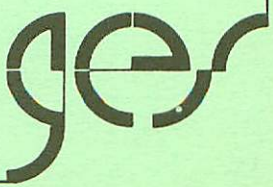
SBC3

Die universelle CPU-Baugruppe mit der CPU Z80

für den NDR-Computer

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 Norddeutschen Rundfunk, vom Sender Freies Berlin, vom Bayerischen 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,
  "Mikrocomputer selbstgebaut und programmiert"
  2., neue bearbeitete und erweiterte Auflage
  ISBN 3-7723-7162-0, DM 38,-
  erschienen im Francis-Verlag, München
  Bestellnummer: B001
  Auf diesem Buch baut die NDR-Serie auf

- Sonderhefte der "mc"
  "Mikrocomputer Schritt für Schritt"
  Bestellnummer: SONDERNDR
  "Mikrocomputer Schritt für Schritt Teil 2"
  Bestellnummer: SONDERNDR2

- 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, dem 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 RAM64/128 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).

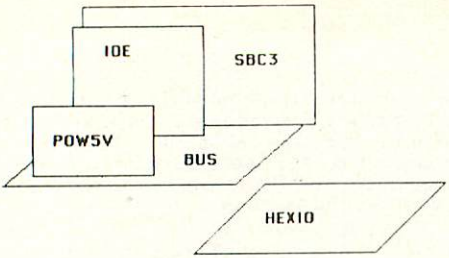


Abb 1: Konfiguration beim Einsteigerpaket

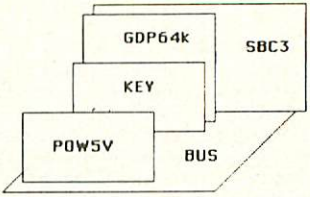
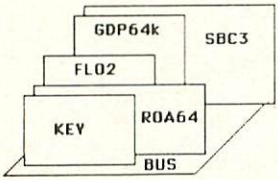


Abb.2: Mögliche Konfiguration beim System ohne Floppy und ohne CP/M



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

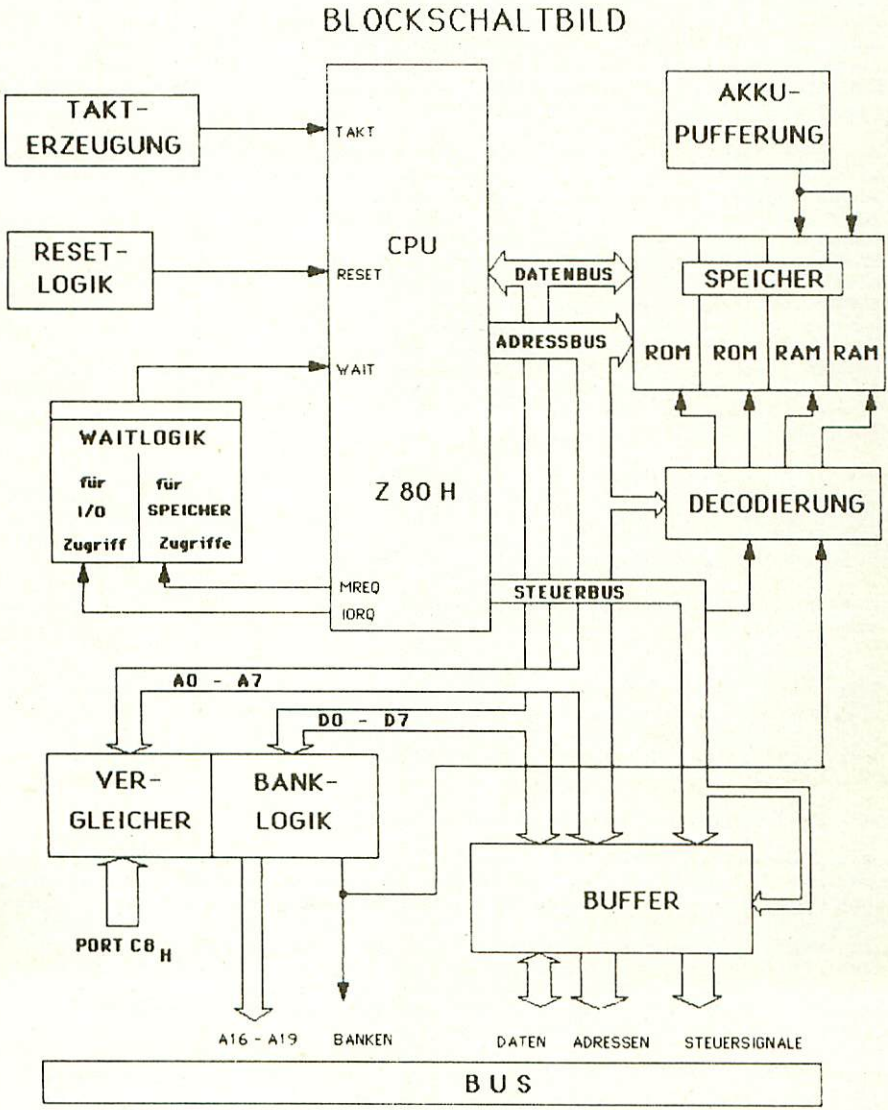
Voll gepufferter Daten-, Adress- und Steuerbus

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	Original GES-Platine mit Lötstoplack (r2)		
1	Handbuch Ausgabe 1		
1	74 04	IC 22	6 Iverter
1	74 121	IC 1	Monoflop
1	74 LS 00	IC 3	4 NAND
1	74 LS 01	IC 23	4 NAND mit offenem Kollektor
1	74 LS 04	IC 13	6 Inverter
1	74 LS 32	IC 10	4 OR
1	74 LS 74	IC 2	2 D-Flip-Flop
1	74 LS 138	IC 15	3 zu 8 Dekoder
2	74 LS 139	IC 4, 12	2 zu 4 Dekoder
2	74 LS 164	IC 11, 14	8-Bit Schieberegister
4	74 LS 245	IC16,18,19,20	8-Bit-Tri-State Bustreiber
1	74 LS 273	IC 21	8-Bit D-Register
1	74 LS 688	IC 17	8-Bit Größenvergleicher
1	CPU Z80A	IC 9	CPU
2	RAM 8k	IC 7, 8	RAM-Speicher 8kbyte
1		Q 1	Quarz 8 MHz
11	100 nF	C2 - C12	Keramik-Kondensatoren 100 nF
2	10 uF	C1, C13	Tantal-Kondensatoren 10 uF
8	1k	R1, R2, R3, R12 - R16	Widerstände 1 kOhm
2	2,2k	R8, R9,	Widerstände 2,2 kOhm
3	4,7k	R5, R6, R10	Widerstände 4,7 kOhm
1	68	R4	Widerstand 68 Ohm
1	220	R7	Widerstand 220 Ohm
1	330	R11	Widerstand 330 Ohm
1	8*1k	RN2	Netzwerkwiderstände 8*1 kOhm
1	8*3.3k	RN1	Netzwerkwiderstand 8*3,3 kOhm
1	1N4148	D1	Si-Diode
4	Mod 225H		Shunt-Stecker
1		JMP12, JMP13	Stiftreihe 2 * 4 gerade
2		JMP1, JMP15	Stiftreihe 2 * 2 gerade
2	BSX 20	TR1, TR2	Transistoren
1	NCM-2,4	Akku	Akku 2,4 V
1		S1	Taster für RESET mit Tastkappe

1	St 1	36-polige Steckerleiste
1	St 1	18-polige Steckerleiste
1		40-polige IC-Fassung
4		28-polige IC-Fassung
6		20-polige IC-Fassung
3		16-polige IC-Fassung
9		14-polige IC-Fassung

