aufnehmen.

Zur Serie gibt es einige Begleitmaterialien, es ist daher nicht unbedingt notwendig, die Fernsehserie gesehen zu haben, um den NDR-Computer zu bauen und zu begreifen:

- Bücher:
  Rolf-Dieter Klein,
  "Mikrocomputer selbstgebaut und programmiert"
  2., neu bearbeitete und erweiterte Auflage
  ISBN 3-7723-7162-0,
  erschienen im Franzis-Verlag, München
  Bestellnummer: 10078
  Auf diesem Buch baut die NDR-Serie auf

- Sonderhefte der "mc"
  "Mikrocomputer Schritt für Schritt"
  Bestellnummer: 10399
  "Mikrocomputer Schritt für Schritt Teil 2"
  Bestellnummer: 10398

- Zeitschriften "mc" und "ELO" des Franzis-Verlages

- Zeitschrift "LOOP" der Firma Graf (siehe Kapitel 11.1)

- Videocassetten:
  lizensierte Originalcassetten für den privaten Gebrauch. Auf diesen zwei Cassetten sind die 26 Folgen der Fernsehserie enthalten.
  Systeme: VHS, Beta, Video 2000
  Bestellnummer: 10439 (VHS, zwei Kassetten)
  10436 (Beta, zwei Kassetten)
  10438 (Video 2000, eine Kassette)
  Preise siehe gültige Preisliste

1.2 Wozu dient die Baugruppe HEXIO2

Die HEXIO2 dient als einfache Ein- und Ausgabeeinheit für die NDR-Single-Board-Computer SBC2 oder SBC3. Die Eingaben erfolgen über eine eingebaute Tastatur (bestehend aus 24 Tasten) oder einem DIL-Schalter.

Die Ausgabe der Informationen erfolgt entweder über eine 8-stellige Siebensegmentanzeige oder über die 8, auf der HEXIO2 aufgebrachten Leuchtdioden.

Sie dient als Einstiegsmöglichkeit in die Maschinenspracheprogrammierung und als Regel- bzw. Steuereinheit für Maschinen, Roboter oder sonstige elektrische Anlagen.

1.3 Wie setzt man die HEXIO2 ein

Die HEXIO2 wird in Verbindung mit den Single-Board-Computern SBC3 oder SBC2 eingesetzt. Auf diesen Baugruppen sind die CPU (Zentraleinheit) und die Speicher angebracht.

Der Unterschied zwischen den beiden angebotenen Paketversionen besteht in den SBC-Karten.

Das erste Einsteigerpaket ist mit der SBC3 ausgestattet. Mit dieser SBC sind folgende Ausbaustufen möglich:

- späterer Ausbau zum Z80 bzw. CP/M 2.2 mit 10MByte Festplatte
- RAM-Kapazität bis 16kByte
- Akku-Pufferung der RAM-Speicher
- integrierte Bank-Umschalte-Logik
- dadurch externe Speichererweiterung bis 1MByte
- BOOT-Logik zum Laden des Betriebssystems CP/M 2.2

Das zweite Einsteigerpakets ist ist mit der SBC2 ausgestattet. Mit dieser SBC sind folgende Ausbaustufen möglich:

- RAM-Kapazität bis 4kByte
- Monitor und Tatatur anschlußfähig
- mit Zero-Power-RAMs; Speicherung der Programme wie bei Akku-Pufferung
- keine externe Speichererweiterung
- kein Ausbau zum CP/M 2.2 Rechner möglich

2. Technische Daten

Format der Baugruppe:	180 x 200 mm
Spannungsversorgung:	+ 5,0 V
Stromverbrauchmit SBC3:	ca. 0,7 A
Stromverbrauch mit SBC2:	ca. 0,4 A
Eingabe:	24 Tasten, DIL-Schalter
Ausgabe:	8 Siebensegment-Anzeigen, 8 LED
Bus:	2 Steckplätze NDR-Bus

3. Baugruppenbeschreibung

3.1 Das Steckernetzteil

Das Industrieeinsteigerpaket wird in der Regel mit einem Steckernetzteil betrieben. Dieses Netzteil genügt zum Betrieb der Baugruppe.

Bei Systemaufbauten mit mehr Systemen als mit der SBC3 bzw. SBC2 betrieben wird ist es empfehlenswert, zum einwandfreien Funktionieren des Systems das Netzgerät neu abzugleichen. Wie Sie hierbei vorgehen müssen ist im nachfolgenden Abschnitt beschrieben.

Zunächst trennen Sie das Netzteil vom 220V-Netz. Lösen Sie mit einem großen Schraubendreher die drei Schrauben mit denen das Steckernetzteil zusammengehalten wird. Jetzt können Sie das Vorderteil und das Hinterteil des Gehäuses abziehen. Nehmen Sie jetzt das Netzteil so in die Hand, daß die von hinten gesehen rechte Seite vor Ihnen liegt. Dort sehen Sie in der linken unteren Ecke ein Potentiometer (Siehe auch Bild1). Dort können Sie den Abgleich vornehmen.

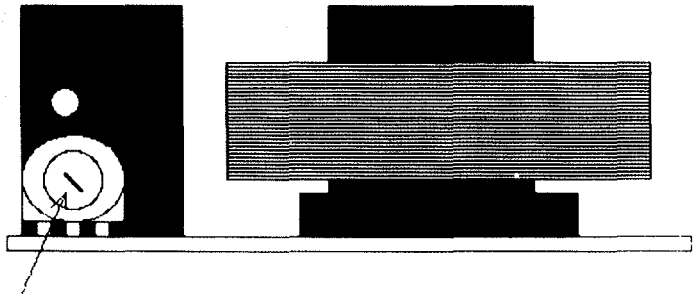


Bild1: Steckernetzteil Seitenansicht von rechts

Nehmen Sie einen kleinen Schraubendreher zur Hand und verstellen Sie vorsichtig die Einstellung des Potis. Eine Drehung nach links senkt die, am Ausgang anliegende Spannung ab. Eine Drehung nach rechts erhöht die Spannung. Schließen Sie anschließend das Gehäuse wieder und stecken Sie es wieder ein.

Unbedingt schließen da sonst Gefahr eines Stromschlages besteht!

Messen Sie die Spannung die nun an Ihrem Sysztem anliegt. Sie sollte zwischen 4,85V und 5,15V liegen. Wiederholen Sie den Abgleich solange bis sich die Spannung im angegebenen Zwischenraum befindet.

3.2 Die Ziffernanzeige

Die Ausgabe der Informationen der Ein- und Ausgabegeräte können mit der Ziffernanzeige sichtbar gemacht werden. Die Anzeige besteht aus acht einzelnen Siebensegmentanzeigen die über die Portadresse 01h angesprochen werden. Wird ein Ausgabebefehl auf diese Portadresse gegeben, werden durch J8 die einzelnen Anzeigenelemente ausgewählt. Siehe auch Kapitel 7.1.3

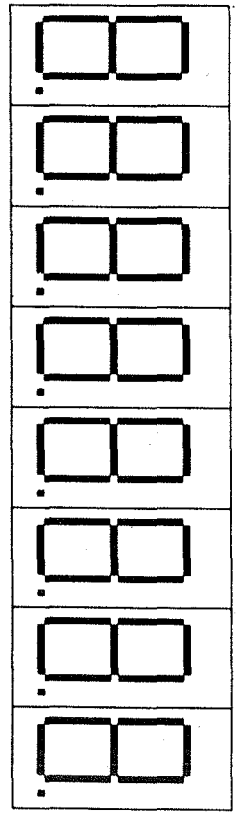


Bild2: Anzeigenelemente (HD1131)mit Anordnung auf der HEXIO2

3.3 Die HEXIO2-Baugruppe

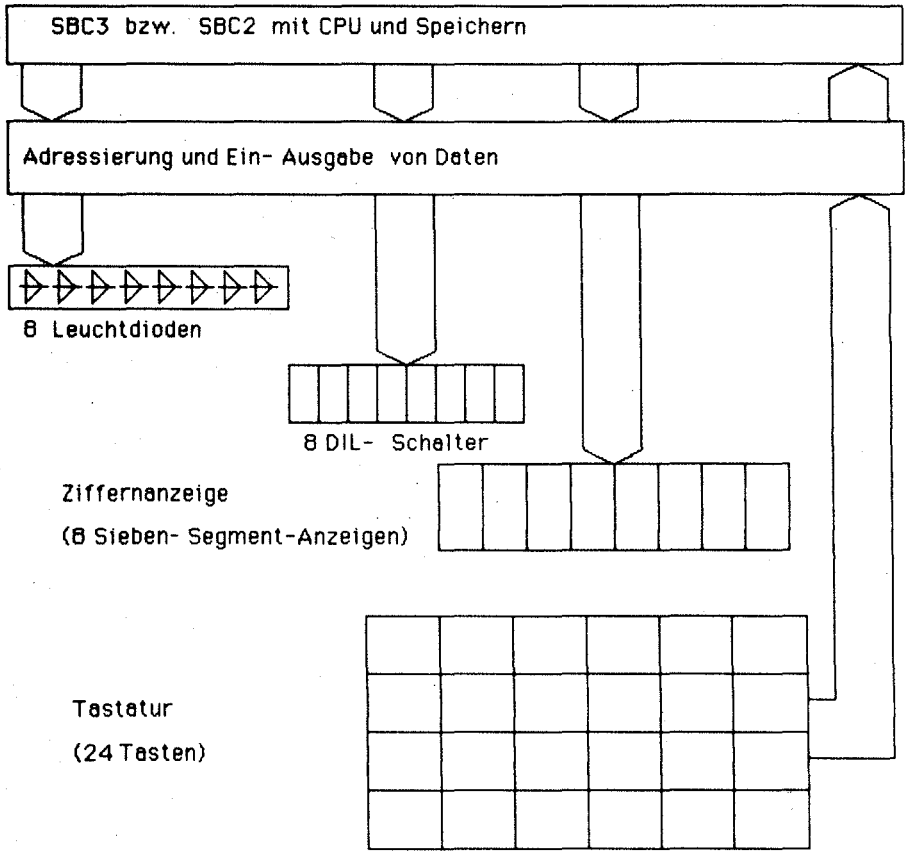


Bild3: Blockschaltbild HEXIO2

Die Eingaben erfolgen auf der HEXIO2 mittels einer einfachen Tastatur (In-Port 00h), die aus 24 Tasten besteht, oder mittels 8-DIL-Schalter (In-Port 02h).

Die Ausgabe erfolgt auf einer 8-stelligen Siebensegmentanzeige (Out-Port 01h) oder auf acht Leuchdioden (Out-Port 02h), die getrennt angesteuert werden können.

Die anderen Bauteile dienen zur Adressierung der einzelnen Portadressen und zur Ein-, Ausgabe der Befehle und Daten.

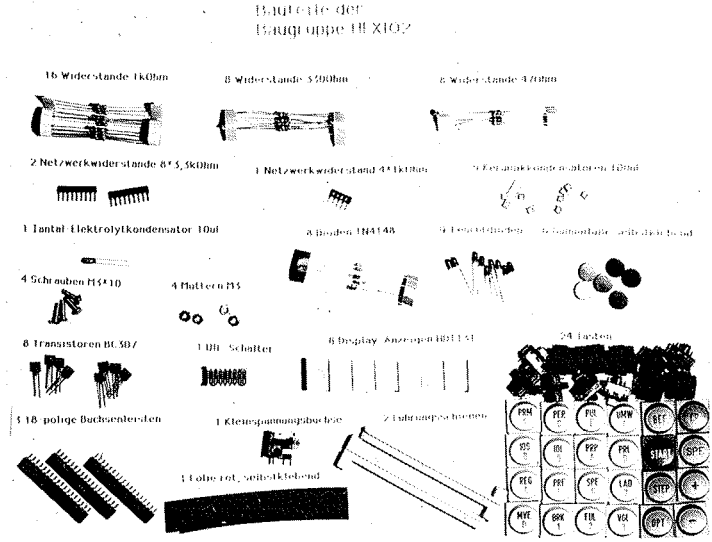
4. Aufbauanleitung

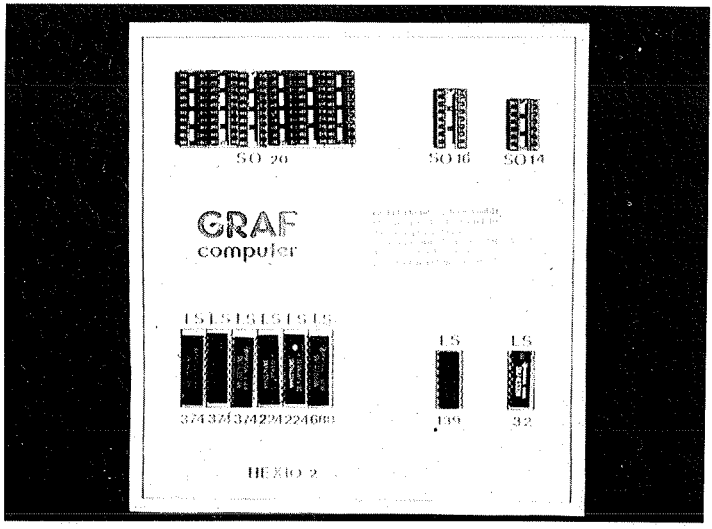
4.1 CMOS-Warnung

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) bevor Sie einen Baustein berühren.

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





4.2 Stückliste

1	Orginal	GES-Leiterplatte mit Lötstoplack und Bestückungsdruck	
1	Handbuch	HEXIO2	
1	74LS32	J1	4 OR mit je zwei Eingängen
1	74LS139	J2	Zwei 2-Bit Binärdekoder
2	74LS244	J4,J5	Acht Bus-Leistungstreiber
3	74LS374	J6-J8	8-Bit D-Register mit Tri-State Ausgängen
1	74LS688	J3	8-Bit Größenvergleicher
16	1k	R18-R33	Widerstand 1kOhm
8	330	R2-R9	Widerstand 330Ohm
8	47	R10-R17	Widerstand 47 Ohm
2	8*3,3k	RN1,RN2	Netzwerkwirsand 8*3,3 kOhm
1	4*1k	RN3	Netzwerkwiderstand 4*1 kOhm
8	1N4148	D7,D8,D15,D16,D23,D24,D31,D32	Siliziumdiode 1N4148
8	BC307	T2-T9	Transistor BC307
9	LED	LD1-LD8	Leuchtdioden
8	HD1131		Display-Anzeigen (7-Segment Anzeigen)
1			Folie rot,selbstklebend
24		TA1-TA24	Tasten mit Tastkappe
1	DIL		DIL-Schalter 8-fach
6	SO20		20-polige IC-Fassung
1	SO16		16-polige IC-Fassung
1	SO14		14-polige IC-Fassung
3			18-polige Buchsenleiste
9	100nF	C2,C3,C5,C7-C12	Keramikkondensatoren 100 nF
1	10 uF	C13	Tantal-Elko 10 uF
2			Führungsschiene
4			Schraube M3*10
4			Mutter M3
1			Kleinspannungsbuchse
6			Gummifüße selbstklebend

4.3.1 Aufbau Schritt für Schritt

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. Beim Einlöten der Bauelemente beginnt man am besten mit den drei grünen 18-poligen Steckerleisten. Diese werden in die zweite Anschlußreihe mit der Bezeichnung ST2 eingesetzt. Dabei sollten zuerst die beiden äußeren Stifte und einer in der Mitte verlötet werden. Dann empfiehlt es sich nachzuschauen, ob die Stecker parallel zur Leiterplatte liegen und ob keine "Bäuche" zwischen den verlöteten Stiften liegen. Sollten Bäuche vorhanden sein, muß wiederum in der Mitte der Bäuche ein Stift unter Druck angelötet werden. Liegt die Steckerleiste dann richtig, können die restlichen Stifte verlötet werden.

Es ist empfehlenswert nach jedem Lötvorgang die überstehenden Drahtstücke mit einem Seitenschneider abzuzwicken.

Die Keramikkondensatoren C2 - C12 sind ungepolt und können ohne auf die Polung zu achten eingelötet 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.

Die Kohleschichtwiderstände R1 - R33 werden alle liegend eingelötet. Es handelt sich um folgende Widerstandsgrößen:

R1 - R9	330 Ohm	orange-orange-braun
R10 - R17	47 Ohm	gelb-violett-schwarz
R18 - R32	1 kOhm	braun-schwarz-rot

Die Netzwerkwiderstände RN1, RN2 und RN3 haben einen Zifferncode. Die drei angegebenen Ziffern nach der Firmen- und Typenbezeichnung sind gleich codiert wie die Farbcodes, nur da^ statt der Farben gleich die Ziffern draufstehen. Die ersten beiden Ziffern sind der Zahlenwert und die dritte Ziffer die Zehnerpotenz, also die Anzahl der Nullen. Z. B. 330 Ohm = 331. Diese Widerstände haben einen gemeinsamen Anschluß der mit einem Punkt auf dem Netzwerkwiderstand vor der Firmenbezeichnung gekennzeichnet ist. Auf dem Bestückungsdruck bzw. Bestückungsplan ist der gemeinsame Anschluß auch mit einem Punkt gekennzeichnet

Jetzt werden die Dioden D1 - D32 eingelötet. Da Dioden nur in einer Richtung Ströme durchlassen muß beim Einbau auf die Polung der Dioden geachtet werden. Die Dioden sind dann richtig eingesteckt, wenn der schwarze Ring auf der Diode und der Querstrich des Aufdrucks auf der Bestückungsseite (siehe auch Anhang D) übereinstimmen.
Die Ihnen ausgelieferte Version der Tasten beinhaltet bereits Dioden, so daß Sie die folgenden Dioden nicht einlöten müssen:

D1 - D6
D9 - D14
D17 - D22
D25 - D30

Für das Einlöten der Leuchtdioden LD1 - LD9 gilt das gleiche wie für die Dioden. Welcher Anschluß der Leuchtdioden die Annode und welcher die Kathode ist entnehmen Sie bitte folgendem Bild (längeres Bein entspricht dem Anschluß -):

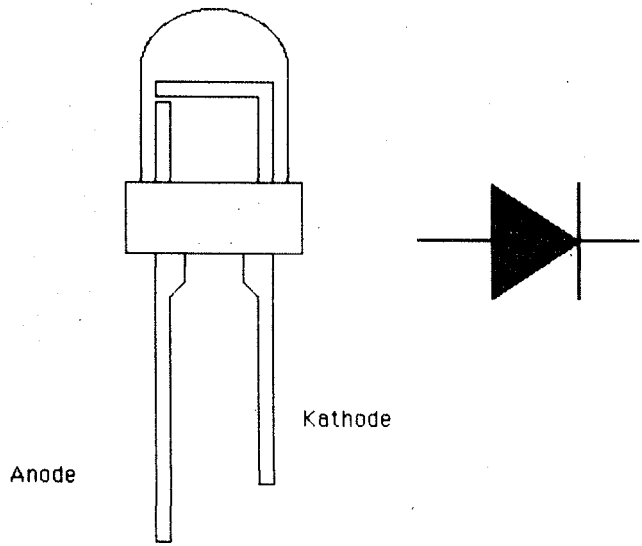


Bild4: Leuchtdiode Anschlüsse und Anschlußbelegung auf der Leiterplatte

Die einzelnen Elemente der Ziffernanzeige werden jetzt eingelötet. Setzen Sie die Siebensegmentanzeigen so auf der Leiterplatte ein, daß der Dezimalpunkt auf jedem Anzeigenelement rechts unten liegt! Halten Sie die einzelnen Elemente solange fest bis einige Pins verlötet sind. Damit wird gewährleistet, daß die Elemente sauber auf der Karte aufliegen. Jetzt wird nur noch die rote Folie auf die Anzeige geklebt, damit die Anzeige besser lesbar ist.

Nun können die Tasten der Tastatur eingesteckt und verlötet werden. Hierzu werden zunächst die Tasten auf die HEXIO2-Karte gesteckt. Dies muß so geschehen, daß die beiden Kontakte in die dafür gebohrten Löcher passen. Beginnen Sie mit dem Einlöten am besten bei Taste 1 TA1 und löten Sie nacheinander alle in einer Zeile liegenden Tasten ein. Auch hier sollten Sie darauf achten, daß die einzelnen Tasten eben auf der Leiterplatte aufliegen. Anschließend können die Tastkappen, wie im Bild5 dargestellt aufgesteckt werden.

TASTATUR HEXIO2

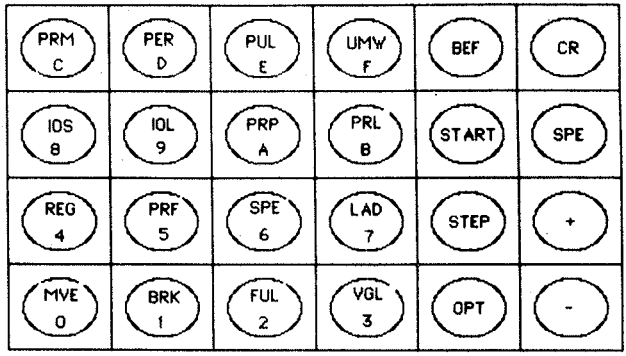


Bild5: Tastatur der HEXIO2

Jetzt werden die Transistoren T2 - T9 eingelötet. Die Lage der Transistoren muß mit den, auf der Bestückungsseite aufgedruckten Anschlüssen übereinstimmen. Siehe Bild6.

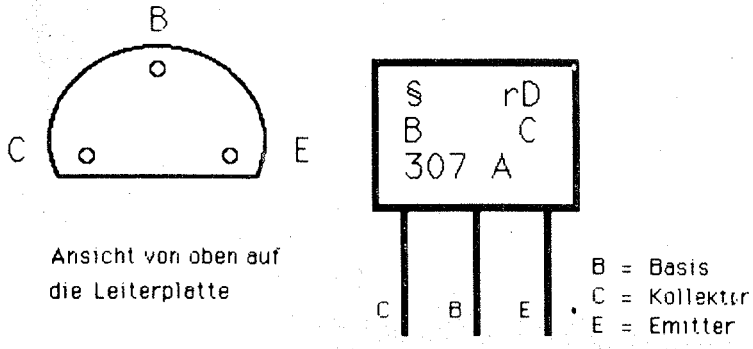


Bild6: Transistor BC 307A

Der DIL-Schalter wird so eingelötet, daß die Zahlen auf den einzelnen Schaltern von der Tastatur aus gesehen auf dem Kopf stehen.

Die Kondensator C13 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.

Es müssen noch die Führungsschienen für die SBC3- bzw. SBC2-Karten mit jeweils zwei Schrauben M3*10 anzuschrauben.

4.3.2 Netzteile

- Steckernetzteil

Das mitgelieferte Steckernetzteil reicht zum Betrieb Ihres Systems, bei einer zusätzlichen Belastung von bis zu 0,5 A aus. Die am Netzteil eingestellte Spannung ist aber auf den "normalen" Betrieb eingestellt. Wie diese Spannung verändert werden kann, entnehmen Sie bitte Kapitel 3.1.

- Trafo2 und interne Spannungsversorgung

Auf Wunsch ist auch eine Spannungversorgung erhältlich die direkt auf der HEXIO2 angebracht ist. Wenn Sie diese Spannungsversorgung bevorzugen entfällt die Kleinspannungsbuchse.

Um diese Spannungversorgung aufzubauen gehen Sie wie folgt vor.

Löten Sie zunächst die beiden Kondensatoren C4 und C6 ein.

Anschließend wird der Elektrolytkondensator C1 eingelötet. Bei diesem Kondensator ist wie bei dem Tantalkondensator C13 auf die Polung zu achten.

Nun wird der Gleichrichter B80C eingelötet. Hierbei muß darauf geachtet werden, daß die vier Anschlüsse an den richtigen Lötstellen angelötet werden. Siehe auch Bild7:

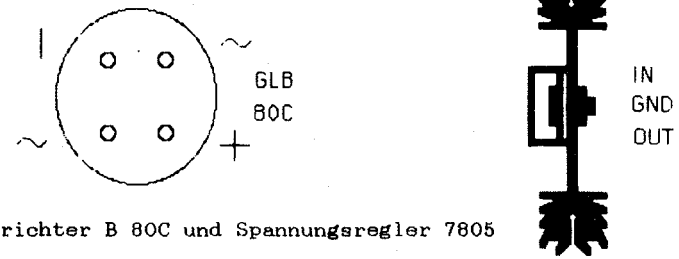
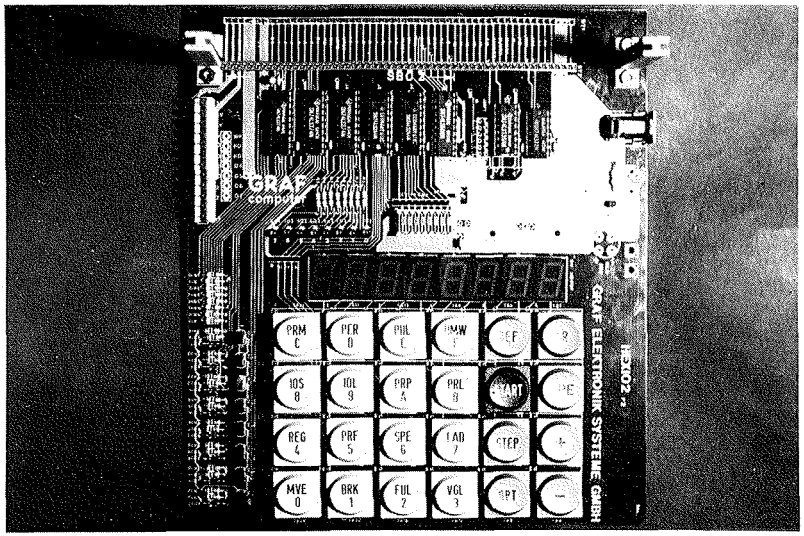


Bild7: Gleichrichter B 80C und Spannungsregler 7805

Jetzt muß noch der Spannungsregler 7805 eingelötet werden. Die Rückseite des Spannungsreglers muß in Richtung des Kühlkörpers zeigen, der anchließend noch eingesetzt und verlötet wird.



5. Testanleitung

5.1 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 mit dem mitgelieferten Netzgerät an das 220V-Netz angeschlossen werden. Dies geschieht durch einstecken des Steckernetzteils in eine Steckdose. Nun verbinden Sie den Kleinspannungsstecker mit der Kleinspannungsbuchse auf der HEXIO2. Diese Buchse befindet sich am hinteren, rechten Rand der HEXIO2.

Man mißt, ob an allen IC-Sockeln die Versorgungsspannung von +5V ankommt. Dabei liegt bei Standard-TTL-Bausteinen jeweils am letzten 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).

Liegt die Versorgungsspannung +5V und 0V (Masse) an den richtigen Pins an, dann können die ICs eingesetzt werden, achten Sie aber darauf,daß die Versorgungsspannung unterbrochen ist. Dabei muß auf die Richtung der ICs geachtet werden. Die Markierung auf dem IC muß mit der Kerbe in der IC-Fassung übereinstimmen.

5.2 Test der HEXIO2 mit anderen Baugruppen

Ist der vorhergehende Test erfolgreich abgeschlossen können Sie die ICs einstecken und die Baugruppe in Verbindung mit einem Single-Board-Computer entweder SBC3 oder SBC2 testen. Hierzu werden die ICs eingesteckt und die SBC3 bzw SBC2 auf die Stiftleiste aufgestek-kt. Der Single-Board-Computer muß so in die Steckerleiste gesteckt werden, daß keiner der Stifte geknickt wird, oder übersteht. Die Bauelemente auf den SBCs müssen in Richtung der Tastatur zeigen.

