Datenbuch

Mikrocomputer-Datenbuch 79/80

Z80/Z8/Z8000
Bausteine
Baugruppen
Systeme
Software

Stand: 1. Mai 1979
Diese Broschüre enthält unverbindliche Beschreibungen und gibt keine Auskunft über Verfügbarkeiten.
Lieferzeiten, Leistungsumfang des Lieferanten und Preise.
Technische Änderungen — auch unangekündigt — bleiben vorbehalten.
Für die angegebenen Schaltungen, Beschreibungen und Tabellen wird keine Gewähr bezüglich der Freiheit von Rechten Dritter übernommen.
Fragen über Technik, Preise und Liefermöglichkeiten richten Sie bitte an die jeweilige Geschäftsstelle von KONTRON in Österreich und Deutschland und an STOLZ AG in der Schweiz.
Keine Gewähr für Irrtümer und Druckfehler.

INHALTSVERZEICHNIS

Z80A-CPU MIKROPROZESSOR

Zilog

Z80A-CPU-Blockschaltbild

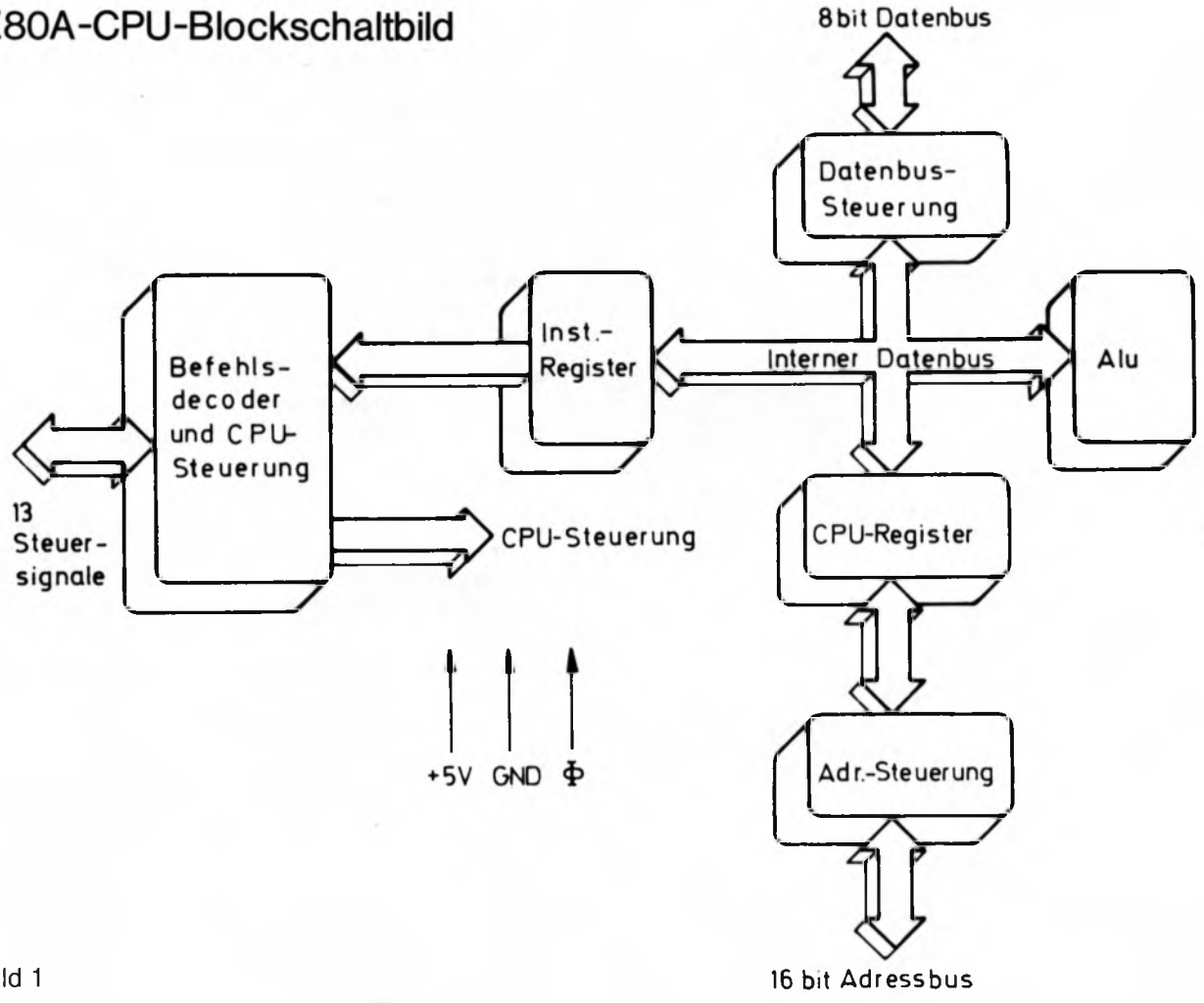


Bild 1

1. ZILOG-Z80A MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Das Computersystem Z80A umfaßt ein komplettes Angebot an MC-Halbleiterbausteinen, fertig aufgebaute MC-Platinen, fertige OEM-Computersysteme und ein Entwicklungshilfssystem mit der dazugehörigen Betriebssoftware.

Dabei ist das Halbleiterbausteinangebot so geartet, daß auch komplexere MC-Systeme mit minimaler MC-Bausteinanzahl zu realisieren sind. Als Speicherbausteine sind sämtliche Standardchips verwendbar, zusätzliche Logik ist im ganzen System praktisch nicht nötig.

Z80A-CPU-Register

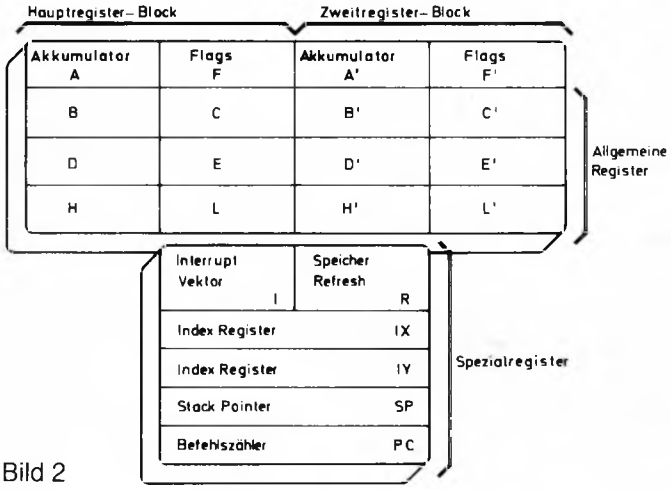


Bild 2

☐ Aufbau

Die Z80A-CPU (Ionenimplantierter n-Kanal-Silicon-Gate-Schaltkreis) ist als Mikroprozessor der dritten Generation ein Bauelement auf dem neuesten Stand der Technik mit bisher unerreichter Leistungsfähigkeit.

