1 Ausgabe
Text. Norbert Grotz, Kempten, Harald Fink, Rettenberg
Layout. Harald Fink
Druck und Bindung Druckerei Rieder, Kempten
Bestellnr 11315

Erstellt mit "Microsoft Word" und "PageMaker" auf einem mc-modular AT/386

Inhaltsverzeichnis

1.2 Wozu dient die Baugruppe

Die Baugruppe KEY 3 ist in erster Linie das Bindeglied zwischen dem Mikrocomputer (SBC2, SBC3, CPU Z80 oder CPU68xx) und einer Tastatur . Mit ihr ist es möglich, sowohl eine herkömmliche ASCII-Tastatur (parallele Tastatur), als auch eine modernere PC oder oder MF-Tastatur (serielle Tastatur) anzuschließen.

Dabei kommen die Vorteile der KEY3 vor allem mit einer seriellen Tastatur zur Geltung, da sie die Möglichkeiten einer solchen Tastatur voll unterstützt.
Aber auch mit einer parallelen Tastatur bietet sie gegenüber den älteren KEY Baugruppen Verbesserungen, wie z. B. den Tastaturpuffer.

Des Weiteren dient die KEY3 zur Aufnahme einer SMART Watch Dies wurde vorgesehen, da die SMART Watch auf einem 8K RAM sitzt, diese RAM-Bausteine aber auf der neueren ROA256/1M von 32k od 256k Bausteinen abgelöst wurden So bestand keine Möglichkeit mehr, die SMART Watch zu betreiben, obwohl sie aber gerade im neuen Multitastking Betriebsystem unbedingt benötigt wird

Auch bietet das akkugepufferte RAM auf der KEY3 die Möglichkeit wichtige Daten, wie z B Setupdaten (Welche Konfiguration, was für Laufwerke, Festplatte .) auch beim Abschalten des Computers zu erhalten

2. Technische Daten

-Tastatur 8-Bit seriell (Pc/XT/MF-Tastatur)
7-Bit parallel (ASCII-Tastatur)

-256 Zeichen Ringpuffer

-40 programmierbare Funktionstasten

-Belegung der Fuktionstasten auch über das Betriebsystem oder das Grundprogramm möglich

-Wiederholungsmöglichkeit für die letzten 7 Befehle

-Tastenvertauschen durch Software möglich

-Bedienung der SMART Watch wird komplett übernommen

-Verwendung als Timer möglich (auch über Interrupt)

3. Funktionsprinzipien von Tastaturen

3.1 ASCII-Code

Da ein Computer keine Buchstaben oder Zeichen, sondern nur Zahlen versteht, ist es notwendig, das ganze Alphabet, alle Sonderzeichen und die Ziffern zu codieren, also für jedes Zeichen eine bestimmte Zahl festzulegen, die im Computer dann anstatt des Zeichens verwendet wird.

Bei den Mikrocomputern hat sich für diese Codierung der ASCII-Code durchgesetzt. ASCII bedeutet American Standard Code for Information Interchange (= Amerikanischer Standardcode für Informationsaustausch)

Dabei wurde von 7 Bit Daten und dem 8ten Bit als Überprüfungsbit (Paritybit), ob die anderen 7 Bit richtig übertragen wurden, ausgegangen. Es stehen so also 128 Codes zur Verfügung. Davon wurden die ersten 32 Codes für Steuerzeichen, wie z.B. CR (Wagenrücklauf) oder LF (line feed) usw. verwendet.

In neuerer Zeit wird das Überprüfungsbit (Paritybit) kaum noch verwendet, da die Übertragung so sicher ist, daß sich eine Überprüfung erübrigt. Von einigen Computerherstellern wird das so freiwerdende 8. Bit mit zur Codierung dazugenommen, so daß 256 Zeichen codiert werden können. Dabei werden die zusätzlichen Codes praktisch ausschließlich für Graphiksonderzeichen verwendet, die nicht unbedingt benötigt werden.

hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII
00	0	NUL	20	32	Space	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

Bild: ASCII Tabelle

3.2 ASCII-Tastataur

Die ASCII-Tastaur war bis vor einiger Zeit die am häufigsten benutzte Mikrocomputertastaur Sie war daher auch die preisgunstigste und wurde deshalb auch im NDR-Computer (mit den Baugruppen KEY und KEY2) verwendet

Bei einer ASCII-Tastatur werden die Tasten schon innerhalb der Tastatur dekodiert und jeder Taste der entsprechende ASCII-Code zugeordnet. Dieser Code wird dann über die Datenleitung zum Computer gesendet. Bei den meisten ASCII-Tastaturen wird diese Ubertragung parallel vorgenommen, d h für jedes der 7 Bit gibt es eine eigene Leitung Aber es ist noch eine Leitung notwendig, um dem Computer zu melden, wenn neue Daten da sind. Die Tastatur legt dann auf diese Leitung einen kurzen Impuls (Strobe) an dem der Computer erkennt, daß jetzt eine Taste gedruckt wurde

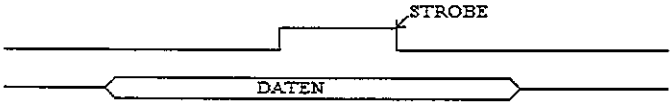


Bild Strobe Timing

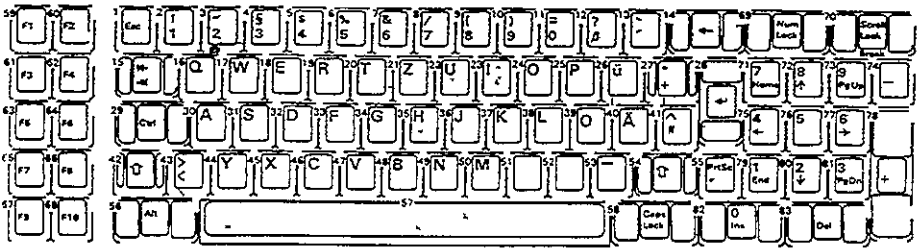
Bei älteren Tastaturen wurde diese Übertragung von der Tastatur zum Rechner auch seriell vorgenommen, d h auf einer Leitung wurden die 7 Bit hintereinander übertragen.

3.3 PC / XT Tastatur

In letzter Zeit hat sich immer mehr die Tastaturart der Personal Computer etabliert. So kam es zu einem Preisverfall dieser Tastaturen, wodurch sie inzwischen gunstiger als ASCII-Tastaturen sind.

Bei diesen Tastaturen wird kein Code für jedes Zeichen (z B ASCII-Code), sondern ein Positionscode fur jede Taste gesendet.

Positionscode heißt, daß eine Taste nicht nach ihrer Bedeutung, sondern nach ihrer Position codiert wird Dabei wird einfach die Taste links oben auf der Tastatur (ESC) mit 1 bezeichnet und dann vorlaufend durchnummeriert.



Dieser Positionscode wird als 8 Bit Datenbyte gesendet und zwar, wenn eine Taste gedrückt wurde (Bit 7 ist dann Null) und nochmal, wenn die Taste wieder losgelassen wurde (Bit 7 ist dann Eins).

Wenn z.B. die Taste "4" gedrückt wird, so sendet die Tastatur den Code 5d (d=deziamal). Beim Loslassen der Taste wird der Code 133d (oder 85h h=hexadezimal) gesendet.

Auch Sondertasten wie "SHIFT" oder "CONTROL" werden genauso behandelt. Dies hat zur Folge, daß z.B. die Unterscheidungen, zwischen Klein- und Großbuchstaben (SHIFT gedrückt) oder zwischen Normal- und Controlebene (Control gedrückt), vom Computer selber übernommen werden müssen und nicht durch die Tastatur erledigt werden, wie bei der ASCII-Tastatur. Dies hört sich zwar im ersten Momment wie ein Rückschritt an, aber dadurch, daß sich viel mehr Möglichkeiten für die Tastatur (z.B. viele Ebenen,verschiedene Belegungen,Funktionstasten ...) ergeben und diese Möglichkeiten auch vom Rechner selber gesteuert werden können, fällt der Nachteil des Mehraufwands für den Computer kaum mehr ins Gewicht.

Auch wird bei einer PC/XT-Tastatur die Übertragung nicht parallel, sondern seriell vorgenommen. Dazu werden zwei Leitungen verwendet. Auf einer Leitung werden die Daten übertragen, auf der anderen Leitung ein Takt, der zur Syncronisation der Daten dient. Bei jeder steigenden Taktflanke wird der momentane Zustand der Datenleitung als ein Bit des Codes übernommen. Dabei werden nicht nur die 8 Datenbit gesendet, sondern immer vor einem Code noch ein Startbit,das immer High ist. So ergibt sich ein 9 Bitwort,dessen erstes Bit allerdings immer eins ist.

Bild: Timing einer Sendung

Da der NDR-Computer aber nicht für die Verwendung einer solchen Tastatur konstruiert wurde, kann diese aufwendige Tastaturkontrolle nicht vom Hauptprozessor übernommmen werden. Um aber trotzdem die Vorteile einer PC/XT-Tastatur (vor allem den Preisvorteil) nützen zu können, wurde auf die neue Tastaturbaugruppe, KEY3, eine eigener Prozessor installiert, der die komplette Tastaturverwaltung übernimmt. Dadurch fällt sogar noch der Mehraufwand für den Hauptprozessor, den eine PC/XT-Tastatur mit sich bringt, weg. So entstehen durch Verwendung dieser Tastatur keine Geschwindigkeitseinbußen, bei kompletter Nutzung der Vorteile solch einer Tastatur.

4. Prinzipbeschreibung

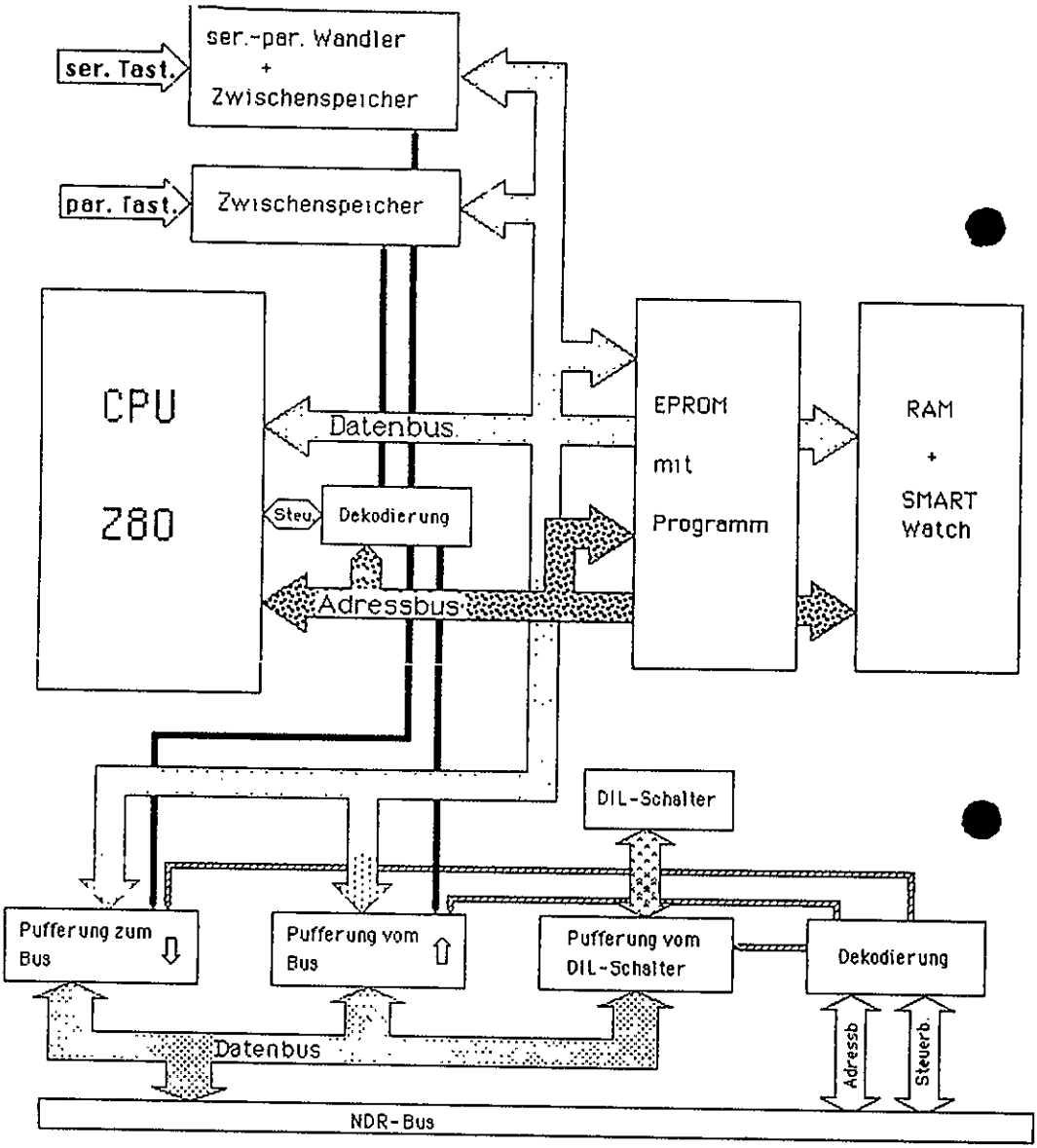
4.1 Tastaturteil

4.1.1 Grundsätzlicher Aufbau

Die KEY3 ist eigentlich ein kleiner Einplatinencomputer Sie besitzt einen eigenen Prozessor, hat einen eigenen Speicherbereich, hat ihre eigenen Ein- und Ausgabeports und benötigt natürlich auch ihr eigenes Programm Der Prozessor steuert mit Hilfe des Programms alle Vorgänge auf der Baugruppe

Und wie beim Hauptcomputer auch, sind, um Daten auszutauschen und Adressen anzugeben, hier natürlich auch ein Daten- und Adressbus vorhanden (Baugruppeninterne Busse) Daneben gibt es für die Ports noch Pufferbausteine, die über Steuerleitungen angesprochen werden.
Zum besseren Verständnis der Baugruppe tragen Kenntnisse über eine andere Prozessorplatine (SBC2, SBC3, CPU680XX) bei

4.1.2 Blockschaltbild



4.1.3 Beschreibung des Blockschaltbildes

Links oben im Blockschaltbild sind die Schnittstellen zu den Tastaturen als Pfeile dargestellt. Der obere Eingang dient zum Anschluß einer seriellen PC/XT-Tastatur Die ankommenden seriellen Daten werden hier mit Hilfe eines Schieberegisters in parallele Daten umgewandelt und zwischengespeichert. Beim Eingang für die parallele ASCII-Tastatur entfällt natürlich die seriell-parallel Wandlung, aber die Daten werden genauso zwischengespeichert.