Bei Verwendung der SBC3 werden alle Pins belegt und Sie wird in beiden Führungsschienen geführt.

Für das Einsetzen der SBC2 ist auf der HEXIO2 eine Markierung angebracht. Die SBC2 muß so aufgesteckt werden, daß sie links in der Führungsschiene geführt wird. Auf der rechten Seite muß die SBC2 bündig mit der Markierung abschließen.

Wird nun das System mit Spannung versorgt meldet es sich mit der "HALLO-1.1"-Meldung. Wenn diese nicht erscheint drücken sie zunächst den RESET-Taster. Nach dem Loslassen muß die Meldung erscheinen. Ist dies nicht der Fall trennen Sie die HEXIO2 vom Netz und untersuchen Sie sie auf eventuelle Fehler. Siehe auch Kapitel 6.

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 verwendeten Bauteile in Ordnung?

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, manch mal 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 (zwischen +5V und Masse müssen genau 5V anliegen) Toleranzen von +- 5% also von 4,75V bis 5,25V sind erlaubt. Falls die Spannung zu gering ist.

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. Schaltungsbeschreibung

7.1 Prinzipbeschreibung

Vereinbarung: Die in den Abbildungen verwendeten Signalbezeichnungen werden wie üblich mit einem Querstrich über der Bezeichnung gekennzeichnet. Dieser Querstrich bedeutet, daß das Signal "Low"-aktiv ist, also seine funktion erfüllt, wenn die Leitung Null-Pegel hat. Im Text ist die Darstellung mit dem Querstrich über dem Signalnamen leider nicht möglich; die "Low-aktivität" wird mit einem vorangestellten "-" kenntlich gemacht, also z.B. -RD und -WR.

7.2 Funktionsbeschreibung der Schaltung

7.2.1 Stromversorgung

Die HEXIO2 benötigt zum Betrieb 5V-Gleichspannung die mit dem 5V-Steckernetzteil erzeugt wird.

Es besteht auch die Möglichkeit das Industrieeinsteigerpaket mit einem Trafo und einer internen (d.h. auf der HEXIO2 angebrachten Spannungversorgung) auszurüsten.

In diesem Fall wird die Kleinspannungsbuchse nicht verwendet. Statt dessen muß der Trafo2 angeschlossen werden. Dieser Trafo erzeugt eine 9V-Wechselspannung die von einer Schaltung aus zwei Kondensatoren, einem Elko, einem Gleichrichter und einem Spannungsregler besteht (siehe auch Kapitel 4.3.2).

7.2.2 Auswahlschaltung für Ein- und Asugabeeinheiten

Die nachfolgend beschriebene Schaltung dient der Auswahl der einzelnen Portadressen. Durch diese Auswahl wird entweder eine Eingabe- oder eine Ausgabe-Baugruppe ausgewählt.

Alle Kontakte des Jumpers (JMP 1) müssen überbrückt sein, um die Q-Eingänge von J3 auf 0 zu setzen. Sind die Kontakte geschlossen, liegen an J3/5/7/9/12/14/16 sämtliche Q-Eingänge auf GROUND. Sind die Kontakte nicht geschlossen, sind die Q-Eingänge durch den Netzwerkwiderstand (RN 1) auf HIGH gelegt. Wenn nun vom Z80 eine Adresse aufgerufen wird, legt er die P-Anschlüsse von J3/4/6/8/11/13/15 auch auf 0. Ist dies geschehen, wird noch das IORQ-Signal an J3/1 angelegt. Sind die P=Q-Eingänge auf 0 gelegt gibt der Größenvergleicher J3 (74LS688) auf 19 ein 0-Signal am Ausgang P=Q an J1 weiter.

Die 0-Signale von J3 liegen an J1/1/11 an. Die Eingänge von J1/11/12/13 sind für die -RD-Anweisungen zuständig. J1/1/2/3 für die WR-Anweisungen. Wird z.B. vom Z80 eine WR-Anweisung ausgegeben liegt an J1/2 eine 0. Von J3 wird jetzt auch an J1/3 ein 0-Signal angelegt. An den anderen Anschlüssen von J1 (74LS32) werden automatisch 1-Signale ausgegeben.

Die -WR- bzw. -RD-Signale von J1 liegen an J2/1/15 (CS-Eingänge) an und aktivieren die Dekoder. Außerdem werden an J2/2/3/14/15 A0 und A1 vom Z80 angelegt. Die angelegten Bitkombinationen A0, A1 können die Werte 00B, 01B, 10B und 11B annehmen. Die Adressierungen der einzelnen Ports sehen wie folgt aus:

IN-Ports:

```
00B       RD 0      J5 (74LS244)       ;Tastatur lesen
10B       RD 2      J4 (74LS244)       ;DIL-Schalter lesen
```

OUT-Ports:

```
00B       WR 0      J7 (74LS374)       ;Tastatur ausgeben
01B       WR 1      J8 (74LS374)       ;Anzeige ausgeben
10B       WR 2      J6 (74LS374)       ;Leuchtdioden ausgeben
```

Diese Signale dienen dazu die von der CPU angesprochenen Ports auszuwählen. Die Auswahl durch J2 (74LS139) erfolgt nach folgender Wahrheitstabelle:

Inputs			Outputs			
CS	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

Tabelle1: Wahrheitstabelle 74LS139

Sind z.B. die Anschlüsse A und B auf 0 gelegt; d.h. wird von der CPU der Port 00H eingelesen, so wird von J1 ein 0-Signal an den CS-Eingang von J2 (J2/15) gelegt. Durch diese Signale wird der Ausgang RD 0 (J2/12) aktiviert (0-Signal).

c

7.2.3 Ein- und Ausgabeschaltungen

Mit diesen Schaltungen werden die Tastatur und der DIL-Schalter abgefragt. Außerdem werden bei entsprechenden Befehlen der CPU die Leuchtdioden bzw. die Ziffernanzeige aktiviert.

Der -RD 2-Befehl der von J2 zu J4 (74LS244) geht, legt dort die 1G- und 2G-Anschlüsse (J4/1/19) auf 0. Wenn einer der DIL-Schalter geschlossen wird, werden die entsprechenden A-Anschlüsse von J4/2/4/6/8/11/13/15/17 auf Ground gelegt. Bei geöffnetem DIL-Schal-

ter werden die A-Anschlüsse mit Hilfe des Netzwerkwiderstandes RN2 auf HIGH (1-Signal) gezogen. Die DIL-Schalter werden über den IN-Port 02H angesprochen. Die dadurch auf der A-Anschlußseite anliegende Bitkombination wird dann von J4 auf den Datenbus übertragen. Die so eingelesenen Bitmuster werden dann vom Datenbus an die CPU weitergegeben.

Auf die gleiche Weise arbeitet auch J5 (74LS244), nur daß bei J5 nur vier Bits beschaltet sind (J5/2/4/6/8). Die restlichen freien Bits J5/11/13/15/17 werden gesondert herausgeführt (D4', D5', D6' und D7').

Das Signal -RD 0 wird von J2 an J5/1/15 weitergegeben. J5 fragt die Tastatur ab. Je nachdem in welcher Zeile eine Taste gedrückt wird, liegt eine entsprechende Bitkombination an J5/2/4/6/8 an. Wird eine Taste in einer bestimmten Zeile gedrückt, dann liegt ein LOW-Signal an dem entsprechenden Pin an.
Die ICs J4 und J5 arbeiten nach folgender Wahrheitstabelle:

Inputs		Outputs
-G	A	Y
H	X	Z
L	L	L
L	H	H

Tabelle2: Wahrheitstabelle 74LS244

Fast gleichzeitig wie J5 die Tatatur abfragt wird J7 (74LS374) mit dem OUT 0-Befehl angesprochen. Wenn jetzt über den Datenbus eine Bitkombination eingelesen wird, fragt J7 die Tastatur ab und gibt die entsprechenden Bytes an die Ziffernanzeige aus. Diese Ausgabe erfolgt über eine Schaltung aus zwei Widerständen und einem PNP-Transistor pro Siebensegmentanzeige. Diese Schaltung wählt eine der Siebensegmentanzeigen aus und versorgt diese mit Strom.

Mit dem Befehl OUT 01H wird J8 (74LS374) angesprochen. J8 gibt dann die durch den Datenbus angelegten Bytes an die Ziffernanzeige aus. Zwischen J8 und die Siebensegmentanzeige sind noch pro Leitung je ein Widerstand geschaltet. Wie die einzelnen Segmente der Siebensegmentanzeigen angesteuert werden ist aus folgender Zeichnung zu entnehmen.

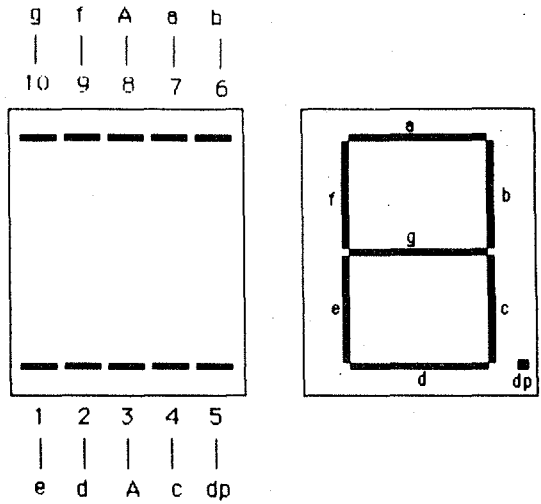


Bild8: Anzeigenelement mit Pinbelegung

Wird z.B. das Byte 00H vom Datenbus auf J8 eingelesen steuert J8 alle acht Siebensegmentanzeigen an. Auf jeder der acht Anzeigen wird dann iene 1 angezeigt.

Die Funktion des ICs J6 ist ähnlich. Liegt ein OUT 02H an J6 (74LS374) und ist auf dem Datenbus ein Byte, wird es von den Leuchtdioden angezeigt. Über die, nach den Leuchtdioden eingebauten Widerstände wird der Strom durch die LEDs und das Latch begrenzt. Werden die Leuchtdioden mit einem 0-Signal angesprochen, fangen die angesprochen LEDs zu leuchten an. Dies ist deßhalb so realisiert, weil die ICs im LOW-Bereich wesentlich größere Ströme verarbeiten können, als im HIGH-Bereich.

Die ICs J6-J8 arbeiten nach folgender Wahrheitstabelle:

Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	QO
H	X	X	Hi Z

Tabelle3: Wahrheitstabelle 74LS374

8. Anwendungsbeispiele

8.1 Mit der SBC3- bzw. SBC2-Karte

Die auf den nächsten Seiten beschriebenen Programme können Sie zu Übungszwecken auf Ihrer HEXIO2 einmal eingeben. Dies geschiet durch Eingabe der Maschinencodes auf der Tastatur der HEXIO2. Jedes eingegebene Byte muß mit der CR-Taste bestätigt werden. Erst nach der Bestätigung stehen die Bitmuster im Speicher.

a) Ein- und Ausgabe von Bitkombinationen vom DIL-Schalter

```
DIL- Schalter                                              PAGE  1

                                TITLE   DIL- Schalter
                                org     8100h
                        start:
8100  DB 02                    in      a,(02h)         ; enlesen von
                                                      ; DIL-Schalter
                                                      ; in den Akku
8102  D3 02                    out     (02h),a         ; Akkuinhalt
                                                      ; auf LEDs
                                                      ; ausgeben
8104  18 FA                    jr      start           ; zurück zum
                                                      ; Anfang
```

b) Sägezahn-Funktion


```
                                TITLE   SÄGEZAHN - FUNKTION

                                org     8100h

                        start:
8100  3E FF                    ld      a,11111111b     ;Akku mit ffh
                                                      ;laden
8102  D3 02                    out     (02H),a         ;ausgeben auf
                                                      ;LEDs
8104  CD 41 81                 call    wait            ;schleife
                                                      ;aufrufen
8107  3E FE                    ld      a,11111110b
8109  D3 02                    out     (02h),a
810B  CD 41 81                 call    wait

810E  3E FC                    ld      a,11111100b
8110  D3 02                    out     (02h),a
8112  CD 41 81                 call    wait

8115  3E F8                    ld      a,11111000b
8117  D3 02                    out     (02h),a
8119  CD 41 81                 call    wait

811C  3E F0                    ld      a,11110000b
811E  D3 02                    out     (02h),a
8120  CD 41 81                 call    wait

8123  3E E0                    ld      a,11100000b
8125  D3 02                    out     (02h),a
8127  CD 41 81                 call    wait

812A  3E C0                    ld      a,11000000b
812C  D3 02                    out     (02h),a
812E  CD 41 81                 call    wait

8131  3E 80                    ld      a,10000000b
8133  D3 02                    out     (02h),a
8135  CD 41 81                 call    wait

8138  3E 00                    ld      a,00000000b
813A  D3 02                    out     (02h),a
813C  CD 41 81                 call    wait

813F  18 BF                    jr      start           ;zurück zum
                                                      ;Anfang

                        wait:
8141  21 00 00                 ld      hl,0000h        ;HL-Registerpaar

                                                      ;mit 00h laden
8144  11 01 00                 ld      de,0001h        ;DE-Registerpaar

                                                      ;mit 01h laden

                        loop:
8147  19                       add     hl,de           ;HL,DE addieren
8148  30 FD                    jr      nc,loop         ;wiederholen bis
                                                      ;Speicher voll

814A  C9                       ret                     ;zurück zum
                                                      ;Hauptprogramm
```

9. Diverses

9.1 Ausblick

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

9.2 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!

9.3 Ausbau des Systems

9.3.1 Weitere Möglichkeiten mit dem Einsteigerpaket

Eine Möglichkeit das Industrieeinsteigerpaket zu erweitern besteht darin, auf den Bussteckplatz weitere Baugruppen zu stecken. Im folgenden sind einige Beispiele aufgezeigt.

z.B.

- CAS-Baugruppe (Abspeichern auf Kassetten)
- ROB2-Baugruppe (Anschluß des Fischer-Technik-Roboters)
- Buserweiterung (Anschluß eines zusätzlichen Busses)

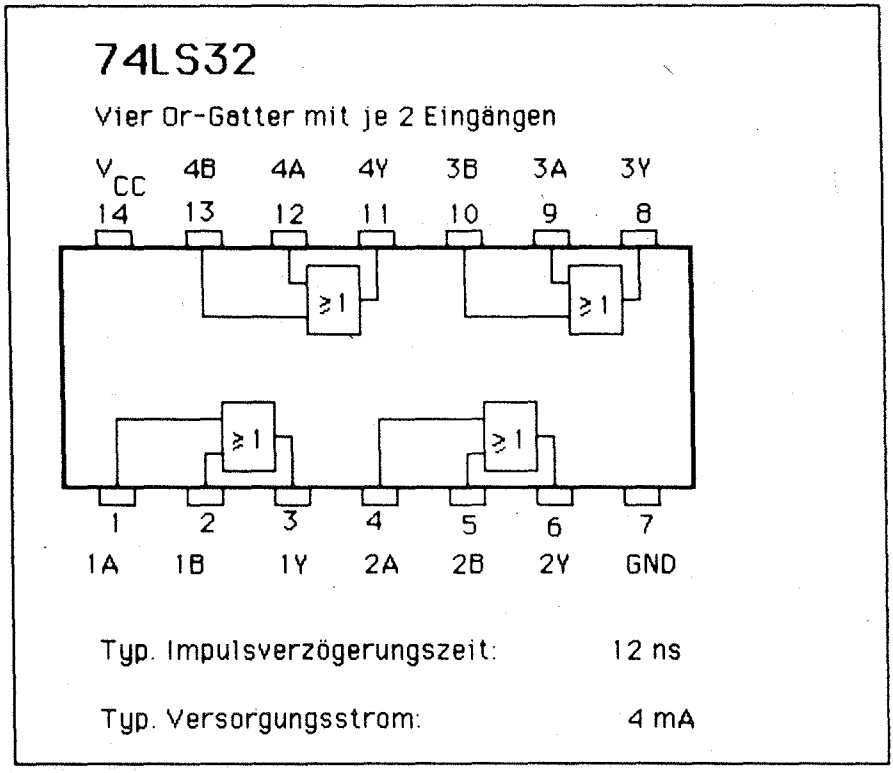
9.3.2 Erweiterung mit Bildschirm und Tastatur

Es ist auch möglich Ihr Einsteigerpaket zum Rechner mit Tastatur und Bildschirm umzurüsten. Die SBC3 bildet für diesen Ausbau die Grundlage. Bei einem derartigen Ausbau benötigen Sie allerdings einen Bus2, Bus3 oder Bus4.

10. Unterlagen zu den verwendeten ICs

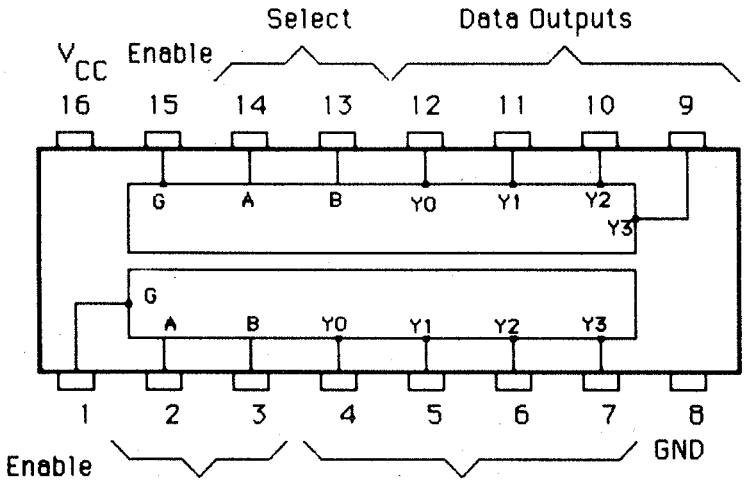
10.1 TTL-ICs

74LS32 vier OR-Gatter mit je 2 Eingängen



74LS139

zwei 2-Bit Binärdekoder/Demultiplexer (2 zu 4)



Wahrheitstabelle:

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

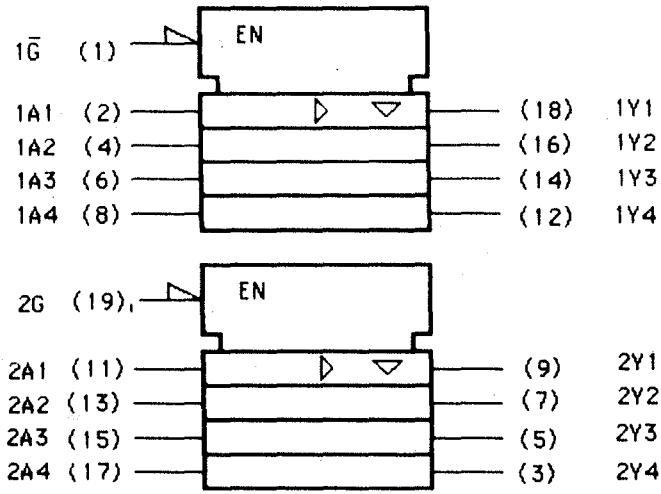
Typ. Impulsverzögerungszeit: 22ns

Typ. Versorgungsstrom: 7mA

Positive Logik: ja

74LS244

Acht Bus-Leitungstreiber (Tri -State)



Wahrheitstabelle:

Inputs		Outputs
$\overline{G}$	A	Y
H	X	Z
L	L	L
L	H	H

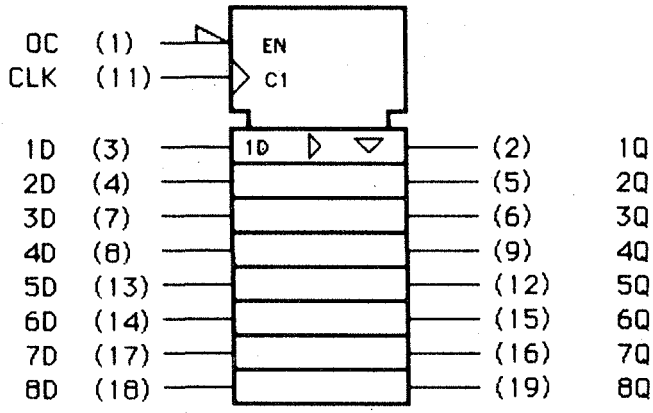
Typ. Impulsverzögerungszeit: 12 ns

Typ. Versorgungsstrom: 27 mA

positive Logik: ja

74LS374

8-Bit D Register mit Tri-State Ausgängen



Wahrheitstabelle:

Output Control	Clock D	Output
L	↑ H	H
L	↑ L	L
L	L X	Q0
H	X X	Hi Z

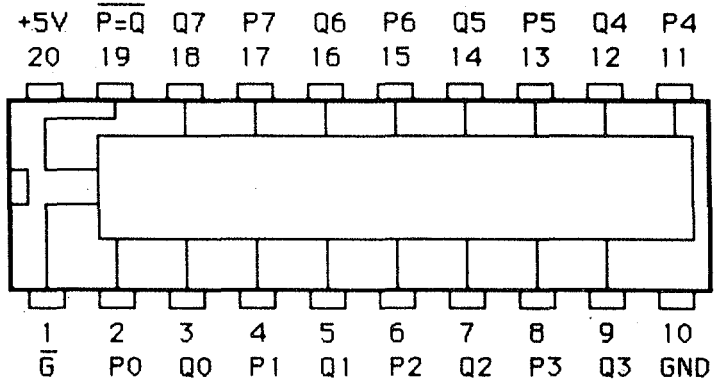
Typ. Impulsverzögerungszeit: 16 ns

Typ. Versorgungsstrom: 26 mA

positive Logik: ja

74LS688

8-Bit Größenvergleicher



Logiktabelle:

INPUT		OUTPUT
$\overline{G}$	P0, P1 ... P7 Q0, Q1 ... Q7	$\overline{P=Q}$
H	X X	H
L	P0≠Q0, P1≠Q1 .... P7≠Q7	H
L	... PY≠QY	H
L	P0=Q0, P1=Q1 .... P7=Q7	L

Typ. Versorgungsstrom: 40 mA

Typ. Impuls-Verzögerungszeit: 15 ns

11. Literatur

11.1 Hinweis auf 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.

11.2 Empfohlene Fachbücher

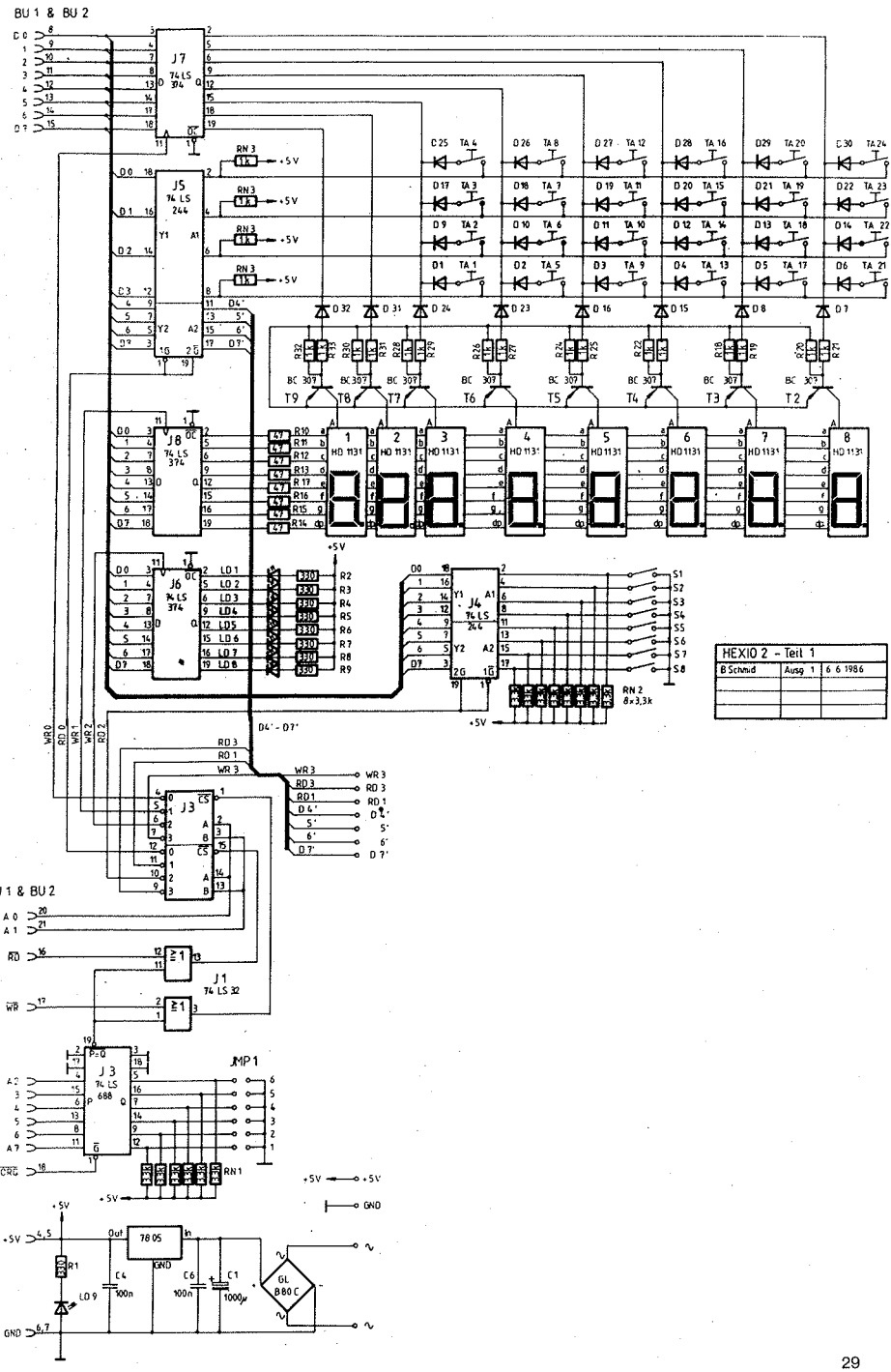
- Hans Fischer:
  Mikroelektronik Einführung mit dem NDR-Computer
  Heft 1 - 4
  Technisches Lehrinstitut Dr.-Ing P. Christiani
  Konstanz
  Bestellnummer: Band1 10144
  Band2 10145
  Band3 10146
  Band4 10147

- Rodnay Zaks:
  Die Programmierung des Z80
  Sybex-Verlag
  Bestellnummer: 10585