Der gegenüber bisher verfügbaren Mikroprozessoren höhere Informations-Durchsatz ("throughput") und die effizientere Programmspeicherausnutzung erlauben die Realisierung von Mikroprozessoren-Systemen mit höheren Anforderungen als bei Systemen mit Mikroprozessoren der 2. Generation.

Darüberhinaus vereinfacht Z80A durch die Notwendigkeit nur einer einzigen Speisespannung und der direkten Anschlußmöglichkeit preiswerter Standardspeicherbausteine den nötigen Schaltungsaufwand.

Bild 1 zeigt ein Blockschaltbild der CPU. In Bild 2 ist die Registerorganisation (= insgesamt 208 Bit dem Anwender zugänglicher Schreib/Lesespeicher) dargestellt.

Hiervon sind zwei gleichaufgebaute Blöcke mit je 6 allgemeinen Registern, die jeweils wahlweise 8- oder 16-Bit-Operationen erlauben.

Hinzu kommen zwei Akkumulatoren und Status(„Flag"-) Register.

Die Operationen der CPU laufen im Haupt-Register/Akku-Block ab; der Zugriff zum zweiten Register/Akku-Block erfolgt durch Übergangs-(„Exchange")-Anweisungen.

Diese alternative Arbeitsweise ermöglicht wechselweises Arbeiten in Haupt- und Hintergrundprogramm ohne Auslagern von Registerinhalten in den Arbeitsspeicher und dadurch besonders schnelle und effiziente Interruptbehandlung.

Der 16-Bit-Stack-Pointer der CPU dient zur Bearbeitung von Mehrebenen („Multilevel-") Interrupts, praktisch unbegrenzte Unterprogramm-Verschachtelung ("Subroutine Nesting") und Zwischenspeicherung von Datenblöcken.

Zwei 16-Bit-Indexregister erlauben die Bearbeitung von Tabellen und relokativen (d.h. im Adreßraum des Arbeitsspeichers verschiebbaren) Informationen.

Ein eigenes Refresh-Register wurde für ein direktes, übersichtliches Arbeiten mit externen dynamischen Speichern ohne Software-Aufwand implementiert.

Das Unterbrechungsregister („I-Register") schließlich liefert zur Realisierung einer besonders leistungsfähigen Art der Interrupt-Behandlung die höherwertigen 8 bits eines Zeigers, der über eine Tabelle den Sprung in Interrupt-Behandlungsprogramme ermöglicht; die niederwertigen 8 Bit der Anfangsadressen werden in üblicher Weise von den anfordernden Schaltung geliefert.

Hervorstechendste Eigenschaft ist die fortschrittliche Architektur, die eine Reduktion der zur Lösung eines bestimmten Problems nötigen Befehle um typisch 50% ermöglicht. Hierdurch werden Programmentwicklungs- und Testkosten eingespart, was besonders bei in kleinen Stückzahlen gefertigten Geräten ausschlaggebend ist; gleichzeitig wird die Anzahl der im System benötigten Programmspeicherbausteine reduziert (wichtig für Systeme in großen Stückzahlen) und die Verarbeitungsgeschwindigkeit in hohem Maß gesteigert. Erzielt wird diese Wirkung durch die bisher genannten und folgenden Eigenschaften der CPU:

- Echte Speicher — indirekte Interruptbearbeitungstechnik möglich (auf Interruptmode 2); dadurch ist eine in der Mikrocomputertechnik bisher unerreichte Flexibilität in der Interruptbehandlung möglich, wie sie bisher nur bei Prozeßrechnern bekannt war.
- In der CPU eingebauter Refresh Controller zum direkten Anschluß von dynamischen Speichern an die CPU ohne zusätzliche Hard- oder Software. Zu den genannten architektonischen Eigenschaften kommen folgende Softwarebesonderheiten:
  - ☐ Befehle zur Behandlung von 4 bit, 8 bit und 16 bit Datenwörter
  - ☐ Befehle zum Register rotieren **und** schieben
- Als einziger Prozessor ist die Z80A-CPU in der Lage, sowohl Einzelbits als auch ganze Dateien mit einem einzigen Befehl zu bearbeiten (Blocktransfers mit einer Blockgröße bis zu 64 kByte bei einer Übertragungsrate von 5,3 μsec/Byte, Blocksuchbefehl, Block-Ein/Ausgaben mit einer Ein/Ausgaberate von 200 kByte/sec, Setzen, Testen oder Rücksetzen irgend eines einzelnen Bits in einem der CPU Register oder einer Speicherstelle).

☐ Kenndaten der Z80A-CPU

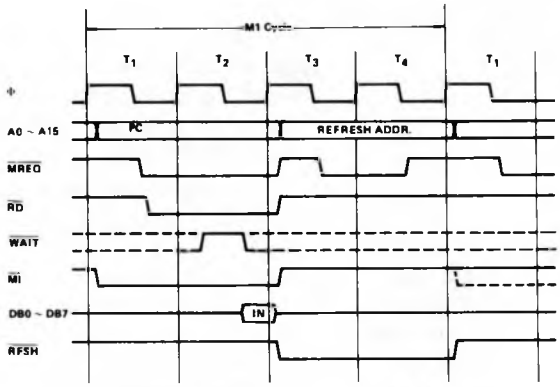
- 1 Chip-Mikroprozessor in n-Kanal-Silicon-Gate-Technologie.
- Befehlssatz umfaßt 158 Befehle, darunter sämtliche 78 8080-Instruktionen (voll Software-kompatibel!).
  Über den 8080-Befehlssatz hinaus verfügt der Z 80 über umfassende 16-, 8-, 4- und Einzel-Bit-Instruktionen und zusätzliche Adressierweisen (indizierte, relative und Bit-Adressierung).
- 17 interne Register.
- 3 schnelle Interrupt-Behandlungsarten und ein zusätzlicher, nichtmaskierbarer Interrupt.
- Direkter Anschluß von dynamischen oder statischen Standardspeicherchips ohne zusätzlichen Bauteileaufwand.
- Eingebaute dynamische Refresh-Hardware.
- Minimale Befehlsausführungsdauer: 1,0 Mikrosekunde.
- Stromversorgung über eine einzige 5-V-Versorgungsspannung.
- 5-V-Einphasen-Takt.
- Alle Anschlüsse TTL-kompatibel.