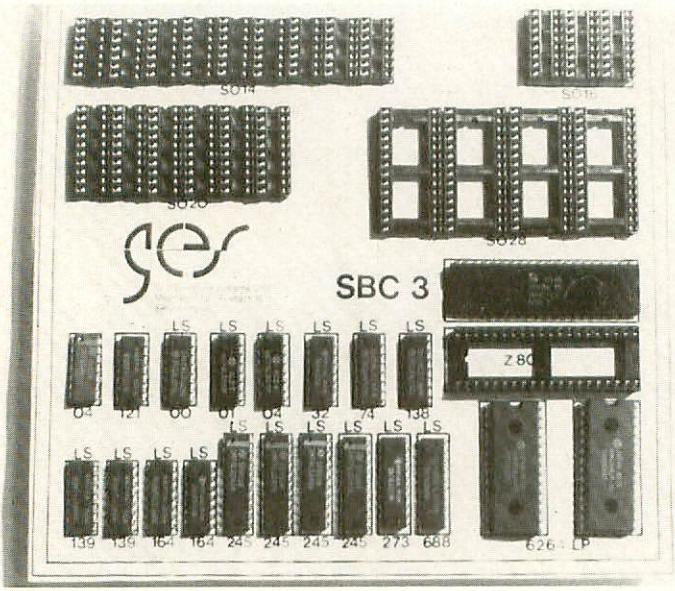


Abb.: ICs und Sockel der SBC3

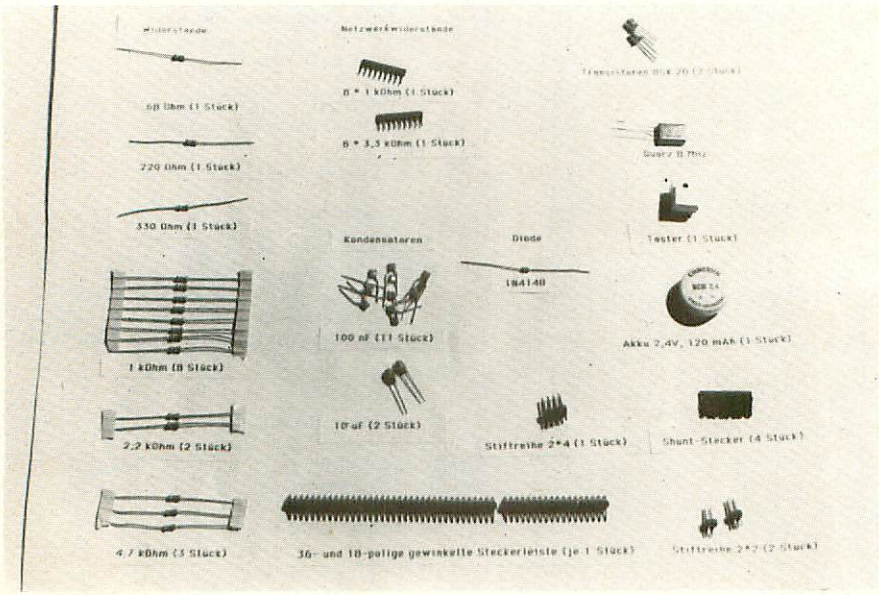
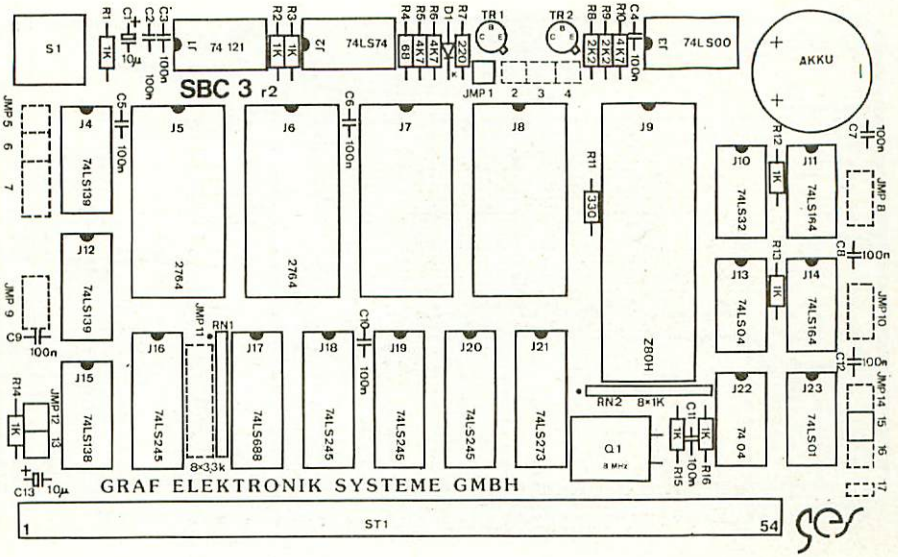
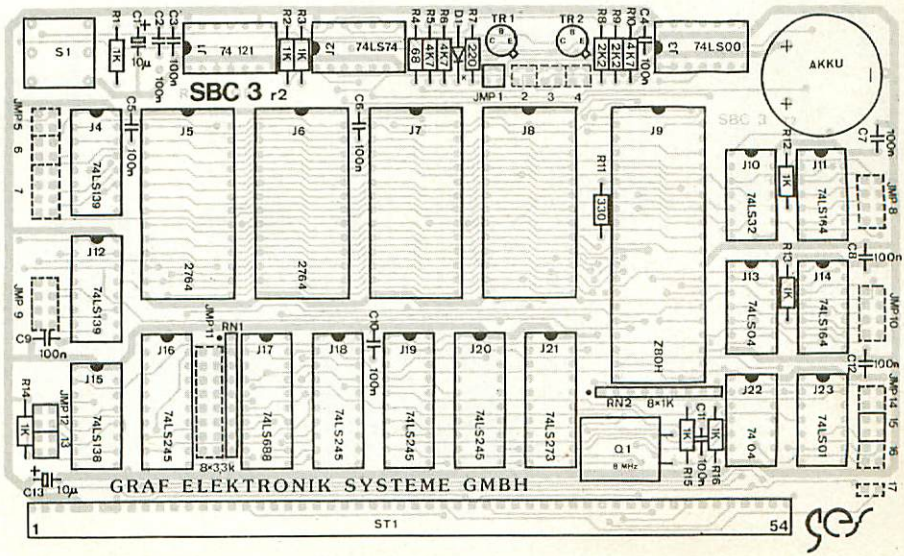


Abb.: Einzelne Bauteile der SBC3

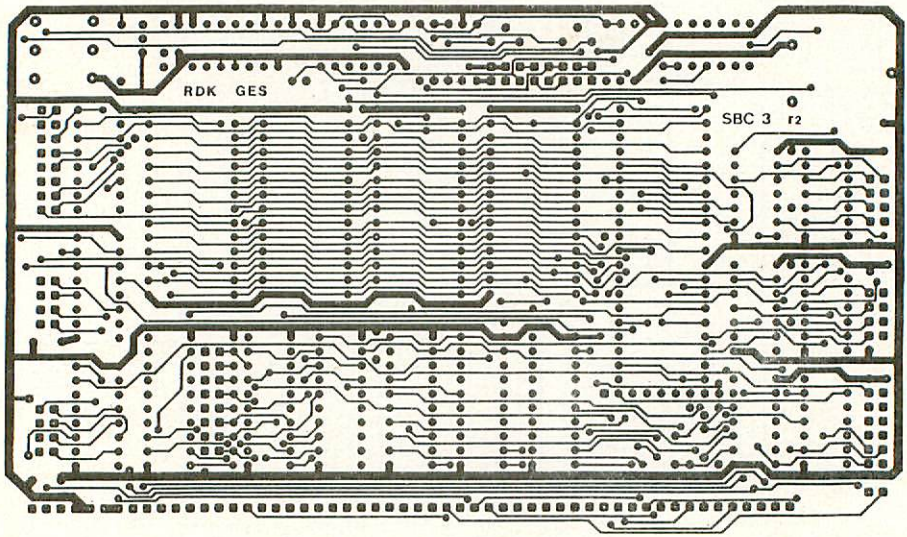
4.3 Bestückungsplan



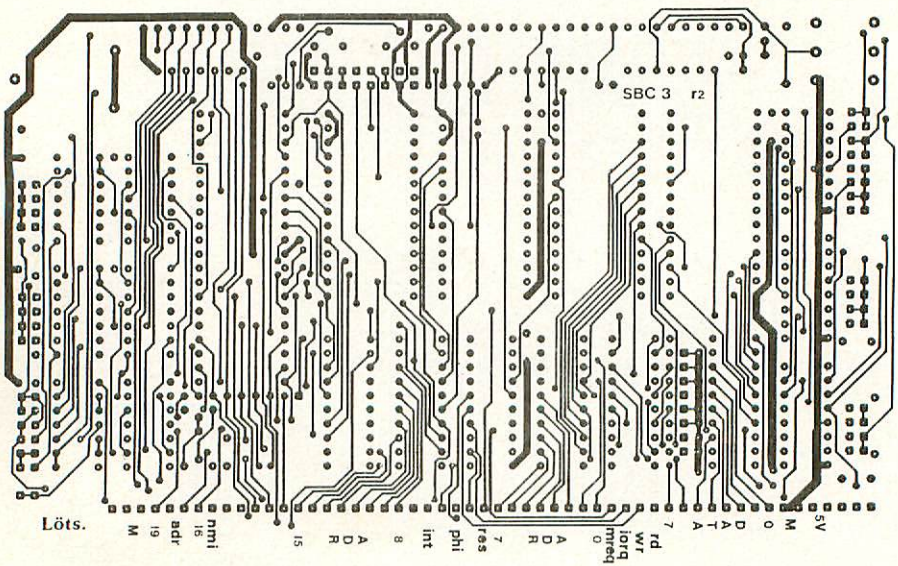
4.4 Layout Bestückungsseite mit Bestückungsplan



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 C1 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 C2 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 R16 sind Einzelwiderstände mit Farbcode. Die verwendeten Widerstände 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

Wie aus der Tabelle hervorgeht sind für die Widerstandswerte die ersten drei Ringe ausschlaggebend. 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 soll so eingelötet werden , daß die beiden Kontaktstifte an jeder Seite des Tasters unten liegen (zur Busleiste hin). Siehe auch Abb.. Dabei müssen die beiden Kunststoffbeinchen (falls vorhanden) abgezwickt werden.

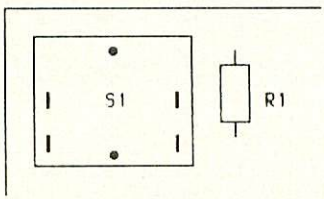


Abb : Einloten des Tasters S1

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.

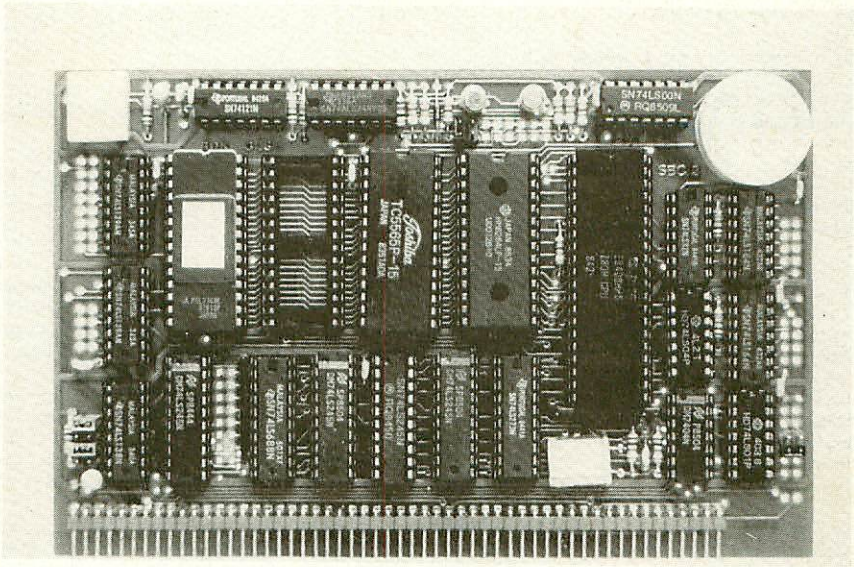


Abb.: Fertig aufgebaute Baugruppe

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 IC7 und IC8 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 7 und 8 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 IC5 gesteckt. Und mindestes ein RAM 8k muß gesteckt werden (IC7). 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 (IC5), der zweite (IC6) kann leer bleiben. Es kann aber auf diesen auch EGOSI2 oder ESPS2 gesteckt werden. Außerdem wird mindestes ein RAM 8k benötigt (IC7). 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