Die in den Zwischenspeichern liegenden Daten können nun vom Prozessor (CPU Z80) abgeholt und weiterverarbeitet werden Um die Daten zu holen, wird vom Prozessor über die Dekodierung der entsprechende Zwischenspeicher angesprochen Der angesprochene Zwischenspeicher legt nun seine Daten auf den Datenbus und sie werden dort vom Z80 abgeholt.

Damit der Prozessor überhaupt weiß, daß er Daten holen soll, benötigt er ein Programm Dieses Programm steht im Eprom

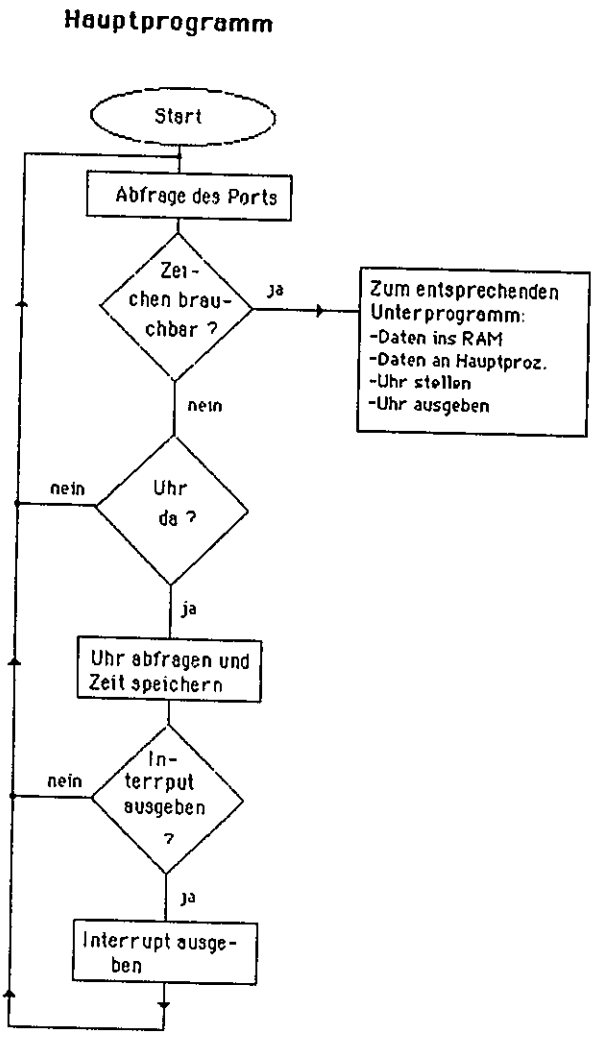
Das Programm ist auch zuständig für die Weiterverarbeitung der eingelesenen Daten. Die Daten vom seriellen Eingang werden zuerst vom Positionscode in den ASCII-Code umgewandelt; dabei werden Sondertasten erkannt und gesondert behandelt (z B SHIFT,ALT oder auch die Funktionstasten usw) Die nun erhaltenen Daten werden im RAM zwischengespeichert. Bei den Daten vom seriellen Port ist eine Umwandlung oder gesonderte Interpretation nicht notwendig Sie werden unverändert zwischengespeichert.

Die im RAM zwischengespeicherten Daten werden vom Z80 nacheinander immer dann an die Pufferung zum Bus weitergegeben, wenn der Hauptprozessor zugreift Mit diesem Zugriff bestätigt der Hauptprozessor, daß er die letzten Daten übernommen hat und bereit ist für neue Daten. Diese Art der "Mitteilung" ist auch bei den alten Tastaturbaugruppen KEY1 und KEY2 so vorgesehen und wurde, um kompatibel zu sein, übernommen.

Die Pufferung vom Bus ist notwendig, daß der Hauptprozessor mit dem KEYProzessor kommunizieren kann.

Die zweite Dekodierung ist für den Hauptprozessor, damit er über den BUS auf die Pufferungsbausteine zugreifen kann und so Daten holen oder abliefern kann.

4.1.4 Flußdiagramme des Programms



4.1.5 Beschreibung des Flußdiagramms

Das Flußdiagramm gliedert sich in drei voneinander unabhänige Teile. Die drei Programmteile Hauptprogramm, Interruptprogramm und NMI-Programm sind deshalb unabhängig voneinander, da sie sich nicht gegenseitig aufrufen. Nomalerweise befindet sich der Prozessor im Hauptprogramm, in dem sich kein Sprung in ein Unterprogramm befindet. Die Unterprogramme können nur durch einen Interrupt aufgerufen werden. Das Interruptprogramm durch den normalen, sperrbaren Interrupt (INT) und das NMI-Programm durch den nicht sperrbaren Interrupt (NMI = Non Maskable Interrupt).

Das Flußdiagramm des Hauptprogramms ist einfach zu durchschauen. Es werden hier eigentlich nur der Eingangsport (im Blockdiagramm: Pufferung vom Bus) abgefragt. Wenn hier ein bestimmtes Zeichen anliegt, wird zum entsprechenden Unterprogramm verzweigt. So z.B. zum Unterprogramm "Daten ins RAM einlesen".

Falls eine SMART Watch vorhanden ist, wird auch diese im Hauptprogramm laufend abgefragt, um immer auf dem aktuellen Stand zu sein und immer, nach einer einstellbaren Zeit, einen Interrupt auf den Bus zu legen (wenn freigegeben).

Etwas komplizierter ist da schon das Flußdiagramm des Interruptprogramms. Hier wird, nachdem die Register gesichert wurden, das Zeichen vom Zwischenpuffer eingelesen und über die Tabelle in ein ASCII-Zeichen umgewandelt. Danach wird abgefragt, ob es ein normales ASCII-Zeichen ist. Wenn ja, wird es gleich in den Puffer geschrieben und falls der Hauptprozessor gerade bereit ist, auch gleich ausgegeben. Ist es kein normales ASCII-Zeichen, wird abgefragt, was für ein Sonderzeichen es ist.

Bei den Sonderzeichen, die in eine andere Ebene umschalten (z.B. SHIFT od. CONTROL), wird der Zeiger auf die Umwandlungstabelle verändert, so daß in der Tabelle eine andere Ebene angesprochen wird. Auch werden die Sonderzeichen beim Loslassen beachtet und der Zeiger wieder auf den normalen Wert gebracht.

Die dritte Möglichkeit ist, daß eine Funktionstaste gedrückt wurde. Hierbei wird noch unterschieden, ob sie nur ausgegeben werden soll, oder ob sie programmiert wird.

Beim Ausgeben wird einfach aus einer anderen Tabelle ein Zeichen nach dem anden n geholt und genauso weiterbehandelt, als ob dieses Zeichen gedrückt worden wäre.

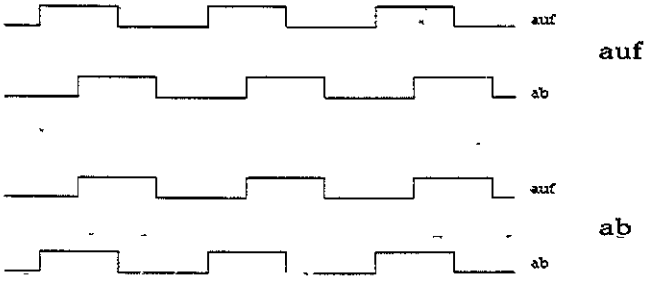
Beim Programmieren werden alle Zeichen, die von der Tastatur kommen, solang die Funktionstaste gedrückt bleibt, in die Tabelle geschrieben und stehen so auch später zur Verfügung.

Der letzte Teil ist das NMI-Programm. Es wird aufgerufen, wenn der Hauptprozessor das Gültigkeitsbit, das Bit 7, ungültig machen will. Dies ist notwenig, daß er dann ein neues Zeichen als neu erkennt. (Sonst würde er nicht erkennen, wenn mehrmals hintereinander dieselbe Taste gedrückt wurde.)

Das Bit 7 wird durch einen Zugriff des Prozessors auf den Dil-Schalterport, sprich Port 69H ungültig gemacht. Dann wird überprüft, ob sich im Tastaturpuffer noch nicht ausgegebene Zeichen befinden. Ist dies der Fall, wird gleich das nächste Zeichen ausgegeben.

4.2 Mausteil

Eine einfache Maus oder ein einfacher Trackball besitzt 4 TTL-Ausgänge, entsprechend den vier möglichen Bewegungsrichtungen (rechts, links, auf und ab) Bei einer Aufwärtsbewegung der Maus erscheinen Rechtecksignale an den beiden Ausgängen für auf und ab Die Anzahl der ausgesandten Impulse wächst proportional mit dem zurückgelegten Weg Bei einer Abwärtsbewegung der Maus erscheinen an den beiden genannten Eingängen ebenfalls Rechteckimpulse Wie läßt sich nun die Bewegungsrichtung ermitteln? Die beiden Signale einer Bewegungsrichtung (auf und ab bzw links und rechts) weisen eine Phasenverschiebung zueinander auf Anhand dieser Phasenverschiebung kann zum Beispiel die Unterscheidung einer Auf- oder Abwärtsbewegung erfolgen Bewegt sich nun die Maus nicht rein waage- oder senkrecht, so kann aus der Zahl der empfangenen Impulse in X- und Y-Richtung die Bewegungsrichtung ermittelt werden und somit eine Positionsbestimmung erfolgen



5. Aufbauanleitung

5.1. CMOS-Warnung

CMOS-Bausteine sind hochempfindlich gegen elektrostatische Aufladung! Verwenden Sie zur Aufbewahrung und zum Transport von CMOS-Bausteinen immer den leitenden Schaumstoff! Alle Pins müssen kurzgeschlossen sein. Bei der KEY3 wird der CMOS-Baustein 6264 (RAM-Speicher) verwendet.

Tip: Fassen Sie an ein geerdetes Teil (z.B. Heizung, Wasserleitung) bevor Sie einen Baustein berühren.

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

5.2.Stückliste

Prüfen Sie zunächst den Bausatz auf Vollständigkeit .

Stückliste KEY3 Ausgabe 1

1 Orginal GES-Leiterplatte mit Lötstoplack r1

1 Handbuch KEY3 Ausgabe 1

7 Sockel 14-polig
9 Sockel 16-polig
6 Sockel 20-polig
2 Sockel 28-polig
1 Sockel 40-polig

2 556
1 74[illegible]4
1 74LS05
1 74LS32
1 74LS125
5 74LS138
1 74LS164
3 74LS245
3 74LS374
4 74LS590
1 RAM 8K NDR statisch (...64)
1 EPROM KEY3 V1.0
1 CPU Z80A 4MHz

14 Con. 100nF Keramik
8 Con. 10nF
1 Con. 10uF Tantal
5 Con. 22nF Wickel

Fortsetzung Stuckleiste!