□ Z80A-CPU-Pinbelegung

Pin	Funktion	Kommentar
A_0-A_{15}	Adreßbus	Tri-State Ausgänge, High-aktiv. Liefern die Adressen für den Speicher (bis zu 64 kByte) und die Ein/Ausgabebausteine.
D_0-D_7	Datenbus	Tri-State Ein/Ausgänge (Bidirektional), High-aktiv. Über den Datenbus erfolgt der Datenaustausch zwischen CPU und Speicher bzw. CPU und E/A-Bausteinen.
$\overline{M1}$	Maschinenzyklus Eins	Ausgang, Low-aktiv. M 1 = aktiv bedeutet, daß der momentane Maschinenzyklus der Operationscode-Lesezyklus (Fetch-Zyklus) der momentan auszuführenden Anweisung ist.
$\overline{MREQ}$	Speicheranforderung (=„Memory-Request“)	Tri-State Ausgang, Low-aktiv. MREQ = aktiv bedeutet, daß auf dem Adreßbus die Adresse für einen Speicherzugriff (Lesen oder Schreiben) ansteht.
$\overline{IOREQ}$	Ein/Ausgabe-Anforderung (=„Input/Output-Request“)	Tri-State-Ausgang, Low-aktiv. IOREQ = aktiv bedeutet, daß an den niederwertigen 8 bits des Adreßbus eine Adresse zur I/O-Portauswahl (Eingabe oder Ausgabe) ansteht. Ein IOREQ-Signal wird auch dann erzeugt, wenn eine Interruptanforderung akzeptiert wurde; in diesem Fall kann dann der zugehörige Interrupt-Vektor auf den Datenbus gelegt werden.
$\overline{RD}$	Lesen (“Read”)	Tri-State-Ausgang, Low-aktiv. RD = aktiv bedeutet, daß die CPU Daten vom Speicher oder von einem I/O-Port lesen soll. Der angesprochene Speicher oder I/O-Baustein interpretiert das Signal als Aufforderung, Daten auf den Datenbus zu legen.
$\overline{WR}$	Schreiben (“Write”)	Tri-State-Ausgang, Low-aktiv. WR = aktiv bedeutet, daß die CPU Daten für den Speicher oder einen I/O-Baustein auf dem Datenbus bereithält.
$\overline{RFSH}$	Refresh	Ausgang, Low-aktiv. RFSH = aktiv bedeutet, daß die niederwertigen 7 bit des Adreßbus eine Refreshadresse für dynamische Speicher führen und das laufende MREQ-Signal zur Einleitung eines Refresh-Zyklus für alle angeschlossenen dynamischen Speicher zu benutzen ist.
$\overline{HALT}$	Halt-Zustand	Ausgang, Low-aktiv. HALT = aktiv bedeutet, daß die CPU einen (Software-)HALT-Befehl ausgeführt hat und zur weiteren Abarbeitung des Programms auf ein Interrupt-Signal wartet (einen nicht-maskierbaren Interrupt oder aber einen freigegebenen maskierbaren Interrupt). Im Halt-Zustand führt die CPU zur Sicherstellung des Refresh-Vorgangs Leerbefehle (NOP's) aus.
$\overline{WAIT}$	Warte-Signal (“Wait”)	Eingang, Low-aktiv. Low-Signal am WAIT-Eingang zeigt der CPU, daß die angesprochenen Speicher- oder I/O-Bausteine noch nicht zur Datenübertragung bereit sind. Die CPU beginnt mit Wait-Zyklen, solange der Eingang aktiviert wird.
$\overline{INT}$	Interrupt-Eingang (“Interrupt Request”)	Eingang, Low-aktiv. Das Interrupt-Anforderungssignal wird von einer peripheren Schaltung erzeugt. die Anforderung wird nach Abarbeitung des in Ausführung befindlichen Befehls berücksichtigt, soweit das interne Software-gesteuerte Interrupt-Freigabe-Flip Flop gesetzt und das BUSRQ-Signal nicht aktiv ist.
$\overline{NMI}$	Nicht maskierbarer Interrupt (“nonmaskable interrupt”)	Eingang, Low-aktiv. Eine Interrupt-Anforderung auf diesem Eingang hat höhere Priorität als Interruptanforderungen auf dem Eingang INT für maskierbare Interrupts und wird durch das interne Interrupt-Freigabe-Flip Flop nicht beeinflußt. Gelangt ein Low-Signal an den NMI-Eingang, so wird entsprechend einer RESTART-Instruktion die Programmbehandlung bei Speicheradresse 0066H fortgesetzt.
$\overline{RESET}$	Rückstellen (“Reset”)	Eingang, Low-aktiv. Bewirkt Rücksetzen (= 0) von Interrupt-Freigabe-Flip-Flop, Befehlszähler, Register I und R und bringt die CPU in die 8080-Interrupt-Betriebsart. Während des Rückstellungsvorgangs befinden sich Daten- und Adreßbus im hochohmigen, sämtliche übrigen Ausgänge im inaktiven Zustand.
$\overline{BUSRQ}$	Bus-Anforderung (“Bus Request”)	Eingang, Low-aktiv. Bringt Adreß-, Daten- und Steuerbus-Signal in den hochohmigen Zustand, sodaß externe Schaltungen diese Leitungen benutzen können.
$\overline{BUSAK}$	Bus-Anforderungsbestätigung (“Bus Acknowledgement”)	Ausgang, Low-aktiv. Bestätigt, daß Adreß-, Daten- und Steuerbus in den hochohmigen Zustand gebracht wurden.