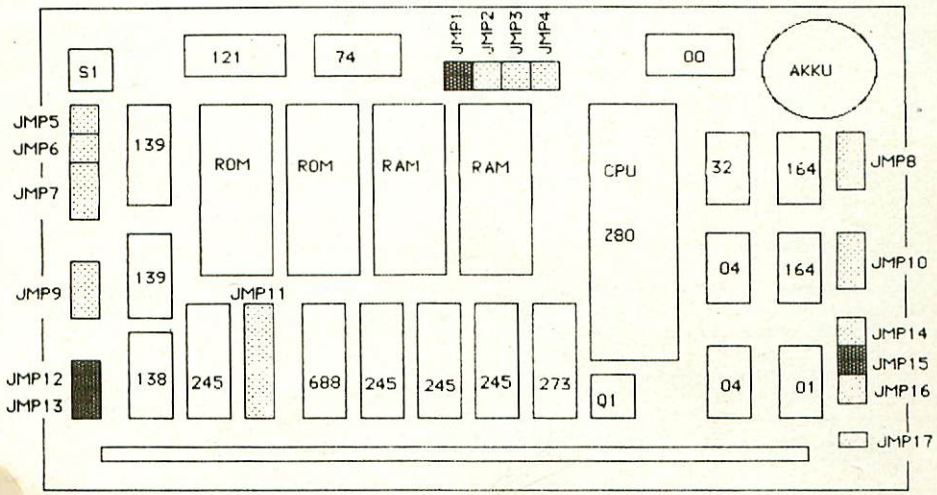
Konfigurieren Sie Ihr System wie unter Punkt 1.3. Auf den ersten Steckplatz (IC5) wird das EPROM "FLOMON" gesteckt. Auf dem dritten und vierten Steckplatz (IC7, IC8) müssen 2 8k RAM stecken. Die JMP müssen dann wie unter 5.3.2 beschrieben gesteckt sein. Es kann auch die gleiche Speicherkonfiguration 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 (IC6, IC7) gesteckt. Ein 8k RAM muß auf Steckplatz 4 (IC8) gesteckt werden. Wenn Sie jetzt einschalten, muß dasselbe Menu wie beim CP/M System ohne ZEAT erscheinen. Wählen Sie Punkt 3 der 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.

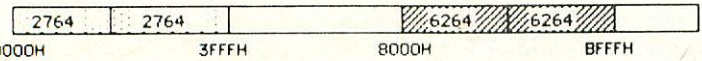
Einstellung der offenen JMP mit "Shunt-Stecker"

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

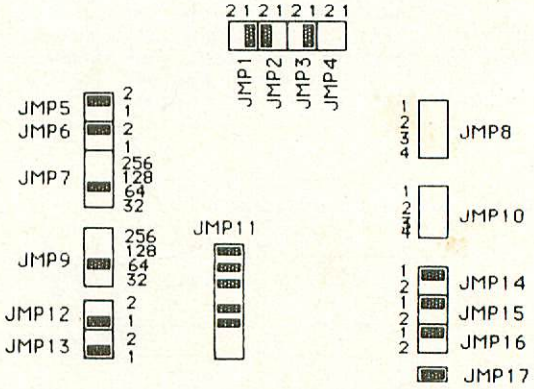
Bei diesem System können maximal 2 EPROMs 2764 und 2 RAM 8k eingesetzt werden (32k).

Speicherkonfiguration:



Einstellung der JMP

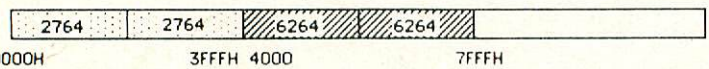
JMP1	Stellung 1
JMP12	Stellung 1
JMP13	Stellung 1
JMP15	Stellung 1



5.3.2 System mit CP/M

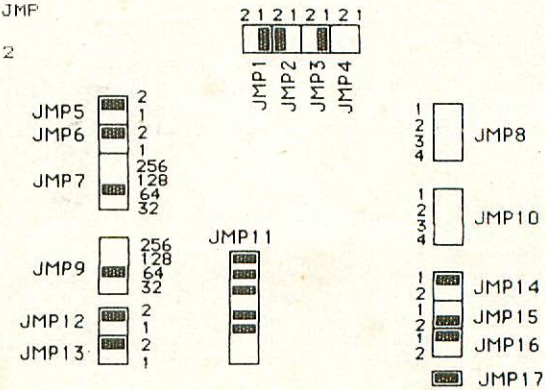
Bei diesem System können auch maximal 2 EPROMs 8k und 2 RAMs 8k eingesetzt werden. Auf der Adresse 2000H kann noch das Grundprogramm EGRU2000, oder EGOSI2 usw. stecken.

Speicherkonfiguration:



Einstellung der einzelnen JMP

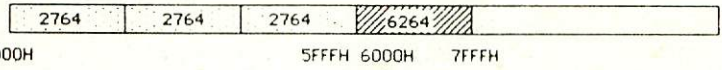
JMP1	Stellung 1 oder 2
JMP12	Stellung 2
JMP13	Stellung 2
JMP15	Stellung 2



5.3.3 System mit CP/M und ZEAT

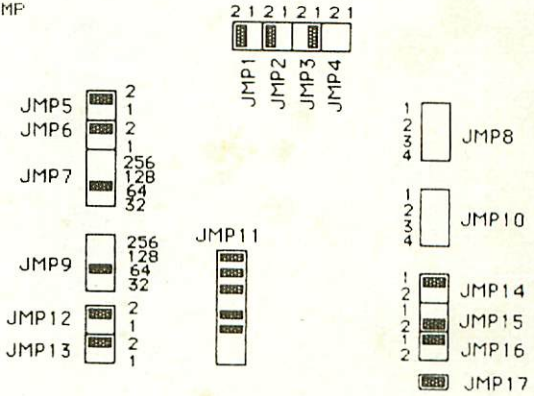
Bei diesem System werden 3 EPROMS 8k und 1 RAM 8k eingesetzt.

Speicherkonfiguration:



Einstellung der einzelnen JMP

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

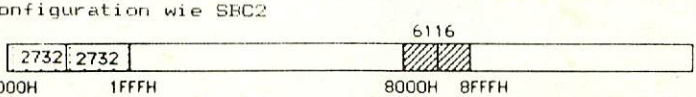


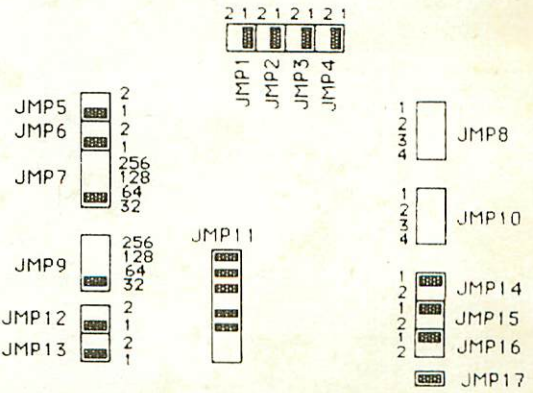
Dies sind die drei Einstellungen, mit denen die SBC3 normalerweise betrieben wird. Wollen Sie andere Speicher als 8k Speicher verwenden oder die CPU Z80H mit 8 MHz dann müssen Sie die voreingestellten JMP "anrühren".

5.4 Speicherkonfigurationen und dazugehörige Jumperstellungen

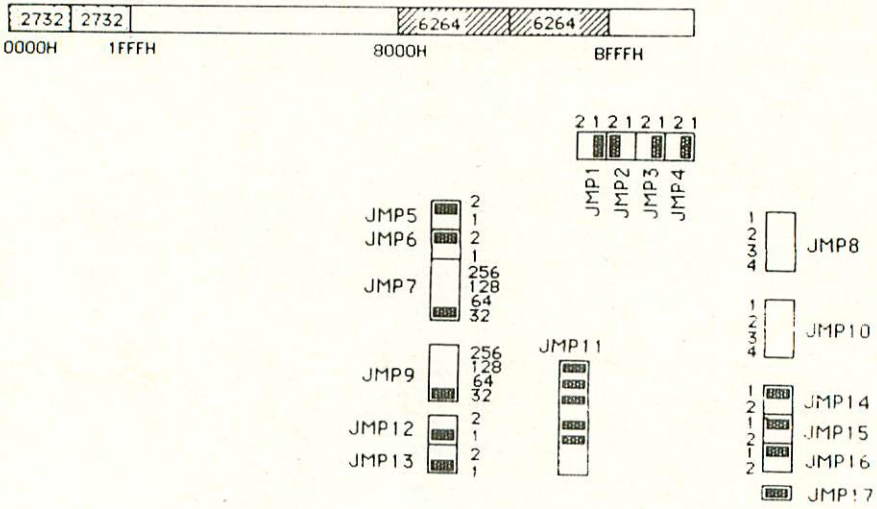
5.4.1. System ohne CP/M mit 8k EPROMs und 8k RAMs (siehe 5.3.1)
5.4.2. System mit CP/M mit 8k EPROMs und 8k RAMs (siehe 5.3.2)
5.4.3. System mit CP/M und ZEAT mit 8k EPROMs und 8k RAM (siehe 5.3.3)

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

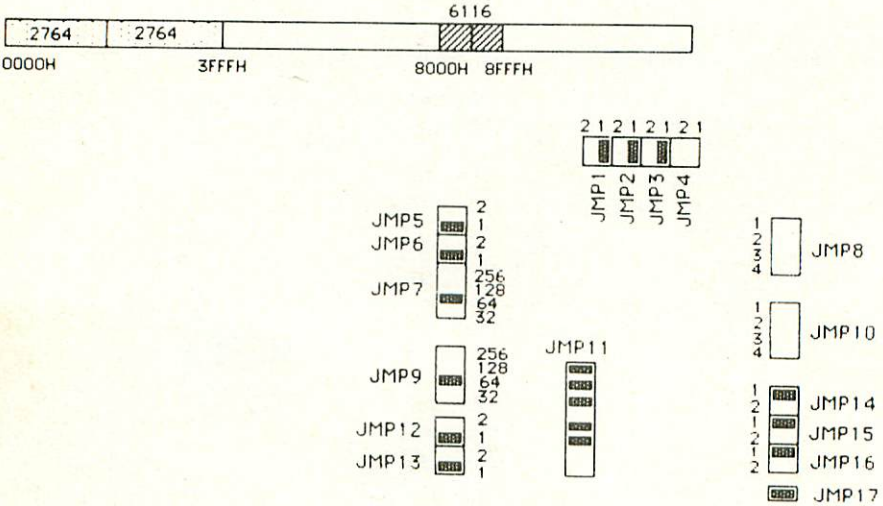




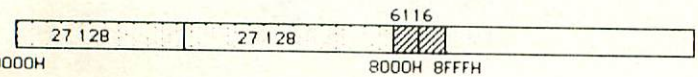
5.4.5. System ohne CP/M mit 4k EPROMs und 8k RAMs