4 10KOhm Widerstand

3 4*1KOhm Netzwerkwiderstand
1 8*1KOhm Netzwerkwiderstand
1 4*10KOhm Netzwerkwiderstand

1 Stiftleiste gew 36-polig
1 Stiftleiste gew 18-polig
1 Canon-Buchse 15-polig gew lötbar/Sc
1 DSUb 9-polig gew Stecker m Halter

1 2*3 polige Stiftleiste gerade
1 2*8 polige Stiftleiste gerade
2 3 polige Stiftleiste gerade
3 2 poligen Shuntstecker

1 Quarz 4 000 MHz
1 Dilschalter 8

5.3. Löten und Lotzinn

Die Baugruppe ist sehr eng bestuckt Zum Aufbau ist daher etwas Sorgfalt und ein wenig Erfahrung nötig

Die Leistung des verwendeten Lötkolbens sollte 50 W nicht überschreiten, da sonst Bauteile und die Leiterplatte zu heiß und damit zerstört werden könnten

Beim Lötzinnn ist darauf zu achten, daß es unbedingt ein Flußmittel enthält und nicht dicker als 1 mm ist (z B SN 60 0 75 mm Durchmesser) Das richtige Lötzinn kann viel Arger mit Nachlöten und Fehlersuche ersparen

5.4 Aufbau Schritt für Schritt

Zum Aufbau der Baugruppe benötigen Sie folgendes Werkzeug

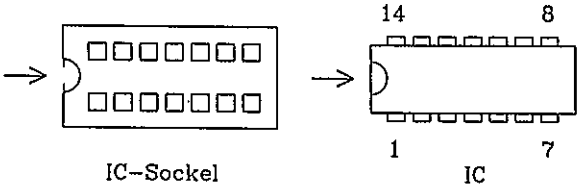
Lötkolben mit temperaturgeregelter Spitze
Lötzinn, säurefrei, mit kolophonium-Seele
Pinzette
Elektroniker-Seitenschneider

Beim Aufbau der Baugruppe sollten Sie sehr sorgfältig arbeiten und sich Zeit lassen Gehen Sie dabei genau nach der Aufbauanleitung vor Am besten ist es, wenn Sie die Anleitung vorher genau durchlesen, um sich einen Uberblick uber die Reihenfolge der Bestuckung zu verschaffen.

Zu Ihrer Orientierung sind bei jedem Abschnitt Kontrollpunkte angegeben, die Sie abhaken können, wenn Sie einen Abschnitt bearbeitet haben.

Die Bauteile werden auf der Bestückungsseite bestückt. Auf der anderen Seite der Leiterplatte, der Lötseite, wird ausschließlich gelötet. Sie erkennen diese Seite an der Aufschrift "Lötseite".

Beginnen Sie mit dem Einlöten der IC-Sockel. Bestücken Sie die Leiterplatte zuerst mit allen Sockeln von der Bauteilseite her. Schauen Sie schon beim Bestücken nach, ob die Lage der Sockel richtig ist (Kerbe in der Fassung muß in dieselbe Richtung weisen, wie die Nase auf dem Bestückungsplan (Siehe Skizze).



Markierung an IC und Sockel

Beachten Sie auch, daß der Platz für den DIL-Schalter (S1) freibleibt. Wenn Sie alle Sockel eingesetzt haben, decken Sie die Seite mit einem Karton ab und drehen die Leiterplatte um. Der Karton hilft Ihnen, die Sockel beim Umdrehen der Leiterplatte in der gewünschten Position zu halten. Verlöten Sie nun von jedem Sockel, am besten diagonal zueinander, zwei Beinchen. Bevor Sie nun die restlichen Beinchen verlöten, drehen Sie die Leiterplatte um und kontrollieren sie nach den folgenden Gesichtspunkten.

( ) Schauen Sie noch einmal nach, ob die IC-Sockel sich auf ihrem richtigen Platz befinden.

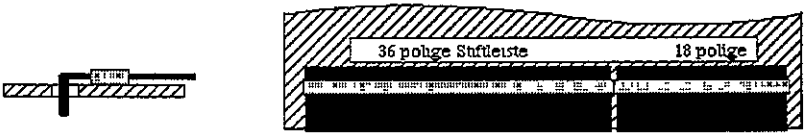
( ) Überprüfen Sie , ob die Kerben der Sockel in die gleiche Richtung weisen, wie auf dem Bestückungsplan.

( ) Die Sockel sollten auf der Leiterplattenoberfläche aufliegen.

( ) Wenn Sie Wert auf eine sehr gute Optik legen, dann können Sie die Sockel so ausrichten, daß sie parallel zu den Seitenkanten verlaufen.

Wenn Sie sich von der richtigen Lage der Sockel überzeugt haben, können Sie diese jetzt vollständig einlöten.

Wenn Sie dies erledigt haben, können Sie die NDR-Leiste aufbringen Gehen Sie ähnlich vor wie bei den IC-Sockeln Die NDR-Leiste setzt sich aus einer 18 poligen und einer 36 poligen gewinkelten Stiftleiste zu einer 54 poligen Leiste zusammen (Siehe Skizze)



AUFBAU NDR-LEISTE

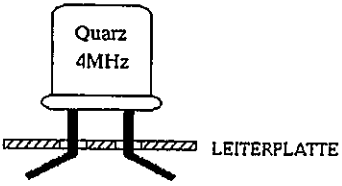
Setzen Sie diese Leisten ein und löten sie am besten nur drei Beine je Leiste fest. Kontrollieren Sie, ob die Leiste nun parallel zur Kante verlauft Wenn nicht, korrekieren Sie dies, indem sie die Leiste leicht in die entsprechende Richtung drucken und den gunstigsten Pol festlöten Wenn die Leiste die gewunschte Parallelität erreicht hat, können Sie die restlichen Beine verlöten

Jetzt können Sie die großen Bauteile einlöten Bei DIL-Schalter verfahren Sie genauso, wie bei den IC-Sockeln Wie Sie sehen können ist der DIL-Schalter ein Bauelement, der mehrere Schalter beinhaltet Diese Schalter sind nummeriert. Der Schalter 1 muß auf der Seite liegen, auf der die Kerbe im Bestuckungsplan eingezeichnet ist (Siehe Skizze)

Der PC/XT-Anschluß (ST4) wird mit dem Aufbringen auf der Platine fest justiert und kann sofort verlötet werden

Fur den Mausanschluß (ST3) ist es ratsam , die Canon-Buchse mittels der Schrauben ebenfalls auszurichten und fest zu justieren, und ihn dann erst zu verlöten

Als letztes Bauelement wird nun der Quarz aufgebracht. Dies geschieht zur Sicherheit des Quarzes, da dieser durch das häufige Umdrehen der Leiterplatte beschädigt werden könnte. Beim Quarz müssen Sie nicht auf die Polung achten. Er wird direkt stehend auf der Leiterplatte aufgebracht (Siehe Skizze).



EINLÖTEN DES QUARZES

6. Testanleitung

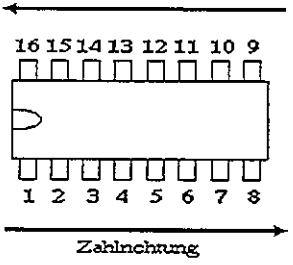
6.1 Erste Prüfung ohne IC's

Der erste Test der Baugruppe sollte immer ohne IC's ausgeführt werden, um eine Zerstörung derselben zu verhindern. Die Leiterplatte ist demnach nur mit den IC-Sockeln und den passiven Elementen bestückt. Sollten sie die IC-Bausteine schon eingesetzt haben, bitten wir sie, diese wieder zu entfernen. Überprüfen sie zunächst den ohmschen Widerstand zwischen Masse (NDR-Leiste/Pin 6, 7, 52, 53) und der Versorgungsspannung +5V (NDR-Leiste/Pin 4, 5) Hier darf der gemessene Wert keinesfalls niederohmig sein. Werte um die 1KOhm sind normal. Wenn sie einen abweichenden Wert feststellen, fahren sie mit dem Kapitel 7 (Fehlersuche) fort.

Sollten sie ein lauffähiges System haben, d h nach dem Einschalten erhalten sie eine Meldung auf dem Bildschirm, stecken sie die halbfertige KEY3 dazu (**vorher ausschalten !!!**).
Wenn sie jetzt das System wieder einschalten, muß dieses noch weiterhin problemlos funktionieren (Test der Daten- und Adressleitungen auf Kurzschluß) Wenn nicht, weiter mit Kapitel 7.

Nun beginnen sie damit, die Spannungsversorgung +5V der einzelnen IC-Bausteine zu messen. Bei den Standard TTL-Bausteinen liegt diese jeweils auf dem letzten Pin der Fassung (z.B bei 14 poligen an Pin 14)

Zu zählen beginnen sie auf der rechten Seite der Einkerbung, und zwar der Reihe nach auf dieser Seite hoch (Siehe Skizze). Wenn sie am letzten Pin auf der rechten Seite angekommen sind, haben sie bei den Standard TTL-Bausteinen den Massepin erreicht. Wenn sie nun weiterzählen mussen, springen sie direkt auf die andere Seite des Sockels und zählen dann auf dieser Seite wieder in Richtung Kerbe Links der Kerbe sitzt nun der Letzte Pin der Fassung Hier liegt die 5V-Versorgungsspannung an.



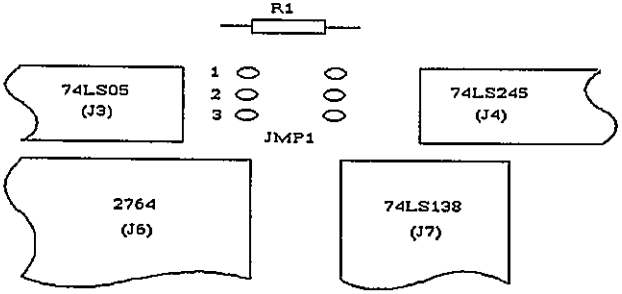
Dieselbe Anordnung gilt für den Speicherbaustein 6264 (J6) und die Timmer 556 (J12, J16)
Beim Prozessorbaustein Z80A (J1) liegt die Versorgungsspannung an Pin 11 an, die Masseverbindung an Pin 29
Beim EPROM muß die Versorgungsspannung +5V am Pin 28 des Sockels liegen, die Masseverbindung an Pin 14 des Sockels

Wenn sie alle Spannungen überprüft haben, können sie das System abschalten, die KEY3 herausnehmen und die IC-Bausteine einsetzen.

VORSICHT: Stecken sie die IC's richtig herum (d.h. IC-Kerbe über Sockel-Kerbe) auf die entsprechenden Sockel. Kontrollieren sie nachdem sie alle eingesteckt haben, die Lage der IC's lieber noch einmal nach. Schauen sie auch, ob alle IC-Beinchen in den Sockeln stecken. Herausstehende Beinchen bewirken, daß die Baugruppe nicht funktioniert und eine eventuelle Fehlersuche erschwert wird.

6.2. Einsetzen der kompletten Baugruppe

Bevor sie die Baugruppe wieder einsetzen, müssen sie den Jumper JMP1 richtig setzen. Mit diesem Jumper entscheiden sie, ob sie eine PC/XT oder eine ASCII-Tastatur verwenden. Das folgende Bild zeigt diesen Jumper JMP1:

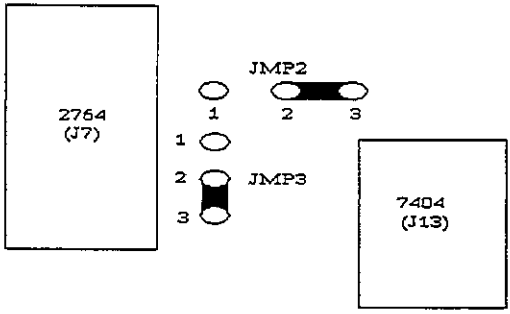


In der Stellung 1 können sie eine ASCII-Tastatur betreiben. Hierbei ist die Voraussetzung, daß die Tastatur ein STROBE-Signal mit HIGH-Pegel aussendet.

Sendet ihre Tastatur das Strobe-Signal mit einem Low-Pegel, müssen sie die Jumperstellung 2 wählen.

Für den Fall, daß sie eine PC/XT Tastatur verwenden, müssen sie die Jumperstellung 3 wählen.

Wie sie wissen, gibt es Speicher mit unterschiedlicher Speicherkapazität. Damit die KEY3 nicht nur mit dem standardmässigen RAM 6264 betrieben werden kann, sondern auch mit dem nächsthöheren RAM, dem 62256, wird der Jumper JMP3 benötigt. Die gleiche Funktion besitzt der Jumper JMP2 für das EPROM. Stanardmässig verwenden wir den EPROM-Typ 2764. Mit Hilfe des JMP2 können auch die Typen 27128 und 27256 eingesetzt werden. Für den Fall, daß Sie einen höherwertigen Speicher oder ein höherwertiges EPROM einsetzen, müssen Sie die Voreinstellung auf der Lötseite der Platine unterbrechen. Auf der folgenden Seite sehen Sie die Standardeinstellung der Jumper und eine Tabelle, wie Sie die Jumper für den jeweiligen Typ setzen müßen.