☐ Befehlslesezyklus ("Instruction Op-Code-Fetch")

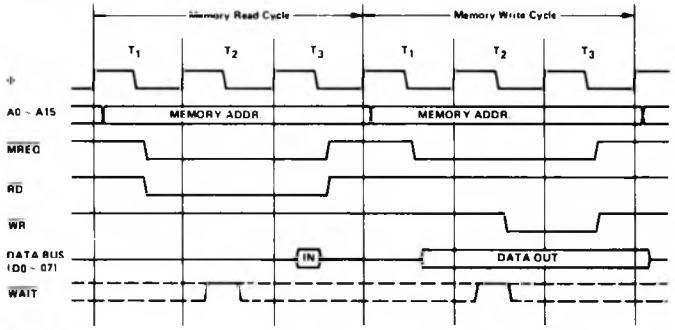
Der Inhalt des Befehlszählers wird bei Beginn des Zyklus auf den Adreßbus gebracht; 1/2 Taktimpuls später wird $\overline{MREQ}$ aktiv, sodaß die fallende Flanke des $\overline{MREQ}$ direkt als Freigabe-Signal ("Chip Enable") für dynamische Speicherbau-



steine benutzt werden kann. Während $\overline{RD}$ = aktiv müssen die Daten aus dem Speicher auf den Datenbus gebracht werden; die CPU holt diese Daten mit der steigenden Flanke des Taktimpulses T_3 ab. Taktimpulse T_3 und T_4 eines Lesezyklus' werden benutzt, um dynamischen Speichern während des Befehlsdekodier- und Ausführungsvorgangs in der CPU das Refresh-Signal zu liefern.
$\overline{RFSH}$ gibt diese Refresh-Signale frei.

☐ Speicherzugriffszyklen ("Memory-Read-or-Write")

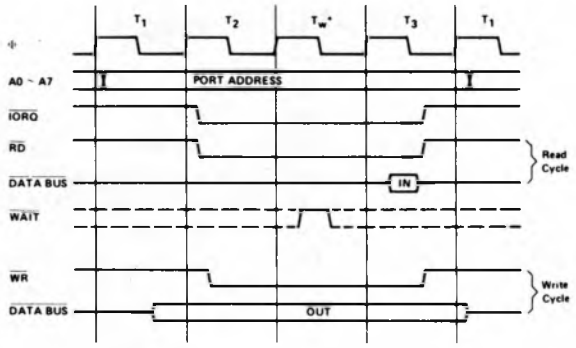
Bei Lese- oder Schreibzyklen verhalten sich die Signale $\overline{MREQ}$ und $\overline{RD}$ genauso wie beim Befehlslesezyklus. Beim Schreibzyklus wird $\overline{MREQ}$ aktiv, sobald die Information auf dem Adreßbus stabil ist, sodaß dieses Signal direkt als Bausteinauswahlsignal ("Chip-Enable") für dynamische Speicher verwendet werden kann.



$\overline{WR}$ ist aktiv, sobald die Informationen auf dem Datenbus stabil sind, sodaß er für praktisch alle Halbleiterspeicher als R/W-Signal zu verwenden ist.

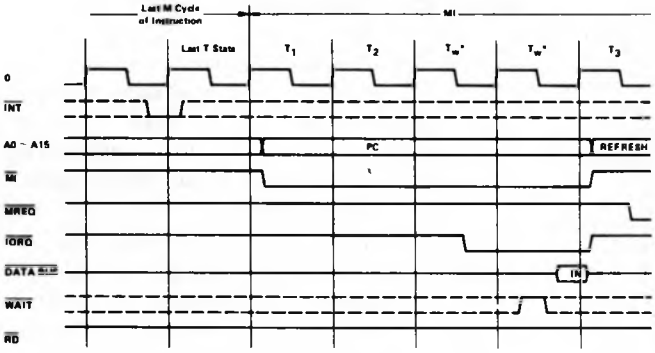
☐ Ein/Ausgabezyklus ("Input or Output Cycles")

Während Ein/Ausgabe Operationen wird automatisch ein Wartezyklus (T_W*) eingefügt, um dem I/O Port genügend Zeit zur Adreßdekodierung und eventueller Aktivierung des WAIT-Signals zu geben.



☐ Unterbrechungsbehandlungs-Zyklen ("Interrupt Request/Acknowledge Cycle")

Der Interrupt-Eingang wird von der CPU bei der steigenden Flanke des letzten Taktimpulses jeder ausgeführten Anweisung abgefragt. Im Interrupt-Fall wird dann ein besonderer M1-Zyklus erzeugt, bei dem $\overline{IORQ}$ statt $\overline{MREQ}$ aktiv wird als Signal für die unterbrechende Schaltung, daß nun ein 8-Bit-Vektor auf den Datenbus zu legen ist. Zwei Wartezyklen T_W* werden hierbei automatisch eingefügt, um der Prioritätsschaltung genügend Zeit zum Durchschalten der Prioritätssignale zu geben.



☐ Z 80-Befehlssatz

Im folgenden wird lediglich ein großer Überblick über den Z 80-Befehlssatz gegeben, Einzelheiten finden Sie im Z-80-Manual. Zu unterscheiden sind folgende Befehlsgruppen:

- 8 Bit-Ladebefehle (="8 bit loads")
- 16 Bit-Ladebefehle (="16 bit loads")
- Austauschbefehle (="Exchanges")
- Blocktransfers im Speicher ("Memory Block Moves")
- Blocksuchbefehle ("Memory Block Searches")
- 8 Bit arthmetische und logische Befehle (="8 bit arithmetic and logic")
- 16 Bit arithmetische Befehle (= 16 bit arithmetic)
- Allgemeine Akkumulator- und Status-Anweisungen (="General purpose Accu and Flag Operations")
- Akku-Rotieren und -Schieben (="Rotate and Shift")
- Bit Setzen, Rücksetzen und Testen (="Bit Set, Reset and Test")
- Ein/Ausgabe (="Input and Output")
- Sprünge (="Jumps")
- Unterprogrammaufrufe ("Calls")
- Restarts ("Restarts")
- Rücksprünge (="Returns")
- Sonstige Befehle ("Miscellaneous Group")