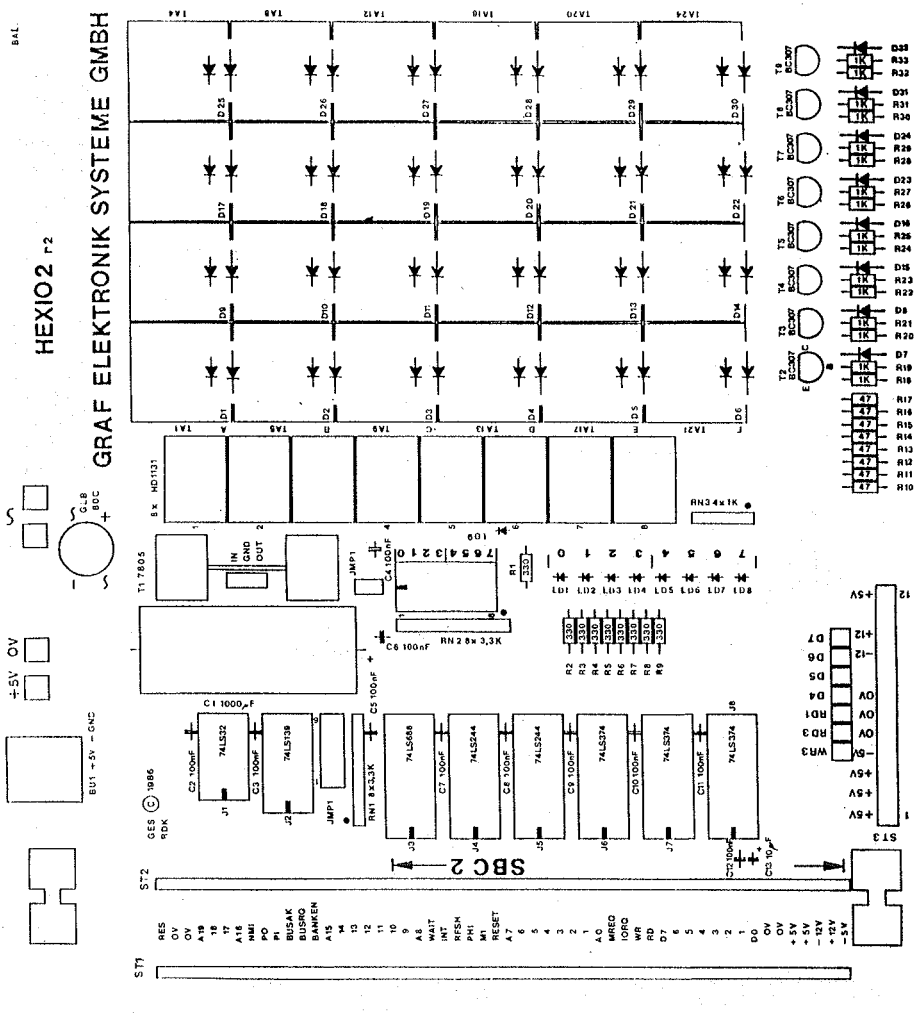
- Rolf-Dieter Klein:
  Mit Hexmon Programme entwickeln
  Bestellnummer: 10286

  Preise siehe gültige Preisliste

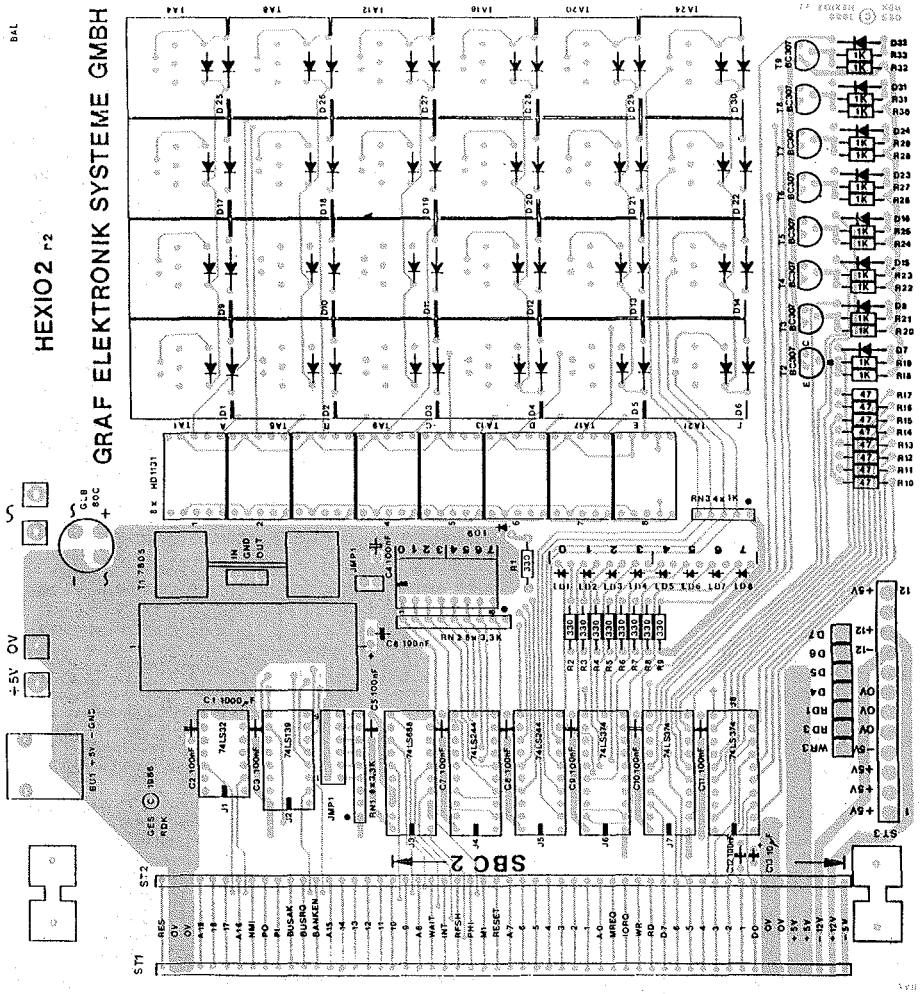
Anhang A: Schaltplan



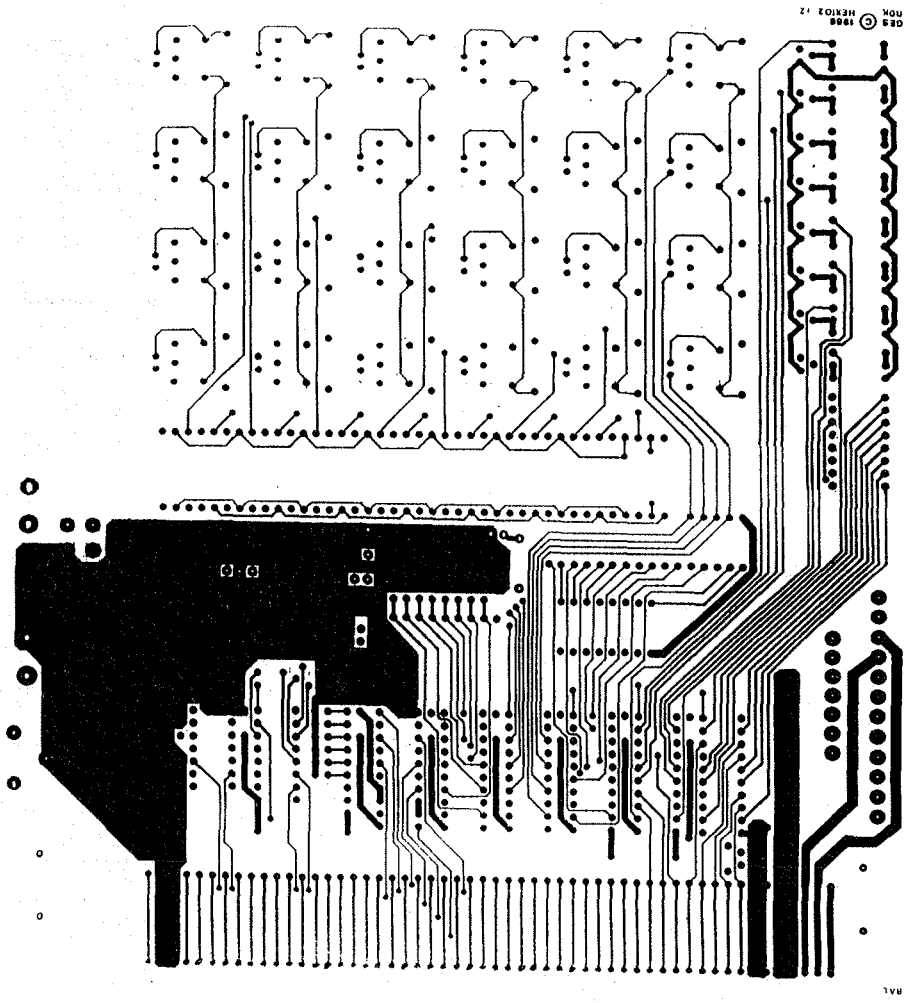
Anhang B: Bestückungsplan



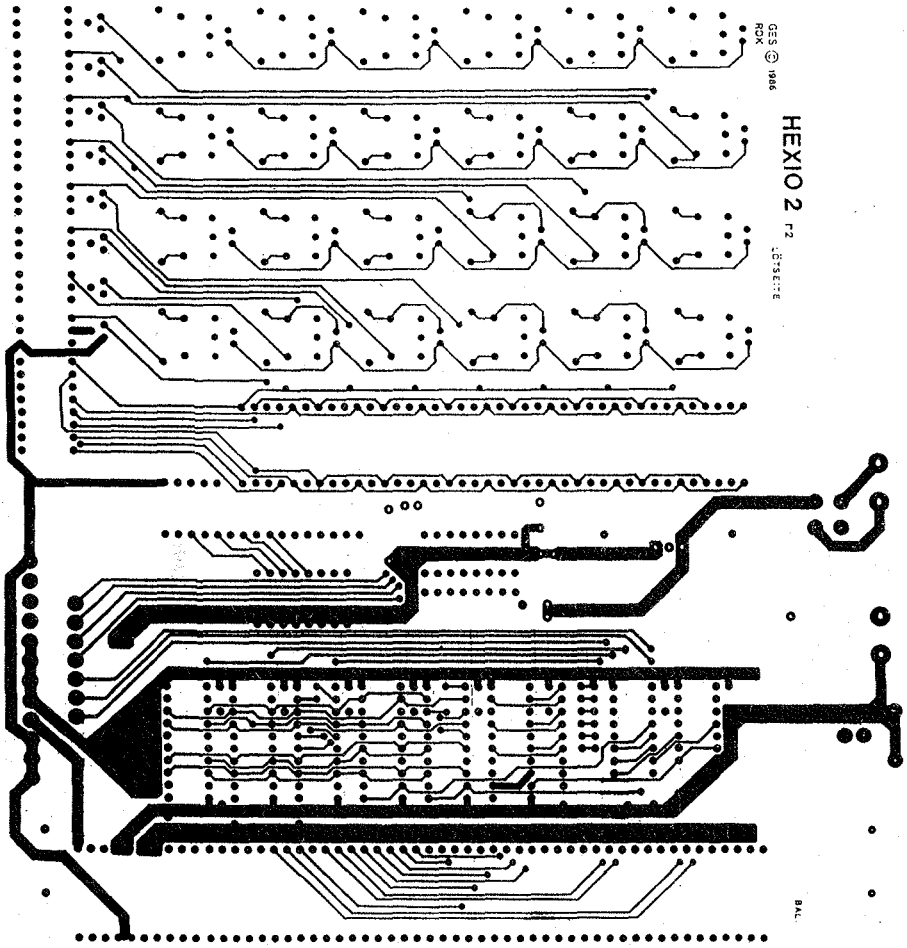
Anhang C: Layout Bestückungsseite mit Bestückungsdruck



Anhang D: Layout Bestückungsseite



Anhang E: Layout Lötseite



Neu!

Telefonservice 08 31- 62 11 jeden Mittwochabend bis 20.00 Uhr

---

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang
Telefon: 08 31 / 6 93 00

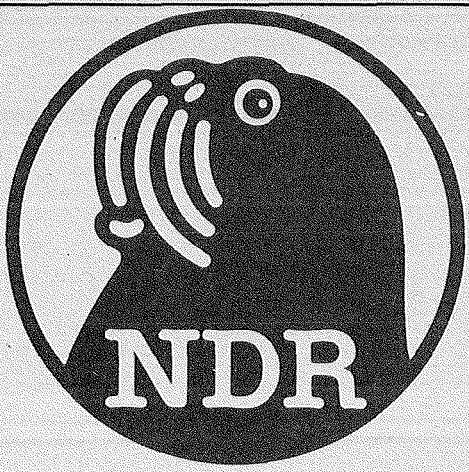
Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr

ges



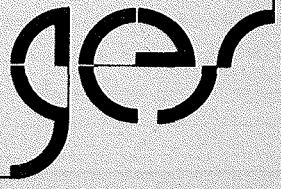
SBC3

Die universelle CPU-Baugruppe mit der CPU Z80

für den NDR-Computer

Ausgabe 3

Graf Elektronik Systeme GmbH



Inhalt Seite

1. Einführung

1.1 Zum NDR-Klein-Computer

Der NDR-Klein-Computer wird in der Fernsehserie "Mikroelektronik" - Mikrocomputer selbstgebaut und programmiert" aufgebaut, erklärt und in Betrieb genommen. Diese Serie wird vom Dorddeutschen Rundfunk, vom Sender Freies Berlin, vom Bayrischen Fernsehen und von Radio Bremen ausgestrahlt. Es werden bald auch die Regionalsender anderer Bundesländer die Sendung in ihr Programm aufnehmen. Zur Serie gibt es einige Begleitmaterialien, es ist daher nicht unbedingt notwendig, die Fernsehserie gesehen zu haben, um den NDR-Klein-Computer zu bauen und zu begreifen:

- Buch

  Rolf-Dieter Klein,
  "Rechner Modular"
  Der NDR-Klein-Computer -
  selbstgebaut und programmiert
  ISBN 3-7723-8721-7, DM 68,-
  erschienen im Franzis-Verlag, München
  Auf diesem Buch baut die NDR-Serie auf

- Zeitschriften "mc" und "ELO" des Franzis-Verlages

- Zeitschrift "LOOP" der Firma Graf (siehe Kapitel 11.1)

- Videocassetten:
  lizensierte Originalcassetten für den privaten Gebrauch.
  Auf diesen zwei Cassetten sind die 26 Folgen der Fernsehserie enthalten.
  Systeme: VHS, Beta, Video 2000
  Preise: siehe gültige Preisliste

1.2 Wozu dient die Baugruppe

Die Baugruppe SBC3 ist ein "Single Board Computer", d.h. soviel wie Einplatinencomputer. Ein "Einplatinencomputer" ist ein für sich allein funktionierender Computer, der nur noch diverse Ein/Ausgabe Einheiten fehlen um vernünftig arbeiten zu können. Auf dieser "SBC" sind CPU, Speicher und Schaltungen zur Steuerung der CPU vorhanden. Bei CP/M-Betrieb ersetzt die Baugruppe SBC3 die Baugruppen BANKBOOT und CPUZ80. Eine ROA64k oder RAM64/256 ist bei CP/M Betrieb schon noch nötig.

1.3 Wie setzt man die Baugruppe SBC3 ein

Da es sich um eine universelle CPU-Baugruppe handelt, kann die Baugruppe mit allen Baugruppen die für die CPU Z80 einsetzbar sind eingesetzt werden. Dies reicht vom Einsteigerpaket mit HEX-Tastatur bis zum kommerziell einsetzbaren CP/M-Computer. Abb. 1 zeigt die Konfiguration beim Einsteigerpaket. Abb. 2 zeigt die SBC3 in einer Konfiguration mit Bildschirm und Tastatur, aber ohne Floppy-Laufwerke. Abb. 3 zeigt die Systemkonfiguration beim vollausgebauten CP/M-System (mit Floppy-Laufwerken und eventuell Festplatte).

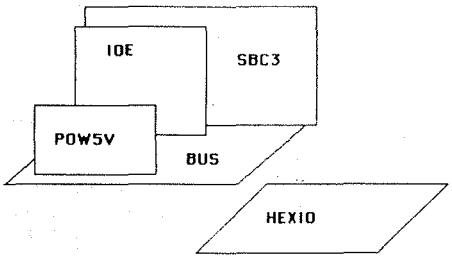


Abb 1: Konfiguration beim Einsteigerpaket

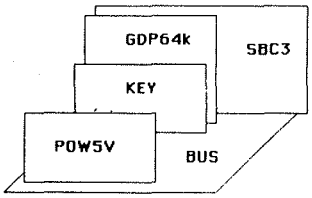
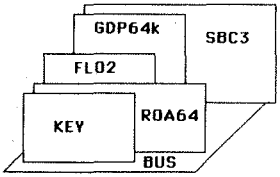


Abb.2: Mögliche Konfiguration beim System ohne Floppy und ohne CP/M



Statt der ROA64k kann auch die dynamische Speicherkarte RAM64/256 verwendet werden.
Als Stromversorgung wird das Netzgerät NE2 verwendet

Abb.3: Mögliche Konfiguration beim System mit Floppy und CP/M

2: Technische Daten

Baugruppengröße: 100 x 160 mm (Europakarte)
Bus: NDR-Bus

Spannungsversorgung: +5V

Stromverbrauch: 500 mA

Durch default-Einstellung der JMP mögliche Speicher: 8k (2764, 6264)

Einsetzbare Speicher:	EPROM:	4K	(2732)	max. 64k aber nur 32k
		8K	(2764)	sind sinnvoll, da
		16K	(27128)	sonst kein RAM
		32K	(27256)	
	RAM:	2K	(6116)	max. 16k
		8K	(6264 bzw. 5565 usw.)	

Akku: Spannung: +2,4V, Kapazität: 120 mAh
ungefähre Haltedauer der RAM-Speicherinformation: 1 Jahr

BANK-Logik zur Adressierung von 1 MByte

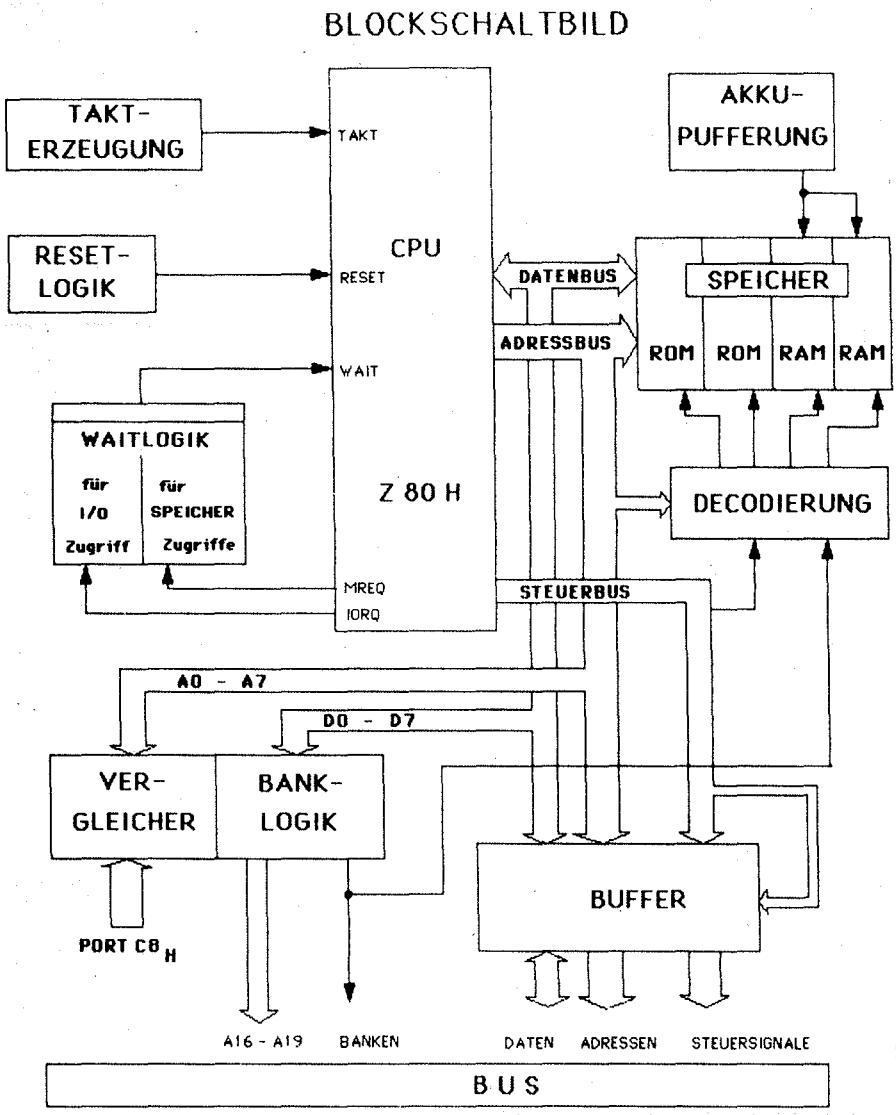
Voll gepufferter Daten-, Adress- und Steuerbus

Taktversorgung: 4 (Z80A) und 8 MHz (Z80H)

Wait-Zyklen einstellbar getrennt für Speicher und I/O

3. Prinzipbeschreibung

3.1 Blockschaltbild SBC3



3.2 Beschreibung des Blockschaltbildes und Schaltungsprinzip

Die CPU Z80 ist das Herz der Schaltung. Sie steuert zentral alle Abläufe des Computers. Einsetzbar sind die Z80-Typen Z80, Z80A, Z80B und Z80H. Damit die CPU vernünftig arbeiten kann braucht sie eine Taktversorgung und eine RESET-Logik. Der Takt bestimmt die Arbeitsgeschwindigkeit der CPU. Die RESET-Logik muß dafür sorgen, daß bei Tastendruck oder beim Einschalten des Computers ein RESET durchgeführt wird. Dazu ist es notwendig, daß für 3 Taktzyklen ein LOW-Signal am RESET-Eingang der CPU anliegt. Erklärung RESET: RESET bedeutet Rücksetzen und setzt die CPU in den Anfangszustand zurück.

Die Speicher sind die Arbeitsebenen der CPU. Vom Speicher holt er sich die Befehle die er dann ausführt, legt dort Daten ab, holt sie wieder etc. Dabei gibt es grob umrissen zwei Typen von Speichern: ROMs und RAMs. ROM ist die Abkürzung für "READ ONLY MEMORY" und bedeutet, daß von diesem Speicher nur gelesen werden kann. Die Speicherinformationen (Programme) sind dort fest eingebrannt und bleibt auch nach Abschalten der Spannungsversorgung erhalten. Das EPROM ist ein weiterentwickeltes ROM, das mit UV-Licht gelöscht werden kann. RAM ist die Abkürzung für "RANDOM ACCESS MEMORY" und bedeutet, daß auf diesen Speicher geschrieben und von ihm gelesen werden kann. Der Nachteil an diesen Speichern ist, daß nach Abschalten der Versorungsspannung die Speicherinformation gelöscht wird. Deshalb wurde hier bei der SBC3 eine Akku-Pufferung vorgesehen, die nach Abschalten der Spannungsversorgung die RAMs mit Strom versorgt.

Die BANK-AUSWAHL-LOGIK wird nur benötigt, wenn die SBC3 im CP/M-System eingesetzt wird. Hier dient sie dazu den Adressraum des Z80 auf 1 Mbyte zu erhöhen. Zum "booten" des Betriebssystems werden die Speicher auf der SBC3 verwendet. Das Betriebssystem wird von der Diskette auf eine Speicherbank geladen, die dann aufgerufen wird.

Die BUFFER dienen dazu den Datenbus, Adressbus und Steuerbus zu verstärken und auf den BUS weiter zu geben.

Die WAIT-Logik muß nur dann eingesetzt werden, wenn sie mit der CPU Z80H mit 8 MHz ihr System betreiben. Dabei können entweder die Speicher oder irgendwelche EIN/AUSGABE-Einheiten für die CPU zu langsam sein. Die WAIT-Zyklen können für Speicher und EIN/AUSGABE-Einheiten getrennt eingestellt werden (bis zu je 4 WAIT-Zyklen).

3.3 Prinzipbeschreibung CP/M und "booten"

Das Betriebssystem CP/M benötigt einen RAM-Bereich von 0000H beginnend. Dies kann mit Hilfe der Bankumschaltung erreicht werden. Das Betriebssystem ist aber auf Diskette gespeichert und muß in den RAM-Bereich geladen werden. Um das Betriebssystem in diesen Bereich zu laden, verwendet man das Prinzip des "Bootstrap-Loaders". D.h. der Monitor (Flomon) lädt das Betriebssystem von der Floppy-Disc in den RAM-Bereich und springt nach dem Laden in den RAM-Bereich und startet das eben geladene Betriebssystem CP/M. Diesen Vorgang bezeichnet man als "booten".

Dabei sitzt auf der SBC3 das "Boot-EPROM" (FLOMON). Der RAM-Bereich in den das CP/M geladen wird, muß durch eine ROA64k oder eine RAM64/256 zur Verfügung gestellt werden. Aber es wird keine BANKBOOT mehr benötigt.

4. Aufbauanleitung

4.1 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.

Bei der SBC3 sind die RAM 8k oder RAM 2k CMOS-Bausteine.