JUMPERSTELLUNGEN JMP2:

SPEICHER	TYP	STELLUNG
8K * 8	6264	VERBINDUNG 2-3
32K * 8	62256	VERBINDUNG 1-2

JUMPERSTELLUNGEN JMP3:

SPEICHER	TYP	STELLUNG
8K * 8	2764	VERBINDUNG 1-2
16K * 8	27128	VERBINDUNG 1-2
32K * 8	27256	VERBINDUNG 2-3

Sobald die Jumper richtig eingestellt sind, können sie die Baugruppe einstecken. Wenn sie ihr System einschalten, muß die Anzeige nach wie vor problemlos erscheinen Sollte dies nicht der Fall sein, schalten sie das System sofort ab und gehen sie zum Kapitel 7

6.3. Tastaturanschluß

Bevor sie die Tastatur anschliesen, sollten sie zuerst das System wieder ausschalten
Die PC-Tastatur wird an der Steckerleiste ST 4 angeschlossen. Sobald der Stecker eingesteckt wurde, schalten sie das System ein Wenn sie nun die Tasten CAPS-Lock (s Kap 3.3) oder NUM-Lock drucken, sehen sie, sofern die Kontrolleuchten rechts oben auf der Tastatur aufleuchten, ob die Tastatur die erforderliche Spannungsversorgung hat.

Wenn sie eine ASCII-Tastatur anschließen wollen, müssen sie die Verbindung zur Steckerleiste ST2 herstellen Hierbei müssen sie darauf achten, daß der Stecker richtig herum aufgesteckt wird.

6.4 Testen mit dem Z80-Grundprogramm

Wenn Sie das System gestartet haben, wartet der Rechner auf eine Eingabe von Ihnen. Diese Eingabe erscheint unten auf dem Bildschirm in einem kleinen Feld. In diesem Feld erscheint maximal 1 Zeichen. Sobald Sie ein neues Zeichen eingeben, verschwindet das alte Zeichen und der Rechner erwartet eine neue Eingabe. Geben Sie nun das nächste Zeichen ein und kontrollieren Sie, ob das von Ihnen eingebene Zeichen auf dem Monitor erscheint. Gehen Sie auf diese Weise alle Tasten der Tastatur durch.

6.5. Test mit dem FLOMON

Der Tastaturtest mit dem FLOMON verläuft ziemlich einfach. Sie müssen nur die Control-Taste und die C-Taste gemeinsam drücken. Sie befinden sich dann im Testmodus, und können alle Tasten ihrer Tastatur drücken und sehen, ob das entsprechende Zeichen auf dem Monitor erscheint. Verwenden sie eine PC/XT Tastatur und gibt der Rechner Zeichen aus, die nicht dem gedrückten Zeichen auf der Tastatur entsprechen, betrachten sie den Punkt 6.6..

6.6. Amerikanischer-Deutscher Zeichensatz beim FLOMON

Sollte auf dem Monitor falsche Zeichen erscheinen und sie benutzen eine PC/XT Tastatur, besteht die Möglichkeit, daß diese Tastatur den amerikanischen Zeichensatz anwendet. Um den deutschen Zeichensatz einzustellen, drücken sie nochmals die Control und die C-Taste. Anschliesend drücken sie die Escape ("ESC")-Taste. Dann betätigen sie die "z"-Taste und anschließend die "1". Nun ist der deutsche Zeichensatz eingegeben. Wenn sie nun wiederum den Testmodus mit Control C aufrufen, müßen alle Tasten ihr richtiges Signal ergeben. Um den amerikanischen Zeichensatz einzugeben, gehen sie den oben gezeigten Weg, nur anstelle des"1" geben sie den Wert "0" ein.

Zusammenfassung:

deutsch	ESC z 1
amerikanisch	ESC z 0

6.7 Testen mit dem 680xx Grundprogamm

Wenn Sie das Grundprogramm 4.3 besitzen, verläuft der Test wie beim Z80 Grundprogramm. Besitzen Sie jedoch die neuere Version 6.0, gehen Sie folgendermaßen vor. Drücken Sie die Taste "E". Mit diser Taste wählen Sie den Editor aus, in welchem Sie sich nun befinden müssten. Der Editor ist ein einfaches Textverarbeitungsprogramm, mit dessen Hilfe Sie nun alle Tasten der Tastatur durcharbeiten und kontrollieren können.

6.8 Testen mit der CPU 8088

Wenn Sie eine CPU 8088 besitzen, benötigen Sie die KEY3, wenn Sie eine ASCII-Tastatur verwenden. Sie haben nun aber zwei Möglichkeiten eine PC-Tastatur anzuschließen, da die CPU einen eigenen Tastaturanschluß besitzt. Die KEY3 funktioniert hier zwar auch, sinnvoller ist es jedoch, die PC-Tastatur direkt an die CPU anzuschließen.

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 der Schaltungsbeschreibung und dem Schaltplan entnehmen.

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

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

8. Schaltungsbeschreibung
8.1 Tastaturteil
8.1.1 Z80 - Mikroprozessor - Herz der Baugruppe

Wie schon in der Prinzipbeschreibung gesagt, ist diese Baugruppe eigentlich ein eigener kleiner Computer. Wie bei allen Computern gibt es auch hier einen Prozessor, den Z80 von Zilog (IC1, Z80).

Dieser Prozessor übernimmt auf der KEY 3 den Löwenanteil der Arbeit.

Er liest von der Tastatur kommende Daten, entweder vom parallelen Port (IC5, 74LS374) oder vom seriellen Port (IC4, 74LS245) ein, wandelt die Daten (wenn notwendig) in, für den Computer verständliche, ASCII-Zeichen um (siehe 3.), speichert sie im Tastaturpuffer zwischen, erkennt Funktionstasten und Sondertasten und gibt die Zeichen über einen Puffer (IC 17, 74LS374) an den Bus des Hauptprozessors weiter. Nebenbei kontrolliert er noch, ob der Hauptprozessor ihm (über einen weiteren Puffer, IC18, 74S374) etwas mitteilen will, und reagiert darauf, indem er z.B. die Belegung der Funktionstasten ändert.

Mit zum Prozessorteil gehören auch die Taktlogik und die Speicherbausteine.
Die Taktlogik besteht aus IC13 74LS04, dem Quarz (X1, 4MHz), zwei Widerständen (R3,R4, 1K) und einem Kondensator (C3, 100nF). Durch beschalten (siehe Schaltplan) des Quarzes mit den anderen Bauteilen wird der Quarz in Schwingung versetzt. Da er nur mit seiner eingeprägten Frequenz von 4MHz schwingen kann, liefert er so den benötigten Takt von 4MHz an den Prozessor (IC1, Z80, Pin1).

Zum Betrieb eines Prozessors wird natürlich auch Speicher benötigt. Im Speicher steht einmal das Programm, ohne das der Prozessor nicht weiß was er tun soll. Außerdem wird der Speicher zur Zwischenspeicherung von Daten (z.B. für Tastaturpuffer) benötigt.

Um das Programm beim Ausschalten des Computers nicht zu verlieren, muß es in einem EPROM gespeichert sein. In einem EPROM gehen Daten auch ohne Versorgungsspannung nicht verloren, aber dafür können in ein EPROM im normalen Betrieb (der hier gegeben ist) keine Daten geschrieben werden, sondern nur Daten (die in einem besonderen Verfahren hineinprogrammiert wurden) ausgelesen werden. Im diesem EPROM (IC7, 2764) ist das Programm gespeichert.

Da wie gesagt in ein EPROM nichts geschrieben werden kann, wird noch ein RAM (IC6, 8464) benötigt. Beim RAM kann geschrieben und gelesen werden, aber beim Abschalten gehen alle gespeicherten Daten verloren. Die KEY3 wird mit den 8KByte-Typen 2764 und 8464 geliefert. Durch Auftrennen und Umstecken der Jumper 2 u. 3 können aber auch 32K-Typen (27256 u. 43256) verwendet werden.

Die beiden Speicherbausteine liegen direkt am interen Datenbus der Baugruppe. Vom internen Adressbus sind die Adressen A0-A13 direkt verbunden. Das RAM fühlt sich nur angesprochen, wenn A15 auf high liegt (über IC3, 74LS05). Es ergeben sich damit für das EPROM der Adressbereich von 0 - 1FFFH (- 7FFFFH beim 27256) und für das RAM von 8000H - 9FFFH (- FFFFFH beim 43256).

Die beiden OR-Gatter (IC2 74LS32) sorgen dafür, daß die beiden Bausteine nur auf Speicherzugriffe des Prozessors (IC1, Z80, Pin19 MERQ ist dann aktiv = low) reagieren.

8.1.2 Adressdekodierungen

Da die Baugruppe sowohl einen eigenen Prozessor hat, als auch über den Bus mit dem Hauptprozessor verbunden ist, sind zwei Adressdekodierungen notwendig

Durch die interne Dekodierung (IC8, 74LS138) werden die Adressen der verschiedenen Puffer und Ports für den internen Prozessor festgelegt.

Durch die zweite Dekodierung (IC14,15,10,11 alle 74S138) werden die Adressen der Baugruppe (einschließlich Mausteil) aus Sicht des Hauptprozessors festgelegt

Tabelle

	Port	Adresse	Eingang
Ausgang			
Intern	parallele Tastatur	5	*
	serielle Tastatur 1	*	
	serielle Tast löschen	0	*
	Ausgabe an BUS 2		*
	Einlesen von BUS	3	*
	Ausgaben von INT an BUS	4	*
Als Baugruppe aus Sicht des Hauptprozessors			
	Daten von KEY3 holen	68H	*
	Daten an KEY3 geben	8CH	*
	Daten von DIL-Schalter	69H	*
	Mausteil		
	Taste gedruckt 8BH	*	
	Steps nach oben 8FH	*	
	unten	8EH	*
	links	8DH	*
	rechts	8CH	*
	Vorbelegen	8DH	*
	Löschen	8EH	*

Etwas verwirrend ist vielleicht die Tatsache, daß die Bausteine IC17 u IC18 (beide 74LS374) sowohl intern, als auch vom BUS aus angesprochen werden können, und zwar unter verschiedenen Adressen.

Der Baustein IC17 dient als Puffer zur Zeichenausgabe der Baugruppe an den BUS So kann der interne Prozessor ein Byte, das er ausgeben will, in den Baustein schreiben. Dabei hat der Baustein die Adresse 2. für den internen Prozessor (interne Dekodierung)

Der Hauptprozessor kann ein Byte, das in diesem IC zwischengespeichert wird, abholen, und zwar unter der Adresse 68h (Baugruppendekodierung).

Beim Baustein IC18 ist es genau umgekehrt. Hier wird das Zeichen vom Hauptprozessor hineingeschrieben (in die Adresse 8Ch) und kann vom internen Prozessor (unter der Adresse 3) abgeholt werden.

8.1.3 Ein- und Ausgänge

Um die Verbindung mit der "Außenwelt", sprich zur Tastatur und zum Hauptprozessor zu bekommen, braucht der Prozessor einige Ein- und Ausgänge, sogenannte Ports. Jeder dieser Ports wird über eine bestimmte Adresse angesprochen (siehe 8.1.2).

Der wichtigste Port ist der Eingabeport von der Seriellen Tastatur. Hier werden die seriell ankommenden Daten (siehe 3.3) zuerst durch ein Schieberegister (IC9, 74S164) in parallele Daten umgewandelt.

Beim 74LS164 handelt es sich um ein 8 Bit-Schieberegister,das die an Pin 1 (oder Pin 2) anliegenden Daten bei jeder steigenden Taktflanke übernimmt. Da die Daten aber mit einem Takt geliefert werden, der bei jeder fallenden Flanke gültig ist (siehe 3.3), wird dieser Takt zuerst durch IC3 (74LS05) invertiert.

Obwohl von der Tastatur 9 Bit gesendet werden, ist hier ein 8 Bit-Schieberegister ausreichend, da das erste der gesendeten Bits, das Startbit, nicht zu den Daten gehört. Es wird durch das ganze Schieberegister durchgeschoben und fällt, wenn das letzte Bit ankommt, wieder heraus. So, daß dann genau die 8 Datenbit im Schieberegister stehen. Da nach 9 Bit auch kein Takt mehr gesendet wird, bleiben die Daten stehen, bis das Schieberegister vom internen Prozessor gelöscht wird, oder bis neue Daten von der Tastatur kommen (was aber solange dauert, daß der Prozessor das Schieberegister längst gelöscht hat). Dieses Löschen, das durch ansprechen (schreiben oder lesen) der Adresse 0 geschieht, ist notwendig, da, falls an Ausgang H (Pin 13) des Schieberegister ein High-Pegel liegt, sonst die Interuptleitung des Prozessors blockiert wäre. Doch zurück zum Startbit. Ganz nutzlos ist es in der Schaltung nicht, denn wenn es an der letzten Stelle (Ausgang H, Pin 13) des Schieberegisters angekommen ist, löst es beim Z80 einen Interrupt aus (siehe 8.1.4). Dem Prozessor wird so gemeldet, daß Daten da sind und er diese bei Adresse 1 abholen kann. Dabei wird der Baustein IC4 (74LS245) angesprochen, der dann die Daten auf den internen Datenbus durchschaltet.