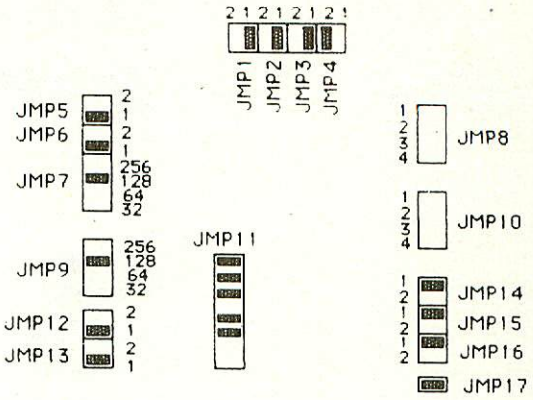


5.4.6. System ohne CP/M mit 8k EPROMs und 2k RAMs

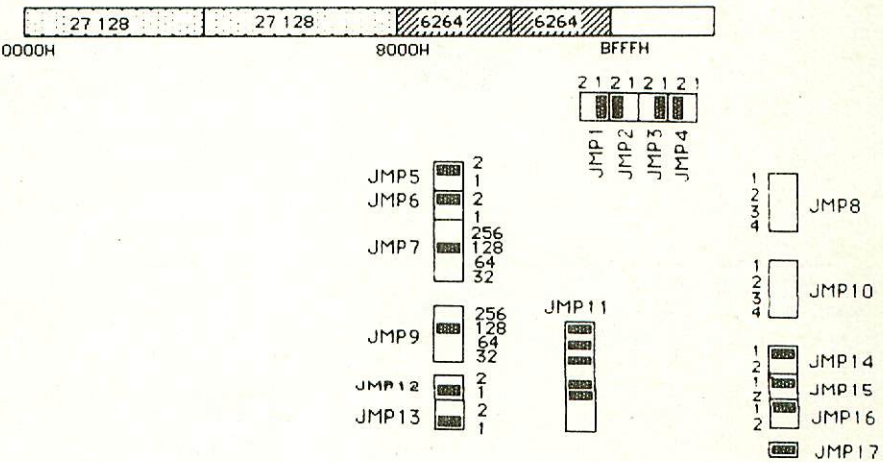


5.4.7 System ohne CP/M mit 16k EPROMs und 2k RAMs

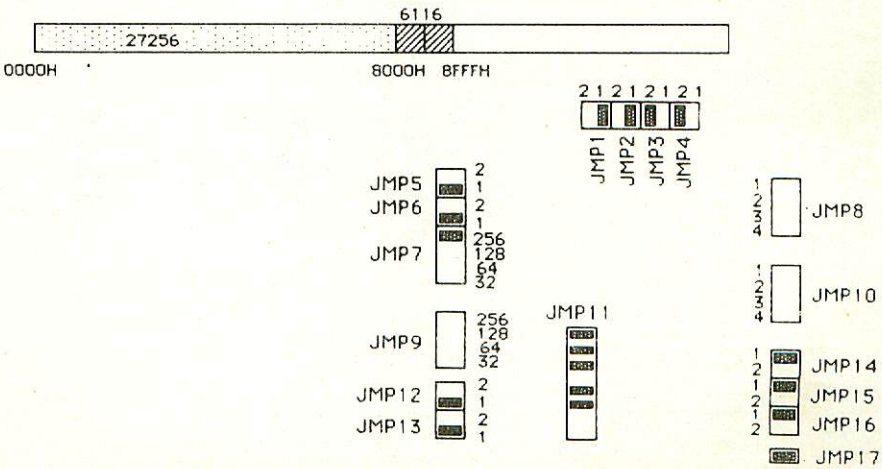




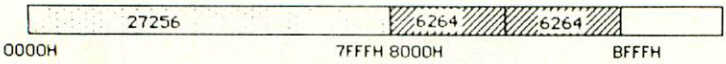
5.4.8. System ohne CP/M mit 16k EPROM und 8k RAMs

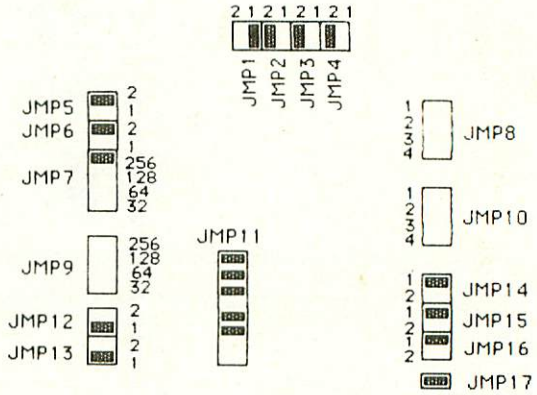


5.4.9. System ohne CP/M mit 32k EPROMs und 2k RAM

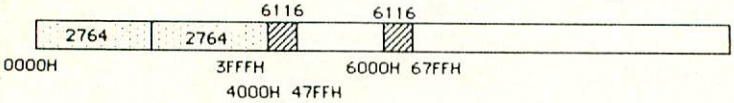


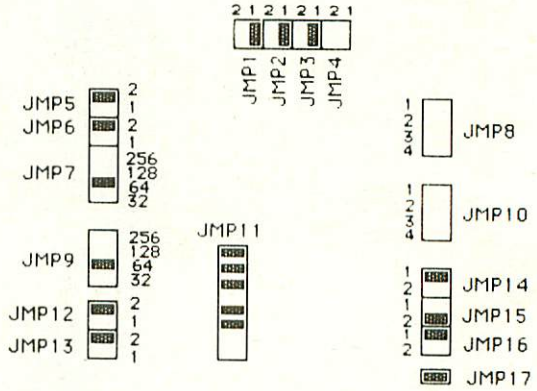
5.4.10. System ohne CP/M mit 32k EPROMs und 8k RAMs



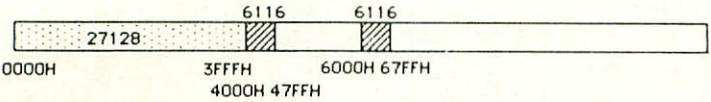


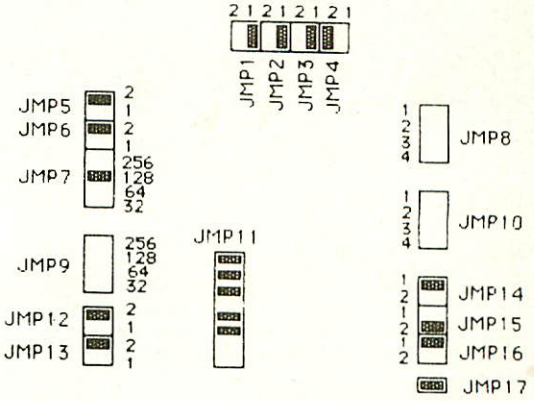
5.4.11. System mit CP/M mit 8k EPROMs und 2k RAMs (wie BANKBOOT)



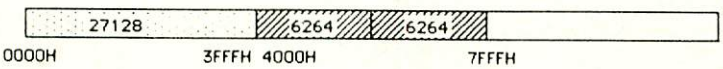


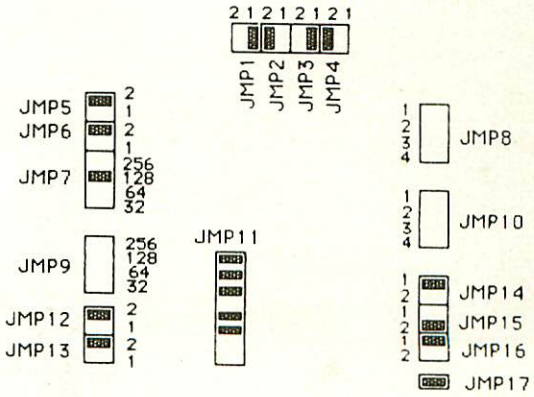
5.4.12. System mit CP/M mit 16k EPROMs und 2k RAMs





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:

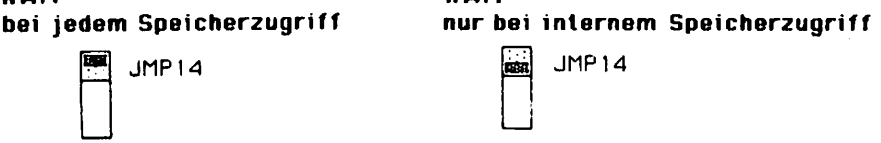
Dieser JMP dient dazu den Takt von 4 MHz auf 8 MHz umzustellen (siehe Abb.)

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 (IC5) gilt JMP7, für das zweite EPROM (IC6) 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.