4.2 Stückliste

1	10389	Original GES-Platine mit Lötstoplack und Bestückungsdruck r4		
1	10388	Handbuch Ausgabe 2		
1	60033	74 04	J23	6 Iverter
1	60038	74 121	J1	Monoflop
1	60075	74 LS 00	J4	4 NAND
1	60076	74 LS 01	J24	4 NAND (open Collector)
1	60079	74 LS 04	J14	6 Inverter
2	60121	74 LS 32	J3,J11	4 OR
1	60137	74 LS 74	J2	2 D-Flip-Flop
1	60094	74 LS 138	J16	3 zu 8 Dekoder
2	60095	74 LS 139	J5,J13	2 zu 4 Dekoder
2	60102	74 LS 164	J12,J14	8-Bit Schieberegister
4	60115	74 LS 245	J17,19,20,21	8-Bit-Tri-State Bustreiber
1	60118	74 LS 273	J22	8-Bit D-Register
1	60135	74 LS 688	J18	8-Bit Größenvergleicher
1	10802	CPU Z80A	J10	CPU
1	10357	RAM 8k	J8, od. J9	RAM-Speicher 8kbyte
1	60179		Q 1	Quarz 8 MHz
11	60239	100 nF	C1,C2,C4-C12	Keramik-Kond. 100 nF
2	60248	10 uF	C3, C13	Tantal-Kond. 10 uF
9	60626	1k	R1, R2, R12 - R18	Widerstände 1 kOhm
1	60617	10k	R10	Widerstand 10 kOhm
2	60627	2,2k	R7, R8,	Widerstände 2,2 kOhm
3	60648	4,7k	R4, R5, R9	Widerstände 4,7 kOhm
1	60746	68	R3	Widerstand 68 Ohm
1	60631	220	R6	Widerstand 220 Ohm
1	60643	330	R11	Widerstand 330 Ohm
1	60519	8*1k	RN2	Netzwerkw. 8*1 kOhm
1	60518	8*3.3k	RN1	Netzwerkw. 8*3,3 kOhm
1	60290	1N4148	D1	Si-Diode
4	60486	Mod 225H		Shunt-Stecker
1	60502		JMP12, JMP13	Stiftreihe 2 * 4 gerade
2			JMP1, JMP15	Stiftreihe 2 * 2 gerade
2	60603	BSX 20	TR1, TR2	Transistoren
1	60221	NCM-2,4	Akku	Akku 2,4 V
1	60301		S1	Taster für RESET

1	10406	ST 1	36-polige Steckerleiste
1	10405	ST 1	18-polige Steckerleiste
1	60193		40-polige IC-Fassung
4	60190		28-polige IC-Fassung
6	60187		20-polige IC-Fassung
3	60185		16-polige IC-Fassung
9	60183		14-polige IC-Fassung

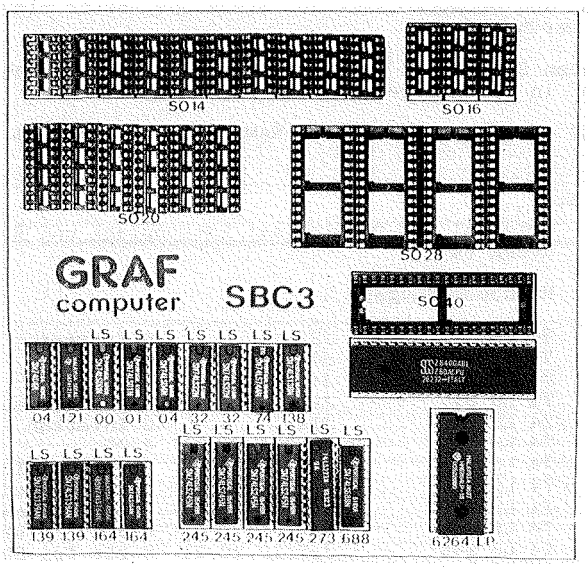


Abb.: ICs und Sockel der SBC3

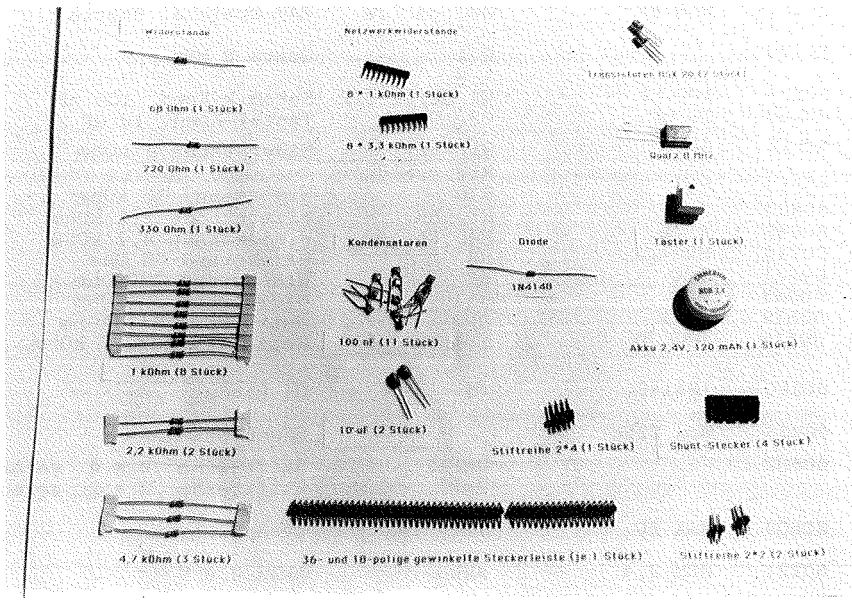
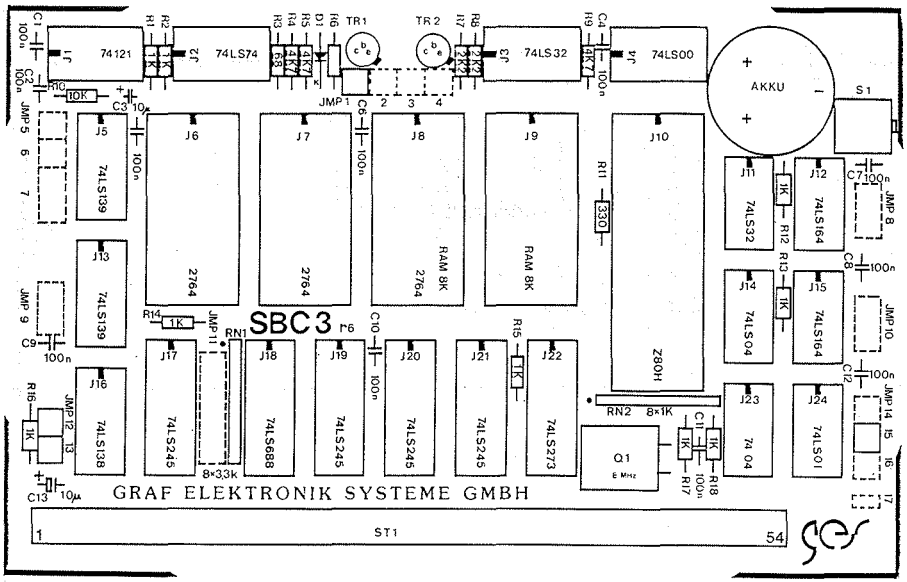
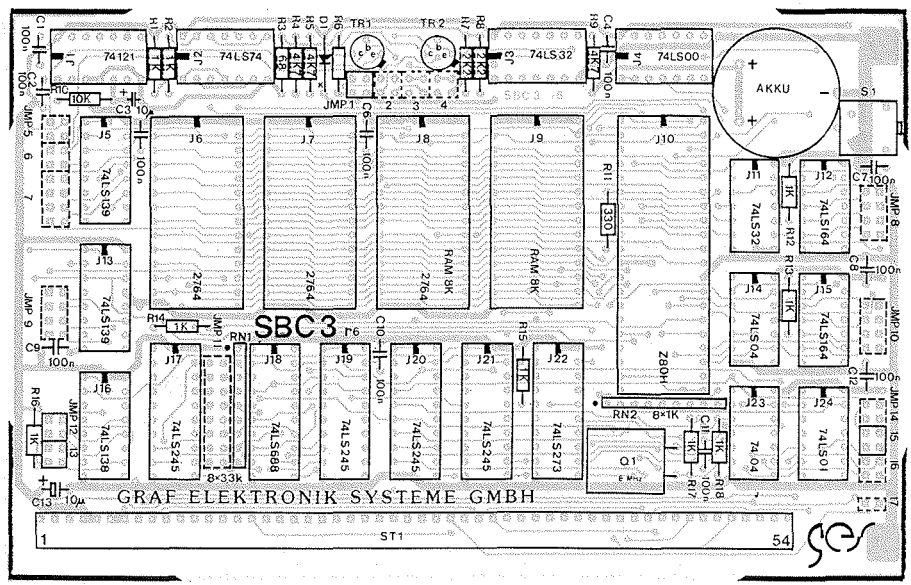


Abb.: Einzelne Bauteile der SBC3

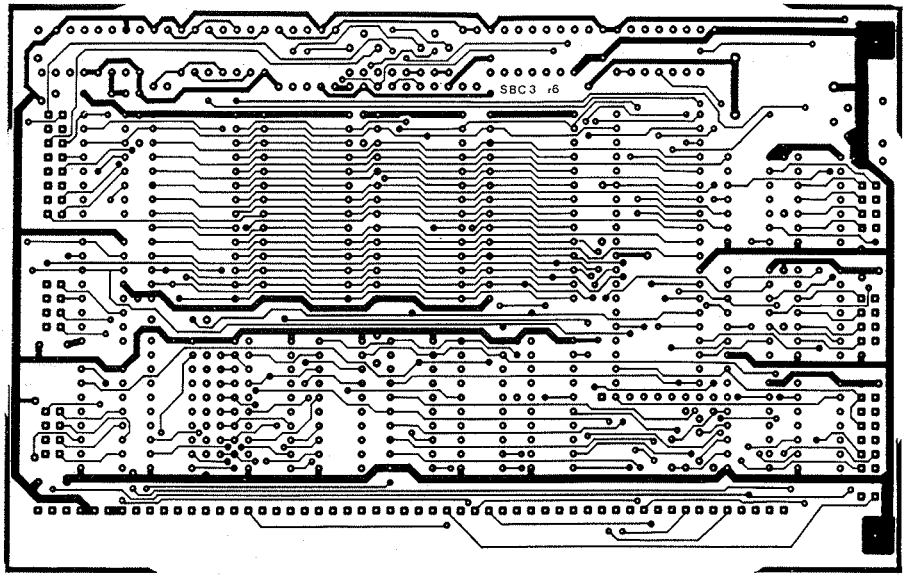
4.3 Bestückungsplan



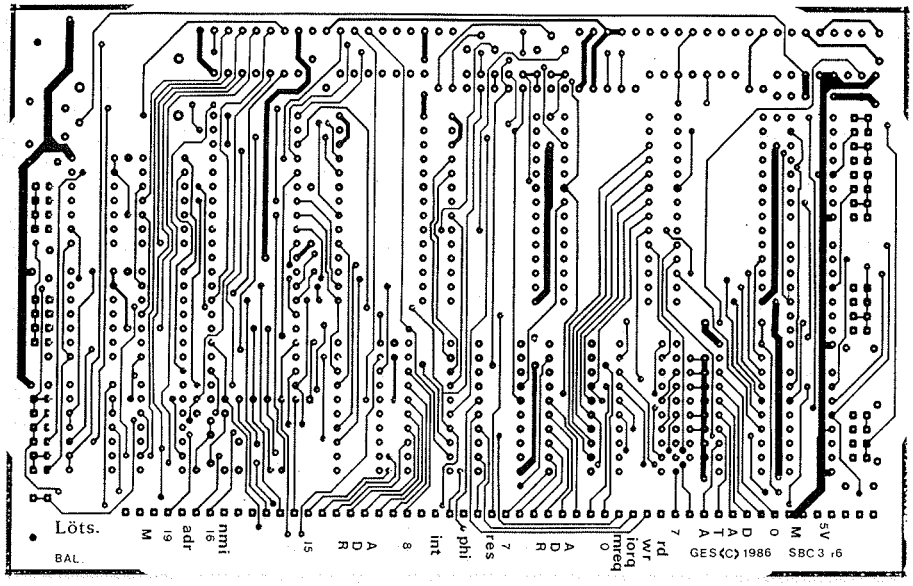
4.4 Layout Bestückungsseite mit Bestückungsplan



4.5 Layout Bestückungsseite



4.6 Layout Lötseite



4.7 Aufbau Schritt für Schritt

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. Beim Einlöten der Bauelemente beginnt man am besten mit der gewinkelten Steckerleiste. Es sollte darauf geachtet werden, daß die Leiste parallel zur Leiterplatte liegt, um gut auf den Bus gesteckt werden zu können. Dabei sollten zuerst die beiden äußeren Stifte und einer in der Mitte verlötet werden. Dann empfiehlt es sich nachzuschauen, ob die Stecker parallel zur Platine liegen und ob keine "Bäuche" zwischen den verlöteten Stiften liegen. Sollten "Bäuche" vorhanden sein, muß wiederum in der Mitte der "Bäuche" ein Stift unter Druck angelötet werden. Liegt die Steckerleiste dann richtig, können die restlichen Stifte verlötet werden.

Nun wird die Leiterplatte mit den IC-Sockel 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 Leiterplatte durch den Aufdruck sehr deutlich zu erkennen, oder auf dem Bestückungsplan, wenn kein Bestückungsdruck vorhanden.

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 Leiterplatte 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.

Die Kondensatoren C3 und C13 sind gepolt und dürfen auf keinen Fall falsch herum eingelötet werden. Der Pluspol ist mit einem "+" gekennzeichnet. Im Bestückungsplan ist der Pluspol ebenfalls mit einem "+" gekennzeichnet.

Die Kondensatoren C1, C2 und C4 bis C12 sind ungepolt und können ohne auf die Polung zu achten eingelötet werden.

Die Netzwerkwiderstände RN1 (8*3,3 kOhm) und RN2 (8*1 kOhm) haben einen gemeisamen Anschluß der mit einem Punkt am Bauelement und auf dem Bestückungsplan gekennzeichnet ist. Die Größe der Netzwerkwiderstände sind nicht durch Farbcode ausgedrückt, sondern durch drei Ziffern. Dabei entsprechen die ersten beiden Ziffern den Anfangsziffern des Widerstandswertes und die dritte Ziffer die Zehnerpotenz, also die Anzahl der anzuhängenden Nullen. Beim 1 kOhm Netzwerkwiderstand steht der Zahlenwert "102" drauf, also 10 und zwei Nullen = 1 kOhm. Beim 3.3 kOhm Netzwerkwiderstand ist der Wert "332" aufgedruckt. Die anderen Aufdrucke mit Ausnahme des Punktes sind hier nicht von Bedeutung.

Die Widerstände R1 bis R18 sind Einzelwiderstände mit Farbcode. Die verwendeten Widersände im Farbcode:

Widerstandswerte	Farbcode
68 Ohm	blau - grau - schwarz
220 Ohm	rot - rot - braun
330 Ohm	orange - orange - braun
1 kOhm	braun - schwarz - rot
2,2 kOhm	rot - rot - rot
4,7 kOhm	gelb - violett - rot
10 kOhm	braum - schwarz - orange

Wie aus der Tabelle hervorgeht sind für die Widerstandswerte die ersten drei Ringe ausschlaggend. Der vierte Ring dient nur zur Toleranzangabe und ist meistens "gold".

Die Diode D1 ist gepolt und darf nicht falsch herum eingelötet werden. Die Kathode ist auf der Diode mit einem Strich gekennzeichnet. Auf dem Bestückungsplan ist die Kathode mit einem "K" beschriftet.

Die beiden Transitoren TR1 und TR2 haben drei Anschlüße: Basis, Emitter und Collector. Auf dem Bestückungsplan sind die Anschlüße mit B, C und E bezeichnet. Am Transistor ist der Anschlußpin der der "Nase" am Transistorgehäuse am nächsten kommt der Emitter. Der mittlere PIN ist die Basis und der dem Emitter gegenüberliegende der Collector. Der Transistor kann einfach eingesetzt werden, ohne daß die Beinchen gekreuzt werden müssen.

Der Quarz Q1 ist ungepolt und sollte möglichst liegend eingelötet werden. Dabei sollten Sie aber darauf achten, daß das Gehäuse des Quarzes nicht die Leiterplatte berührt, und damit Kurzschlüsse zwischen Leiterbahnen verursachen könnte. Sie könnten aber auch etwas Isoliermaterial (z.B. etwas Papier) unter den Quarz legen um dies sicher zu verhindern.

Die beiden Stiftreihen 2*2 werden an JMP1 und an JMP15 eingesetzt. Die Stiftreihe 4*2 wird an JMP12 und JMP13 eingesetzt. Die Shuntstecker werden je nach Systemkonfiguration gesteckt (siehe Einstellung der JMP).

Der Taster S1 kann aufgrund der Bohrungen auf der Leiterplatte nicht falsch herum eingelötet werden.

Falls die SBC3 ins GEH3 (Art. Nr. 10673) eingebaut werden soll, benötigen Sie das Rückwandblech für CPU's (Art. Nr. 10828). Dieses Rückwandblech wird zum einen an die Leiterplatte mit zwei Schrauben festgeschraubt, und zum anderen an die Rückwand des Gehäuses. Damit ist die Baugruppe fest justiert.

Der Akku "NCM 2,4" hat drei Beinchen wobei 2 für den "+" Pol sind und einer der "-" Pol. Infolge der Rasterung auf der Platine kann der Akku nicht falsch herum eingelötet werden. Falls die Bohrungen für den Akku zu klein sind, müssen Sie die Beinchen des Akku mit einer Flachzange etwas quetschen, oder Sie löten den Akku einfach nur auf die Bohrungen auf.

**** ACHTUNG ****

Betrifft: SBC 3 - JUMPER - Voreinstellung

Die in der Stückliste (Seite 5) aufgeführten Stiftleisten

1 Stück 2 x 4 gerade (JMP 12, JMP 13)
und
2 Stück 2 x 2 gerade (JMP 1, JMP 15)

mit der Bestell- Nr. 60502 werden nicht mitgeliefert, da diese direkt auf der Platine voreingestellt sind.

Bei der Abbildung auf Seite 6 unten sind diese zwar abgebildet, aber nicht im Bausatz enthalten.

Demzufolge brauchen Sie die Einstellung dieser Jumper nicht mehr vorzunehmen und können die in Abschnitt 5.3 geforderten Arbeiten überspringen. (Siehe Seite 13)
Die stark schraffierten JMP im SBC 3 - Handbuch Seite 13 (JMP1, JMP 12, JMP 13, JMP 15) sind also jetzt nicht mehr offen...

Wollen Sie später Ihr System weiter ausbauen, so ist es eventuell erforderlich, diese fest eingestellten Jumper aufzutrennen und Drahtbrücken einzulöten. Dazu dann bitte den Artikel 5.3. genau durchlesen.

Im Ihnen vorliegenden System stehen die Jumper JMP 1, JMP 12, JMP 13 und JMP 15 auf 1 (siehe Skizze S. 14 oben). Das heißt, die RAM's liegen auf Adresse 8000h bis 9FFFh und A000h bis BFFFh. Die SBC 3 kann jetzt als Single Board Computer ohne CP/M verwendet werden.

Die Platinenlayouts auf Seite 8 und der Bestückungsdruck auf Seite 7 tragen die Bezeichnung SBC 3 r6 (Revision 6). Ihnen liegt jedoch die Revision 7 vor.
Die Ausführung des Gerätes der r6 ist identisch mit r7, lediglich die oben beschriebenen Jumper sind bei Revision 7 im Layout (Kapitel 4.6) voreingestellt.

5. Testanleitung

5.1 Erste Prüfung ohne ICs

Die Platine ist bis jetzt erst mit den Sockeln und mit den passiven Bauelementen bestückt. Mit diesem Aufbau werden die ersten Tests durchgeführt.

Wenn Sie den Akku und die Diode richtig bestückt haben müssen an Pin 28 der IC-Sockel IC8 und IC9 jeweils ca. 2,5 V anliegen. Dies ist die Pufferspannung für die RAMs.

Zum nächsten Test muß die Baugruppe in den Bus gesteckt werden. Achten Sie beim Einstecken in den Bus, 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.

Man mißt, ob an allen IC-Sockeln die Versorgungsspannung von +5V ankommt. Dabei liegt jeweils am letzen Pin eines (z.B. bei 14-poligen an Pin 14) liegt 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).
An Pin 28 der ICs 8 und 9 liegen jetzt ca. 4,5 V. Das kommt davon, daß an der Diode ca. 0.7V abfallen. Liegt die Spannung unter 4,5 V ist die Spezifikation für die Speicher nicht mehr erfüllt. Trotzdem läuft die Schaltung fast in jedem Falle, da diese CMOS-RAM relativ unempfindlich gegen Versorgungsspannungserniedrigung sind.

Liegt die Versorgungsspannung +5V und 0V (Masse) an den richtigen Pins an können die ICs eingesetzt werden. Dabei muß auf die Richtung der ICs geachtet werden. Die Markierung auf dem IC muß mit der Kerbe in der Fassung übereinstimmen.

5.2 Test im System

5.2.1 Test im HEX-System

Konfigurieren Sie Ihr System, wie in Punkt 1.3 dargestellt. Das EPROM EHEX2 wird auf den Einbauplatz IC6 gesteckt. Und mindestes ein RAM 8k muß gesteckt werden (IC8). Die JMP müssen jeweils in Stellung 1 gestellt sein (siehe 5.3.1). Sie können auch die Speicher die auf der SBC2 verwendet werden, allerdings müssen Sie dann die entsprechende Speicherkonfiguration an den JMP einstellen (siehe 5.4). Funktioniert die Baugruppe, so muß nach dem Einschalten das "Hallo- 1.1" erscheinen.

5.2.2 Test im System mit Tastatur und Bildschirm ohne CP/M

Konfigurieren Sie Ihr System wie unter 1.3 erläutert. Stecken Sie das EPROM EGRUND2 auf den ersten Einbauplatz (IC6), der zweite (IC7) kann leer bleiben. Es kann aber auf diesen auch EGOSI2 oder ESPS2 gesteckt werden. Außerdem wird mindestes ein RAM 8k benötigt (IC8). Die JMP müssen wie unter 5.3.1 gesteckt werden. Verwenden Sie andere Speicher müssen Sie die JMP entsprechend der Speicherkonfiguration (siehe unter 5.4) eingestellt werden. Wenn Sie das System jetzt einschalten, muß das Grundmenu auf dem Bildschirm erscheinen.

5.2.3 Test im System mit CP/M

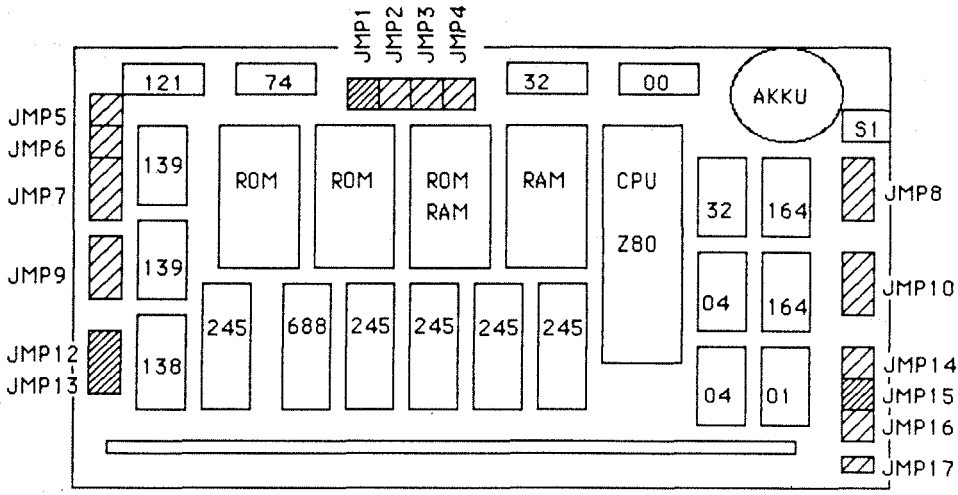
Konfigurieren Sie Ihr System wie unter Punkt 1.3. Auf dem ersten Steckplatz (J6) wird das EPROM "FLOMON" gesteckt. Auf dem vierten Steckplatz (J9) muß ein 8k RAM stecken. Haben Sie noch das alte FLOMON V1.5 müssen zwei 8k RAMs (J8 und J9) gesteckt sein. Die JMP müssen dann wie unter 5.3.2 beschrieben gesteckt sein. Es kann auch

die gleiche Speicherkonfiguration wie auf der BANKBOOT gewählt werden, dann muß aber die entsprechende Speicherkonfiguration mit den JMP eingestellt werden (siehe unter 5.4). Wenn Sie jetzt einschalten muß auf dem Bildschirm das kleine Menu mit "1 = Floppy Boot" "2 = Bank E0000H" usw, erscheinen.

5.2.4 Test mit dem ZEAT-System

Konfigurieren Sie das System wie beim CP/M-System. Das 8k EPROM "FLOMON" wird auf den ersten Steckplatz gesteckt. Die beiden EPROMs "ZEAT A" und "ZEAT B" werden auf die Einbauplätze 2 und 3 (J7, J8) gesteckt. Ein 8k RAM muß auf Steckplatz 4 (J9) gesteckt werden. Wenn Sie jetzt einschalten, muß dasselbe Menu wie beim CP/M System erscheinen. Wählen Sie Punkt 4 des Menu an, muß das Christiani ZEAT Menu erscheinen.

5.3 Einstellung der JMP



Skizze: Bestückungsplan mit JMP

Die stark schraffierten JMP (JMP1, JMP12, JMP13 und JMP15) sind "offene" JMP und müssen von Ihnen eingestellt werden. Die restlichen JMP (leicht schraffiert) sind auf der Lötseite voreingestellt und müssen normalerweise nicht geändert werden. Diese JMP müssem sie ändern, wenn Sie keine 8k Speicher verwenden, wenn Sie die CPU Z80H mit 8 MHz einsetzen wollen oder wenn Sie den Takt zum Bus trennen wollen usw.

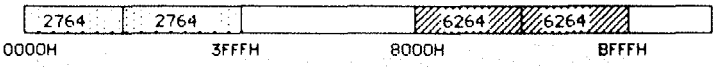
Einstellung der offenen JMP mit "Shunt-Stecker"

Dabei handelt es sich um die JMP1, JMP12, JMP13 und JMP15. Mit diesen 4 JMP kann die SBC3 als "Single Board Computer", ohne CP/M (bisher SBC2 oder CPU Z80 mit ROA64k) oder als zentrale Baugruppe für CP/M (bisher CPU Z80 und BANKBOOT) mit FLOMON oder mit ZEAT (Assembler von Christiani) verwendet werden. Vorausgesetzt ist dabei, daß 8k-EPROMs (2764) und 8k-RAMs eingesetzt werden.

5.3.1 System mit Tastatur und Bildschirm ohne CP/M

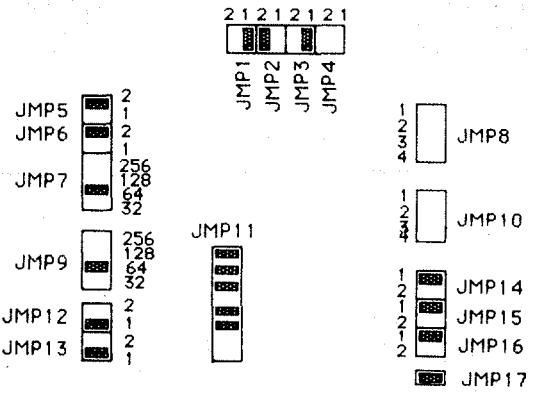
Bei diesem System können maximal 2 EPROMs 2764 und 2 RAM 8k eingesetzt werden (32k).

Speicherkonfiguration:



Einstellung der JMP

JMP1	Stellung 1
JMP12	Stellung 1
JMP13	Stellung 1
JMP15	Stellung 1



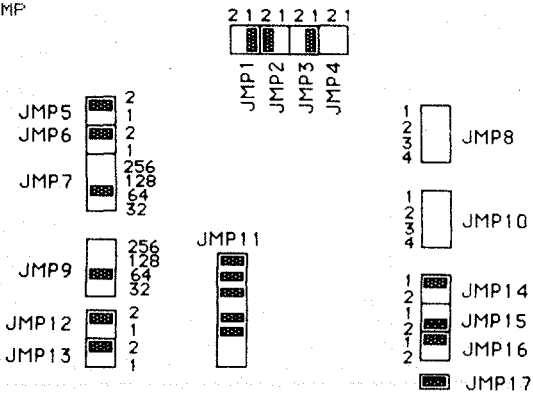
5.3.2 System mit CP/M

Bei diesem System können auch maximal 2 EPROMs 8k und 2 RAMs 8k eingesetzt werden. Auf der Adresse 2000H kann noch das Grundprogramm EGRU2000, oder EGOSI2 usw. stecken.

Speicherkonfiguration:

Einstellung der einzelnen JMP

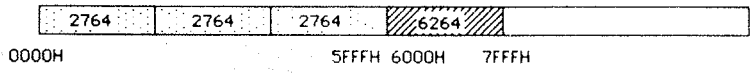
JMP1	Stellung 1 oder 2
JMP12	Stellung 2
JMP13	Stellung 2
JMP15	Stellung 2



5.3.3 System mit CP/M und ZEAT

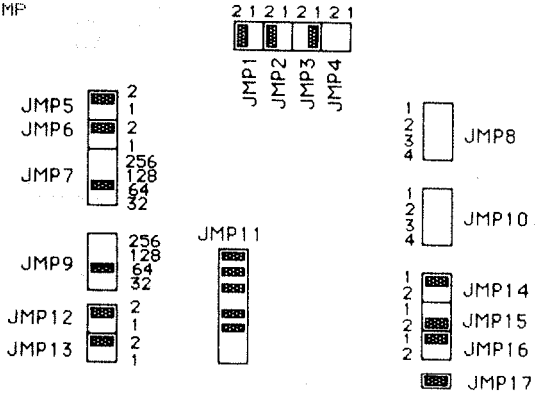
Bei diesem System werden 3 EPROMS 8k und 1 RAM 8k eingesetzt.