Etwas einfacher geht die Sache am Port für die parallele Tastatur vor sich. Hier gibt es den Baustein 74S374 (IC5), der am Eingang anliegenden Daten aufnimmt und zwischenspeichert sobald am CLK-Eingang (Pin 11) eine steigende Flanke auftritt. Diese Flanke wird von der parallelen Tastatur als Strobe (siehe 3.2) mitgeliefert. Außerdem erzeugt dieser Strobe gleich auch noch den Interrupt für den Prozessor.

Dieser kann die Daten jetzt abholen, und zwar bei Adresse 5. Dabei schaltet IC 5 seine zwischengespeicherten Daten auf den Datenbus durch.

Ein Löschen der Daten ist nicht notwendig. Wenn wieder eine Taste gedrückt wird , werden die alten Daten einfach überschrieben.

Dann gibt es noch zwei Ports (IC17, IC18, beide 74LS374) mit dem der interne Prozessor mit dem Hauptprozessor in Verbindung treten kann. Dies ist notwendig da z.B. eine gedrückte Taste dem Hauptprozessor mitgeteilt werden muß.

Es sind zwei Ports notwendig, da beide nur in einer Richtung betrieben werden können und zwar aus der Sicht des internen Prozessors IC17 als Ausgang und IC18 als Eingang.

Da es hier, im Gegensatz zu den Tastaturen, keine Meldeleitung (Interrupt o ä.) gibt, wissen die Prozessoren voneinander nicht , wann der andere etwas will

Beim Ausgang (aus der Sicht des Internen Prozessors) ist das Problem so gelöst. Der Hauptprozessor fragt immer wieder den Port ab, d h. er liest die zwischengespeicherten Daten ein und schaut nach ob sie brauchbar sind (ASCII-Zeichen) Wenn ja, merkt er sich die Daten, die ja für eine gedruckte Taste stehen Wenn er jetzt aber gleich wieder mit dem Abfragen beginnt, erhält er sofort wieder brauchbare Daten,da sich am Port noch gar nichts geändert hat. Für den Hauptprozessor sieht es aber so aus, als sei die gleiche Taste schon wieder gedrückt worden Um dies zu vermeiden bedient er sich eines Tricks Er liest die Stellung des DIL-Schalters ein Dabei interessiert ihn die Stellung der DIL-Schalter überhaupt nicht, aber durch das Ansprechen des DIL-Schalterports wird über die Gatter IC2 (74LS32) u IC13 (74LS04) das höchstwertige Bit am Ausgangsport (IC17) gesetzt.

Wenn der Hauptprozessor jetzt wieder mit dem Abfragen beginnt, erhält er nicht sofort wieder brauchbare Daten, da jetzt das höchstwertige Bit gesetzt ist, was einer Zahl von größer 127 entspricht. (Brauchbare Zeichen sind ASCII-Zeichen, die nur im Bereich von 0-127 liegen) Außerdem wird durch den Zugriff auf den DIL-Schalter noch dem internen Prozessor über die NMI-Leitung (siehe 8 4) mitgeteilt, daß der Hauptprozessor das ASCII-Zeichen abgeholt hat. Der interne Prozessor kann dann gleich das nächste Zeichen aus dem Tastaturpuffer (sofern eines darin ist) ausgeben Für den Hauptprozessor ist dann wieder ein gültiges Zeichen da und der Kreislauf beginnt von vorne

Beim Eingangsport (aus der Sicht des internen Prozessors) ist das Problem anders gelöst. Es wird in Kapitel 10 genau beschrieben

Auch für den Hauptprozessor enthält die Schaltung einige Ports, und zwar die gerade beschriebenen Verbindungsports zum internen Prozessor Sie haben aus Sicht des Hauptprozessors nur andere Adressen Über den DIL-Schalter Port können bestimmte Zahlen, die z.B. Aussagen über die vorhandene Computerkonfiguration machen eingestellt und vom Hauptprozessor eingelesen werden

Und natürlich gibt es auch für den Mausteil eigene Ports, und zwar gleich sieben (siehe 8.1 2 und 8 2)

8.1.4. Interrupts

Wie in 8.3 beschrieben werden dem internen Prozessor Mitteilungen über die INT-Leitung bzw. über NMI-Leitung gemacht. Wenn auf einer dieser Leitungen ein Signal ankommt, hört das Programm sofort mit der Abarbeitung seines momentan laufenden Programms auf und macht an einer bestimmten Adresse (unterschiedliche Adresse bei NMI und INT) weiter.

Während INT einfach die Abkürzung von Interrupt ist, bedeutet NMI Non Maskable Interrupt, auf deutsch: Nicht sperrbarer Interrupt.

Dies bedeutet, daß der NMI nicht per Software gesperrt, also verhindert werden kann. Dagegen ist es beim INT möglich, ihn durch einen Befehl im Programm zu sperren und ihn durch einen anderen Befehl wieder zuzulassen.

8.2 Mausteil

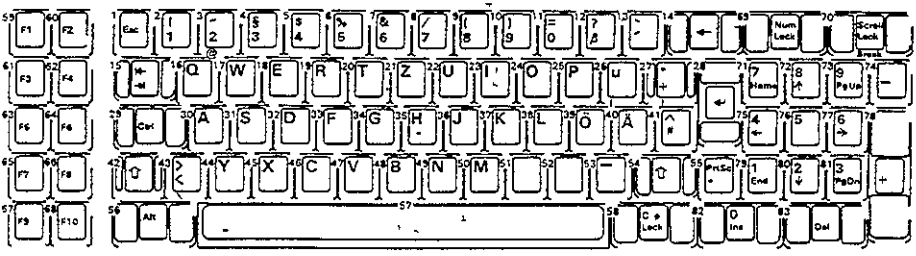
Wie bereits erwähnt, besitzt die Maus für die horizontale und vertikale Bewegung jeweils zwei Ausgänge, die zueinander phasenverschobene Rechtecksignale liefern. Die vier Ausgangssignale der Maus führen zum Triggereingang des Timerbausteins NE556. Entsprechend seiner externen Beschaltung arbeitet der Baustein als monostabiler Impulsgeber. Jede negative Flanke am Triggereingang bewirkt einen kurzen positiven Impuls am Ausgang. Diese Ausgangssignale dienen als Takt für die 8 Bit Zähler 74LS590. Da bei einer Aufwärtsbewegung beide Zähler für die Horizontalbewegung, wegen der Phasenverschiebung, einen Taktimpuls erhalten, jedoch nur einer zählen darf, muß der andere Zähler gesperrt werden. Durch Zuführung der paarweise vertauschten Rechtecksignale auf die Freigabeeingänge des Taktes wird entsprechend der Bewegung nur ein Zähler freigegeben. Zur Erfassung der Bewegung der Maus müssen die Inhalte der vier Zähler regelmäßig ausgelesen werden.

Da die Zählerbausteine mit Ausgaberegistern arbeiten, bedarf es vor dem Auslesen der Zählerinhalte eines Schreibzugriffs (beliebiges Datenwort) auf die Adresse 8DH, um eine Übernahme des aktuellen Zählerstandes in das Ausgaberegister zu veranlassen. Der Zugriff auf die Adresse 8DH bewirkt gleichzeitig eine Speicherung der Zählerstände bei allen vier Zählern. Ebenso bewirkt ein Schreibzugriff (beliebiges Datenwort) auf die Adresse 8EH ein gleichzeitiges Löschen aller vier Zähler. Über den Baustein 74LS125 besteht die Möglichkeit, den Zustand von maximal vier Tasten der Maus abzufragen. Bei der standardmäßigen Belegung führen allerdings nur zwei Eingänge zum 9-poligen Maus-Stecker.

9 Bedienungsanleitung

9.1 Grundfunktionen

Im Grundzustand (z B nach dem Einschalten) verhält sich eine, an die (funktionstuchtige) Baugruppe, angeschlossene XT-Tastatur genauso wie eine ganz normale Schreibmaschine Es erscheint immer das Zeichen auf dem Bildschirm, das auch auf der gedruckten Taste aufgedruckt ist Um diesen und die folgenden Test durchzufuhren sollten Sie sich am Besten in einem Editor (z B 680xx-Grundprogramm Editor) oder in einem Textverarbeitungsprogramm (z.B. Wordstar) befinden, da auf Menu- oder Betriebssystemebene manche Zeichen (z B ÜOAu .) vom Computer nicht beachtet werden



Solange Sie jetzt keine Sondertaste (im Bild hervorgehoben) drucken bleiben Sie in der Grundebene, d.h alle Buchstaben erscheinen (wie bei der Schreibmaschine) als Kleinbuchstaben Bei den Zahlen und beim Zehnerblock erscheinen die Ziffern und nicht die zusàtzlich aufgedruckten Zeichen (wie z B !"$%).

9.2 Ebenen

Wenn Sie jetzt z B einen Großbuchstaben erzeugen wollen, mussen Sie dazu die 'Ebene' wechseln. Dies hört sich vielleicht etwas verwirrend an, geschieht aber ganz einfach durch Drücken einer der zwei SHIFT-Tasten Auch dies ist genau wie bei einer Schreibmaschine Durch das Drücken der SHIFT-Taste haben jetzt alle (fast) Tasten eine andere Bedeutung, und zwar nur solange wie die SHIFT-Taste gedruckt bleibt

Genauso wird die Ebene durch die Tasten < CTRL > und < ALT > gewechselt. Auch hier ergeben sich für die Tasten andere Zeichen oder Funktionen. Die CTRL-Ebene wird z.B. oft als Befehlsebene (z B im Grundpgm -Editor oder in Wordstar) verwendet, d h eine Taste erzeugt auf dem Bildschirm keine Zeichen, sondern löst eine Funktion (wie z B 'gehe eine Zeile nach oben') aus.

Die ALT-Ebene wird bei der KEY3 auch zur Befehlswiederholung verwendet (siehe 9 4).
Auch verschiedene Kombinationen der Sondertasten (z.B SHIFT und CTRL gedruckt) ergeben verschieden Ebenen

Die Zeichen, die zu einer Ebene gehören, können hier nicht alle aufgeführt werden, aber sie sind leicht durch Ausprobieren zu erhalten und lassen sich auch dem Programm (im Teil Datentabelle) leicht entnehmen.

Wichtig sind auch noch die Tasten < CAPSLOCK > und < NUMLOCK >. 'LOCK' bedeutet hier soviel wie 'eingeklinkt', d.h. ihre Wirkung bleibt auch nach dem Loslassen der Taste erhalten.

Die CAPSLOCK-Taste hat die gleiche Wirkung wie die SHIFT-Taste, nur eben auch nach dem Loslassen. Sie kann nur durch Drücken einer SHIFT-TASTE wieder gelöst werden.

Die NUMLOCK-Taste bewirkt ein Umschalten des Zehnerblocks von Zahlentasten zu Steuertasten. Dabei sind nun die aufgedruckten Pfeile als Cursorsteuerung wirksam (funktioniert mit den meisten Textverarbeitungsprogrammen). Alle der so erhaltenen Funktionen stehen in der CTRL-Ebene auch auf dem normalen Tastenfeld zur Verfügung. Im einzelnen gilt:

7 = CTRL -q 8 = CTRL -e 9 = CTRL -r + = CTRL -a
4 = CTRL -s 5 = CTRL -a 6 = CTRL -d
1 = CTRL -k 2 = CTRL -x 3 = CTRL -c - = CTRL -f
0 = CTRL -v . = CTRL -g

Zurück zur Grundebene kommt man durch erneutes Drücken von NUMLOCK.

9.3 Funktionstasten

Es werden von der KEY3 auch die auf jeder XT- oder MF-Tastatur vorhandenen Funktionstasten unterstützt.

Die Funktionstasten sind auf der Tastatur mit F1 - F10 bezeichnet. Bei verschiedenen Tastaturen gibt es auch noch F11 und F12, welche aber von der KEY3 nicht unterstützt wird.

Die zehn vorhanden Funktionstasten stehen in vier Ebenen zur Verfügung (Grundebene, SHIFT gedrückt, CTRL gedrückt, SHIFT und CTRL gedrückt), so daß insgesamt 40 Funktionstasten zur Verfügung stehen. Dabei sind die Funktionstasten vorbelegt, können aber alle ganz einfach selber progammiert werden.