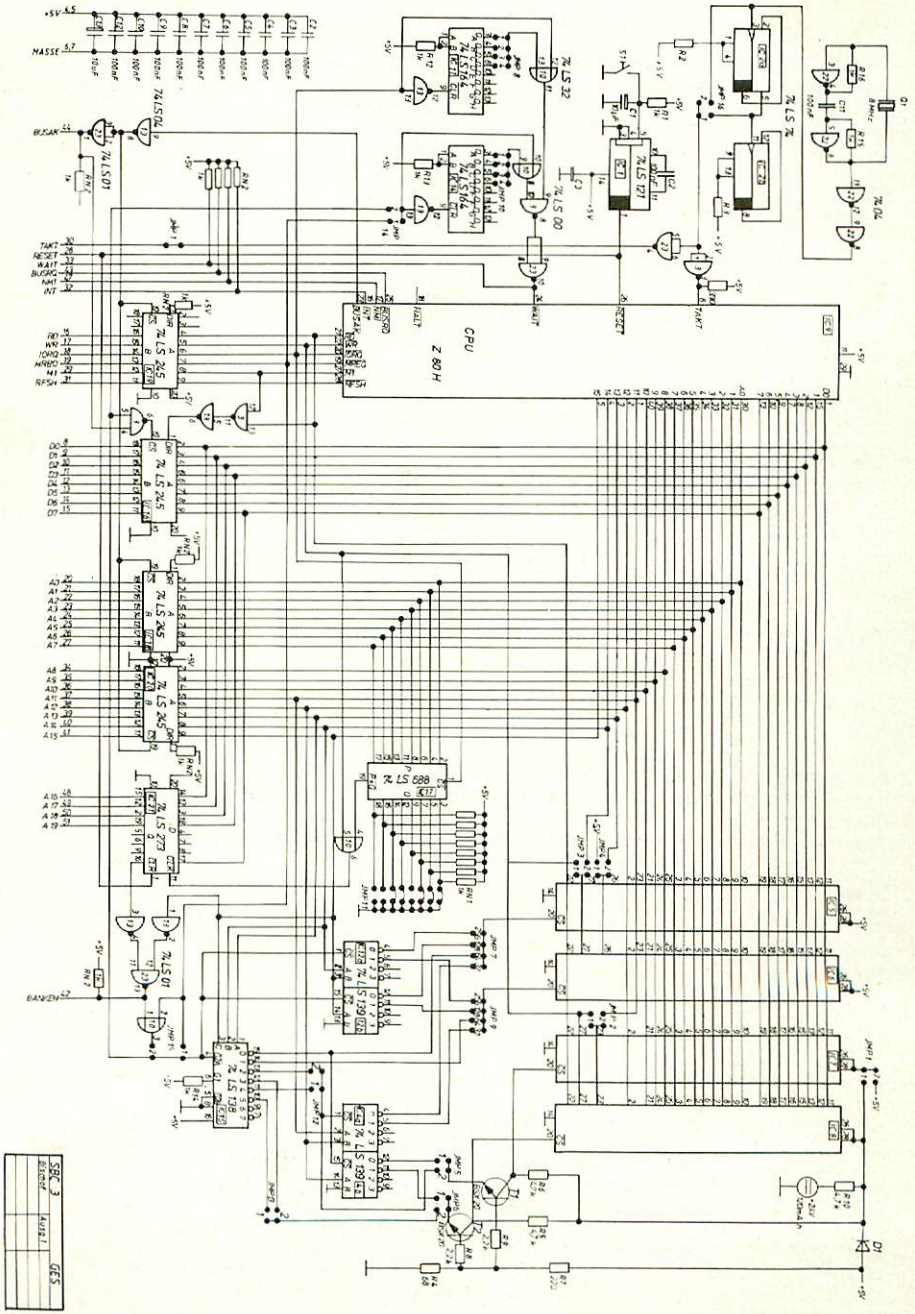
6.1.9 Nehmen Sie alle ICs aus ihren Fassungen. Nehmen Sie sich die Layouts zur Hand und kontrollieren Sie alle Leiterbahnen, mit einem Durchgangsprüfer oder mit einem Ohmmeter, auf Durchgang. Bereits kontrollierte Leiterbahnen können Sie, der Übersicht wegen, auf Layout mit Bleistift durchstreichen oder 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 (IC22) geben dem Takt eine schöne Rechteckform. Die beiden D-Flip-Flop (IC2) teilen die Taktfrequenz jeweils durch zwei. Am Ausgang des ersten Flip-Flops (IC2/5) liegen 4 MHz und am Ausgang des zweiten (IC2/9) 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 (IC1) erzeugt. Das Monoflop wird getriggert mit einem Taster. Um dabei ein prellfreies Triggersignal zu erhalten, wird die Triggerspannung über dem ELKO C1 abgegriffen. Dadurch steigt die Spannung am Elko C1 beim Loslassen des Tasters und beim Einschalten der Spannung langsam an und 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 C2 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 IC14 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 (IC10). 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 (IC23/10) 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 Verfugung 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 (IC21/16) 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* (IC17/19). Dieses Signal wird noch mit WR* ODER verknüpft. Dadurch wird der Ausgang des ODER (IC10/6) nur LOW, wenn auf den Port C8H geschrieben wird. Mit diesem Signal (IC10/6) 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" (IC21/1) 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" (IC21/12) 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 (IC16, IC18, IC19 und IC20). Die Treiber für den Datenbus (IC16) müssen die Daten in beide Richtungen weitergeben können. Daher wird der Eingang DIR (IC16/1) mit dem RD* und dem M1*-Signal der CPU gesteuert. Bei RD ist LOW werden die Daten vom Bus zur CPU übertragen, und bei RD* ist HIGH werden Daten von der CPU auf den Bus übertragen. Die NAND Verknüpfung des Signales RD* mit dem Signal M1 dient lediglich dazu den Eingang DIR beim M1-Zyklus der CPU (siehe Datenblatt CPU) schnell genug zu aktivieren, da dieser Zyklus besonders zeitkritisch ist. 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 von Bus getrennt werden, wenn entweder das Signal BUSAK LOW ist oder wenn auf interne Speicher zugegriffen wird. Dies wird durch eine NAND Verknüpfung der beiden Signale erreicht (IC3/6).
Die Steuersignale und die Adressen müssen nur in einer Richtung verkehren; deshalb kann der Eingang DIR bei diesen LATCHES (IC18, IC19 und IC20) 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 (IC15/4) 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 (IC138/1/2/3) 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 (IC5, IC6, IC7, IC8). Bei Adresse 2000H bis 3FFFH wird z.B. A15 LOW, A14 LOW, A13 HIGH, damit wird der Ausgang 1 (IC15/14) aktiviert und damit das 2. EPROM (IC6). 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 (IC4 und IC12) 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 (IC7/20 und IC8/20) 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 (R5 und R6) fallen nicht ins Gewicht, da der CS Eingang der Speicher extrem hochohmig

ist. Die Basisspannung der Transistoren wird mit R4 und R7 eingestellt. Die Widerstände R8 und R9 (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 R10 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 (R10 = 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.

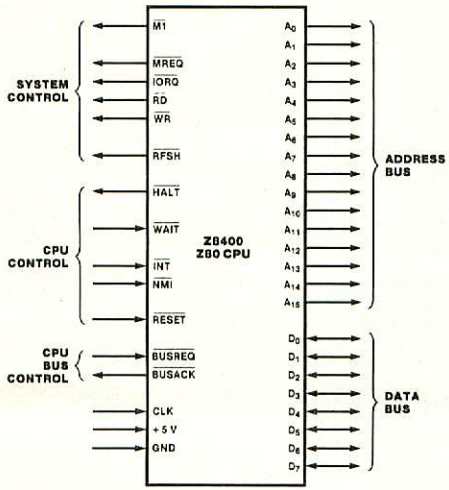


Figure 1. Pin Functions

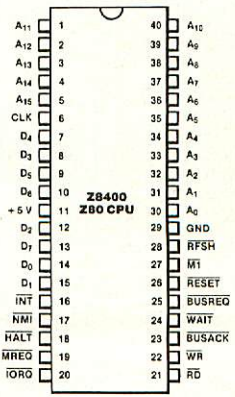


Figure 2a. 40-Pin Dual-In-Line Package (DIP) Pin Assignments

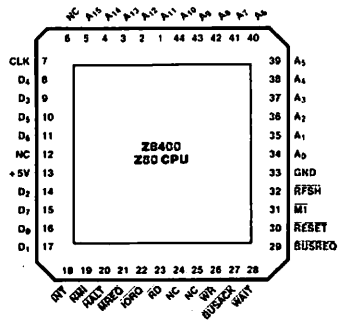


Figure 2b. 44-Pin Chip Carrier Pin Assignments

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.

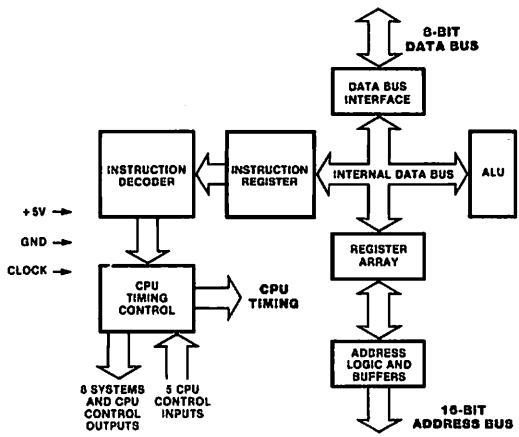


Figure 3. Z80 CPU Block Diagram

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.

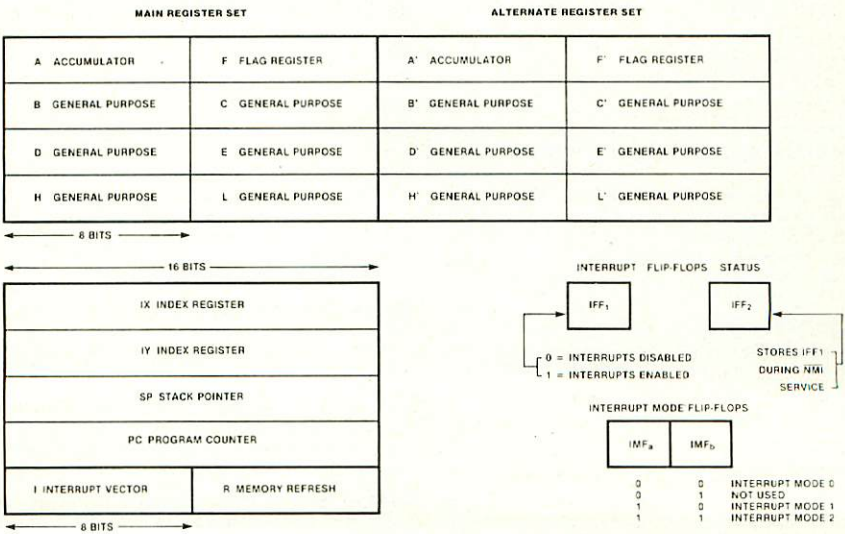


Figure 4. CPU Registers

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	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	X	0	IFF	0	•	11	101	101	ED	2	2	9	
										01	010	111	57				
LD A, R	A ← R	‡	‡	X	X	0	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	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	$qq_H \leftarrow (SP+1)$ $qq_L \leftarrow (SP)$ $SP \rightarrow SP+2$	•	•	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	

Comments	
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	② 0	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	② 0	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										
CPR	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	‡	X	V	0	‡	10	000	r		1	1	4	r Reg.
ADD A, n	A ← A + n	‡	‡	X	‡	X	V	0	‡	11	000	110		2	2	7	000 B
											← n →						001 C
																	010 D
ADD A, (HL)	A ← A + (HL)	‡	‡	X	‡	X	V	0	‡	10	000	110		1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	‡	‡	X	‡	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	‡	X	V	0	‡	11	111	101	FD	3	5	19	
										10	000	110					
											← d →						
ADC A, s	A ← A + s + CY	‡	‡	X	‡	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	‡	X	V	1	‡		010						
SBC A, s	A ← A – s – CY	‡	‡	X	‡	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	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

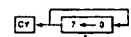
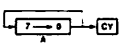
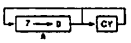
Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	@	‡	‡	X	‡	X	P	•	‡	00	100	111	27	1	1	4	Decimal adjust accumulator.
CPL	A ← Ā	•	•	X	1	X	•	1	•	00	101	111	2F	1	1	4	Complement accumulator (one's complement).
NEG	A ← 0 − A	‡	‡	X	‡	X	V	1	‡	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	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
	mode 0									01	000	110	46				
IM 1	Set interrupt	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
	mode 1									01	010	110	56				
IM 2	Set interrupt	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	8	
	mode 2									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.