Speicherkonfiguration:



Einstellung der einzelnen JMP

JMP1 Stellung 2
JMP12 Stellung 2
JMP13 Stellung 2
JMP15 Stellung 2

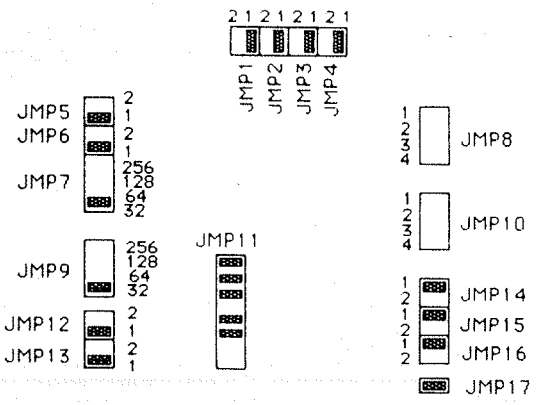


Dies sind die drei Einstellungen, mit denen die SBC3 normalerweise betrieben wird. Wollen Sie andere Speicher als 8k Speicher verwenden oder die CPU Z80H mit 8 MHz dann müssen Sie die voreingestellten JMP "anrühren".

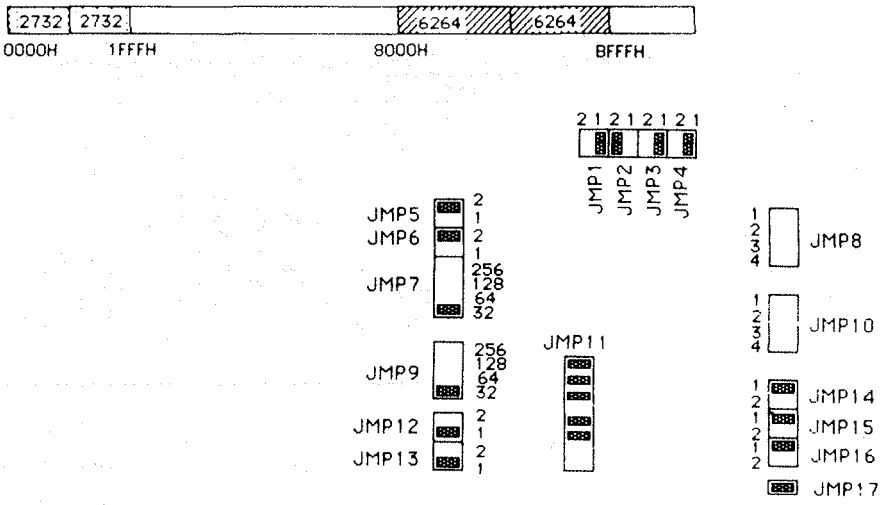
5.4 Speicherkonfigurationen und dazugehörige Jumperstellungen

5.4.1. System ohne CP/M mit 8k EPROMs und 8k RAMs (siehe 5.3.1)
5.4.2. System mit CP/M mit 8k EPROMs und 8k RAMs (siehe 5.3.2)
5.4.3. System mit CP/M und ZEAT mit 8k EPROMs und 8k RAM (siehe 5.3.3)

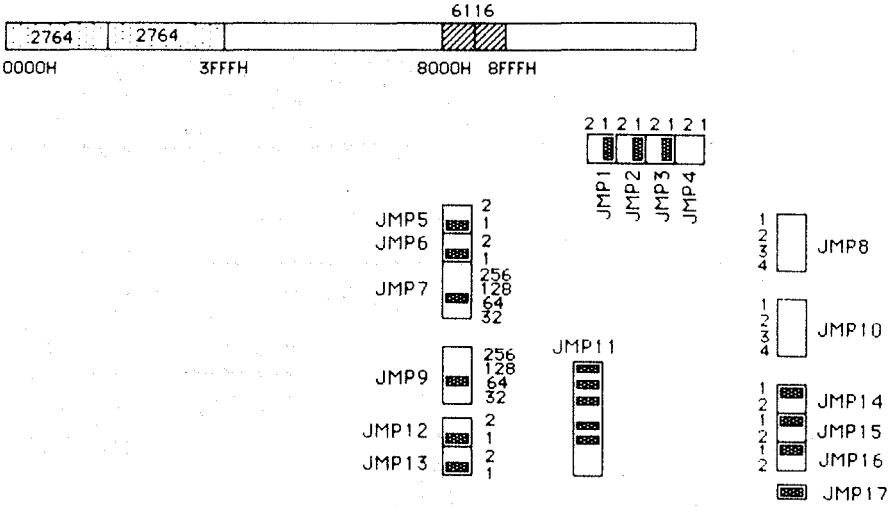
5.4.4 System ohne CP/M mit 4k EPROMS (2732) und 2k RAMs (6116): Konfiguration wie SBC2



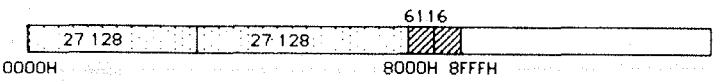
5.4.5. System ohne CP/M mit 4k EPROMs und 8k RAMs

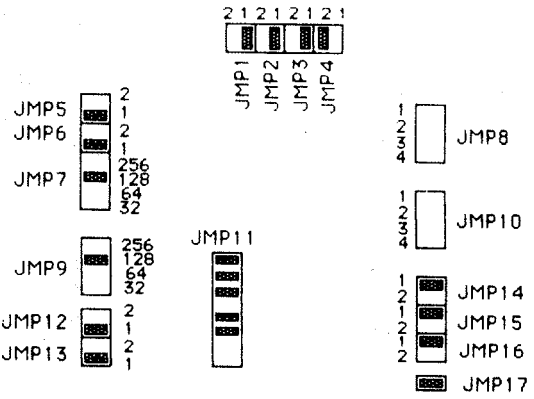


5.4.6. System ohne CP/M mit 8k EPROMs und 2k RAMs

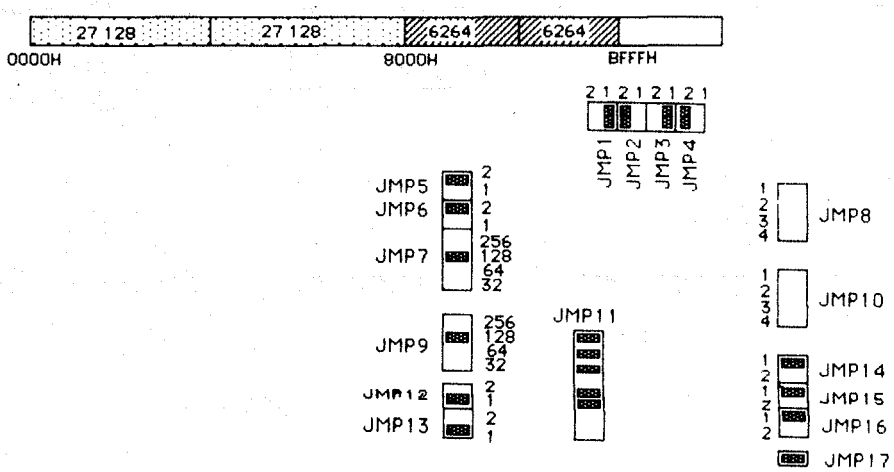


5.4.7 System ohne CP/M mit 16k EPROMs und 2k RAMs

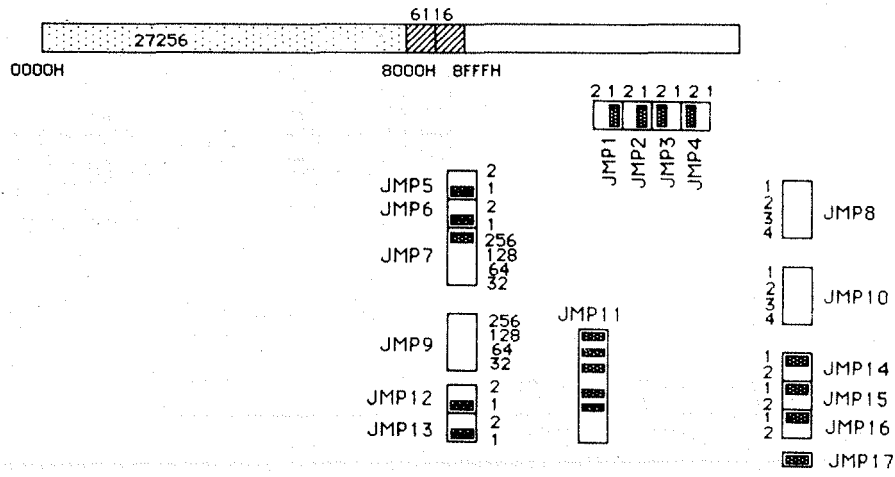




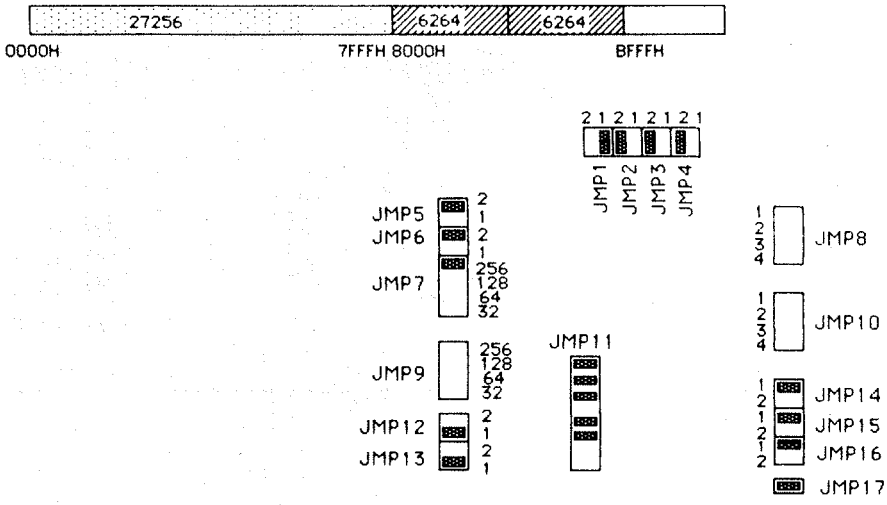
5.4.8. System ohne CP/M mit 16k EPROM und 8k RAMs



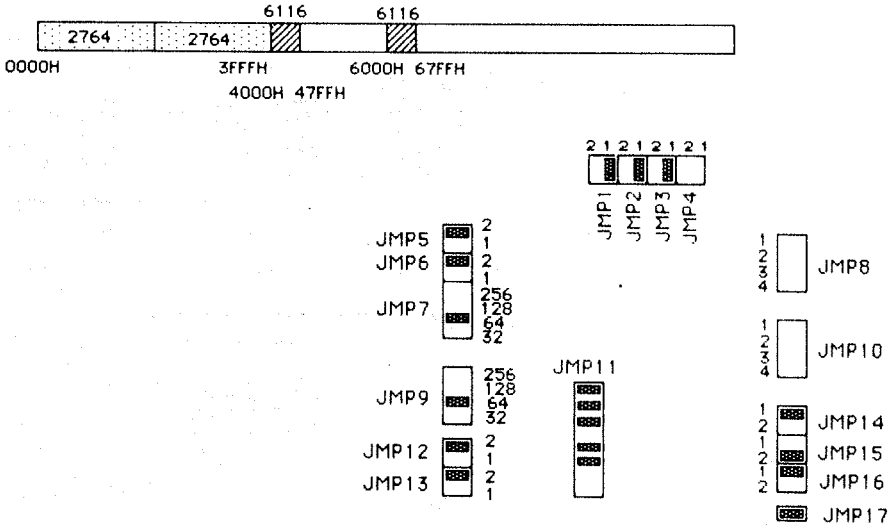
5.4.9. System ohne CP/M mit 32k EPROMs und 2k RAM



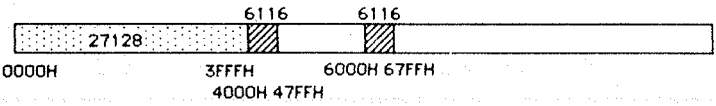
5.4.10. System ohne CP/M mit 32k EPROMs und 8k RAMs

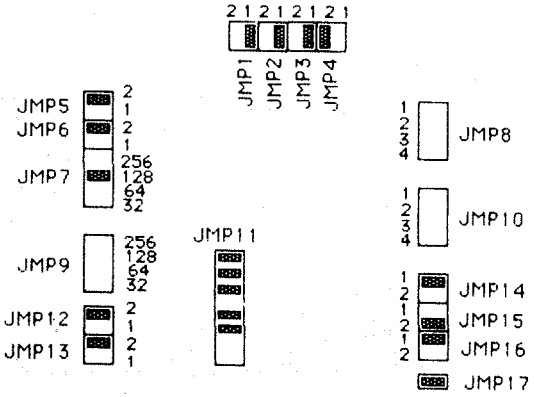


5.4.11. System mit CP/M mit 8k EPROMs und 2k RAMs (wie BANKBOOT)

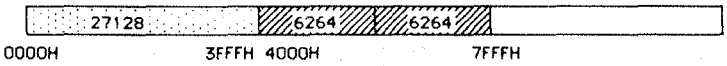


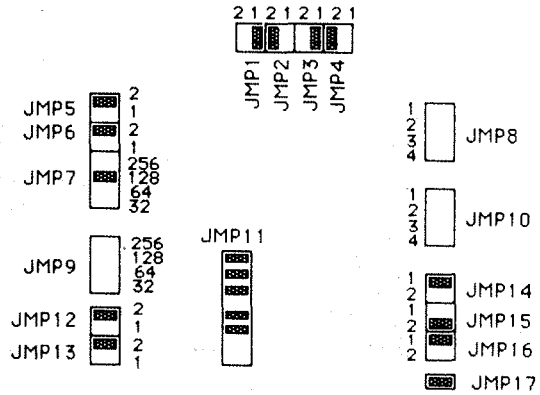
5.4.12. System mit CP/M mit 16k EPROMs und 2k RAMs





5.4.13 System mit CP/M mit 16k EPROMs und 8k RAMs



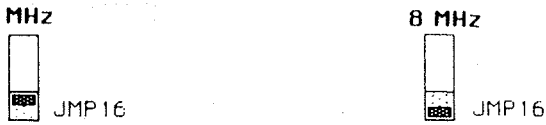


Verwenden Sie das System ohne CP/M und ohne Floppy-Laufwerke können Sie im Höchstfall 1 EPROM 27256 verwenden, da nur 32k ROM-Bereich sinnvoll sind. Beim CP/M-System sind nur 16k ROM vorgesehen, deshalb kann im Höchstfall nur 1 EPROM 27128 und kein EPROM 27256 verwendet werden. Ausnahme: Beim ZEAT-System von der Firma Christiani werden 3 EPROMs 2764 (8K) verwendet, also ein ROM/Bereich von 24k und ein RAM Bereich von 8k mit einem RAM 6264 (8k). Dies funktioniert aber nur beim ZEAT-System; bei diesem System wurde auch das FLOMON entsprechend geändert. Für dieses ZEAT-System muß der JMP1 in Stellung 2 stehen (Spannungsversorgung vom Akku trennen).

5.5 Allgemeine Erklärung der JMP

JMP16:

Dieser JMP dient dazu den Takt von 4 MHz auf 8 MHz umzustellen (siehe Abb.)



JMP10 und JMP11:

Mit diesen beiden JMP können Sie ein bis vier WAIT-Zyklen, getrennt für I/O-Zugriff und Speicherzugriff einstellbar. Die WAIT-Zyklen werden nur benötigt, wenn Sie mit der CPU Z80H arbeiten (siehe Punkt 5.6).

JMP14:

JMP14 dient dazu, daß die WAIT-Logik für Speicher grundsätzlich bei jedem Speicherzugriff oder nur bei Speicherzugriff der auf der Baugruppe befindlichen Speicher, gestartet wird. In Stellung 1 wird die WAIT-Logik bei jedem Speicherzugriff gestartet; in Stellung 2 nur bei Zugriff auf Baugruppeninterne Speicher. Verwenden Sie die CPU Z80A brauchen Sie normalerweise keine WAIT-Zyklen und deshalb ist dann dieser JMP für Sie nicht von Bedeutung.

JMP17:

Dieser JMP ist lediglich dazu da den Takt der CPU auf den BUS zu legen oder ihn vom BUS zu trennen. Interessant ist der JMP wenn Sie die Taktleitung des BUSSES mit dem Takt einer anderen Baugruppe belegen wollen. In diesem Fall müssen Sie JMP17 auftrenen.

JMP2, JMP3 und JMP4:

Diese drei Jumper sind für dazu da die verschiedenen Einsetzbaren Speichertypen "Pinkompatibel" zu machen. Verwenden Sie nur 8k EPROMs und 8k RAMs sind diese JMP voreingestellt und müssen nicht geändert werden. Verwenden Sie aber andere Speicher müssen diese JMP eingestellt werden. Siehe hierzu "Speicherkonfiguration und dazugehörige Jumperstellungen".

JMP6 und JMP7:

Diese beiden JMP dienen dazu den CS für die verschiedenen einsetzbaren EPROM-Typen einzustellen. Hier sind die Stellungen 32, 64, 128 und 256 für die EPROMs 2732, 2764, 27128 und 27256. Für das erste EPROM (J6) gilt JMP7, für das zweite EPROM (J7) gilt JMP9. Hier muß allerdings gesagt werden, daß immer nur zwei gleiche EPROM-Typen verwendet werden dürfen. Diese JMP sind für 8k EPROMs (2764) in Stellung "64" voreingestellt. Genaue Einstellung siehe unter "Speicherkonfigurationen und dazugehörige Jumperstellung".

JMP5, JMP6, JMP12, JMP13 und JMP15:

Diese 5 Jumper dienen dazu den CS für die verschiedenen einsetzbaren RAM-Typen einzustellen. Da sich der RAM Bereich beim System mit CP/M und beim System ohne CP/M ändert sind hierzu fünf Jumper nötig. JMP5 und JMP6 sind für 8k RAM-Bausteine (z.B. 6264) fest eingestellt. Die anderen drei Jumper sind variabel (siehe oben unter "Systemkonfiguration und dazugehörige Jumperstellung").

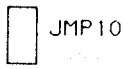
5.6 Betrieb mit der CPU Z80H mit 8 MHz

Wollen Sie die CPU Z80H einsetzen, müssen Sie folgende JMP ändern.

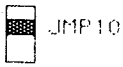
1. Der Takt-Jumper (JMP16) muß auf der Lötseite aufgekratzt werden, und dann mit Lötbrücke in Stellung 2 einstellen (siehe Abb.).

2. Sie müssen, wenn Sie langsame Speicher oder langsame I/O Einheiten haben, WAIT-Zyklen einfügen. Dies geschieht mit den JMP8 und JMP10 und zwar getrennt für I/O-Zugriffe und für Speicherzugriffe. Sollte also die GDP64k oder die FLO2 Schwierigkeiten machen empfiehlt es sich zwei oder drei WAIT-Zyklen für I/O-Zugriffe einzufügen. Ein WAIT-Zyklus bei I/O-Zugriffen ist nicht sinnvoll einzustellen, da die CPU bei jedem I/O-Zugriff automatisch einen WAIT-Zyklus einfügt. Einstellung WAIT für I/O-Zugriff an JMP8: siehe Abb..

Beispiel 1:
kein WAIT-Zyklus für Speicherzugriff
2 WAIT-Zyklen für I/O-Zugriff



Beispiel 2:
2 WAIT-Zyklen für Speicherzugriff
2 WAIT-Zyklen für I/O-Zugriff



Sind Ihre Speicher zu langsam (kann nur bei 8 MHz (Z80H) vorkommen) müssen sie an JMP1 ein oder zwei WAIT-Zyklen einstellen. Meistens genügt ein WAIT-Zyklus. Mehr als zwei WAIT-Zyklen werden so gut wie nie benötigt. Einstellung WAIT für Speicherzugriffe an JMP10 siehe Abb..

6. Fehlersuchanleitung

6.1Mögliche Fehler und ihre Behebung

6.1.1 Sind die bisher verwendeten Baugruppen in Ordnung?
(Funktioniert 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 richtg 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.

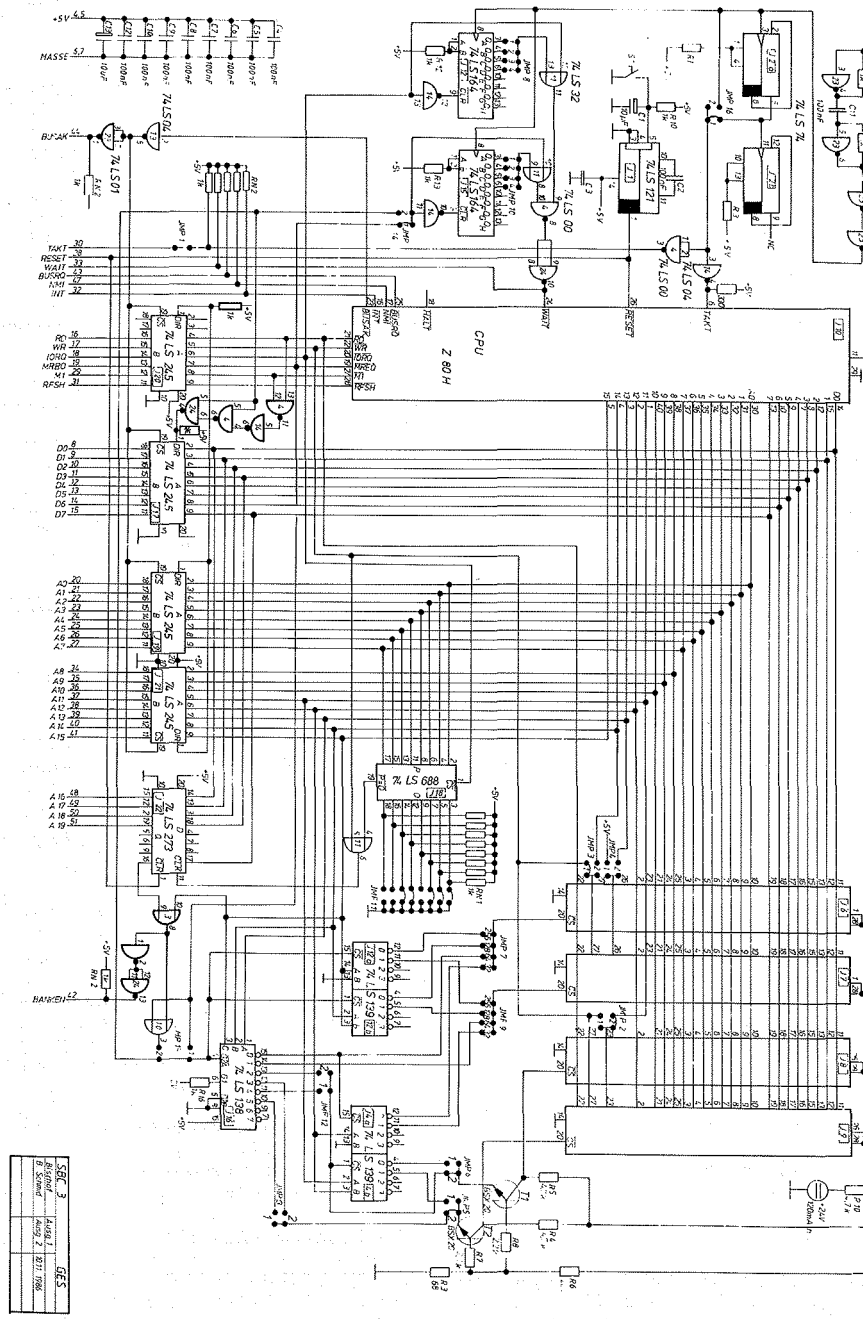
6.1.9 Nehmen Sie alle ICs aus ihren Fassungen. Nehmen Sie sich die Layouts zur Hand und kontrollieren Sie alle Leiterbahnen, mit einem Durchgangsprüfer oder mit einem Ohmmeter, auf Durchgang. Bereits kontrollierte Leiterbahnen können Sie, der Übersicht wegen, auf Layout mit Bleistift durchstreichen oder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

6.1.10 Prüfen sie die Versorgungsspannung mit einem Digitalvoltmeter (am Bus +5 Volt, nicht am Netzgerät, da am Kabel bei starker Belastung bis zu 0.5 Volt 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 dicker Litze (mind. 2 mm Quadrat) erfolgt ist. Gegebenenfalls müssen Sie Ihr Netzteil nachregeln. Vorsicht: Nie über 5.25V nachregeln!

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 Sie an den jeweiligen Ausgängen die richtigen Signale haben. Welche Signale wo anliegen müssen können Sie aus der Schaltungsbeschreibung, aus dem Schaltplan und Ihren eigenen Überlegungen folgern.
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. Schaltungsbeschreibung

7.1 Schaltplan



7.2 Beschreibung der Schaltung SBC3

Wie aus dem Blockschaltbild ersichtlich kann die Schaltung in Blöcke aufgespalten werden. In der folgenden Schaltungsbeschreibung wird jeder Block getrennt beschrieben.

7.2.1 Erzeugung des Taktes

Der Takt wird mit einem einfachen Prinzip erzeugt. Ein Inverter wird von seinem Ausgang auf den Eingang über einen 1k Widerstand zurückgeführt. Von diesen "Schwingern" sind zwei auf der Baugruppe, die mit einem 100 nF Kondensator (C11) miteinander verkoppelt sind. Der Quarz Q1 stabilisiert die Schwingung auf 8 MHz. Die Inverter (J23) geben dem Takt eine schöne Rechteckform. Die beiden D-Flip-Flop (J2) teilen die Taktfrequenz jeweils durch zwei. Am Ausgang des ersten Flip-Flops (J2/5) liegen 4 MHz und am Ausgang des zweiten (J2/9) 2 MHz (hier nicht beschaltet). Über JMP16 kann der Takt (4 oder 8 MHz) eingestellt werden. JMP16 ist hier voreingestellt auf 4 MHz.

7.2.2 RESET-Logik

Zum Rücksetzen der CPU in den Anfangszustand benötigt die CPU am RESET-Eingang einen negativen Impuls mit einer Mindestzeit von 3 Taktzyklen. Dieser negative Impuls wird mit dem Monoflop 74 121 (J1) erzeugt. Das Monoflop wird getriggert mit einem Taster. Um dabei ein prellfreies Triggersignal zu erhalten, wird die Triggerspannung über dem ELKO C3 abgegriffen. Dadurch steigt die Spannung am Elko C3 beim Loslassen des Tasters und beim Einschalten der Spannung langsam an und wird triggert bei einem best. Spannungswert das Monoflop. Bei diesem Triggereingang handelt es sich um einen Schmitt-Trigger Eingang, der die ansteigende Spannung sicher ab einem best. Spannungswert als HIGH erkennt. Ist das Monoflop getriggert, geht der Ausgang Q* auf LOW. Die Länge dieses Signals wird durch den Kondensator C1 festgelegt.

7.2.3 WAIT-Logik

Die WAIT-Logik wird nur benötigt wenn die 8 MHz CPU verwendet wird. Die WAIT-Zyklen können getrennt für I/O und Speicherzugriff eingestellt werden. Bei den WAIT-Zyklen für Speicherzugriff kann noch einmal eingeschränkt werden: Mit JMP 14 kann die WAIT-Logik für Speicherzugriff bei jedem Zugriff auf Speicher, oder aber nur bei Zugriff auf die Speicher auf der Baugruppe, gestartet werden. Dieser JMP14 ist so voreingestellt, daß bei jedem Speicherzugriff die WAIT-Logik gestartet wird.