Vorbelegungstabelle:

Grundebene:			CTRL-Ebene:		
F1	=	'dir'	F1	=	'ws'
F2	=	'pip'	F2	=	'm80'
F3	=	'era'	F3	=	
F4	=	'ren'	F4	=	
F5	=	'stat'	F5	=	
F6	=	'type'	F6	=	
F7	=	'a:'	F7	=	
F8	=	'b:'	F8	=	
F9	=	'.*'	F9	=	
F10	=	'*.'	F10	=	

SHIFT-Ebene.			SHIFT-CTRL-Ebene		
F1	=	'dir'	F1	=	'ws'
F2	=	'pip'	F2	=	'm80'
F3	=	'era'	F3	=	
F4	=	'ren'	F4	=	
F5	=	'stat'	F5	=	
F6	=	'type'	F6	=	
F7	=	'a '	F7	=	
F8	=	'b '	F8	=	
F9	=	'*'	F9	=	
F10	=	'*'	F10	=	

Um eine Funktionstaste zu programmieren, also einer beliebigen Zeichenfolge gleichzusetzen, gibt es zwei Möglichkeiten: Erstens direkt von der Tastatur aus durch bestimmte Tastenfolgen (wird im folgenden beschrieben) oder durch ein Programm vom Hauptprozessor aus (wird im Kapitel 10 'Kommunikation' beschrieben)

Um eine Funktionstaste (z B F5) von der Tastatur aus zu programmieren geht man wie folgt vor:
- < ALT > drucken und gedruckt halten
- Ebene wählen und gedruckt halten (entfällt bei Grundebene)
- < F5 > drucken und gedruckt halten
- < ALT > loslassen
- gewählte Ebene loslassen
- Zeichenfolge die programmiert werden soll eingeben
- < F5 > loslassen

Jetzt ist F5 neu programmiert. Immer wenn jetzt F5 gedrückt wird, gibt die KEY3 die eingegebene Zeichenfolge aus

Als Länge der Zeichenfolge sind für jede Funktionstaste 16 Zeichen vorgeben. Es ist zwar möglich, die Tasten mit mehr als 16 Zeichen zu programmieren, nur reicht dann der zum Speichern vorgesehene Platz nicht mehr aus und die nächste Funktionstaste wird überschrieben Wenn z B in F6 'Donaudampfschiffahrt' programmiert werden soll, so erscheint beim Drucken von F6 richtig 'Donaudampfschiffahrt' und beim Drücken von F7 erscheint dann der Teil, der über die vorgesehenen 16 Zeichen hinausragt, hier 'fahrt' Das, was vorher in F7 gespeichert war ist gelöscht.

Eine Ausnahme bildet F10 (in allen Ebenen) Da nach F10 keine weitere Funktionstaste mehr kommt, können hier die Zeichenfolgen bis zu 80 Zeichen lang sein.

Zu Definition des Begriffs "Zeichenfolge".

Hierzu zählen nicht nur Buchstaben und Ziffern, sondern auch Steuerzeichen (z.B < ENTER > oder < CTRL - e >) und sogar andere Funktionstasten Nur < ALT > darf während des Programmierens nicht gedrückt werden

Dadurch, daß auch andere Funktionstasten zu einer Zeichenfolge zählen, können die Funktionstasten auch ineinander programmiert werden
Eigene Versuche schaffen hier schnell Klarheit

9.4 Befehls- und Wortwiederholung

Die KEY3 hat eine eingebaute Befehls- und Wortwiederholung. Dabei können auf Tastendruck einer(s) der letzten sieben Befehle (Worte) wiederholt werden.
Diese Möglichkeit erweist sich vor allem auf Betriebsystemebene als sehr bequem und zeitsparend.

Die Erklärung ist an einem Beispiel am verständlichsten:

> Folgende Eingaben wurden gemacht (Beispiel auf CP/M)
> A> dir
> A> dir *.com
> A> dir *.exe
> A> B:
> B>
>
> Jetzt möchte man auf Laufwerk B "dir *.com" aufrufen. Dazu drückt man jetzt < Alt > und < 3 > (3, da "dir *.com" der drittletzte Befehl war). Sofort erscheint "dir *.com". Genauso kann man bis zu sieben Befehle zurück alle Befehle durch < Alt > und < 1 - 7 > wieder aufrufen. Wenn jetzt in unserem Beispiel "dir *.exe" aufgerufen werden soll, so geschieht dies wieder durch < Alt > und < 3 >, da durch das Wiederholen von "dir *.com", "dir *.exe" jetzt der drittletzte Befehl ist.

Auch hier erhält man durch eigene Versuche schnell Klarheit.

Genauso, wie beschrieben, verhält es sich bei der Wortwiederholung, nur daß hier nicht die letzten sieben Befehle sondern die letzten sieben Worte wiederholt werden können. Der Aufruf der Wortwiederholung erfolgt durch die Tastenkombination < Alt > und < "Taste unter der Zahl" >, also z.B. beim vorletzten Wort < Alt > und < w >.

9.5 Reset

Genauso, wie ein Computer, kann sich auch die Tastaturkarte einmal "aufhängen", z.B. durch elektromagnetische Störungen (Lichteinschalten, o.ä.) oder durch nicht definierte Tastenkombinationen. Dies wirkt sich so aus, daß von der KEY3 kein Zeichen mehr an den Computer weitergegeben wird. Um dann nicht einen kompletten Reset des ganzen Computers machen zu müssen, was den Verlust der gerade bearbeiteten Daten nach sich zieht, gibt es auf der KEY3 eine Resettaste: Dies ist die Taste < SCROLL LOCK >.

Dabei wird der Reset nur von der Tastaturbaugruppe durchgeführt, der restliche Computer bleibt davon unbeeindruckt.

Wenn also einmal Nichts mehr geht, zuerst die < SCROLL LOCK > Taste betätigen. Sofern der Fehler von der KEY3 kam, kann dann ganz normal weitergearbeitet werden.

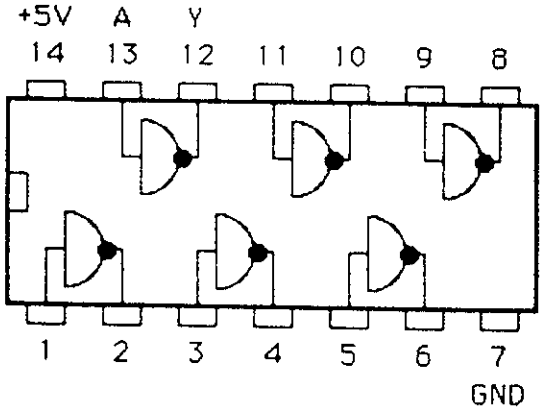
11. Bauelemente

11.1 TTL-Bausteine

11.1 1. 7404

7404

6 Inverter



Logiktabelle

A	Y
0	1
1	0

Typ Impuls-Verzogerungszeit 9 ns

Typ Versorgungsstrom 25 mA

positive Logik

$Y = \overline{A}$

10. Kommunikation

Wie schon mehrmals angedeutet, ist es auch möglich, vom Hauptprozessor aus mit dem KEY3-Prozessor in Verbindung zu treten. Dabei ist ein Datenaustausch in beide Richtungen möglich. Der Austausch erfolgt, wie schon unter 8.1.3 beschrieben, über die Ein-, Ausgabeports. Jeder Prozessor hat dabei einen Port, in den er schreibt und einen Port, vom dem er liest. Der Schreibport des einen ist der Leseport des anderen Prozessors.

Die Ports haben die Adressen:

KEY3-Prozessor:	Schreibport:	2
	Leseport:	3
Hauptprozessor:	Schreibport:	8CH
	Leseport:	68H

Ein Datenaustausch kann nur vom Hauptprozessor aus begonnen werden. In der jetzigen Programmversion gibt es vier verschiede Arten des Datenaustausches. Jede dieser Möglichkeiten wird durch Anlegen eines bestimmen Codes vom Hauptprozessor an Port 8Ch aktiviert.

Die Codes sind:

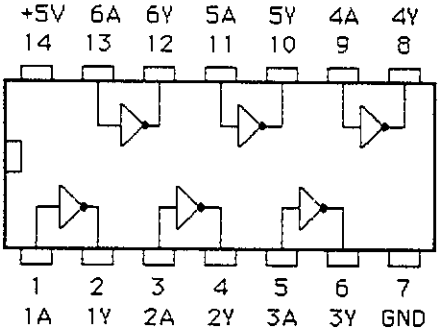
"01"	Daten von der KEY3 auslesen
"02"	Daten in die KEY3 einlesen
"03"	Uhr auf der KEY3 stellen
"04"	Uhr von der KEY3 auslesen

Der Datenaustausch der HOST-CPU mit der KEY3 ist bei der jetzt ausgelieferten Version 1.0 noch nicht implementiert.

11.1.2 74LS05

74LS 05

Sechs Inverter (open Collector)



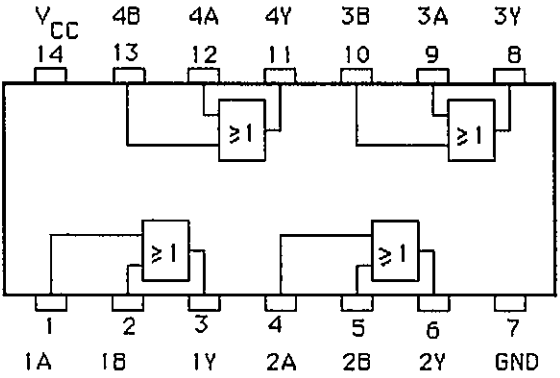
Positive Logik $Y = \overline{A}$

Typ Impuls-Verzogerungszeit	22 ns
Typ Leistungs-aufnahme	60 mW

11.1.3 74LS32

74LS32

Vier Or-Gatter mit je 2 Eingängen



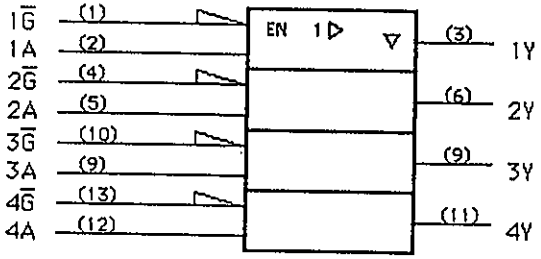
Typ Impulsverzögerungszeit: 12 ns

Typ Versorgungsstrom: 4 mA

11.1.4 74LS125

74LS125

4 Bus-Leitungstreiber



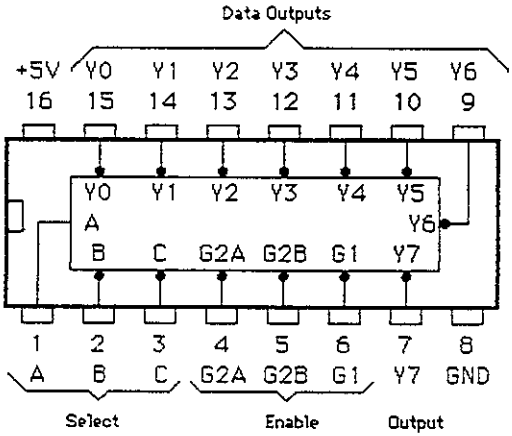
Typ. Impulsverzögerungszeit: 10 ns

Typ. Leistungsaufnahme: 55 mW

11.1.5 74LS138

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

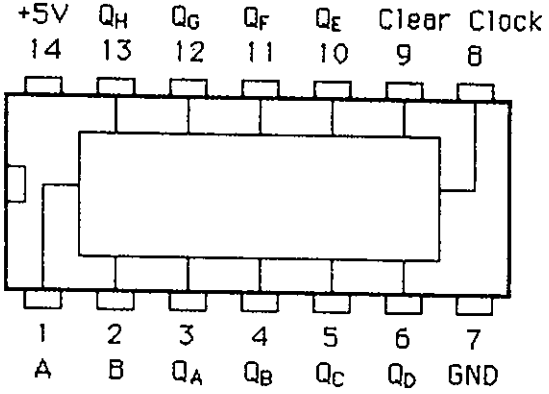
Typ Impulsverzögerungszeit 22 ns

Typ Versorgungsstrom 7 mA

11.1.6 74LS164

74LS164

Schieberegister mit 8-Bit paralleler Ausgabe



Function Table

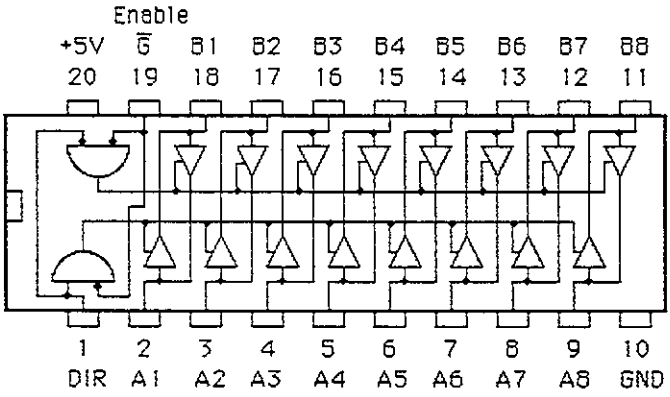
INPUTS				OUTPUTS		
Clear	Clock	A	B	Q_A	Q_B	Q_H
L	x	x	x	L	L	L
H	L	x	x	Q_{A0}	Q_{B0}	Q_{H0}
H	↑	H	H	H	Q_{An}	Q_{Gn}
H	↑	L	x	L	Q_{An}	Q_{Gn}
H	↑	x	L	L	Q_{An}	Q_{Gn}