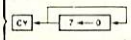
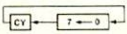
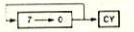
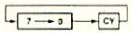
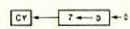
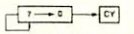
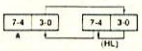
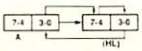
16-BIT ARITHMETIC GROUP

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	ss1	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)

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
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	r Reg.
											00	000	110					000 B
																		001 C
RLC (IX + d)		‡	‡	X	0	X	P/V	0	‡		11	011	101	DD	4	6	23	010 D
	r,(HL),(IX + d),(IY + d)										11	001	011	CB				011 E
												← d →						001 H
											00	[000]	110					101 L
																		111 A
RLC (IY + d)		‡	‡	X	0	X	P/V	0	‡		11	111	101	FD	4	6	23	
											11	001	011	CB				
												← d →						Instruction format and states are as shown for RLCs. To form new opcode replace [000] or RLCs with shown code.
RL m		‡	‡	X	0	X	P/V	0	‡		00	[000]	110					
	m = r,(HL),(IX + d),(IY + d)											[010]						
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	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
															b Bit Tested
BIT b, (IY + d)$_b$	$Z \leftarrow (IY+d)_b$	X	‡	X 1	X	X	0	• 11	111	101	FD	4	5	20	000 0
								11	001	011	CB				001 1
									←d→						010 2
								01	b	110					011 3
SET b, r	$r_b \leftarrow 1$	•	•	X •	X	•	•	• 11	001	011	CB	2	2	8	100 4
								[11]	b	r					101 5
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X •	X	•	•	• 11	001	011	CB	2	4	15	110 6
								[11]	b	110					111 7
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	X •	X	•	•	• 11	011	101	DD	4	6	23	
								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)
																	110 P (sign positive)
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00	011	000	18	2	3	12	111 M (sign negative)
											← e − 2 →						
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	S	Z		H		P/V	N	C	76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11	011	01	DB	2	3	11	n to $A_0 \sim A_7$
											← n →						Acc. to $A_8 \sim A_{15}$
IN r, (C)	r ← (C)	‡	‡	X	‡	X	P	0	•	11	101	101	ED	2	3	12	C to $A_0 \sim A_7$
	if r = 110 only									01	r	000					B to $A_8 \sim A_{15}$
	the flags will																
	be affected																
			①														
INI	(HL) ← (C)	X	‡	X	X	X	X	1	X	11	101	101	ED	2	4	16	C to $A_0 \sim A_7$
	B ← B – 1									10	100	010	A2				B to $A_8 \sim A_{15}$
	HL ← HL + 1		②														
INIR	(HL) ← (C)	X	1	X	X	X	X	1	X	11	101	101	ED	2	5	21	C to $A_0 \sim A_7$
	B ← B – 1									10	110	010	B2		(If B≠0)		B to $A_8 \sim A_{15}$
	HL ← HL + 1													2	4	16	
	Repeat until														(If B = 0)		
	B = 0																
			①														
IND	(HL) ← (C)	X	‡	X	X	X	X	1	X	11	101	101	ED	2	4	16	C to $A_0 \sim A_7$
	B ← B – 1									10	101	010	AA				B to $A_8 \sim A_{15}$
	HL ← HL – 1		②														
INDR	(HL) ← (C)	X	1	X	X	X	X	1	X	11	101	101	ED	2	5	21	C to $A_0 \sim A_7$
	B ← B – 1									10	111	010	BA		(If B≠0)		B to $A_8 \sim A_{15}$
	HL ← HL – 1													2	4	16	
	Repeat until														(If B = 0)		
	B = 0																
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11	010	011	D3	2	3	11	n to $A_0 \sim A_7$
											← n →						Acc. to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11	101	101	ED	2	3	12	C to $A_0 \sim A_7$
										01	r	001					B to $A_8 \sim A_{15}$
			①														
OUTI	(C) ← (HL)	X	‡	X	X	X	X	1	X	11	101	101	ED	2	4	16	C to $A_0 \sim A_7$
	B ← B – 1									10	100	011	A3				B to $A_8 \sim A_{15}$
	HL ← HL + 1		②														
OTIR	(C) ← (HL)	X	1	X	X	X	X	1	X	11	101	101	ED	2	5	21	C to $A_0 \sim A_7$
	B ← B – 1									10	110	011	B3		(If B≠0)		B to $A_8 \sim A_{15}$
	HL ← HL + 1													2	4	16	
	Repeat until														(If B = 0)		
	B = 0																
			①														
OUTD	(C) ← (HL)	X	‡	X	X	X	X	1	X	11	101	101	ED	2	4	16	C to $A_0 \sim A_7$
	B ← B – 1									10	101	011	AB				B to $A_8 \sim A_{15}$
	HL ← HL – 1																
			②														
OTDR	(C) ← (HL)	X	1	X	X	X	X	1	X	11	101	101	ED	2	5	21	C to $A_0 \sim A_7$
	B ← B – 1									10	111	011			(If B≠0)		B to $A_8 \sim A_{15}$
	HL ← HL – 1													2	4	16	
	Repeat until														(If B = 0)		
	B = 0																

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 of 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 operations 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

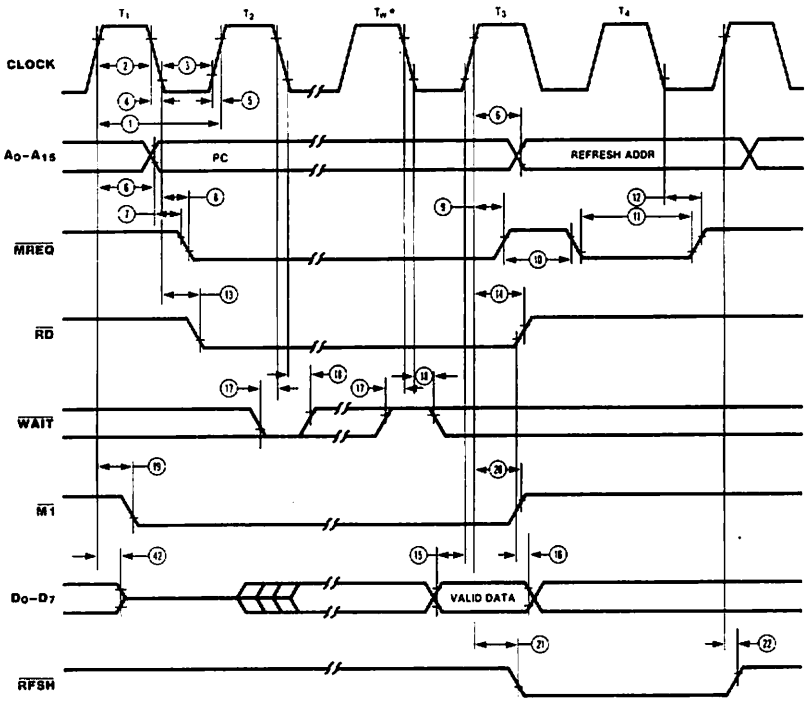
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- 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 ancillary 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.

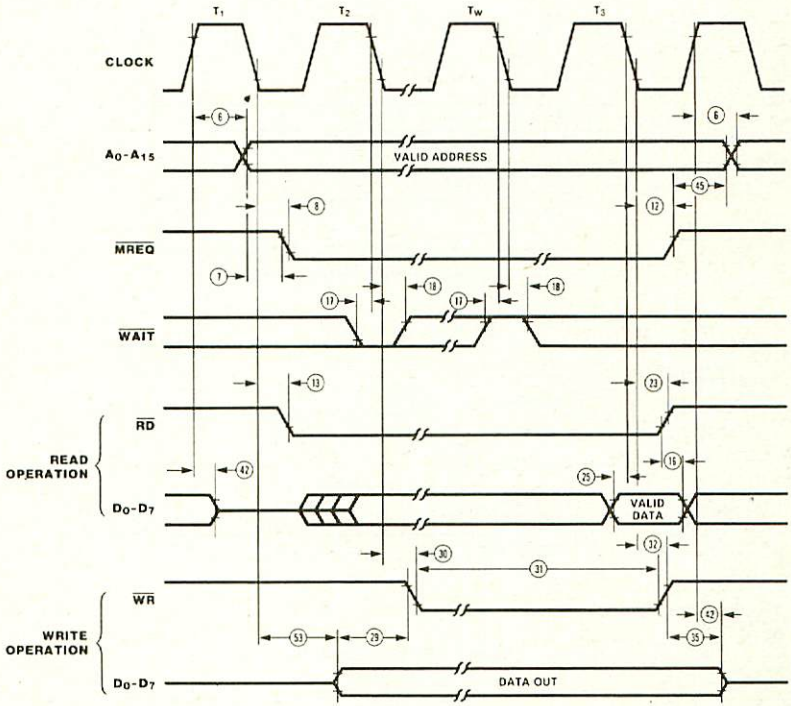
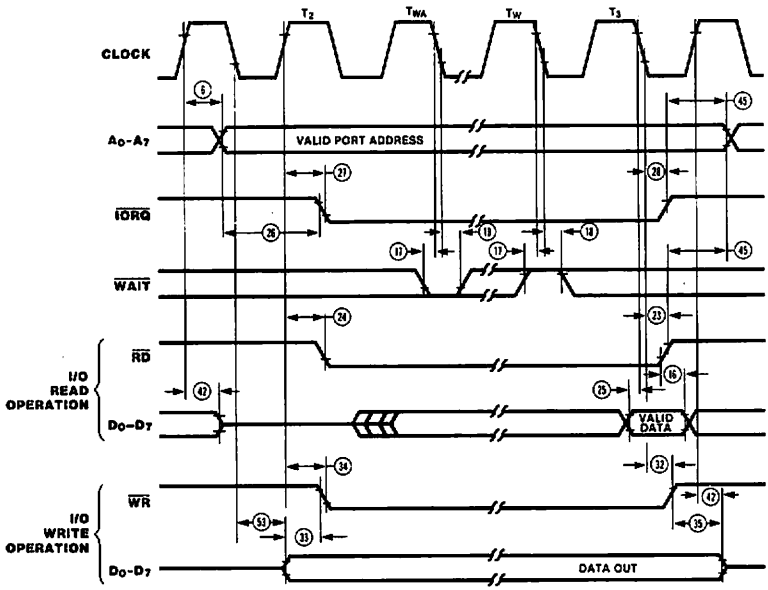