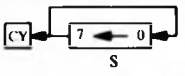
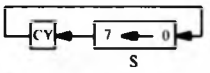
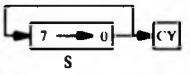
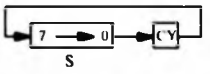
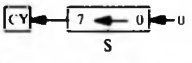
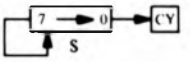
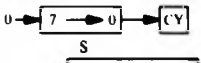
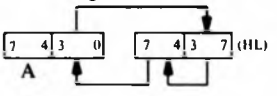
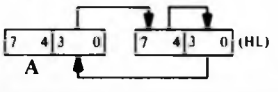
ZEICHENERKLÄRUNG

b	bezeichnet eine Bit-Position in einem Register oder einer Speicherstelle
cc	Statusbedingungscode ("Flag condition") Erlaubte Bedingungen: NZ: ungleich Null (= "Nonzero") Z: gleich Null (= "Zero") NC: Kein Übertrag (= "Non carry") C: Übertrag (= "Carry") PO: Ungerade oder kein Überlauf ("Parity Odd") PE: Gerade oder Überlauf ("Parity Even") P: Positiv N: Negativ
d	Zielregister (8 bit)
dd	16 bit-Zielregister oder Zieladresse im Speicher
e	8 bit vorzeichenbehaftetes Zweierkomplement der Distanz bei relativen Sprüngen oder indizierter Adressierung
L	bezeichnet die 8 speziellen Zieladressen in Seite 0 (dezimal 0, 8, 16, 32, 40, 48 und 56).
n	8 bit-Binärzahl.
nn	16 bit-Binärzahl.
r	allgemeines 8 bit-Register (A, B, C, D, E, H oder L)
s	8 bit Senderregister oder Speicherstelle.
s_b	ein Bit in einem bestimmten 8 bit-Register oder Speicherstelle.
ss	16 bit Senderegister oder Speicherstelle.
Index „L"	Niederwertige (= "Low order") 8 bits eines 16-bit-Registers
Index „H"	Höherwertige (= "High order") 8 bits eines 16-bit-Registers
( )	„Inhalt von . . . ." Zeichen zwischen den Klammern stellen einen Zeiger auf eine Speicherstelle oder ein I/O Port dar.

8 bit Register sind: A, B, C, D, E, H, L, I und R.
16 bit „Paare": AF, BC, DE und HL.
16 bit Register: SP, PC, IX und IY.

Als Adressierweisen kommen (auch Kombinationen) in Frage:
Direkt ("immediate")
Erweitert direkt ("immediate extended")
Modifizierte Seite Null ("Modified Page Zero")
Relativ ("relative")
Erweitert ("extended")
Indiziert ("indexed")
über Register:
impliziert ("implied")
indirekt über Register ("register indirect")
Adressierung eines Bits.

	Mnemo	Durchgeführte Operation	Bemerkungen
8 bit-Ladebefehle	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16 bit-Ladebefehle	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H$; $(SP-2) \leftarrow ss_L$	ss = BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP)$; $dd_H \leftarrow (SP+1)$	dd = BC, DE, HL, AF, IX, IY
Registeraustausch	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix} BC \\ DE \\ HL \end{pmatrix} \leftrightarrow \begin{pmatrix} BC' \\ DE' \\ HL' \end{pmatrix}$	
	EX (SP), ss	$(SP) \leftrightarrow ss_L$, $(SP+1) \leftrightarrow ss_H$	ss ≡ HL, IX, IY
Block-Suchbefehle	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
	CPI	A-(HL), HL ← HL+1 BC ← BC-1	
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	A-(HL) sets the flags only. A is not affected
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, Repeat until BC= 0 or A = (HL)	
8 bit-arithmetische und logische Operationen	ADD s	A ← A + s	
	ADC s	A ← A + s + CY	CY is the carry flag
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ⊕ s	

	Mnemo	Durchgeführte Operation	Bemerkungen
16 bit-arithmetische Operationen	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d - 1	
	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
BCD-, Akku- und Flag-Operationen	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	A ← 00 – A	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
Verschiedene	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemo	Durchgeführte Operation	Bemerkungen
Bit Setzen, Rücksetzen, Testen	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag
	SET b, s	$s_b \leftarrow 1$	s ≡ r, (HL) (IX+e), (IY+e)
	RES b, s	$s_b \leftarrow 0$	
Ein/Ausgabe	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
Sprungbefehle	JP nn	PC ← nn	cc ≡ NZ PO, Z PE, NC P, C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	
	JR kk, e	If condition kk is true PC ← PC + e, else continue	kk ≡ NZ NC, Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
Unterprogrammaufruf	CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, PC ← nn	cc ≡ NZ PO, Z PE, NC P, C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
Restarts	RST L	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$ $PC_L \leftarrow L$	
Rücksprünge	RET	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$	cc ≡ NZ PO, Z PE, NC P, C M
	RET cc	If condition cc is false continue, else same as RET	
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

Z80A-CPU-Befehlssatz, alphabetisch sortiert

ALPHABETICAL ASSEMBLY MNEMONIC	OPERATION
ADC HL,ss	Add with Carry Reg. pair ss to HL
ADC A,s	Add with carry operand s to Acc.
ADD A,n	Add value n to Acc.
ADD A,r	Add Reg. r to Acc.
ADD A,(HL)	Add location (HL) to Acc.
ADD A,(IX+d)	Add location (IX+d) to Acc.
ADD A,(IY+d)	Add location (IY+d) to Acc.
ADD HL,ss	Add Reg. pair ss to HL
ADD IX,pp	Add Reg. pair pp to IX
ADD IY,rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b,(HL)	Test BIT b of location (HL)
BIT b,(IX+d)	Test BIT b of location (IX+d)
BIT b,(IY+d)	Test BIT b of location (IY+d)
BIT b,r	Test BIT b of Reg. r
CALL cc,nn	Call subroutine at location nn if condition cc is true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B≠0
EI	Enable interrupts
EX (SP),HL	Exchange the location (SP) and HL
EX (SP),IX	Exchange the location (SP) and IX
EX (SP),IY	Exchange the location (SP) and IY
EX AF,AF'	Exchange the contents of AF and AF'
EX DE,HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC,DE,HL with contents of BC',DE',HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A,(n)	Load the Acc. with input from device n
IN r,(C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditional Jump to (IY)
JP cc,nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JR C,e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JR NC,e	Jump relative to PC+e if carry=0
JR NZ,e	Jump relative to PC+e if non zero (Z=0)
JR Z,e	Jump relative to PC+e if zero (Z=1)
LD A,(BC)	Load Acc. with location (BC)
LD A,(DE)	Load Acc. with location (DE)
LD A,I	Load Acc. with I
LD A,(nn)	Load Acc. with location nn
LD A,R	Load Acc. with Reg. R
LD (BC),A	Load location (BC) with Acc.
LD (DE),A	Load location (DE) with Acc.
LD (HL),n	Load location (HL) with value n
LD dd,nn	Load Reg. pair dd with value nn
LD dd,(nn)	Load Reg. pair dd with location (nn)
LD HL,(nn)	Load HL with location (nn)
LD (HL),r	Load location (HL) with Reg. r
LD I,A	Load I with Acc.
LF IX,nn	Load IX with value nn
LD IX,(nn)	Load IX with location (nn)
LD (IX+d),n	Load location (IX+d) with value n
LD (IX+d),r	Load location (IX+d) with Reg. r
LD IY,nn	Load IY with value nn
LD IY,(nn)	Load IY with location (nn)
LD (IY+d),n	Load location (IY+d) with value n
LD (IY+d),r	Load location (IY+d) with Reg. r
LD (nn),A	Load location (nn) with Acc.
LD (nn),dd	Load location (nn) with Reg. pair dd
LD (nn),HL	Load location (nn) with HL
LD (nn),IX	Load location (nn) with IX
LD (nn),IY	Load location (nn) with IY
LD R,A	Load R with Acc.
LD r,(HL)	Load Reg. r with location (HL)
LD r,(IX+d)	Load Reg. r with location (IX+d)
LD r,(IY+d)	Load Reg. r with location (IY+d)
LD r,n	Load Reg. r with value n
LD r,r'	Load Reg. r with Reg. r'
LD SP,HL	Load SP with HL
LD SP,IX	Load SP with IX
LD SP,IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE,HL and BC
LDDR	Load location (DE) with location (HL), decrement DE,HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE,HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE,HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' of operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C),r	Load output port (C) with Reg. r
OUT (n),A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B
POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b,m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circclar
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A,s	Subtract operand s from Acc. with carry
SBC HL,ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b,(HL)	Set Bit b of location (HL)
SET b,(IX+d)	Set Bit b of location (IX+d)
SET b,(IY+d)	Set Bit b of location (IY+d)
SET b,r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