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

11.1.7 74LS245

74LS245

8-fach Bus-Transceiver mit 3-state Ausgangen



Function Table

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

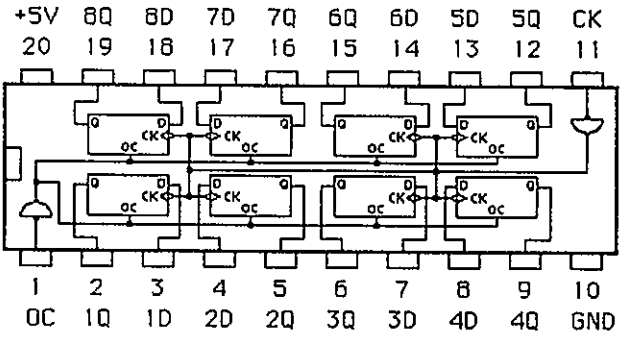
Typ Impuls-Verzogerungszeit 20 ns

Typ Versorgungsstrom 75 mA

11.1.8 74LS374

74LS374

8-Bit D Register mit 3-state-Ausgängen



Logiktabelle

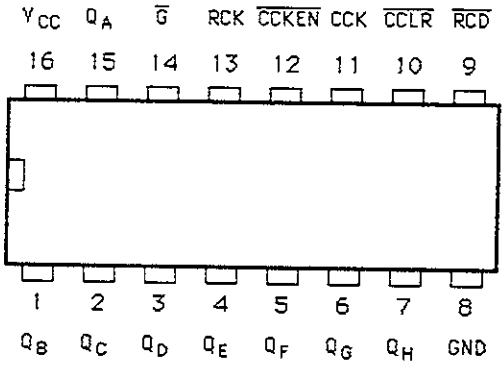
OC	CK	D	Q
0	↑	1	1
0	↑	0	0
0	0	X	Q_0
1	X	X	Z

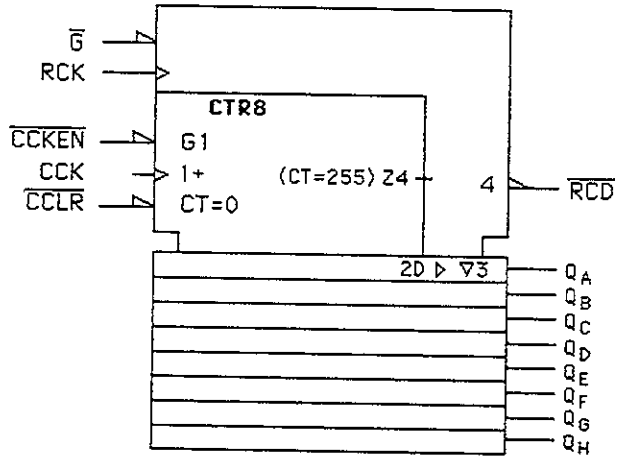
Typ. Impuls-Verzögerungszeit.	16 ns
Typ. Versorgungsstrom	2.6 mA
positive Logik	ja

11.1.9 74LS590

74LS590

8 Bit Binarzähler mit Ausgangsregister



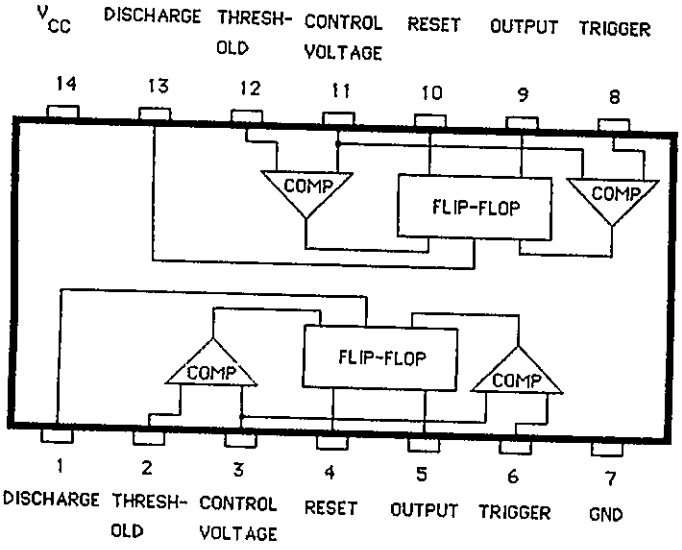


Min. garantierte Zählfrequenz. 20 MHz
Typ. Leistungsaufnahme: 205 mW

11.2 LM556

LM556

Doppel-Timer



Maximaler Versorgungsstrom: ca. 30 mA

Maximale Versorgungsspannung: 18 V

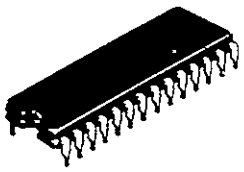
11 3 CMOS STATC RAM 6264

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

■ FEATURES

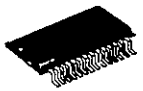
- Fast access Time — 100ns/120ns/150ns (max)
- Low Power Standby — Standby 0 1mW (typ)
  10μW (typ) L /LL version
- Low Power Operation — Operating 700mW/MHz (typ)
- Single +5V Supply
- Completely Static Memory — No clock or Timing Strobe Required
- Equal Access and Cycle Time
- Common Data Input and Output Three State Output
- Directly TTL Compatible All Input and Output
- Standard 28pin Package Configuration
- Pin Out Compatible with 64K EPROM HN482764
- Capability of Battery Back Up Operation (L /LL version)

HM6264P Series



(DP 28)

HM6264FP Series



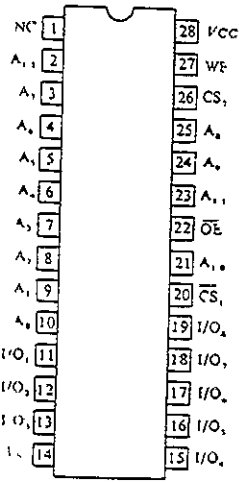
(F P 28D/DA)

■ ORDERING INFORMATION

Type No	Access Time	Package
HM6264P 10	100ns	600 mil 28 pin Plastic DIP
HM6264P 12	120ns	
HM6264P 15	150ns	
HM6264LP 10	100n	
HM6264LP 12	120n	
HM6264LP 15	150n	
HM6264LP 10L	100ns	
HM6264LP 12L	120ns	
HM6264LP 15L	150ns	
HM6264FP 10	100ns	28 pin Plastic SOP (Note)
HM6264FP 12	120ns	
HM6264FP 15	150ns	
HM6264LFP 10	100ns	
HM6264LFP 12	120ns	
HM6264LFP 15	150ns	
HM6264LFP 10L	100ns	
HM6264LFP 12L	120ns	
HM6264LFP 15L	150ns	

Note) A character T is added to the end of Type No for SOP of 3 00 mm (max) thickness

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

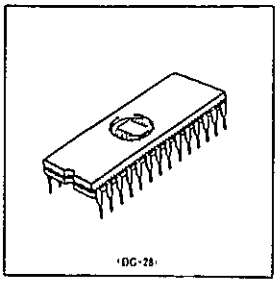
Item	Symbol	Rating	Unit
Terminal Voltage *1	V_T	−0.5 *2 to +7.0	V
Power Dissipation	P_T	1.0	W
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature	T_{stg}	−55 to +125	°C
Storage Temperature Under Bias	T_{bias}	−10 to +85	°C

Notes) *1 With respect to V_{SS}
*2 −3.0V for pulse width ≥ 50ns

11.4 EPROM 2764

8192-word x 8-bit UV Erasable and Programmable Read Only Memory

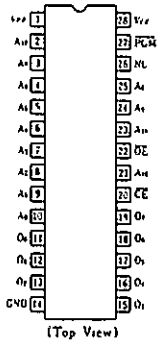
The HN482764G is a 8192 word by 8 bit erasable and electrically programmable ROM. This device is packaged in a 28 pin dual-in-line package with transparent lid. The transparent lid on the package allows the memory content to be erased with ultraviolet light.



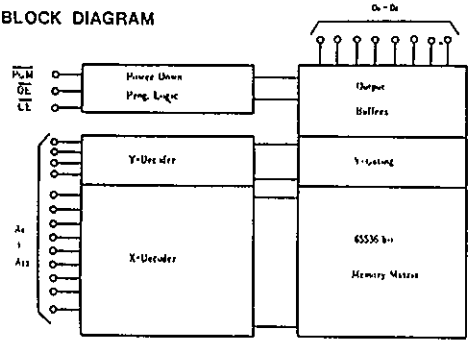
■ FEATURES

- Single Power Supply .... +5V ± 5%
- Simple Programming ..... Program Voltage: +21V D.C.
  Program with one 50ms Pulse
- Static ..... No Clocks Required
- Inputs and Outputs TTL Compatible During Both Read and Program Mode.
- Access Time ..... HN482764G-2 200ns max
  HN482764G 250ns max
  HN482764G-3 300ns max
- High Performance Programming Available
- Low Standby Current ..... 35mA max.
- Compatible with Intel 2764

■ PIN ARRANGEMENT



■ BLOCK DIAGRAM



■ MODE SELECTION

Mode \ Pins	$\overline{CE}$ (20)	$\overline{OE}$ (22)	$\overline{PGM}$ (27)	V_{PP} (1)	V_{CC} (28)	Outputs (11~13, 15~19)
Read	V_{IL}	V_{IL}	V_{IH}	V_{CC}	V_{CC}	Dout
Stand-by	V_{IH}	×	×	V_{CC}	V_{CC}	High Z
Program	V_{IL}	×	V_{IL}	V_{PP}	V_{CC}	Din
Program Verify	V_{IL}	V_{IL}	V_{IH}	V_{PP}	V_{CC}	Dout
Program Inhibit	V_{IH}	×	×	V_{PP}	V_{CC}	High Z

× don't care

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Operating Temperature Range	T_{opr}	0 to +70	°C
Storage Temperature Range	T_{stg}	−65 to +125	°C
All Input and Output Voltage*	V_T	−0.6 to +7	V
V_{PP} Voltage*	V_{PP}	−0.6 to +26.5	V

* with respect to GND

Note: This device is not available for new application

Z8410 Z80® DMA Direct Memory Access Controller

Zilog

Product Specification

April 1985

FEATURES

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

GENERAL DESCRIPTION

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

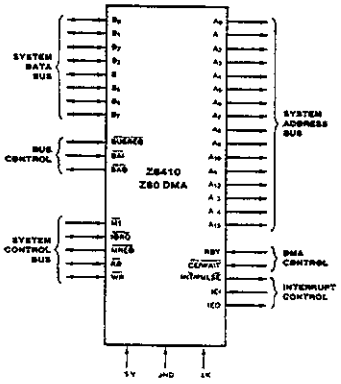


Figure 1 Pin Functions

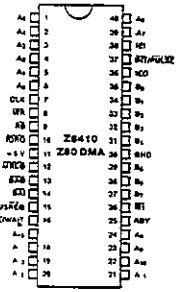


Figure 2 40 pin Dual In Line Package (DIP)
Pin Assignments

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

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

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

FUNCTIONAL DESCRIPTION

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

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

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

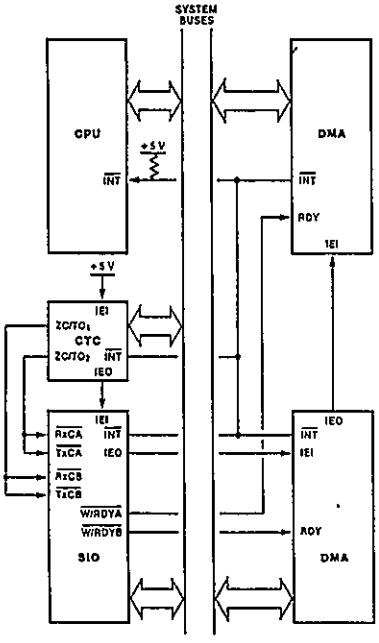


Figure 3. Typical Z80 Environment

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

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

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

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

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

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

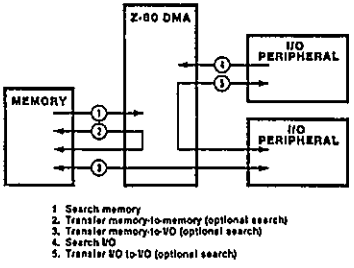


Figure 4. Basic Functions of the Z80 DMA

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

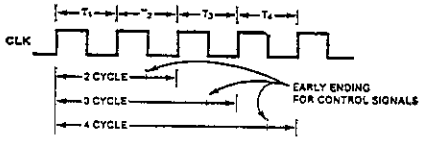


Figure 5. Variable Cycle Length

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

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

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

PIN DESCRIPTION

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

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

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

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

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

CLK. *System Clock* (input) Standard Z80 single-phase clock at 2 5 MHz (Z80 DMA) or 4 0 MHz (Z80A DMA). For 2.5 MHz clocks, a TTL gate with pullup resistor may be adequate to meet the timing and voltage level specification For 4.0 MHz clocks, use a clock driver with an active pullup to meet the V_{IH} specification and rise-time requirements In all cases there should be a resistive pullup to the power supply of 10K ohms (max) to ensure proper power when the DMA is reset.