Wird auf keinen Speicher zugegriffen, dann ist das MREQ*-Signal HIGH und das Schieberegister J15 wird gelöscht. Dadurch sind alle Ausgänge auf LOW. Wird dann auf einen Speicher zugegriffen, wird das Schieberegister nicht mehr gelöscht und schiebt ein HIGH Signale vom Eingang A und B mit jedem Taktsignal auf die parallelen Ausgänge QA bis QH. Bei jedem Taktimpuls wird ein Ausgang aktiviert, zuerst QA dann QB usw. Will man nun 2 WAIT-Zyklen einfügen, wird der Ausgang QB mit dem gemeinsamen Anschluß gebrückt. Dadurch erfährt das Signal eine Verzögerung von 2 Taktzyklen. Da dieses WAIT-Signal nur bei Speicherzugriff anliegen darf wird dieses Signal noch mit ODER verknüpft (J11). Die WAIT-Logik für I/O Zugriffe funktioniert im Prinzip genauso, nur wird hier zum Starten der WAIT-Logik das IORQ*-Signal des Prozessors verwendet. Die beiden WAIT-Signale werden mit AND verknüpft, (dies wird durch 2 NAND realisiert) d.h. sobald eines der beiden WAIT-Signale aktiviert (LOW) ist, wird der Ausgang des zweiten NAND (J24/10) und damit der Eingang WAIT LOW. Da noch das WAIT-Signal vom Bus auf diesen Eingang der CPU geht muß ein "Open Collector" NAND verwendet werden. Ein "Open Collektor" Gatter muß an seinem Ausgang aber mit einem 1k Widerstand auf +5V abgeschlossen werden. Einstellung der WAIT-Zyklen siehe unter 5.6.

7.2.4 BANK-Auswahl-Logik

Da der Z80 nur 64 kbyte adressieren kann und bei CP/M 2.2 64k RAM zur Verfügung stehen müssen, wird eine Logik benötigt, die mehr als 64 kbyte adressieren kann. Zu diesem Zweck werden mit Hilfe des Datenbusses die Adressen A16 bis A19 erzeugt, und auf einem LATCH gespeichert. Außerdem wird noch das Steuersignal "Bank" auf diesselbe Weise erzeugt und ebenfalls auf den Bus gelegt. Zur Definition der Signale "Bank" und "Banken": Das Signal "Bank" liegt am Latch (J22/16) an, und das Signal "Banken" führt auf den Bus (ST1/42). Die einzelnen Speicherkarten (z.B ROA64k oder RAM64/256) dekodieren die Adressen A16 bis A19 wenn das Signal "Banken" auf HIGH liegt und je nach dem welche Kombination (A16 bis A19) anliegt, wird eine von 16 64 kbyte Banken angewählt. Ist das Signal "Banken" LOW, so werden die Speicher auf der SBC3 ausgewählt.
Der Bank-Port auf dem die Adressen A16 bis A19 und das Signal "Bank" abgespeichert werden, wird über den Port C8H aktiviert. Die Adresse C8H ist an JMP11 voreingestellt. Der Baustein 74 LS 688 vergleicht die an JMP11 eingestellte Adresse mit der an den Adressen A0 bis A7 anliegenden Adressen und aktiviert, wenn der Eingang G (mit IORQ* beschaltet) LOW ist, den Ausgang P=Q* (J18/19). Dieses Signal wird noch mit WR* ODER verknüpft. Dadurch wird der Ausgang des ODER (J11/6) nur LOW, wenn auf den Port C8H geschrieben wird. Mit diesem Signal (J11/6) wird der CLK-Eingang des LATCHES 74 LS 273 gesteuert; d.h. wenn auf Port C8H geschrieben wird nimmt das LATCH Daten auf und gibt Sie an den Bus weiter. Die Daten (A16 bis A19 und das Signal "Bank") bleiben auf den Ausgängen gespeichert und können mit einem LOW-Signal am Eingang "CLEAR" (J21/1) gelöscht werden. Dieser Eingang ist mit RESET verbunden, d.h. bei RESET werden die Adressen A16 bis A19 und das Bank-Signal auf LOW gesetzt. Das Signal "Bank" (J21/12) wird noch negiert und mit der negierten Adresse A15 NAND verknüpft, was einer OR Verknüpfung von A15 und "Bank" gleichkommt. Das NAND wurde deshalb gewählt, weil das Signal "Banken" auf dem Bus auch von anderen Baugruppen gesteuert wird (COL256) und deshalb hier ein "Open Collektor" Gatter eingesetzt werden muß. Diese Bank-Auswahl-Logik wird nur bei CP/M benötigt und bei Betrieb ohne CP/M wird diese Logik mit dem JMP15 außer Kraft gesetzt.

7.2.5 Pufferung des Daten-, Adress- und Steuerbusses

Da die CPU auf der SBC3 sitzt und Daten, Adress- und Steuersignale erzeugt, die auf den Bus führen und von mehreren Baugruppen gelesen werden sollen, müssen diese Signale durch Treiber verstärkt werden. Diese Aufgabe übernehmen für den Daten-, Adress- und Steuerbus die bidirektionalen Tri-State Treiber 74 LS 245 (J17, J19, J20 und J21). Die Treiber für den Datenbus (J17) müssen die Daten in beide Richtungen weitergeben können. Daher wird der Eingang DIR (J17/1) mit dem RD* ,dem M1*-Signal und dem Aktivierungssignal für die internen Speicher (J16/4) gesteuert. Diese Verknüpfung der drei Signale hat zur Folge, daß bei internem Speicherzugriff und bei externem Speicherschreibzugriff die Daten auf den Bus gelegt werden. Nur bei Lesezugriffen auf externe Speicherbanken werden die Daten vom Bus eingelesen. Die Verknüpfung mit dem M1-Signal bewirkt einerseits ein schnelleres einlesen vom Bus bei einem M1-Zyklus und andererseits ist dadurch das Einlesen eines Interrupt-Vektors möglich.
Mit dem Eingang CS können die A Ein- bzw Ausgänge von den B Ein- bzw Ausgängen getrennt werden (siehe Datenblatt 74 LS 245). Der Datenbus soll vom Bus getrennt werden, wenn das Signal BUSAK LOW ist.

Die Steuersignale und die Adressen müssen nur in einer Richtung verkehren; deshalb kann der Eingang DIR bei diesen LATCHES (J19, J20 und J21) auf festes Potential (+5V) gelegt werden. Der Eingang CS wird mit dem negierten BUSAK Signal gesteuert, d.h. wenn das BUSAK Signal LOW (aktiv) ist, werden sämtliche Steuersignale, die Adressen A0 bis A15 und der Datenbus getrennt. Das Steuersignal BUSAK* ist LOW wenn eine andere Einheit (sonst. CPU etc.) auf den Bus zugreifen will.

7.2.6 Adressdekodierung

Die Adressdekodierung der Baugruppe ist sehr umfangreich, da 6 verschiedene Speichertypen einsetzbar sein sollen. Verwenden Sie die Speicherkonfiguration die von uns vorgeschlagen ist (8k EPROMs und 8k RAM) wird zur Speicherdekodierung nur noch der Dekoder 74 LS 138 benötigt. Dieser Dekoder hat die Aufgabe die einzelnen Speicherbausteine der Adresse nach anzuordnen. Am Eingang CS des Dekoders (J16/4) liegt entweder das MREQ*-Signal der CPU, beim System ohne CP/M, oder die ODER Verknüpfung des Banken-Signales mit dem MREQ*-Signal, beim System mit CP/M (JMP15 Stellung 2). Ist dieses "Veroderte" Signal LOW, so wird der Dekoder aktiviert und damit die Speicher auf der SBC3 angesprochen. Ist das Signal HIGH, werden externe Speicher (ROA64k oder RAM64/256) angesprochen und der Dekoder wird nicht aktiviert; d.h. die Speicher auf der SBC3 werden nicht angesprochen.

Nun zur eigentlichen Dekodierung: Die Eingänge A, B und C (J16/1/2/3) bestimmen, je nach der anliegenden binären Kombination welcher Ausgang aktiviert (LOW) wird. Diese Ausgänge sind jeweils mit dem CS-Eingang der Speicher verbunden. Bei den RAM-Bausteinen hängt noch ein Transistor dazwischen, der aber nur für die Akku-Pufferung benötigt wird. Wird nun auf einen Speicher einer bestimmten Adresse zugegriffen, so wird einer der Ausgänge aktiviert und damit einer der vier Speicher (J6, J7, J8 und J9). Bei Adresse 2000H bis 3FFFH wird z.B. A15 LOW, A14 LOW, A13 HIGH, damit wird der Ausgang 1 (J16/14) aktiviert und damit das 2. EPROM (J7). Da beim System mit CP/M und beim System ohne CP/M verschiedene Speicherkonfigurationen der RAM vorliegen, muß der CS Eingang der RAM Bausteine auf verschiedene Ausgänge der Dekoder gelegt werden. Dies geschieht mit JMP12 und JMP13. Sind diese beiden JMP auf Stellung 2, so sind die RAMs von Adresse 4000H bis 5FFFH und 6000H bis 7FFFH. Sind die JMP in Stellung 1, so liegen die RAMs von Adresse 8000H bis 9FFFH und A000H bis BFFFH.

Dekodierung von anderen Speichern
Bisher wurde nur über die Dekodierung von 8k-Speichern gesprochen. Werden aber andere Speicher (EPROMs: 2732, 27128, 27256; RAMs: 6116) verwendet, die mehr oder weniger Speicherkapazität als 8k Speicher haben, so muß die Dekodierung geändert werden. Zu diesem Zweck wurden die beiden ICs 74 LS 139 (J5 und J13) eingesetzt. In diesen beiden ICs sind 4 "zwei zu vier Dekoder" enthalten und zwar für jeden der obengenannten Speicher Typen wird ein solcher Dekoder verwendet. Auf diese vier Dekoder soll hier nicht näher eingegangen werden, da sie prinzipiell genauso aufgebaut ist, wie bei 8k Speichern; mehr zu diesen Speicherkonfigurationen JMP-Stellungen finden Sie unter 5.3 und folgende).

7.2.7 Akku-Pufferung der RAMs

RAMs sind bekanntlich flüchtige Speicher und haben beim Ausschalten der Versorgungsspannung die negative Eigenschaft ihre Daten zu verlieren. Dies geschieht hier auf der SBC3, dank der Akku-Pufferung, nicht. Beim Ausschalten des Computers versorgt ein Akku (2,4 V und 120 mAh) die RAM-Bausteine mit Strom. Dazu müssen die RAMs aber im "Standby-Mode" sein, d.h. die CS Eingänge der RAMs (J8/20 und J9/20) müssen HIGH sein. Trifft dies zu, sind die RAMs im "Standby-Mode" und benötigen in diesem Zustand mind. 2.0 V Spannung und ca. 0.5 - 2 uA Strom um die Speicherinformation zu halten. Um diesen Standby-Mode sicher zu erreichen, werden schnelle Schalttransistoren zwischen die Dekoderausgänge und den CS-Eingängen der Speicher geschaltet. Fällt nun die Spannung ab, (Ausschalten des Computer, Stromausfall etc.) fällt die Basisspannung ab und die Transistoren sperren. Nun liegt an den CS-Eingängen durch den Akku ein HIGH-Signal. Die beiden 4,7 kOhm Widerstände (R4 und R5) fallen nicht ins Gewicht, da der CS Eingang der Speicher extrem hochohmig

ist. Die Basisspannung der Transistoren wird mit R3 und R6 eingestellt. Die Widerstände R7 und R8 (2,2k) begrenzen den Basisstrom der Transistoren. Diese Begrenzung ist nötig, um die Dekoderausgänge nicht zu stark zu belasten. Die Diode D1 sorgt dafür, daß der Akku bei Spannungsabfall nicht den ganzen Computer mit Strom versorgt. Der Widerstand R9 dient dazu, den Ladestrom für den Akku einzustellen. Durch den eingesetzten Widerstand von 4,7 kOhm beträgt der Ladestrom ca. 1 mA. Der Akku hält die Speicherinformation ohne Nachladen bei Zimmertemperatur ca. 1 Jahr. Da der Computer aber sicherlich ab und zu läuft wird der Akku automatisch nachgeladen. Um den Akku voll aufzuladen, benötigt man bei einem eingestellten Ladestrom von 1 mA (R9 = 4,7 kOhm) ca. 120 Stunden. Der Akku kann also normalerweise nicht leer werden. Verwenden Sie 2k RAMs müssen Sie darauf achten, daß Sie CMOS RAMs benutzen, andernfalls ist der Akku innerhalb einiger Stunden leer. Verwenden Sie die mitgelieferten 8k RAMs oder andere 8k RAMs brauchen Sie sich diesbezüglich keine Sorgen machen, da es nur CMOS 8k RAMs gibt.

7.2.8 Speicher

Wie unter 5.3 und folgende schon erläutert sind 6 verschiedene Speichertypen möglich. Die Baugruppe hat eine Grundeinstellung, die 8k Speicher vorsieht (EPROM 2764 und RAM 6264). Die vier verschiedenen einsetzbaren EPROM-Typen sind bis auf Pin 26 und 27 Pinkompatibel. Um Pinkompatibilität zu erreichen werden diese beiden Pins über JMP4 und JMP3 so eingestellt, daß die jeweiligen Speicher gesteckt werden können. Die beiden RAM-Typen sind bis auf Pin 23 Pinkompatibel. Hier wird mit JMP2 die verschiedene Pinbelegung ausgeglichen. Mögliche Speicherkonfigurationen siehe unter 5.3 und folgende.

7.2.9 Die CPU (Central Processing Unit) Z80

Siehe Datenblatt CPU Z80 (unter 10. Bauelemente)

8. Anwendungsbeispiele

8.1 Einsatz im Einsteigerpaket (siehe unter 5.3.1)
8.2 Einsatz im System mit Bildschirm und Tastatur ohne CP/M (siehe unter 5.3.1)
8.3 Professionelles System mit CP/M (siehe unter 5.3.2)
8.4 CP/M-System mit Christiani ZEAT (siehe unter 5.3.3)

9. Ausblick, Diverses

Falls irgendwelche Änderungen auf dieser Baugruppe auftreten, erfahren Sie diese durch die Zeitschrift LOOP (siehe unter Punkt 11).

Bitte senden Sie uns die dem Bausatz oder Fertiggerät beiliegende Kritikkarte ausgefüllt zurück, denn nur so können wir Fehler oder Mißverständnisse irgendwelcher Art erkennen und verbessern! Vielen Dank schon im Voraus für Ihre Hilfe!

Z8400
Z80® CPU Central Processing Unit

Zilog

Product Specification

April 1985

FEATURES

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Eight MHz, 6 MHz, 4 MHz, and 2.5 MHz clocks for the Z80H, Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers, together with indexed and relative addressing, result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 similar, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

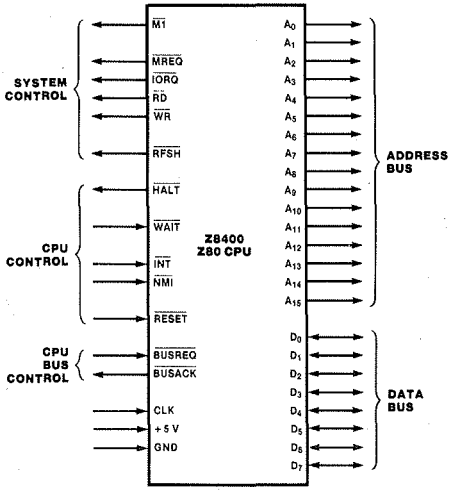


Figure 1. Pin Functions

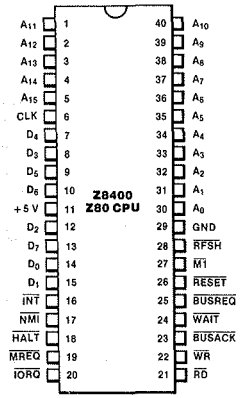


Figure 2a. 40-Pin Dual-In-Line Package (DIP) Pin Assignments

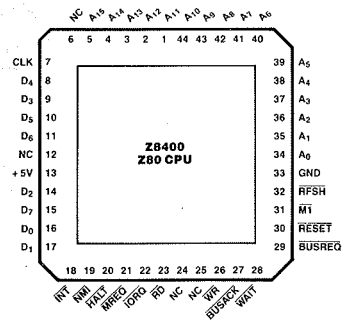


Figure 2b. 44-Pin Chip Carrier Pin Assignments

GENERAL DESCRIPTION

The Z80, Z80A, Z80B, and Z80H CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single + 5V power source. All output signals are fully decoded and timed to control standard memory or peripheral circuits; the CPU is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

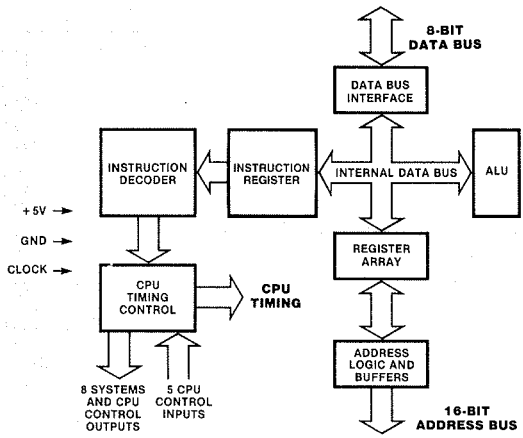


Figure 3. Z80 CPU Block Diagram

Z80 MICROPROCESSOR FAMILY

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU REGISTERS

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

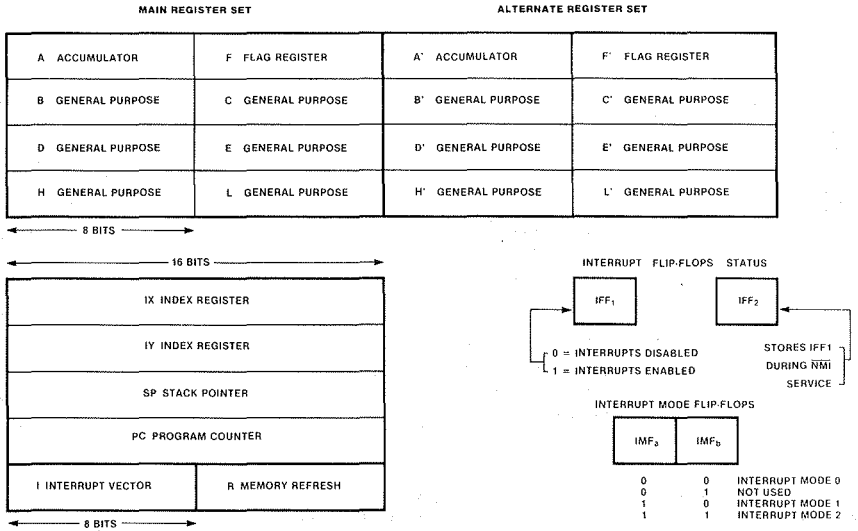


Figure 4. CPU Registers

Z80 CPU REGISTERS (Continued)

Table 1. Z80 CPU Registers

Register		Size (Bits)	Remarks
A, A′	Accumulator	8	Stores an operand or the results of an operation.
F, F′	Flags	8	See Instruction Set.
B, B′	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C′	General Purpose	8	See B, above.
D, D′	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E′	General Purpose	8	See D, above.
H, H′	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L′	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Used for indexed addressing
SP	Stack Pointer	16	Holds address of the top of the stack. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

INTERRUPTS: GENERAL OPERATION

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The $\overline{NMI}$ is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt and it requires that interrupts be enabled in software in order to operate. $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — similar to the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Non-Maskable Interrupt ($\overline{NMI}$). The nonmaskable interrupt cannot be disabled by program control and therefore will be accepted at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the

interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request.

Mode 0 Interrupt Operation. This mode is similar to the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus. This is normally a Restart instruction, which will initiate a call to the selected one of eight restart locations in page zero of memory. Unlike the 8080, the Z80 CPU responds to the Call instruction with only one interrupt acknowledge cycle followed by two memory read cycles.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has only one restart location, 0038H.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit vector on the data bus during the interrupt acknowledge cycle. The CPU forms a pointer using this byte as the lower 8 bits and the contents of the I register as the upper 8 bits. This points to an entry in a table of addresses for interrupt service routines. The CPU then jumps to the routine at that address. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwired to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description, are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* (03-0029-01) and *Z80 Assembly Language Programming Manual* (03-0002-01).

Table 2. State of Flip-Flops

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt INT disabled
DI instruction execution	0	0	Maskable interrupt INT disabled
EI instruction execution	1	1	Maskable interrupt INT enabled
LD A,I instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
LD A,R instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	IFF_1	$IFF_1 \rightarrow IFF_2$ (Maskable interrupt INT disabled)
RETN instruction execution	IFF_2	•	$IFF_2 \rightarrow IFF_1$ at completion of an NMI service routine.

INSTRUCTION SET

The Z80 microprocessr has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory, or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set which shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. For an explanation of flag notations and symbols for mnemonic tables, see the Symbolic Notations section which follows these tables. The *Z80 CPU Technical Manual* (03-0029-01), the *Programmer's Reference Guide* (03-0012-03), and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- 8-bit loads
- 16-bit loads
- Exchanges, block transfers, and searches
- 8-bit arithmetic and logic operations
- General-purpose arithmetic and CPU control
- 16-bit arithmetic operations
- Rotates and shifts
- Bit set, reset, and test operations
- Jumps
- Calls, returns, and restarts
- Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- Immediate
- Immediate extended
- Modified page zero
- Relative
- Extended
- Indexed
- Register
- Register indirect
- Implied
- Bit

8-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments	
LD r, r'	r ← r'	•	•	X	X	•	•	01	r	r'		1	1	4	r, r'	Reg.
LD r, n	r ← n	•	•	X	X	•	•	00	r	110		2	2	7	000	B
									← n →						001	C
LD r, (HL)	r ← (HL)	•	•	X	X	•	•	01	r	110		1	2	7	010	D
LD r, (IX + d)	r ← (IX + d)	•	•	X	X	•	•	11	011	101	DD	3	5	19	011	E
								01	r	110					100	H
									← d →						101	L
LD r, (IY + d)	r ← (IY + d)	•	•	X	X	•	•	11	111	101	FD	3	5	19	111	A
								01	r	110						
									← d →							
LD (HL), r	(HL) ← r	•	•	X	X	•	•	01	110	r		1	2	7		
LD (IX + d), r	(IX + d) ← r	•	•	X	X	•	•	11	011	101	DD	3	5	19		
								01	110	r						
									← d →							
LD (IY + d), r	(IY + d) ← r	•	•	X	X	•	•	11	111	101	FD	3	5	19		
								01	110	r						
									← d →							
LD (HL), n	(HL) ← n	•	•	X	X	•	•	00	110	110	36	2	3	10		
									← n →							
LD (IX + d), n	(IX + d) ← n	•	•	X	X	•	•	11	011	101	DD	4	5	19		
								00	110	110	36					
									← d →							
									← n →							

8-BIT LOAD GROUP (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD (IY + d), n	(IY + d) ← n	•	•	X	X	•	•	11	111	101	FD	4	5	19	
								00	110	110	36				
									← d →						
									← n →						
LD A, (BC)	A ← (BC)	•	•	X	X	•	•	00	001	010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	X	•	•	00	011	010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	X	•	•	00	111	010	3A	3	4	13	
									← n →						
									← n →						
LD (BC), A	(BC) ← A	•	•	X	X	•	•	00	000	010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	X	•	•	00	010	010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	X	•	•	00	110	010	32	3	4	13	
									← n →						
									← n →						
LD A, I	A ← I	‡	‡	X	0	X	IFF	0	•	11	101	101	ED	2	2	9	
										01	010	111	57				
LD A, R	A ← R	‡	‡	X	0	X	IFF	0	•	11	101	101	ED	2	2	9	
										01	011	111	5F				
LD I, A	I ← A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9	
										01	000	111	47				
LD R, A	R ← A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9	
										01	001	111	4F				

NOTE: IFF, the content of the interrupt enable flip-flop, (IFF_2), is copied into the P/V flag.

16-BIT LOAD GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	X	•	•	00	dd0	001		3	3	10	dd Pair
									← n →						00 BC
									← n →						01 DE
LD IX, nn	IX ← nn	•	•	X	X	•	•	11	011	101	DD	4	4	14	10 HL
								00	100	001	21				11 SP
									← n →						
									← n →						
LD IY, nn	IY ← nn	•	•	X	X	•	•	11	111	101	FD	4	4	14	
								00	100	001	21				
									← n →						
									← n →						
LD HL, (nn)	H ← (nn + 1)	•	•	X	X	•	•	00	101	010	2A	3	5	16	
	L ← (nn)								← n →						
									← n →						
LD dd, (nn)	dd_H ← (nn + 1)	•	•	X	X	•	•	11	101	101	ED	4	6	20	
	dd_L ← (nn)							01	dd1	011					
									← n →						
									← n →						

NOTE: $(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively. e.g., BC_L = C, AF_H = A.

16-BIT LOAD GROUP (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD IX, (nn)	$IX_H \leftarrow (nn+1)$ $IX_L \leftarrow (nn)$	•	•	X	•	X	•	•	•	11 00	011 101 ←n→ ←n→	101 010	DD 2A	4	6	20	
LD IY, (nn)	$IY_H \leftarrow (nn+1)$ $IY_L \leftarrow (nn)$	•	•	X	•	X	•	•	•	11 00	111 101 ←n→ ←n→	101 010	FD 2A	4	6	20	
LD (nn), HL	$(nn+1) \leftarrow H$ $(nn) \leftarrow L$	•	•	X	•	X	•	•	•	00	100 ←n→ ←n→	010	22	3	5	16	
LD (nn), dd	$(nn+1) \leftarrow dd_H$ $(nn) \leftarrow dd_L$	•	•	X	•	X	•	•	•	11 01	101 dd0 ←n→ ←n→	101 011	ED	4	6	20	
LD (nn), IX	$(nn+1) \leftarrow IX_H$ $(nn) \leftarrow IX_L$	•	•	X	•	X	•	•	•	11 00	011 100 ←n→ ←n→	101 010	DD 22	4	6	20	
LD (nn), IY	$(nn+1) \leftarrow IY_H$ $(nn) \leftarrow IY_L$	•	•	X	•	X	•	•	•	11 00	111 100 ←n→ ←n→	101 010	FD 22	4	6	20	
LD SP, HL	$SP \leftarrow HL$	•	•	X	•	X	•	•	•	11	111	001	F9	1	1	6	
LD SP, IX	$SP \leftarrow IX$	•	•	X	•	X	•	•	•	11 11	011 111	101 001	DD F9	2	2	10	
LD SP, IY	$SP \leftarrow IY$	•	•	X	•	X	•	•	•	11 11	111 111	101 001	FD F9	2	2	10	
PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$ $SP \rightarrow SP-2$	•	•	X	•	X	•	•	•	11	qq0	101		1	3	11	
PUSH IX	$(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$ $SP \rightarrow SP-2$	•	•	X	•	X	•	•	•	11 11	011 100	101 101	DD E5	2	4	15	
PUSH IY	$(SP-2) \leftarrow IY_L$ $(SP-1) \leftarrow IY_H$ $SP \rightarrow SP-2$	•	•	X	•	X	•	•	•	11 11	111 100	101 101	FD E5	2	4	15	
POP qq	$qq_H \leftarrow (SP+1)$ $qqL \leftarrow (SP)$ $SP \rightarrow SP+2$	•	•	X	•	X	•	•	•	11	qq0	001		1	3	10	
POP IX	$IX_H \leftarrow (SP+1)$ $IX_L \leftarrow (SP)$ $SP \rightarrow SP+2$	•	•	X	•	X	•	•	•	11 11	011 100	101 001	DD E1	2	4	14	
POP IY	$IY_H \leftarrow (SP+1)$ $IY_L \leftarrow (SP)$ $SP \rightarrow SP+2$	•	•	X	•	X	•	•	•	11 11	111 100	101 001	FD E1	2	4	14	

qq	Pair
00	BC
01	DE
10	HL
11	AF

NOTE: $(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively. e.g., BC_L = C, AF_H = A.