Befehlsvergleichsliste der Systeme Z80A und 8080A

Opcode	8080A		Z80A		Opcode	8080A		Z80A	
00	NOP		NOP		30	----		JR	NC,disp
01	LXI	B,dddd	LD	BC,dddd	31	LXI	SP,dddd	LD	SP,dddd
02	STAX	B	LD	(BC),A	32	STA	adr	LD	(adr),A
03	INX	B	INC	BC	33	INX	SP	INC	SP
04	INR	B	INC	B	34	INR	M	INC	(HL)
05	DCR	B	DEC	B	35	DCR	M	DEC	(HL)
06	MVI	B,dd	LD	B,dd	36	MVI	M,dd	LD	(HL),dd
07	RLC		RLCA		37	STC		SCF	
08			EX	AF,AF'	38	----		JR	C,disp
09	DAD	B	ADD	HL,BC	39	DAD	SP	ADD	HL,SP
0A	LDAX	B	LD	A,(BC)	3A	LDA	adr	LD	A,(adr)
0B	DCX	B	DEC	BC	3B	DCX	SP	DEC	SP
0C	INR	C	INC	C	3C	INR	A	INC	A
0D	DCR	C	DEC	C	3D	DCR	A	DEC	A
0E	MVI	C,dd	LD	C,dd	3E	MVI	A,dd	LD	A,dd
0F	RRC		RRCA		3F	CMC		CCF	
10	----		DJNZ	disp	40	MOV	B,B	LD	B,B
11	LXI	D,dddd	LD	DE,dddd	41	MOV	B,C	LD	B,C
12	STAX	D	LD	(DE),A	42	MOV	B,D	LD	B,D
13	INX	D	INC	DE	43	MOV	B,E	LD	B,E
14	INR	D	INC	D	44	MOV	B,H	LD	B,H
15	DCR	D	DEC	D	45	MOV	B,L	LD	B,L
16	MVI	D,dd	LD	D,dd	46	MOV	B,M	LD	B,(HL)
17	RAL		RLA		47	MOV	B,A	LD	B,A
18	----		JR	disp	48	MOV	C,B	LD	C,B
19	DAD	D	ADD	HL,DE	49	MOV	C,C	LD	C,C
1A	LDAX	D	LD	A,(DE)	4A	MOV	C,D	LD	C,D
1B	DCX	D	DEC	DE	4B	MOV	C,E	LD	C,E
1C	INR	E	INC	E	4C	MOV	C,H	LD	C,H
1D	DCR	E	DEC	E	4D	MOV	C,L	LD	C,L
1E	MVI	E,dd	LD	E,dd	4E	MOV	C,M	LD	C,(HL)
1F	RAR		RRA		4F	MOV	C,A	LD	C,A
20	----		JR	NZ,disp	50	MOV	D,B	LD	D,B
21	LXI	H,dddd	LD	HL,dddd	51	MOV	D,C	LD	D,C
22	SHLD	adr	LD	(adr),HL	52	MOV	D,D	LD	D,D
23	INX	H	INC	HL	53	MOV	D,E	LD	D,E
24	INR	H	INC	H	54	MOV	D,H	LD	D,H
25	DCR	H	DEC	H	55	MOV	D,L	LD	D,L
26	MVI	H,dd	LD	H,dd	56	MOV	D,M	LD	D,(HL)
27	DAA		DAA		57	MOV	D,A	LD	D,A
28	----		JR	Z,disp	58	MOV	E,B	LD	E,B
29	DAD	H	ADD	HL,HL	59	MOV	E,C	LD	E,C
2A	LHLD	adr	LD	HL,(adr)	5A	MOV	E,D	LD	E,D
2B	DCX	H	DEC	HL	5B	MOV	E,E	LD	E,E
2C	INR	L	INC	L	5C	MOV	E,H	LD	E,H
2D	DCR	L	DEC	L	5D	MOV	E,L	LD	E,L
2E	MVI	L,dd	LD	L,dd	5E	MOV	E,M	LD	E,(HL)
2F	CMA		CPL		5F	MOV	E,A	LD	E,A