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

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

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

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

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

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

fetched An interrupt acknowledge is indicated when both M1 and IORQ are active

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

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

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

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

INTERNAL STRUCTURE

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

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

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

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

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

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

RR0-RR6—Read Registers 0 through 6

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

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

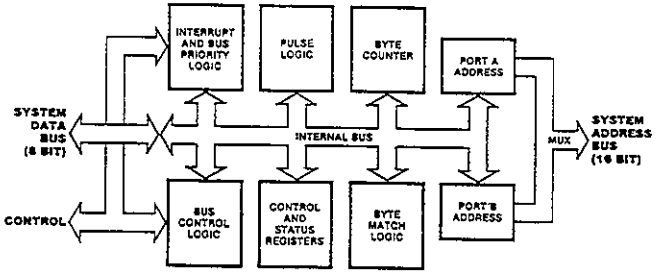


Figure 6 Block Diagram

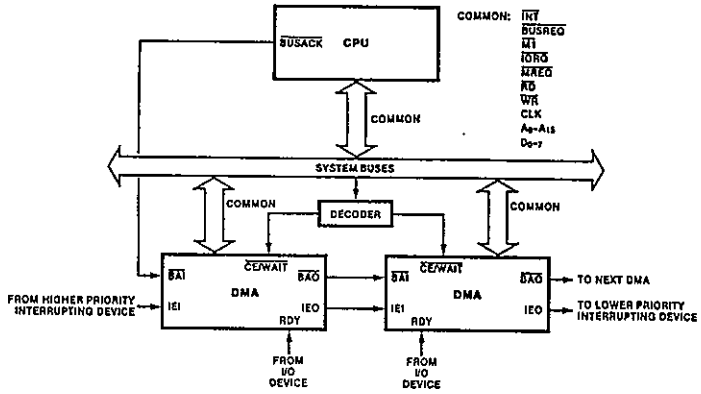


Figure 7. Multiple-DMA Interconnection to the Z80 CPU

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

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

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

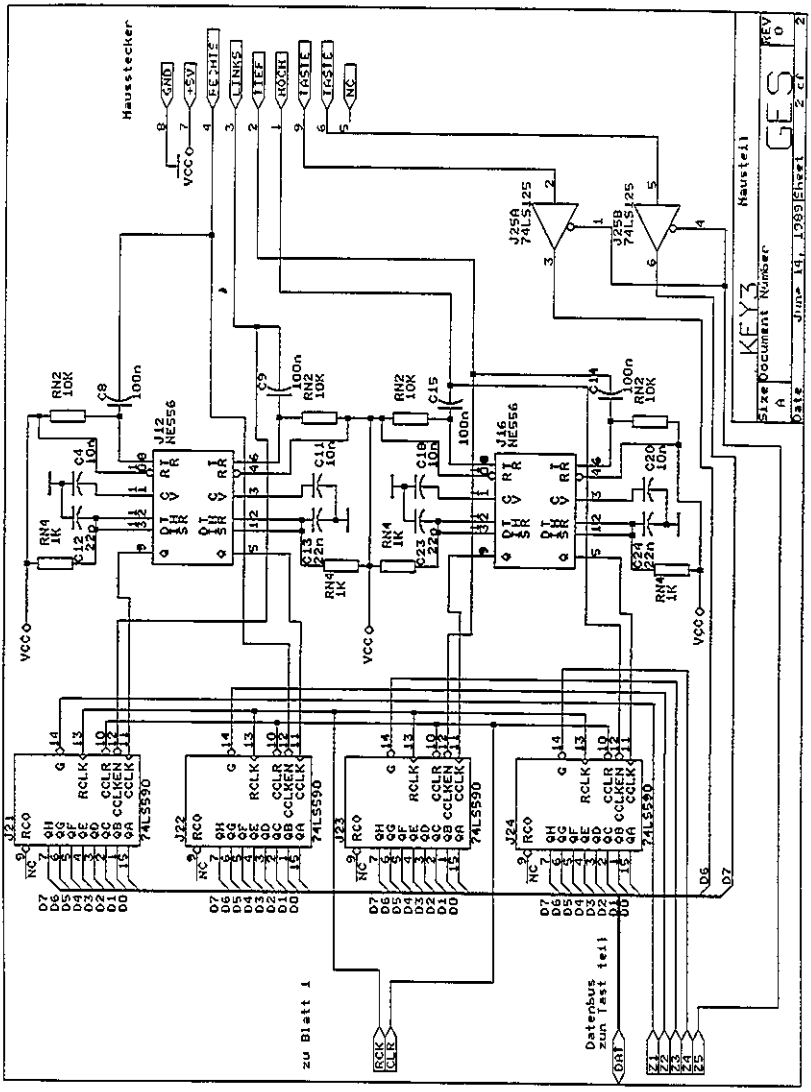
Comments	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HEX
WR0 sets DMA to receive block length Port A starting address and temporarily sets Port B as source	0	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B ⟶ A Temporary for Loading B Address*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

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

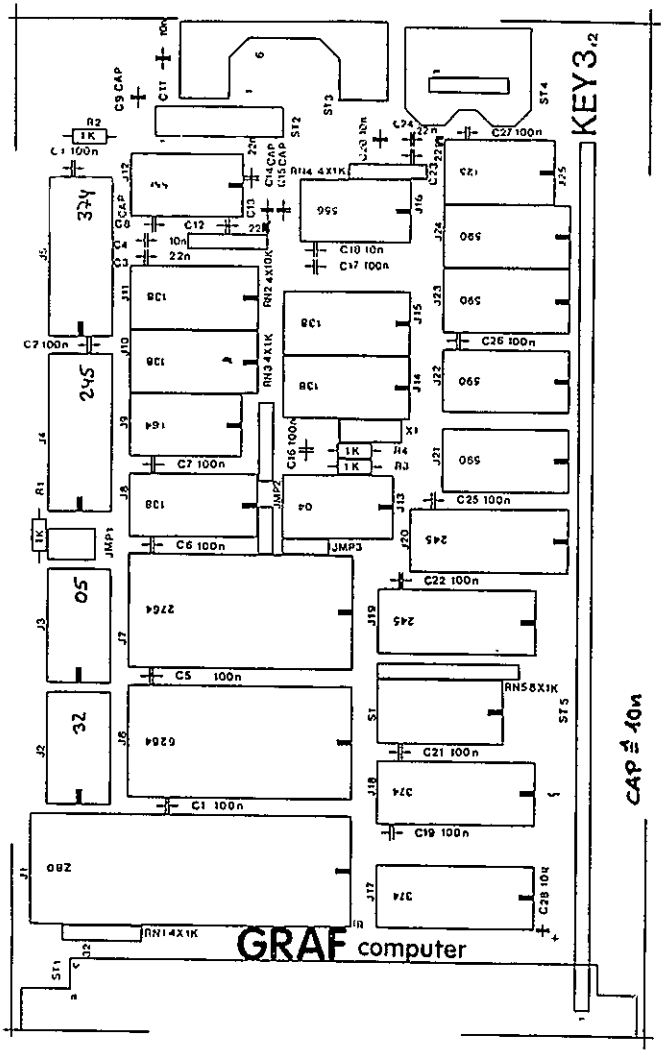
Figure 9 Sample DMA Program

ANHANG A: SCHALTPLAN (TASTATURTEIL)

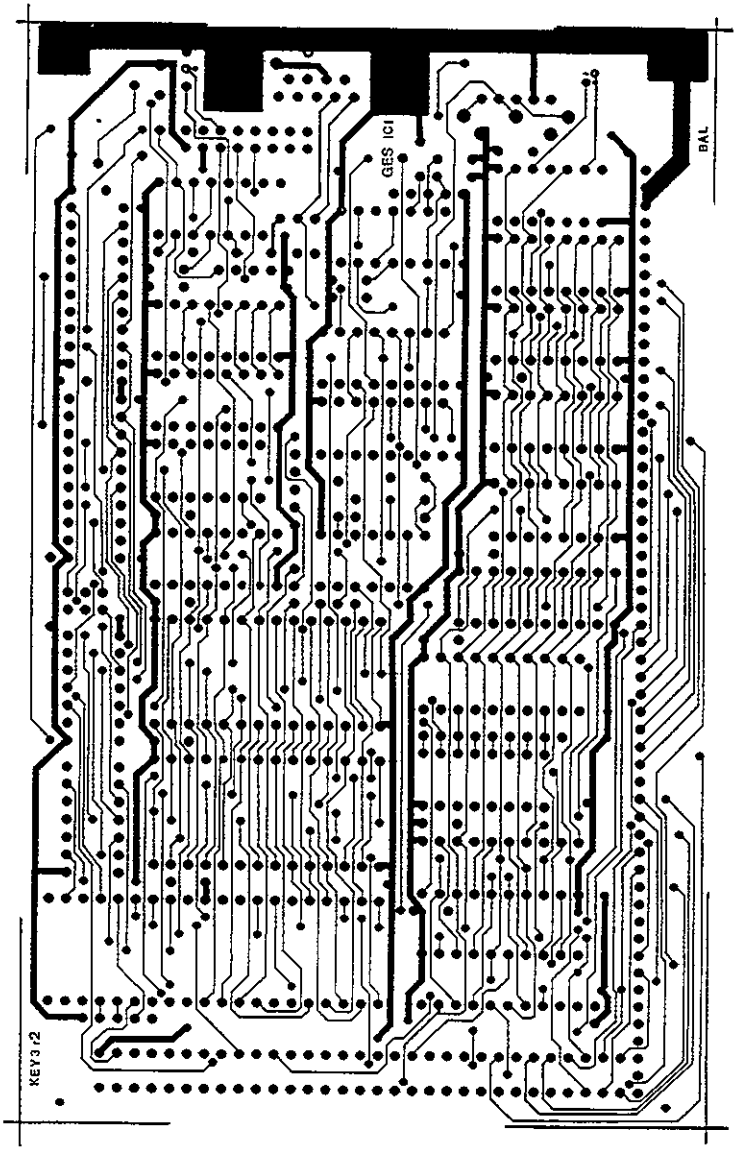
ANHANG B: SCHALTPLAN (MAUSTEIL)



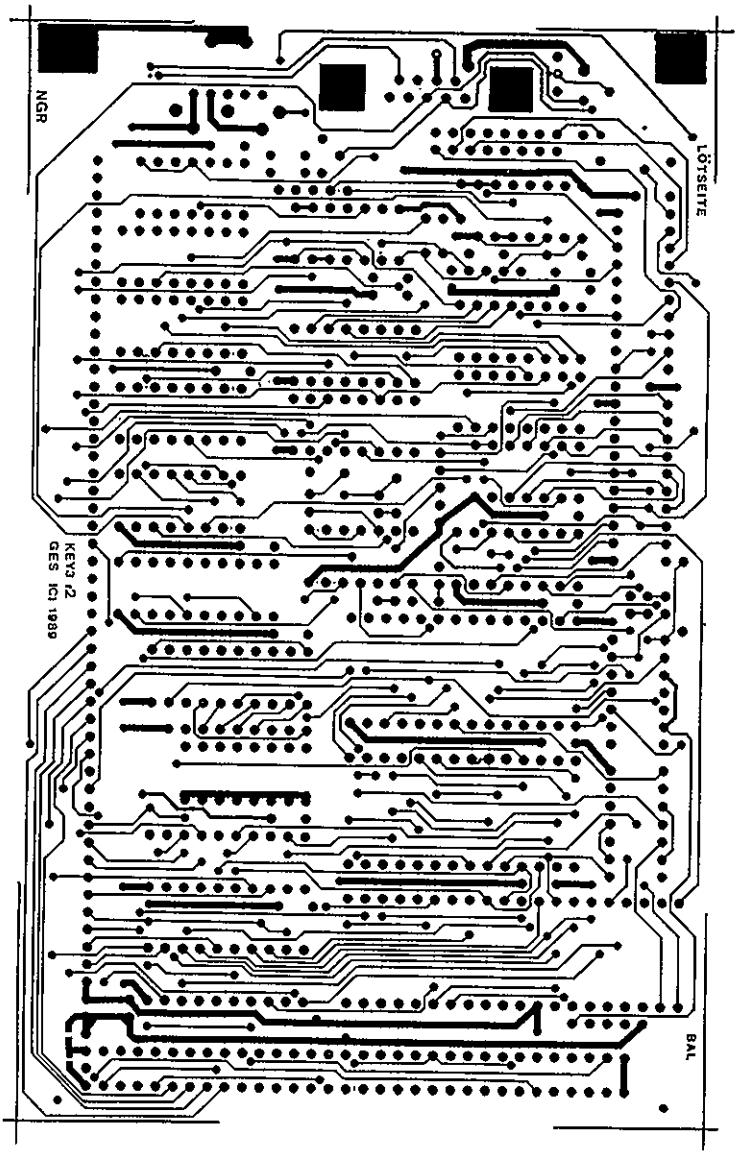
ANHANG C: BESTÜCKUNGSPLAN



ANHANG D: LAYOUT (BESTUCKUNGSSEITE)



ANHANG E: LAYOUT LÖTSEITE



ANHANG F: LISTINGS