EXCHANGE, BLOCK TRANSFER, BLOCK SEARCH GROUPS

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11	101	011	EB	1	1	4	
EX AF, AF′	AF ↔ AF′	•	•	X	•	X	•	•	•	00	001	000	08	1	1	4	
EXX	BC ↔ BC′ DE ↔ DE′ HL ↔ HL′	•	•	X	•	X	•	•	•	11	011	001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	•	X	•	•	•	11	100	011	E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 11	011 100	101 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IYL ↔ (SP)	•	•	X	•	X	•	•	•	11 11	111 100	101 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	•	•	X	0	X	↕ ①	0	•	11 10	101 100	101 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	0 ②	0	•	11 10	101 110	101 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X	0	X	↕ ①	0	•	11 10	101 101	101 000	ED A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	0 ②	0	•	11 10	101 111	101 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	↕ ③	X	↕	X	↕ ①	1	•	11 10	101 100	101 001	ED A1	2	4	16	

NOTE: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
② P/V flag is 0 only at completion of instruction.
③ Z flag is 1 if A = HL, otherwise Z = 0.

EXCHANGE, BLOCK TRANSFER, BLOCK SEARCH GROUPS (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
			③		①										
CPIR	A − (HL)	‡	‡	X	X	1	•	11	101	101	ED	2	5	21	If BC ≠ 0 and A ≠ (HL)
	HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0							10	110	001	B1	2	4	16	If BC = 0 or A = (HL)
			③		①										
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	‡	‡	X	X	1	•	11 10	101 101	101 001	ED A9	2	4	16	
			③		①										
CPDR	A − (HL)	‡	‡	X	X	1	•	11	101	101	ED	2	5	21	If BC ≠ 0 and A ≠ (HL)
	HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0							10	111	001	B9	2	4	16	If BC = 0 or A = (HL)

NOTE: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
② P/V flag is 0 only at completion of instruction.
③ Z flag is 1 if A = (HL), otherwise Z = 0.

8-BIT ARITHMETIC AND LOGICAL GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	‡	‡	X	‡	V	0	‡	10	[000]	r	1	1	4	r Reg.
ADD A, n	A ← A + n	‡	‡	X	‡	V	0	‡	11	[000]	110	2	2	7	000 B
										← n →					001 C
															010 D
ADD A, (HL)	A ← A + (HL)	‡	‡	X	‡	V	0	‡	10	[000]	110	1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	‡	‡	X	‡	V	0	‡	11	011	101	DD	3	5	19	100 H
									10	[000]	110					101 L
										← d →						111 A
ADD A, (IY + d)	A ← A + (IY + d)	‡	‡	X	‡	V	0	‡	11	111	101	FD	3	5	19	
									10	[000]	110					
										← d →						
ADC A, s	A ← A + s + CY	‡	‡	X	‡	V	0	‡		[001]						s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A − s	‡	‡	X	‡	V	1	‡		[010]						
SBC A, s	A ← A − s − CY	‡	‡	X	‡	V	1	‡		[011]						
AND s	A ← A ∧ s	‡	‡	X	1	X	P	0	0	[100]						
OR s	A ← A ∨ s	‡	‡	X	0	X	P	0	0	[110]						
XOR s	A ← A ⊕ s	‡	‡	X	0	X	P	0	0	[101]						
CP s	A − s	‡	‡	X	‡	X	V	1	‡	[111]						

8-BIT ARITHMETIC AND LOGICAL GROUP (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
INC r	r ← r + 1	‡	‡	X	‡	X	V	0	•	00	r	100		1	1	4	
INC (HL)	(HL) ←																
	(HL) + 1	‡	‡	X	‡	X	V	0	•	00	110	100		1	3	11	
INC (IX + d)	(IX + d) ←	‡	‡	X	‡	X	V	0	•	11	011	101	DD	3	6	23	
	(IX + d) + 1									00	110	100					
											← d →						
INC (IY + d)	(IY + d) ←	‡	‡	X	‡	X	V	0	•	11	111	101	FD	3	6	23	
	(IY + d) + 1									00	110	100					
											← d →						
DEC m	m ← m − 1	‡	‡	X	‡	X	V	1	•			101					

NOTE: m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace 100 with 101 in opcode.

GENERAL-PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

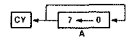
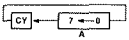
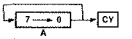
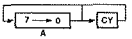
Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	@	‡	‡	X	‡	X	P	•	‡	00	100	111	27	1	1	4	Decimal adjust accumulator.
CPL	A ← A	•	•	X	1	X	•	1	•	00	101	111	2F	1	1	4	Complement accumulator (one's complement).
NEG	A ← 0 − A	‡	‡	X	‡	X	V	1	‡	11	101	101	ED	2	2	8	Negate acc. (two's complement).
										01	000	100	44				
CCF	CY ← CY	•	•	X	X	X	•	0	‡	00	111	111	3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00	110	111	37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00	000	000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01	110	110	76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11	110	011	F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11	111	011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
										01	000	110	46				
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
										01	010	110	56				
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
										01	011	110	5E				

NOTES: @ converts accumulator content into packed BCD following add or subtract with packed BCD operands.
IFF indicates the interrupt enable flip-flop.
CY indicates the carry flip-flop.
★ indicates interrupts are not sampled at the end of EI or DI.

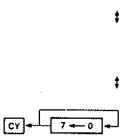
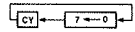
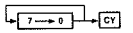
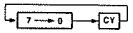
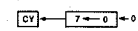
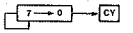
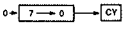
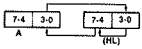
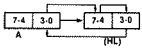
16-BIT ARITHMETIC GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	‡	00	ssl	001		1	3	11	ss Reg.
																	00 BC
ADC HL, ss	HL ←																01 DE
	HL + ss + CY	‡	‡	X	X	X	V	0	‡	11	101	101	ED	2	4	15	10 HL
										01	ss1	010					11 SP
SBC HL, ss	HL ←																
	HL − ss − CY	‡	‡	X	X	X	V	1	‡	11	101	101	ED	2	4	15	
										01	ss0	010					
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	‡	11	011	101	DD	2	4	15	pp Reg.
										01	pp1	001					00 BC
																	01 DE
																	10 IX
																	11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	‡	11	111	101	FD	2	4	15	rr Reg.
										00	rr1	001					00 BC
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00	ss0	011		1	1	6	01 DE
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	10	10 IY
										00	100	011	23				11 SP
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	10	
										00	100	011	23				
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00	ss1	011		1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	10	
										00	101	011	2B				
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	10	
										00	101	011	2B				

ROTATE AND SHIFT GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	‡	00	000	111	07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	‡	00	010	111	17	1	1	4	Rotate left accumulator.
RRCA		•	•	X	0	X	•	0	‡	00	001	111	0F	1	1	4	Rotate right circular accumulator.
RRA		•	•	X	0	X	•	0	‡	00	011	111	1F	1	1	4	Rotate right accumulator.

ROTATE AND SHIFT GROUP (Continued)

Symbolic Mnemonic	Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLC r		↕	↕	X	0	X	P/V	0	•	↕	11	001	011	CB	2	2	8	Rotate left circular register r.
											00	000	r					
RLC (HL)		↕	↕	X	0	X	P/V	0		↕	11	001	011	CB	2	4	15	
											00	000	110					
RLC (IX + d)		↕	↕	X	0	X	P/V	0		↕	11	011	101	DD	4	6	23	
	r,(HL),(IX + d),(IY + d)										11	001	011	CB				
												← d →						
											00	000	110					
RLC (IY + d)		↕	↕	X	0	X	P/V	0		↕	11	111	101	FD	4	6	23	
											11	001	011	CB				
												← d →						
											00	000	110					
RL m		↕	↕	X	0	X	P/V	0		↕		010						
	m = r,(HL),(IX + d),(IY + d)																	
RRC m		↕	↕	X	0	X	P/V	0		↕		001						
	m = r,(HL),(IX + d),(IY + d)																	
RR m		↕	↕	X	0	X	P/V	0		↕		011						
	m = r,(HL),(IX + d),(IY + d)																	
SLA m		↕	↕	X	0	X	P/V	0		↕		100						
	m = r,(HL),(IX + d),(IY + d)																	
SRA m		↕	↕	X	0	X	P/V	0		↕		101						
	m = r,(HL),(IX + d),(IY + d)																	
SRL m		↕	↕	X	0	X	P/V	0		↕		111						
	m = r,(HL),(IX + d),(IY + d)																	
RLD		↕	↕	X	0	X	P/V	0		•	11	101	101	ED	2	5	18	Rotate digit left and right between the accumulator and location (HL).
											01	101	111	6F				
RRD		↕	↕	X	0	X	P/V	0		•	11	101	101	ED	2	5	18	The content of the upper half of the accumulator is unaffected.
											01	100	111	67				

BIT SET, RESET AND TEST GROUP

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow r_b$	X	‡	X	1	X	X	0	•	11	001	011	CB	2	2	8	r Reg.
										01	b	r					000 B
BIT b, (HL)	$Z \leftarrow (HL)_b$	X	‡	X	1	X	X	0	•	11	001	011	CB	2	3	12	001 C
										01	b	110					010 D
BIT b,(IX + d)$_b$	$Z \leftarrow (IX+d)_b$	X	‡	X	1	X	X	0	•	11	011	101	DD	4	5	20	011 E
										11	001	011	CB				100 H
											← d →						101 L
										01	b	110					111 A
BIT b, (IY + d)$_b$	$Z \leftarrow (IY+d)_b$	X	‡	X	1	X	X	0	•	11	111	101	FD	4	5	20	b Bit Tested
										11	001	011	CB				000 0
											← d →						001 1
										01	b	110					010 2
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11	001	011	CB	2	2	8	011 3
										11	b	r					100 4
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11	001	011	CB	2	4	15	101 5
										11	b	110					110 6
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11	011	101	DD	4	6	23	111 7
										11	001	011	CB				
											← d →						
										11	b	110					
SET b, (IY + d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11	111	101	FD	4	6	23	
										11	001	011	CB				
											← d →						
										11	b	110					
RES b, m	$m_b \leftarrow 0$ m≡r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	10							To form new opcode replace 11 of SET b, s with 10. Flags and time states for SET instruction.

NOTE: The notation m_b indicates location m, bit b (0 to 7).

JUMP GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11	000	011	C3	3	3	10	cc Condition
										← n →							000 NZ (non-zero)
										← n →							001 Z (zero)
JP cc, nn	If condition cc	•	•	X	•	X	•	•	•	11	cc	010		3	3	10	010 NC (non-carry)
	is true PC←nn,									← n →							011 C (carry)
	otherwise									← n →							100 PO (parity odd)
	continue																101 PE (parity even)
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00	011	000	18	2	3	12	110 P (sign positive)
										← e − 2 →							111 M (sign negative)
JR C, e	If C = 0,	•	•	X	•	X	•	•	•	00	111	000	38	2	2	7	If condition not met.
	continue									← e − 2 →							
	If C = 1,													2	3	12	If condition is met.
	PC ← PC + e																
JR NC, e	IF C = 1,	•	•	X	•	X	•	•	•	00	110	000	30	2	2	7	If condition not met.
	continue									← e − 2 →							
	If C = 0,													2	3	12	If condition is met.
	PC ← PC + e																
JP Z, e	If Z = 0	•	•	X	•	X	•	•	•	00	101	000	28	2	2	7	If condition not met.
	continue									← e − 2 →							
	If Z = 1,													2	3	12	If condition is met.
	PC ← PC + e																
JR NZ, e	If Z = 1,	•	•	X	•	X	•	•	•	00	100	000	20	2	2	7	If condition not met.
	continue									← e − 2 →							
	If Z = 0,													2	3	12	If condition is met.
	PC ← PC + e																
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11	101	001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	8	
										11	101	001	E9				
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	8	
										11	101	001	E9				
DJNZ, e	B ← B − 1	•	•	X	•	X	•	•	•	00	010	000	10	2	2	8	If B = 0
	If B = 0,									← e − 2 →							
	continue																
	If B≠0,													2	3	13	If B≠0,
	PC ← PC + e																

NOTES: e represents the extension in the relative addressing mode.
e is a signal two's complement number in the range < − 126, 129 >.
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

CALL AND RETURN GROUP

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC \leftarrow nn$,	•	•	X	•	X	•	•	•	11	001 ← n → ← n →	101	CD	3	5	17	
CALL cc,nn	If condition cc is false	•	•	X	•	X	•	•	•	11	cc ← n →	100		3	3	10	If cc is false.
	continue, otherwise same as CALL nn										← n →			3	5	17	If cc is true.
RET	$PC_L \leftarrow (SP)$ $PC_H\leftarrow(SP+1)$	•	•	X	•	X	•	•	•	11	001	001	C9	1	3	10	
RET cc	If condition cc is false	•	•	X	•	X	•	•	•	11	cc	000		1	1	5	If cc is false.
	continue, otherwise same as RET													1	3	11	If cc is true.
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 01	101 001	101 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 01	101 000	101 101	ED 45	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	•	X	•	•	•	11	t	111		1	3	11	

cc	Condition
000	NZ (non-zero)
001	Z (zero)
010	NC (non-carry)
011	C (carry)
100	PO (parity odd)
101	PE (parity even)
110	P (sign positive)
111	M (sign negative)

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

NOTE: [1]RETN loads $IFF_2 \rightarrow IFF_1$

INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Opcode 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	•	•	11	011	01	DB	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
									← n →						
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	‡	‡	X	‡	P	0	•	11 01	101 r	101 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			①												
INI	(HL) ← (C) B ← B – 1 HL ← HL + 1	X	‡	X	X	X	1	X	11 10	101 100	101 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			②												
INIR	(HL) ← (C) B ← B – 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	1	X	11 10	101 110	101 010	ED B2	2 2	5 (If B≠0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			①												
IND	(HL) ← (C) B ← B – 1 HL ← HL – 1	X	‡	X	X	X	1	X	11 10	101 101	101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			②												
INDR	(HL) ← (C) B ← B – 1 HL ← HL – 1 Repeat until B = 0	X	1	X	X	X	1	X	11 10	101 111	101 010	ED BA	2 2	5 (If B≠0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11	010	011	D3	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
									← n →						
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 01	101 r	101 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			①												
OUTI	(C) ← (HL) B ← B – 1 HL ← HL + 1	X	‡	X	X	X	1	X	11 10	101 100	101 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			②												
OTIR	(C) ← (HL) B ← B – 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	1	X	11 10	101 110	101 011	ED B3	2 2	5 (If B≠0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			①												
OUTD	(C) ← (HL) B ← B – 1 HL ← HL – 1	X	‡	X	X	X	1	X	11 10	101 101	101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
			②												
OTDR	(C) ← (HL) B ← B – 1 HL ← HL – 1 Repeat until B = 0	X	1	X	X	X	1	X	11 10	101 111	101 011	ED	2 2	5 (If B≠0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

NOTES: ① If the result of B – 1 is zero, the Z flag is set; otherwise it is reset.
② Z flag is set upon instruction completion only.

SUMMARY OF FLAG OPERATION

Instructions	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	‡	‡	X	‡	X	V	0	‡	8-bit add or add with carry.
SUB s; SBC A, s; CP s; NEG	‡	‡	X	‡	X	V	1	‡	8-bit subtract, subtract with carry, compare and negate accumulator.
AND s	‡	‡	X	1	X	P	0	0	Logical operation.
OR s, XOR s	‡	‡	X	0	X	P	0	0	Logical operation.
INC s	‡	‡	X	‡	X	V	0	•	8-bit increment.
DEC s	‡	‡	X	‡	X	V	1	•	8-bit decrement.
ADD DD, ss	•	•	X	X	X	•	0	‡	16-bit add.
ADC HL, ss	‡	‡	X	X	X	V	0	‡	16-bit add with carry.
SBC HL, ss	‡	‡	X	X	X	V	1	‡	16-bit subtract with carry.
RLA; RLCA; RRA; RRCA	•	•	X	0	X	•	0	‡	Rotate accumulator.
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	‡	‡	X	0	X	P	0	‡	Rotate and shift locations.
RLD; RRD	‡	‡	X	0	X	P	0	•	Rotate digit left and right.
DAA	‡	‡	X	‡	X	P	•	‡	Decimal adjust accumulator.
CPL	•	•	X	1	X	•	1	•	Complement accumulator.
SCF	•	•	X	0	X	•	0	1	Set carry.
CCF	•	•	X	X	X	•	0	‡	Complement carry.
IN r (C)	‡	‡	X	0	X	P	0	•	Input register indirect.
INI; IND; OUTI; OUTD	X	‡	X	X	X	X	1	•	Block input and output. Z = 1 if B ≠ 0, otherwise Z = 0.
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	•	Block input and output. Z = 1 if B ≠ 0, otherwise Z = 0.
LDI; LDD	X	X	X	0	X	‡	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR; LDDR	X	X	X	0	X	0	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
CPI; CPIR; CPD; CPDR	X	‡	X	X	X	‡	1	•	Block search instructions. Z = 1 if A = (HL), otherwise Z = 0. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LD A; I, LD A, R	‡	‡	X	0	X	IFF	0	•	IFF, the content of the interrupt enable flip-flop, (IFF_2), is copied into the P/V flag.
BIT b, s	X	‡	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag.

SYMBOLIC NOTATION

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity: P/V = 1 if the result of the operation is even; P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow. If P/V does not hold overflow, P/V = 0.
H*	Half-carry flag. H = 1 if the add or subtract operation produced a carry into, or borrow from, bit 4 of the accumulator.
N*	Add/Subtract flag. N = 1 if the previous operation was a subtract.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
‡	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is indeterminate.
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one o the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

*H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.

PIN DESCRIPTIONS

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

BUSACK. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR have entered their high-impedance states. The external circuitry can now control these lines.

BUSREQ. *Bus Request* (input, active Low). Bus Request has a higher priority than NMI and is always recognized at the end of the current machine cycle. BUSREQ forces the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR to go to a high-impedance state so that other devices can control these lines. BUSREQ is normally wired-OR and requires an external pullup for these applications. Extended BUSREQ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

Halt. *Halt State* (output, active Low). HALT indicates that the CPU has executed a Halt instruction and is awaiting either a nonmaskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

INT. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. INT is normally wired-OR and requires an external pullup for these applications.

IORQ. *Input/Output Request* (output, active Low, 3-state). IORQ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. IORQ is also generated concurrently with M1 during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

M1. *Machine Cycle One* (output, active Low). M1, together with MREQ, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. M1, together with IORQ, indicates an interrupt acknowledge cycle.

MREQ. *Memory Request* (output, active Low, 3-state). MREQ indicates that the address bus holds a valid address for a memory read or memory write operation.

NMI. *Non-Maskable Interrupt* (input, negative edge-triggered). NMI has a higher priority than INT. NMI is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

RD. *Read* (output, active Low, 3-state). RD indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

RESET. *Reset* (input, active Low). RESET initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that RESET must be active for a minimum of three full clock cycles before the reset operation is complete.

RFSH. *Refresh* (output, active Low). RFSH, together with MREQ, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

WAIT. *Wait* (input, active Low). WAIT indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended WAIT periods can prevent the CPU from refreshing dynamic memory properly.

WR. *Write* (output, active Low, 3-state). WR indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CPU TIMING

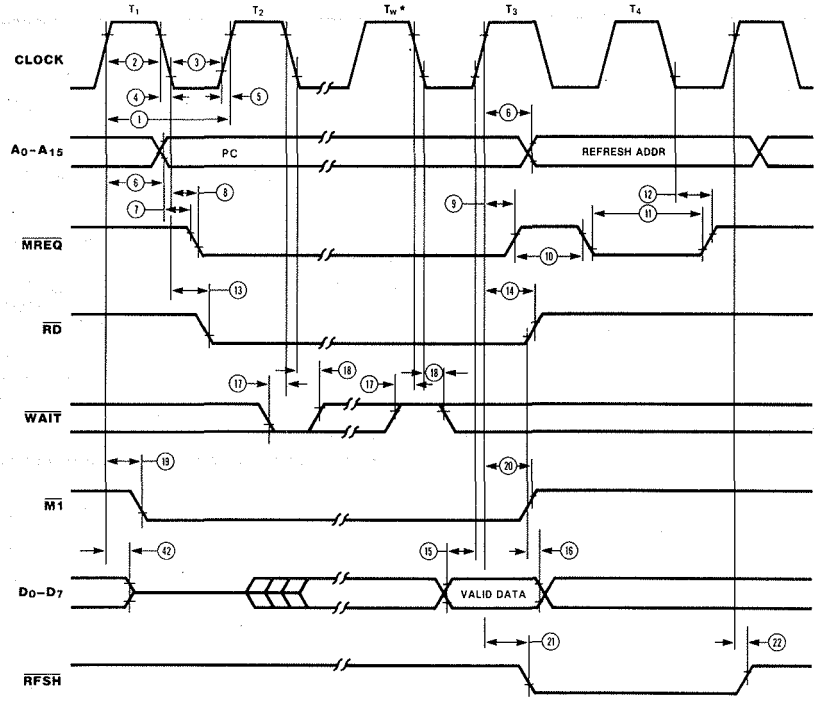
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the falling edge of clock state T_2. During clock states T_3 and T_4 of an $\overline{M1}$ cycle, dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



*T_W = Wait cycle added when necessary for slow ancilliary devices.

Figure 5. Instruction Opcode Fetch

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

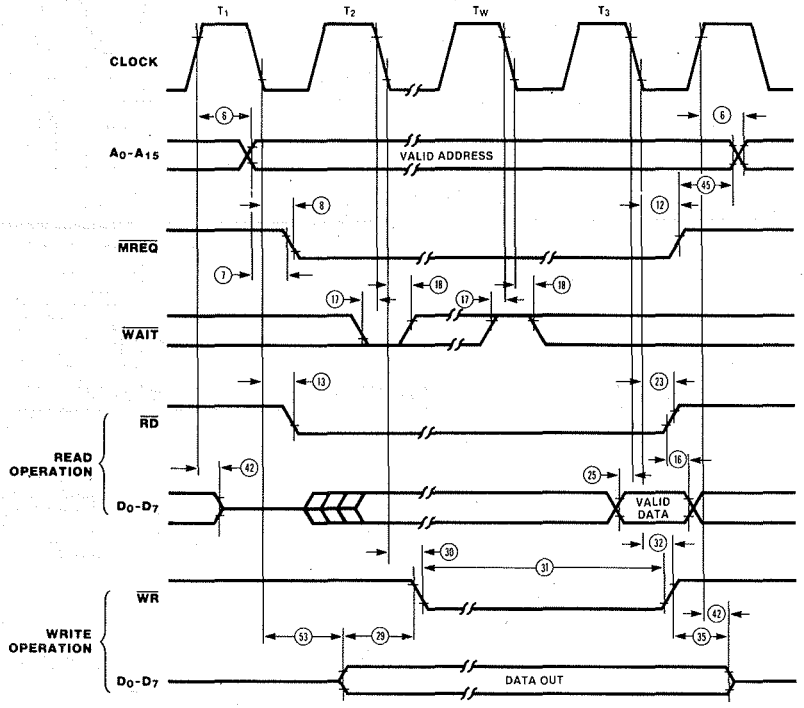
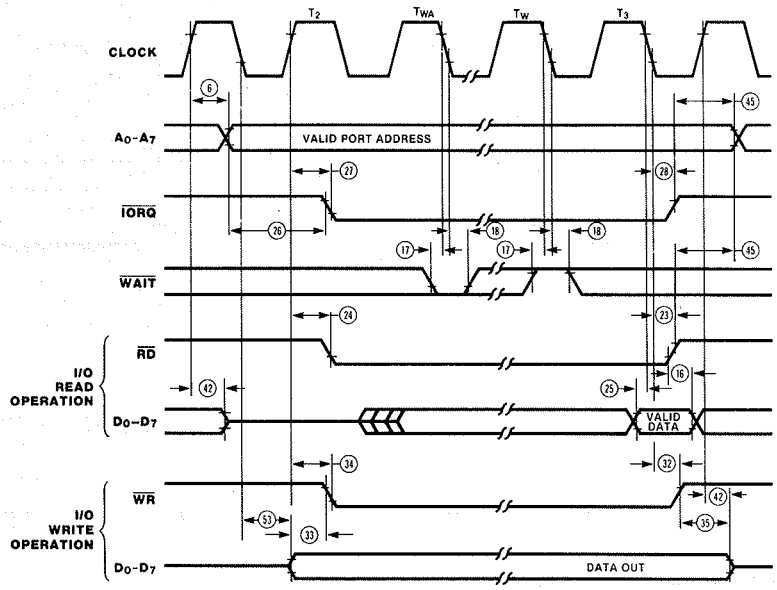


Figure 6. Memory Read or Write Cycles

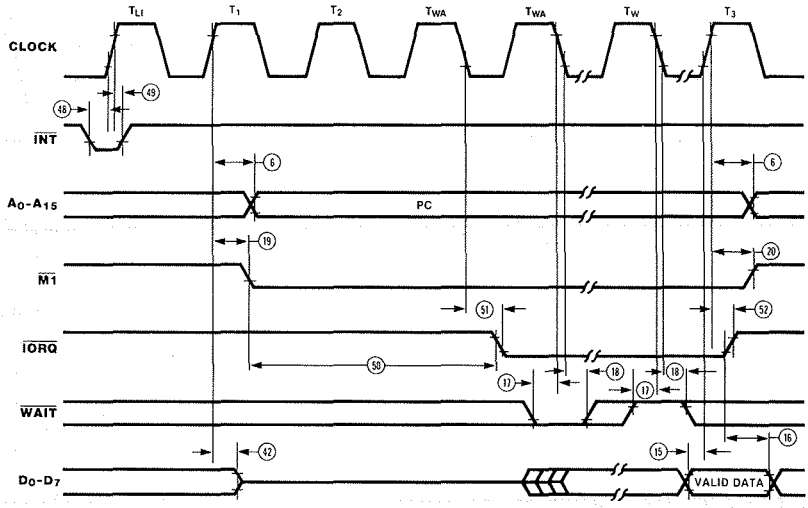
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_{WA}). This extra Wait state allows sufficient time for an I/O port to decode the address from the port address lines.



T_{WA} = One wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

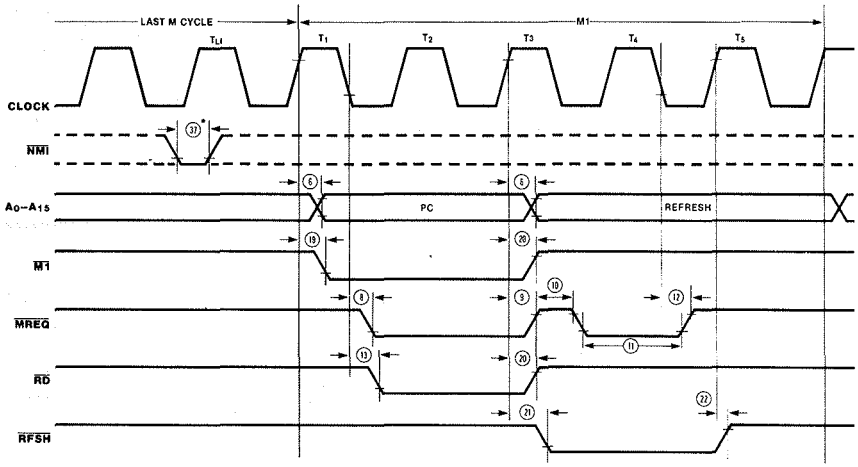
Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.