Figure 6. Memory Read or Write Cycles

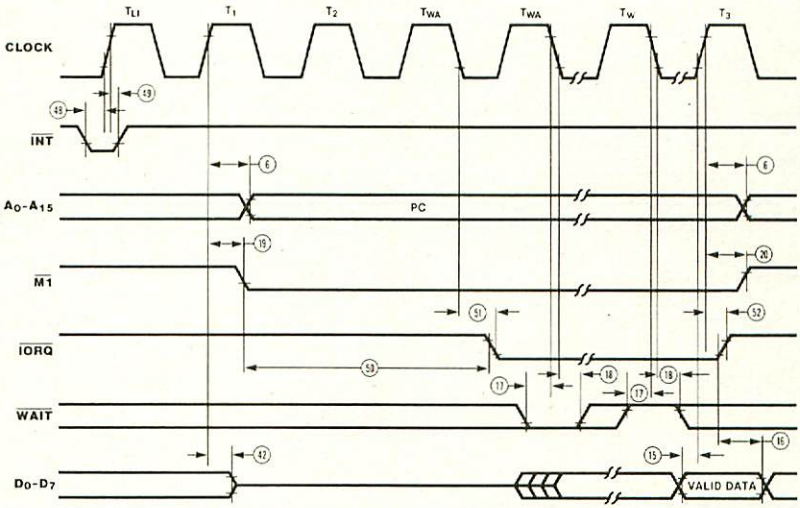
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_{WA}). This extra Wait state allows sufficient time for an I/O port to decode the address from the port address lines.



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

Figure 7. Input or Output Cycles

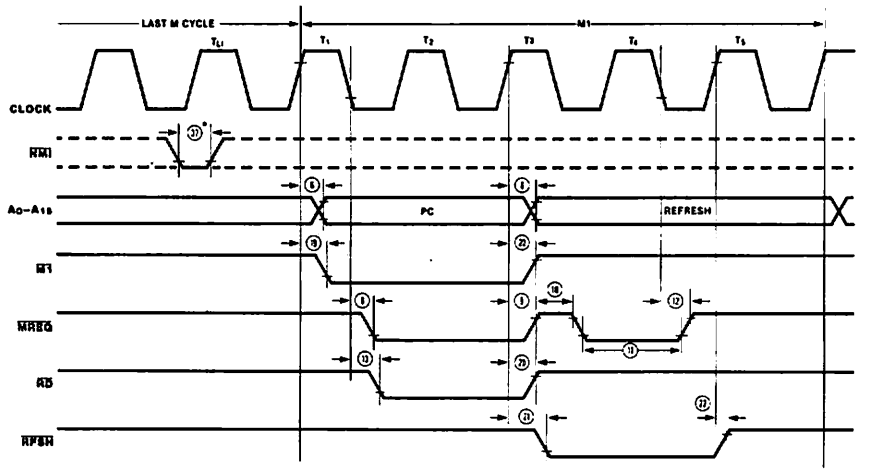
Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (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

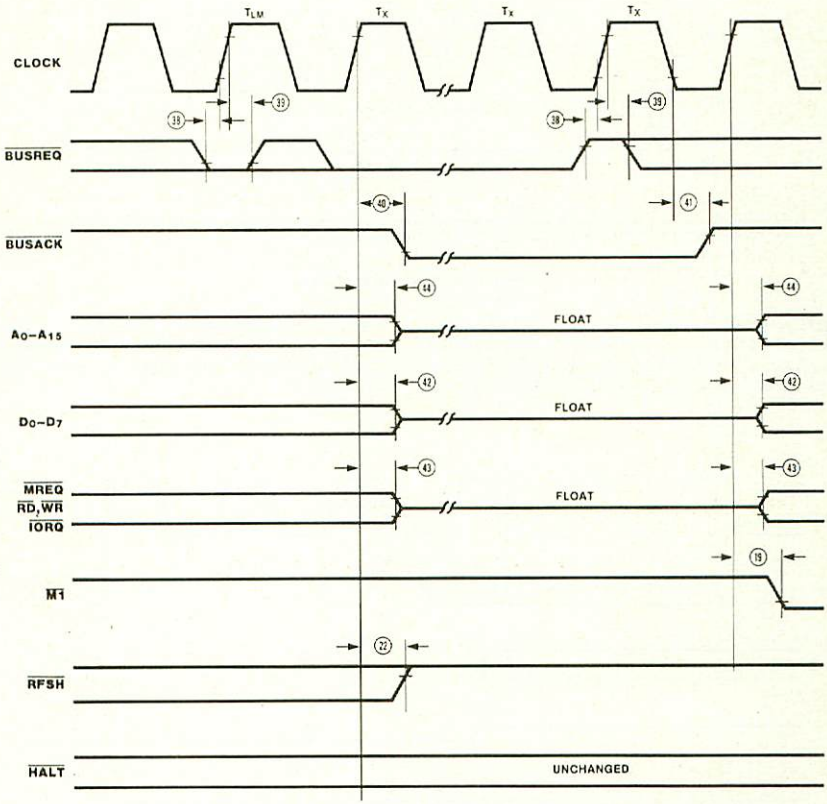
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on 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

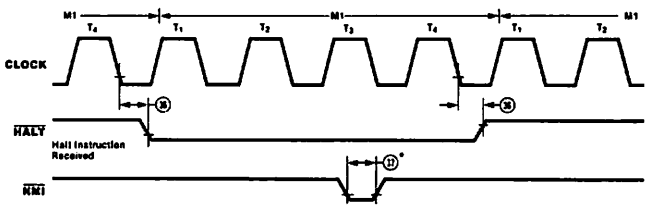
Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to 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_{LO}).

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).

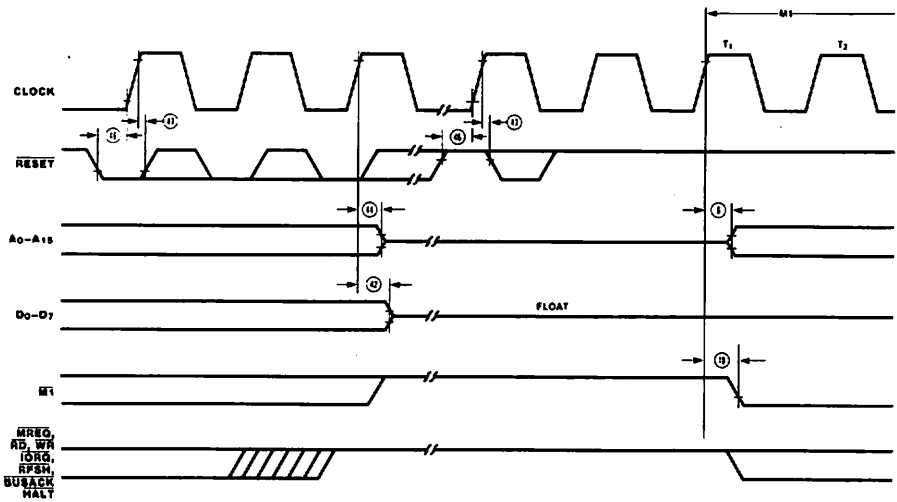


Figure 12. Reset Cycle

AC CHARACTERISTICS†

Number	Symbol	Parameter	Z80 CPU		Z80A CPU		Z80B CPU		Z80H CPU	
			Min	Max	Min	Max	Min	Max	Min	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*		

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†

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*		

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)

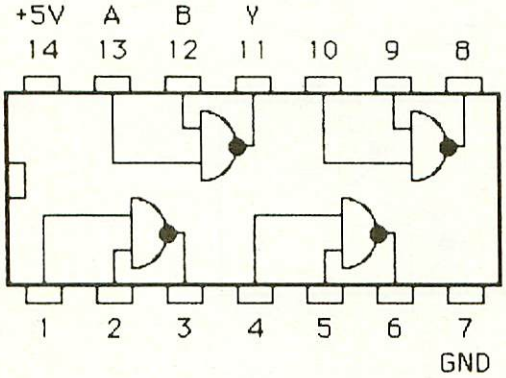
TEMPERATURE
S = 0°C to +70°C
E = −40°C to +85°C
M* = −55°C to +125°C

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

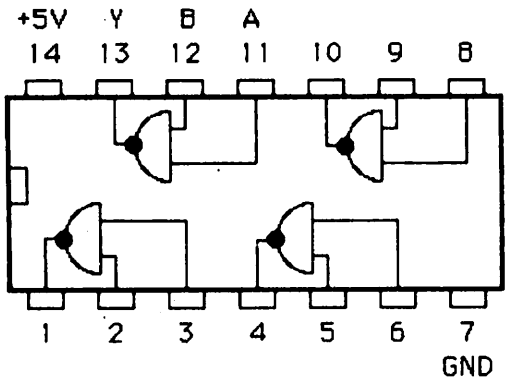
Typ. Impuls-Verzögerungszeit: 9,5 ns

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

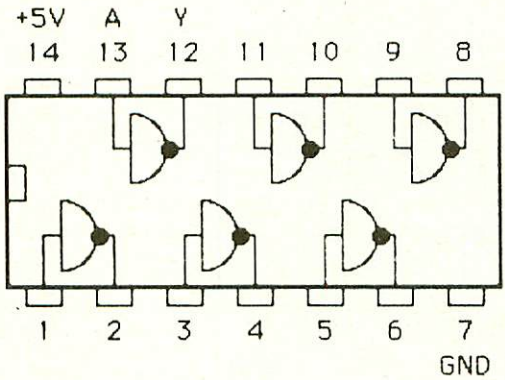
Typ. Impuls-Verzögerungszeit: 22 ns

Typ. Leistungsaufnahme: 40 mW

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

7404

6 Inverter



Logiktabelle

A	Y
0	1
1	0

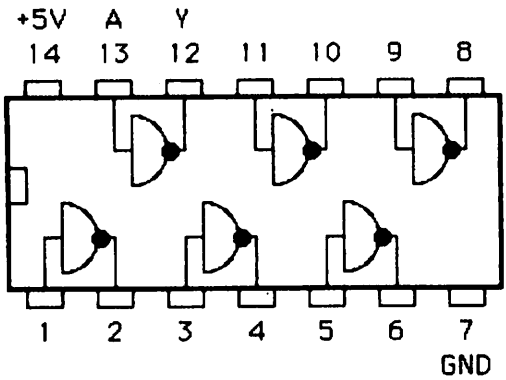
Typ Impuls-Verzögerungszeit: 9 ns

Typ. Versorgungsstrom: 25 mA

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

74LS04

6 Inverter



Logiktabelle:

A	Y
0	1
1	0

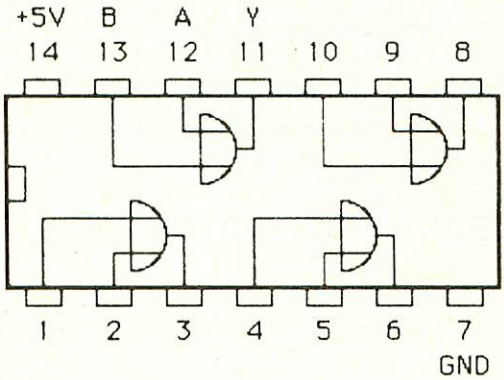
Typ. Impuls-Verzögerungszeit: 10 ns

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

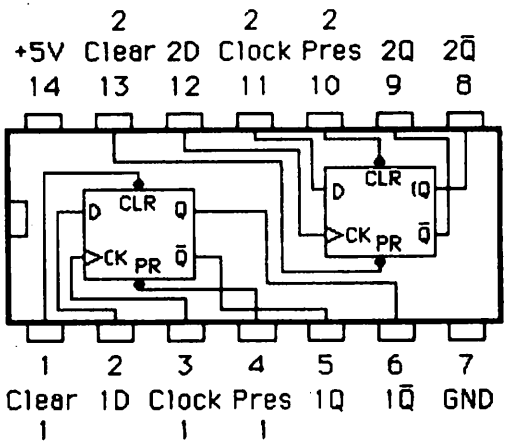
Typ. Impuls-Verzögerungszeit: 12 ns

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

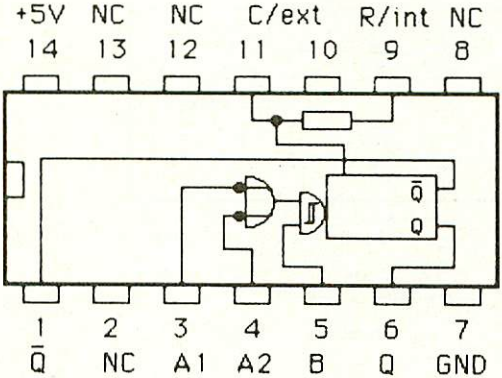
*Dieser Zustand ist nicht stabil; d.h. er bleibt nicht erhalten, wenn Preset und/oder Clear inaktiv (High) werden.

Typ. Impulsverzögerungszeit : 17 ns

Typ. Versorgungsstrom : 16 mA

74LS121

Monoflop mit Schmitt-Trigger-Eingang



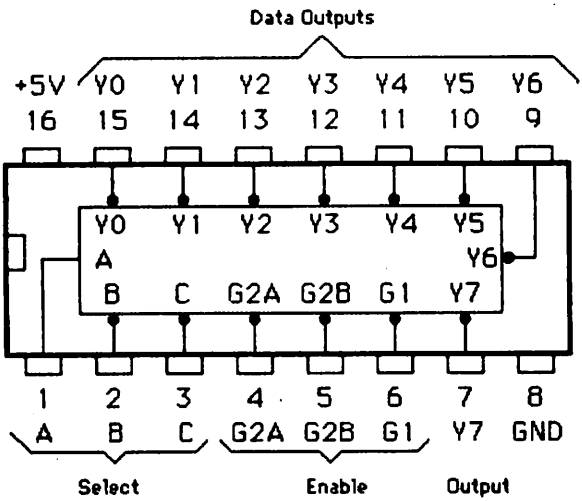
Wahrheitstabelle:

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

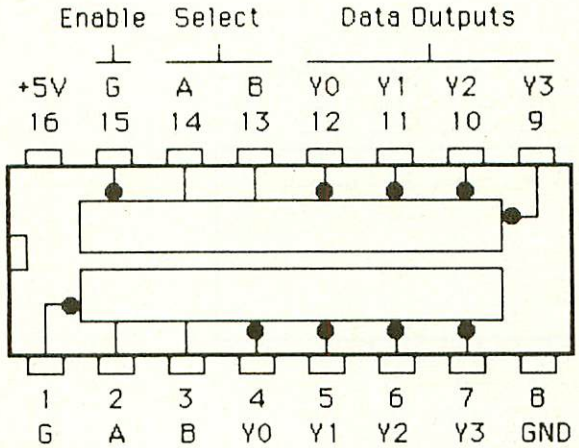
Positive Logik *G2 = G2A + G2B

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	L

H = high level
L = low level
x = irrelevant

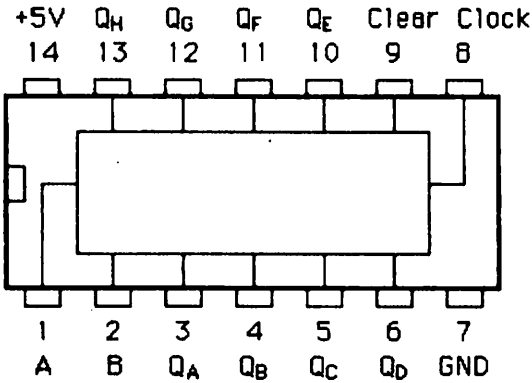
Typ. Impuls-Verzögerungszeit: 20 ns

Typ. Versorgungsstrom: 7 mA

positive Logik: siehe Tabelle

74LS164

Schieberegister mit 8-Bit paralleler Ausgabe



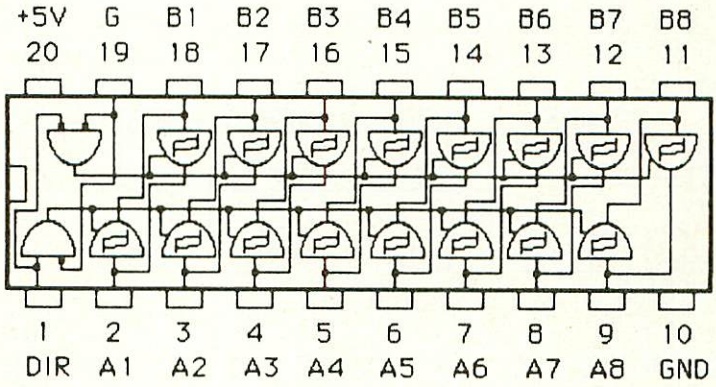
Function Table:

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:

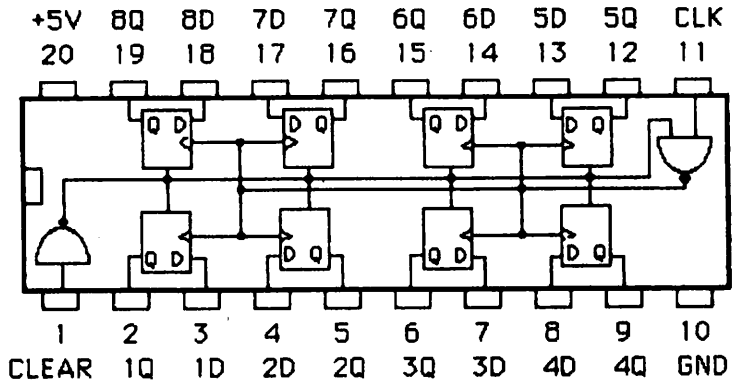
Enable $\overline{G}$	Direktion DIR	Operation
L	L	B data to A Bus
L	H	A data to B Bus
H	X	Isolation

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

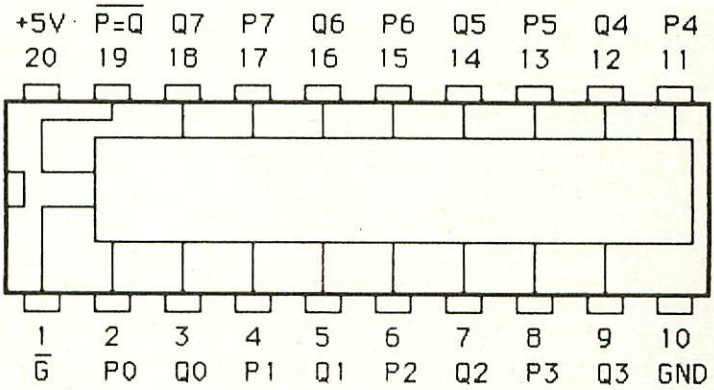
positive Logik

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

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

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

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang
Öffnungszeiten:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
langer Samstag 10.00 – 14.00 Uhr