NOTES: 1) T_{LI} = Last state of any instruction cycle.
2) T_{WA} = Wait cycle automatically inserted by CPU.

Figure 8. Interrupt Request/Acknowledge Cycle

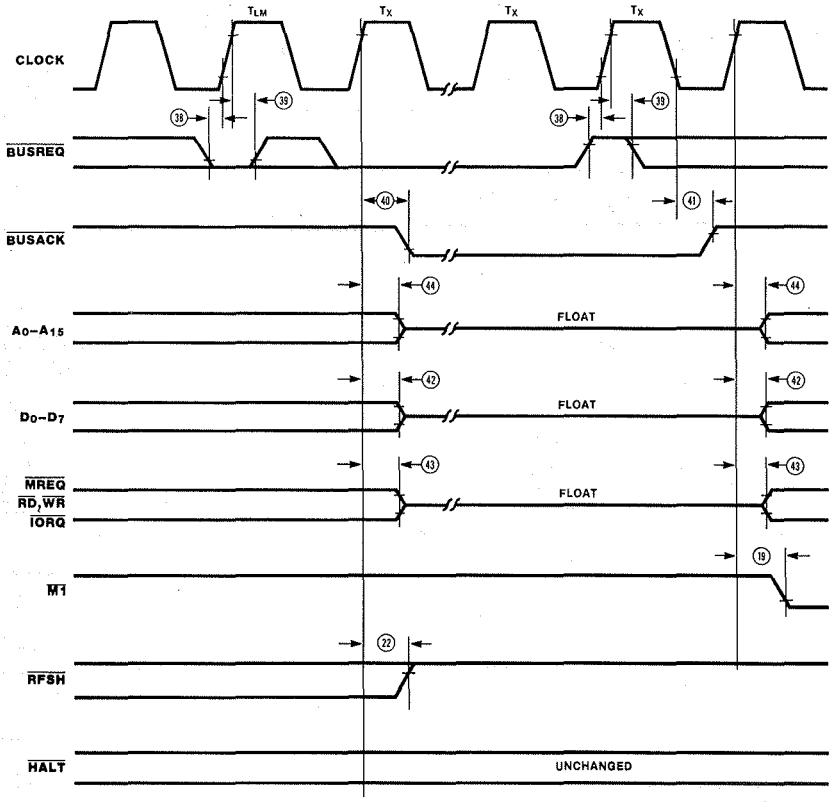
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding the last state of any instruction cycle (T_{LI}).

Figure 9. Non-Maskable Interrupt Request Operation

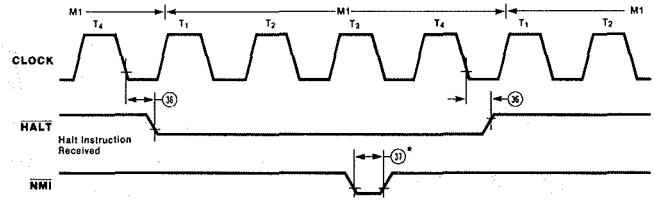
Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.



NOTES: 1) T_{LM} = Last state of any M cycle.
2) T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Z-BUS Request/Acknowledge Cycle

Halt Acknowledge Cycle. When the CPU receives a HALT instruction, it executes NOP states until either an INT or NMI input is received. When in the Halt state, the HALT output is active and remains so until an interrupt is received (Figure 11). INT will also force a Halt exit.



*Although NMI is an asynchronous input, to guarantee its being recognized on the following machine cycle, NMI's falling edge must occur no later than the rising edge of the clock cycle preceding the last state of any instruction cycle (T_{LI}).

Figure 11. Halt Acknowledge Cycle

Reset Cycle. RESET must be active for at least three clock cycles for the CPU to properly accept it. As long as RESET remains active, the address and data buses float, and the control outputs are inactive. Once RESET goes inactive, two internal T cycles are consumed before the CPU resumes normal processing operation. RESET clears the PC register, so the first opcode fetch will be to location 0000H (Figure 12).

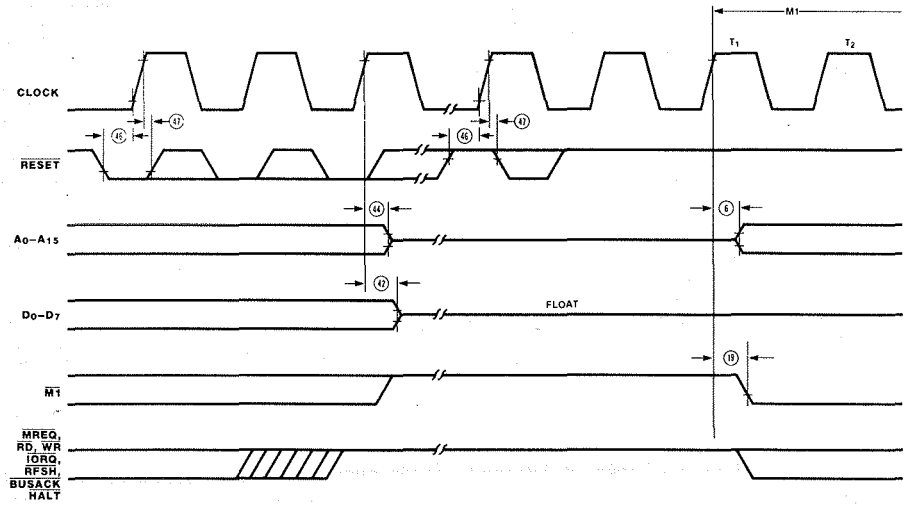


Figure 12. Reset Cycle

AC CHARACTERISTICS†

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU Min	Z80B CPU Max	Z80H CPU Min	Z80H CPU Max
1	TcC	Clock Cycle Time	400*		250*		165*		125*	
2	TwCh	Clock Pulse Width (High)	180	2000	110	2000	65	2000	55	2000
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000	55	2000
4	TfC	Clock Fall Time		30		30		20		10
5	TrC	Clock Rise Time		30		30		20		10
6	TdCr(A)	Clock ↑ to Address Valid Delay		145		110		90		80
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*		65*		35*		20*	
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		100		85		70		60
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay		100		85		70		60
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*		45*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*		220*		135*		100*	
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		100		85		70		60
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay		130		95		80		70
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay		100		85		70		60
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑		0		0		0		0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70		70		60		50	
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓		0		0		0		0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay		130		100		80		70
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay		130		100		80		70
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay		180		130		110		95
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay		150		120		100		85
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay		110		85		70		60
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay		100		85		70		60
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4, or M_5 Cycles	60		50		40		30	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*		180*		110*		75*	
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay		90		75		65		55
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay		110		85		70		60
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*		80*		25*		5*	
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80		70		60
31	TwWR	$\overline{WR}$ Pulse Width	360*		220*		135*		100*	
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay		100		80		70		60
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*		−10*		−55*		55*	
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay		80		65		60		55
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*		15*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓		300		300		260		225
37	TwNMI	$\overline{NMI}$ Pulse Width	80		80		70		60*	
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80		50		50		40	

*For clock periods other than the minimums shown, calculate parameters using the table on the following page. Calculated values above assumed TrC = TfC = 20 ns.
†Units in nanoseconds (ns).

AC CHARACTERISTICS† (Continued)

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU Min	Z80B CPU Max	Z80H CPU Min	Z80H CPU Max
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0		0		0		0	
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90		80
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay		110		100		90		80
42	TdCr(Dz)	Clock ↑ to Data Float Delay		90		90		80		70
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)		110		80		70		60
44	TdCr(Az)	Clock ↑ to Address Float Delay		110		90		80		70
45	TdCTr(A)	$\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑ to Address Hold Time	160*		80*		35*		20*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90		60		60		45	
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time		0		0		0		0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80		80		70		55	
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time		0		0		0		0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*		270*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay		110		85		70		60
52	TdCf(IORQr)	Clock ↑ $\overline{IORQ}$ ↑ Delay		100		85		70		60
53	TdCf(D)	Clock ↓ to Data Valid Delay		230		150		130		115

*For clock periods other than the minimums shown, calculate parameters using the following table. Calculated values above assumed TrC = TfC = 20 ns.
†Units in nanoseconds (ns).

FOOTNOTES TO AC CHARACTERISTICS

Number	Symbol	General Parameter	Z80	Z80A	Z80B	Z80H
1	TcC	TwCh + TwCl + TrC + TfC				
7	TdA(MREQf)	TwCh + TfC	− 75	− 65	−50	−45
10	TwMREQh	TwCh + TfC	− 30	− 20	−20	−20
11	TwMREQl	TcC	− 40	− 30	−30	−25
26	TdA(IORQf)	TcC	− 80	− 70	−55	−50
29	TdD(WRf)	TcC	− 210	− 170	−140	−120
31	TwWR	TcC	− 40	− 30	−30	−25
33	TdD(WRf)	TwCl + TrC	− 180	− 140	−140	−120
35	TdWRr(D)	TwCl + TrC	− 80	− 70	−55	−50
45	TdCTr(A)	TwCl + TrC	− 40	− 50	−50	−45
50	TdM1f(IORQf)	2TcC + TwCh + TfC	− 80	− 65	−50	−45

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
$V_{IHC} = V_{CC} - 0.6$ V
V_{ILC} = 0.45 V
V_{OH} = 1.5 V
V_{OL} = 1.5 V
FLOAT = ±0.5 V

ORDERING INFORMATION

Z80 CPU, 2.5 MHz

40-pin DIP	44-pin LCC	44-pin PCC
Z8400 PS	Z8400 LM*	Z8400 VS†
Z8400 CS	Z8400 LMB*†	
Z8400 PE		
Z8400 CE		
Z8400 CM*		
Z8400 CMB*		
Z8400 CMJ*		

Z80A CPU, 4.0 MHz

40-pin DIP	44-pin LCC	44-pin PCC
Z8400A PS	Z8400A LM*	Z8400A VS†
Z8400A CS	Z8400A LMB*†	
Z8400A PE		
Z8400A CE		
Z8400A CM*		
Z8400A CMB*		
Z8400A CMJ*		

Z80B CPU, 6.0 MHz

40-pin DIP	44-pin PCC
Z8400B PS	Z8400B VS†
Z8400B CS	
Z8400B PE	

Z80H CPU, 8.0 MHz

40-pin DIP	44-pin PCC
Z8400H PS	Z8400H VS†

Codes

First letter is for package; second letter is for temperature.

C = Ceramic DIP
P = Plastic DIP
L = Ceramic LCC
V = Plastic PCC

R = Protopack
T = Low Profile Protopack
DIP = Dual-In-Line Package
LCC = Leadless Chip Carrier
PCC = Plastic Chip Carrier (Leaded)

TEMPERATURE
S = 0°C to +70°C
E = −40°C to +85°C
M* = −55°C to +125°C

FLOW
B = 883 Class B
J = JAN 38510 Class B

*For Military Orders, refer to the Military Section.
†Available soon.

ORDERING INFORMATION

Z80 CPU, 2.5 MHz

40-pin DIP	44-pin LCC	44-pin PCC
Z8400 PS	Z8400 LM*	Z8400 VS†
Z8400 CS	Z8400 LMB*†	
Z8400 PE		
Z8400 CE		
Z8400 CM*		
Z8400 CMB*		
Z8400 CMJ*		

Z80A CPU, 4.0 MHz

40-pin DIP	44-pin LCC	44-pin PCC
Z8400A PS	Z8400A LM*	Z8400A VS†
Z8400A CS	Z8400A LMB*†	
Z8400A PE		
Z8400A CE		
Z8400A CM*		
Z8400A CMB*		
Z8400A CMJ*		

Z80B CPU, 6.0 MHz

40-pin DIP	44-pin PCC
Z8400B PS	Z8400B VS†
Z8400B CS	
Z8400B PE	

Z80H CPU, 8.0 MHz

40-pin DIP	44-pin PCC
Z8400H PS	Z8400H VS†

Codes

First letter is for package; second letter is for temperature.

C = Ceramic DIP
P = Plastic DIP
L = Ceramic LCC
V = Plastic PCC

R = Protopack
T = Low Profile Protopack
DIP = Dual-In-Line Package
LCC = Leadless Chip Carrier
PCC = Plastic Chip Carrier (Leaded)

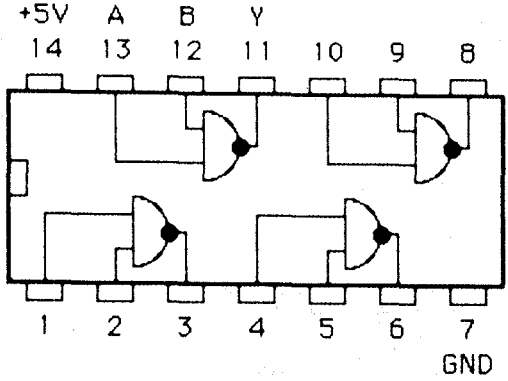
TEMPERATURE
S = 0°C to +70°C
E = −40°C to +85°C
M*= −55°C to +125°C

FLOW
B = 883 Class B
J = JAN 38510 Class B

*For Military Orders, refer to the Military Section.
†Available soon.

74LS00

4 NAND-Gatter mit je zwei Eingängen



Logiktabelle:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

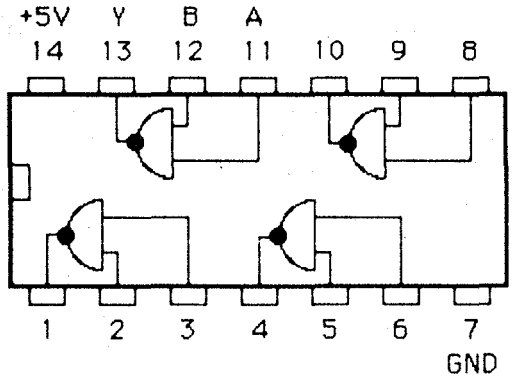
Typ. Impuls-Verzögerungszeit: 9,5 ns

Typ. Leistungs-aufnahme: 8 mW

positive Logik:
$Y = \overline{AB}$

74LS01

4 NAND-Gatter mit je zwei Eingängen



Offene Kollektorausgänge

Logiktabelle:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

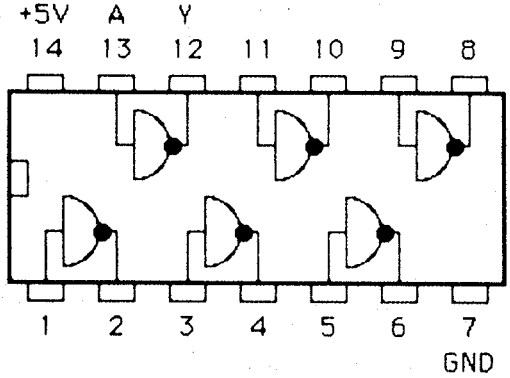
Typ. Impuls-Verzögerungszeit: 22 ns

Typ. Leistungsaufnahme: 40 mW

positive Logik:
$Y = \overline{AB}$

7404

6 Inverter



Logiktabelle:

A	Y
0	1
1	0

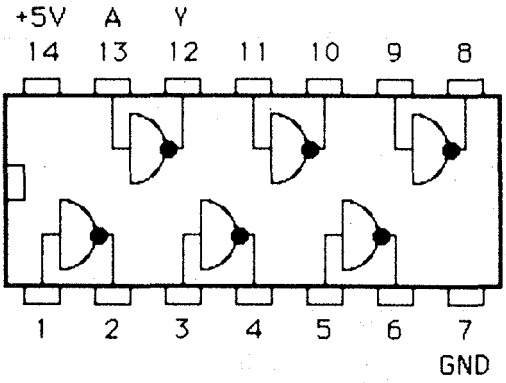
Typ. Impuls-Verzögerungszeit: 9 ns

Typ. Versorgungsstrom: 25 mA

positive Logik:
$Y = \overline{A}$

74LS04

6 Inverter



Logiktabelle:

A	Y
0	1
1	0

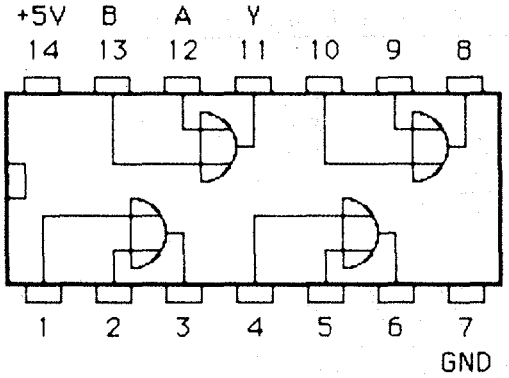
Typ. Impuls-Verzögerungszeit: 10 ns

Typ. Versorgungsstrom: 4 mA

positive Logik:
$Y = \overline{A}$

74LS32

4 OR-Gatter mit je zwei Eingängen



Logiktabelle:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

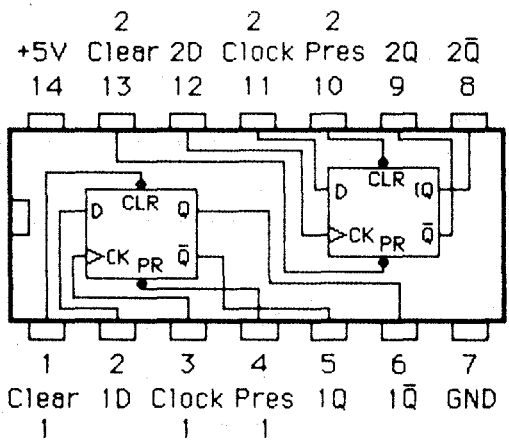
Typ. Impuls-Verzögerungszeit: 12 ns

Typ. Leistungs-aufnahme: 20 mW

positive Logik.
Y = A + B

7474

Zwei D-Flipflops mit Preset und Clear



Wahrheitstabelle:

Inputs				Outputs	
Preset	Clear	Clock	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

Positive Logik

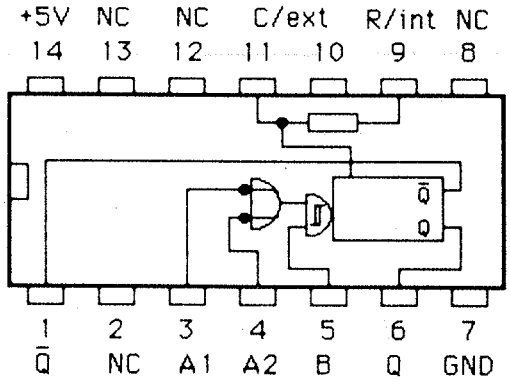
*Dieser Zustand ist nicht stabil; d.h. er bleibt nicht erhalten, wenn Preset und/oder Clear inaktiv (High) werden.

Typ. Impulsverzögerungszeit : 17 ns

Typ. Versorgungsstrom : 16 mA

74LS121

Monoflop mit Schmitt-Trigger-Eingang



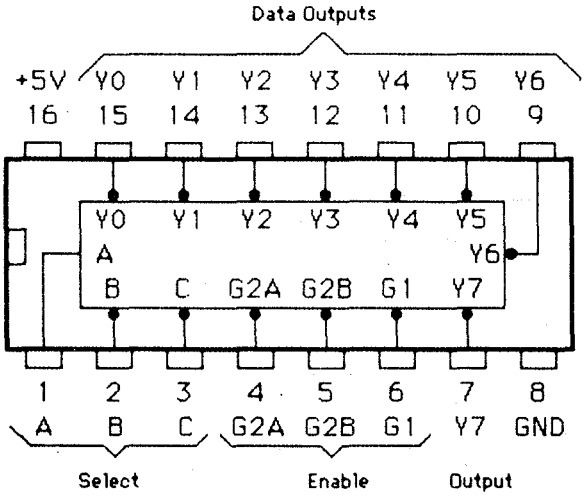
Wahrheitstabelle:

Inputs			Outputs	
A1	A2	B	Q	Q̄
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⊓	⊔
↓	H	H	⊓	⊔
↓	↓	H	⊓	⊔
L	X	↑	⊓	⊔
X	L	↑	⊓	⊔

Typ. Impulsverzögerungszeit von A1,A2 : 47,5 ns
Typ. Impulsverzögerungszeit von B : 37,5 ns
Typ. Versorgungsstrom 16 mA

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

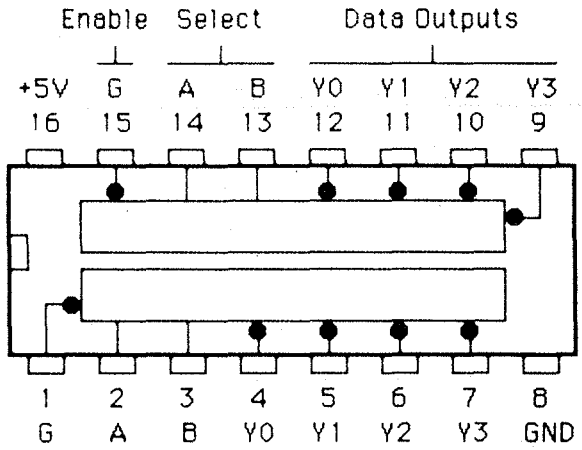
Positive Logik *G2 = G2A + G2B

Typ. Impulsverzögerungszeit : 22 ns

Typ. Versorgungsstrom 7 mA

74LS139

2 2-zu-4 Decoder/Demultiplexer



Logiktabelle:

INPUTS			OUTPUTS			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	x	x	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	H

H = high level
L = low level
x = irrelevant

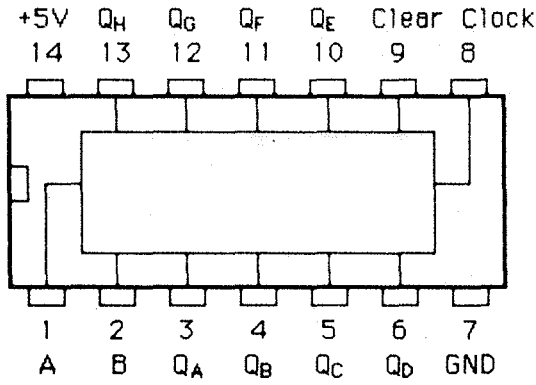
Typ. Impuls-Verzögerungszeit: 20 ns

Typ. Versorgungsstrom: 7 mA

positive Logik: siehe Tabelle

74LS164

Schieberegister mit 8-Bit paralleler Ausgabe



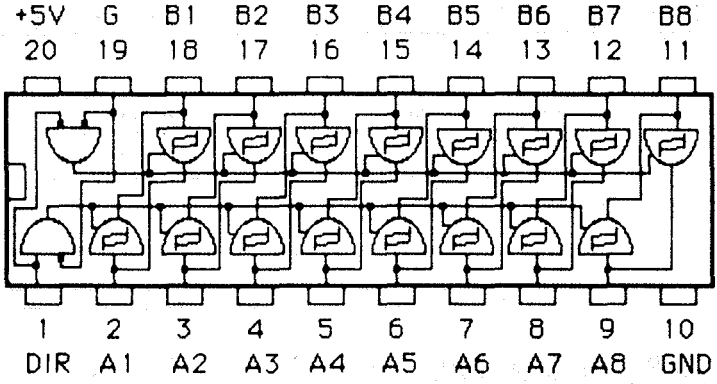
Function Table:

INPUTS				OUTPUTS			
Clear	Clock	A	B	Q_A	Q_B	...	Q_H
L	x	x	x	L	L		L
H	L	x	x	Q_{A0}	Q_{B0}		Q_{H0}
H	↑	H	H	H	Q_{An}		Q_{Gn}
H	↑	L	x	L	Q_{An}		Q_{Gn}
H	↑	x	L	L	Q_{An}		Q_{Gn}

Typ. Impulsverzögerungszeit: 15 ns
Typ. Versorgungsstrom: 20 mA

74LS245

Acht Bus-Transceiver (Tri-State)



Logiktabelle:

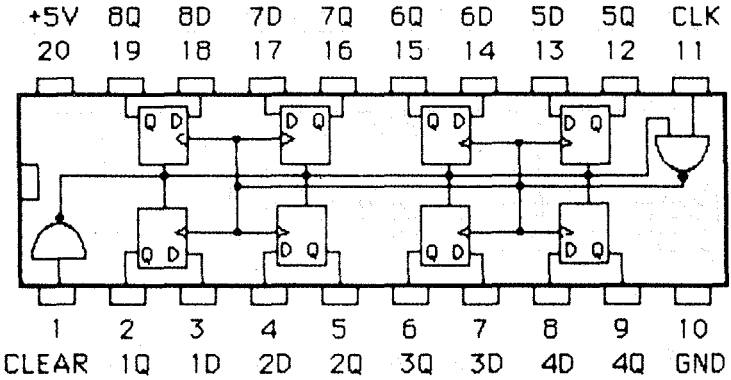
Enable $\overline{G}$	Direktion DIR	Operation
L	L	B data to A Bus
L	H	A data to B Bus
H	X	Isolation

Typ. Impuls-Verzögerungszeit: 8 ns

Typ. Versorgungsstrom: 62 mA

74LS273

8-Bit D-Register mit Clear



Logiktabelle:

INPUT			OUTPUT
CLEAR	CLOCK	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	L	Q0

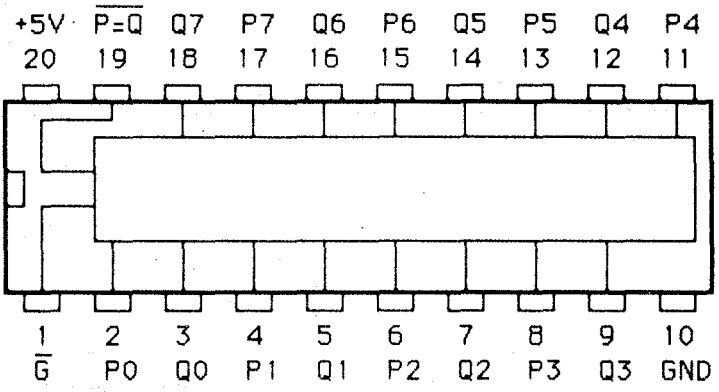
positive Logik

Typ. Impuls-Verzögerungszeit: 17,5 ns

Typ. Versorgungsstrom: 17,5 mA

74LS688

8-Bit Größenvergleicher



Logiktabelle:

INPUT		OUTPUT
$\overline{G}$	P0, P1 ... P7 Q0, Q1 ... Q7	$\overline{P=Q}$
H	X X	H
L	P0≠Q0, P1≠Q1 .... P7≠Q7	H
L	... PY≠QY	H
L	P0=Q0, P1=Q1 .... P7=Q7	L

Typ. Versorgungsstrom: 40 mA

Typ. Impuls-Verzögerungszeit: 15 ns

11. Literatur

11.1 Hinweis auf 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 Ingormationen verfügen.

Ein LOOP-ABO können Sie bei jeder Bestellung einfach mitbestellen. Auch auf der Kritikkarte können Sie ein LOOP-ABO ganz einfach mitbestellen.

Telefonservice
08 31- 62 11
jeden Mittwochabend
bis 20.00 Uhr

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang
Telefon: 08 31 / 6 93 00

Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr

ges