Opcode	8080A		Z80A		Opcode	8080A		Z80A	
60	MOV	H,B	LD	H,B	90	SUB	B	SUB	B
61	MOV	H,C	LD	H,C	91	SUB	C	SUB	C
62	MOV	H,D	LD	H,D	92	SUB	D	SUB	D
63	MOV	H,E	LD	H,E	93	SUB	E	SUB	E
64	MOV	H,H	LD	H,H	94	SUB	H	SUB	H
65	MOV	H,L	LD	H,L	95	SUB	L	SUB	L
66	MOV	H,M	LD	H,(HL)	96	SUB	M	SUB	(HL)
67	MOV	H,A	LD	H,A	97	SUB	A	SUB	A
68	MOV	L,B	LD	L,B	98	SBB	B	SBC	A,B
69	MOV	L,C	LD	L,C	99	SBB	C	SBC	A,C
6A	MOV	L,D	LD	L,D	9A	SBB	D	SBC	A,D
6B	MOV	L,E	LD	L,E	9B	SBB	E	SBC	A,E
6C	MOV	L,H	LD	L,H	9C	SBB	H	SBC	A,H
6D	MOV	L,L	LD	L,L	9D	SBB	L	SBC	A,L
6E	MOV	L,M	LD	L,(HL)	9E	SBB	M	SBC	A,(HL)
6F	MOV	L,A	LD	L,A	9F	SBB	A	SBC	A,A
70	MOV	M,B	LD	(HL),B	A0	ANA	B	AND	B
71	MOV	M,C	LD	(HL),C	A1	ANA	C	AND	C
72	MOV	M,D	LD	(HL),D	A2	ANA	D	AND	D
73	MOV	M,E	LD	(HL),E	A3	ANA	E	AND	E
74	MOV	M,H	LD	(HL),H	A4	ANA	H	AND	H
75	MOV	M,L	LD	(HL),L	A5	ANA	L	AND	L
76	HLT		HALT		A6	ANA	M	AND	(HL)
77	MOV	M,A	LD	(HL),A	A7	ANA	A	AND	A
78	MOV	A,B	LD	A,B	A8	XRA	B	XOR	B
79	MOV	A,C	LD	A,C	A9	XRA	C	XOR	C
7A	MOV	A,D	LD	A,D	AA	XRA	D	XOR	D
7B	MOV	A,E	LD	A,E	AB	XRA	E	XOR	E
7C	MOV	A,H	LD	A,H	AC	XRA	H	XOR	H
7D	MOV	A,L	LD	A,L	AD	XRA	L	XOR	L
7E	MOV	A,M	LD	A,(HL)	AE	XRA	M	XOR	(HL)
7F	MOV	A,A	LD	A,A	AF	XRA	A	XOR	A
80	ADD	B	ADD	A,B	B0	ORA	B	OR	B
81	ADD	C	ADD	A,C	B1	ORA	C	OR	C
82	ADD	D	ADD	A,D	B2	ORA	D	OR	D
83	ADD	E	ADD	A,E	B3	ORA	E	OR	E
84	ADD	H	ADD	A,H	B4	ORA	H	OR	H
85	ADD	L	ADD	A,L	B5	ORA	L	OR	L
86	ADD	M	ADD	A,(HL)	B6	ORA	M	OR	(HL)
87	ADD	A	ADD	A,A	B7	ORA	A	OR	A
88	ADC	B	ADC	A,B	B8	CMP	B	CP	B
89	ADC	C	ADC	A,C	B9	CMP	C	CP	C
8A	ADC	D	ADC	A,D	BA	CMP	D	CP	D
8B	ADC	E	ADC	A,E	BB	CMP	E	CP	E
8C	ADC	H	ADC	A,H	BC	CMP	H	CP	H
8D	ADC	L	ADC	A,L	BD	CMP	L	CP	L
8E	ADC	M	ADC	A,(HL)	BE	CMP	M	CP	(HL)
8F	ADC	A	ADC	A,A	BF	CMP	A	CP	A

Opcode	8080A		Z80A	
C0	RNZ		RET	NZ
C1	POP	B	POP	BC
C2	JNZ	adr	JP	NZ,adr
C3	JMP	adr	JP	adr
C4	CNZ	adr	CALL	NZ,adr
C5	PUSH	B	PUSH	BC
C6	ADI	dd	ADD	A,dd
C7	RST	0	RST	0
C8	RZ		RET	Z
C9	RET		RET	
CA	JZ	adr	JP	Z,adr
CB	----		see below	
CC	CZ	adr	CALL	Z,adr
CD	CALL	adr	CALL	adr
CE	ACI	dd	ADC	A,dd
CF	RST	1	RST	8
D0	RNC		RET	NC
D1	POP	D	POP	DE
D2	JNC	adr	JP	NC,adr
D3	OUT	port	OUT	port,A
D4	CNC	adr	CALL	NC,adr
D5	PUSH	D	PUSH	DE
D6	SUI	dd	SUB	dd
D7	RST	2	RST	10H
D8	RC		RET	C
D9	----		EXX	
DA	JC	adr	JP	C,adr
DB	IN	port	IN	A,port
DC	CC	adr	CALL	C,adr
DD	----		see below	
DE	SBI	dd	SBC	A,dd
DF	RST	3	RST	18H
E0	RPO		RET	PO
E1	POP	H	POP	HL
E2	JPO	adr	JP	PO,adr
E3	XTHL		EX	(SP),HL
E4	CPO	adr	CALL	PO,adr
E5	PUSH	H	PUSH	HL
E6	ANI	dd	AND	dd
E7	RST	4	RST	20H
E8	RPE		RET	PE
E9	PCHL		JP	(HL)
EA	JPE	adr	JP	PE,adr
EB	XCHG		EX	DE,HL
EC	CPE	adr	CALL	PE,adr
ED	----		see below	
EE	XRI	dd	XOR	dd
EF	RST	5	RST	28H
F0	RP		RET	P
F1	POP	PSW	POP	AF
F2	JP	adr	JP	P,adr
F3	DI		DI	
F4	CP	adr	CALL	P,adr
F5	PUSH	PSW	PUSH	AF
F6	ORI	dd	OR	dd
F7	RST	6	RST	30H
F8	RM		RET	M
F9	SPHL		LD	SP,HL
FA	JM	adr	JP	M,adr
FB	EI		EI	
FC	CM	adr	CALL	M,adr
FD	----		see below	
FE	CPI	dd	CP	dd
FF	RST	7	RST	38H

Folgende Anweisungen sind nur im Befehlssatz des Systems Z80A implementiert

Opcode

CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
CB06	RLC	(HL)
CB07	RLC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
CB0E	RRC	(HL)
CB0F	RRC	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
CB16	RL	(HL)
CB17	RL	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
CB1E	RR	(HL)
CB1F	RR	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB26	SLA	(HL)
CB27	SLA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB2E	SRA	(HL)
CB2F	SRA	A

CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
CB3E	SRL	(HL)
CB3F	SRL	A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB46	BIT	0,(HL)
CB47	BIT	0,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB4E	BIT	1,(HL)
CB4F	BIT	1,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB56	BIT	2,(HL)
CB57	BIT	2,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB5E	BIT	3,(HL)
CB5F	BIT	3,A

Opcode			Opcode		
CB60	BIT	4,B	CB90	RES	2,B
CB61	BIT	4,C	CB91	RES	2,C
CB62	BIT	4,D	CB92	RES	2,D
CB63	BIT	4,E	CB93	RES	2,E
CB64	BIT	4,H	CB94	RES	2,H
CB65	BIT	4,L	CB95	RES	2,L
CB66	BIT	4,(HL)	CB96	RES	2,(HL)
CB67	BIT	4,A	CB97	RES	2,A
CB68	BIT	5,B	CB98	RES	3,B
CB69	BIT	5,C	CB99	RES	3,C
CB6A	BIT	5,D	CB9A	RES	3,D
CB6B	BIT	5,E	CB9B	RES	3,E
CB6C	BIT	5,H	CB9C	RES	3,H
CB6D	BIT	5,L	CB9D	RES	3,L
CB6E	BIT	5,(HL)	CB9E	RES	3,(HL)
CB6F	BIT	5,A	CB9F	RES	3,A
CB70	BIT	6,B	CBA0	RES	4,B
CB71	BIT	6,C	CBA1	RES	4,C
CB72	BIT	6,D	CBA2	RES	4,D
CB73	BIT	6,E	CBA3	RES	4,E
CB74	BIT	6,H	CBA4	RES	4,H
CB75	BIT	6,L	CBA5	RES	4,L
CB76	BIT	6,(HL)	CBA6	RES	4,(HL)
CB77	BIT	6,A	CBA7	RES	4,A
CB78	BIT	7,B	CBA8	RES	5,B
CB79	BIT	7,C	CBA9	RES	5,C
CB7A	BIT	7,D	CBAA	RES	5,D
CB7B	BIT	7,E	CBAB	RES	5,E
CB7C	BIT	7,H	CBAC	RES	5,H
CB7D	BIT	7,L	CBAD	RES	5,L
CB7E	BIT	7,(HL)	CBAE	RES	5,(HL)
CB7F	BIT	7,A	CBAF	RES	5,A
CB80	RES	0,B	CBB0	RES	6,B
CB81	RES	0,C	CBB1	RES	6,C
CB82	RES	0,D	CBB2	RES	6,D
CB83	RES	0,E	CBB3	RES	6,E
CB84	RES	0,H	CBB4	RES	6,H
CB85	RES	0,L	CBB5	RES	6,L
CB86	RES	0,(HL)	CBB6	RES	6,(HL)
CB87	RES	0,A	CBB7	RES	6,A
CB88	RES	1,B	CBB8	RES	7,B
CB89	RES	1,C	CBB9	RES	7,C
CB8A	RES	1,D	CBBA	RES	7,D
CB8B	RES	1,E	CBBB	RES	7,E
CB8C	RES	1,H	CBBC	RES	7,H
CB8D	RES	1,L	CBBD	RES	7,L
CB8E	RES	1,(HL)	CBBE	RES	7,(HL)
CB8F	RES	1,A	CBBF	RES	7,A

Opcode

```
CBC0    SET   0,B
CBC1    SET   0,C
CBC2    SET   0,D
CBC3    SET   0,E
CBC4    SET   0,H
CBC5    SET   0,L
CBC6    SET   0,(HL)
CBC7    SET   0,A
CBC8    SET   1,B
CBC9    SET   1,C
CBCA    SET   1,D
CBCB    SET   1,E
CBCC    SET   1,H
CBCD    SET   1,L
CBCE    SET   1,(HL)
CBCF    SET   1,A

CBD0    SET   2,B
CBD1    SET   2,C
CBD2    SET   2,D
CBD3    SET   2,E
CBD4    SET   2,H
CBD5    SET   2,L
CBD6    SET   2,(HL)
CBD7    SET   2,A
CBD8    SET   3,B
CBD9    SET   3,C
CBDA    SET   3,D
CBDB    SET   3,E
CBDC    SET   3,H
CBDD    SET   3,L
CBDE    SET   3,(HL)
CBDF    SET   3,A

CBE0    SET   4,B
CBE1    SET   4,C
CBE2    SET   4,D
CBE3    SET   4,E
CBE4    SET   4,H
CBE5    SET   4,L
CBE6    SET   4,(HL)
CBE7    SET   4,A
CBE8    SET   5,B
CBE9    SET   5,C
CBEA    SET   5,D
CBEB    SET   5,E
CBEC    SET   5,H
CBED    SET   5,L
CBEE    SET   5,(HL)
CBEF    SET   5,A
```

Opcode

```
CBF0    SET   6,B
CBF1    SET   6,C
CBF2    SET   6,D
CBF3    SET   6,E
CBF4    SET   6,H
CBF5    SET   6,L
CBF6    SET   6,(HL)
CBF7    SET   6,A
CBF8    SET   7,B
CBF9    SET   7,C
CBFA    SET   7,D
CBFB    SET   7,E
CBFC    SET   7,H
CBFD    SET   7,L
CBFE    SET   7,(HL)
CBFF    SET   7,A

DD09    ADD   IX,BC

DD19    ADD   IX,DE

DD21    LD    IX,dddd
DD22    LD    (adr),IX
DD23    INC   IX
DD29    ADD   IX,IX
DD2A    LD    IX,(adr)
DD2B    DEC   IX

DD34    INC   (IX+offset)
DD35    DEC   (IX+offset)
DD36    LD    (IX+offset),dd
DD39    ADD   IX,SP

DD46    LD    B,(IX+offset)
DD4E    LD    C,(IX+offset)

DD56    LD    D,(IX+offset)
DD5E    LD    E,(IX+offset)

DD66    LD    H,(IX+offset)
DD6E    LD    L,(IX+offset)

DD70    LD    (IX+offset),B
DD71    LD    (IX+offset),C
DD72    LD    (IX+offset),D
DD73    LD    (IX+offset),E
DD74    LD    (IX+offset),H
DD75    LD    (IX+offset),L
DD77    LD    (IX+offset),A
DD7E    LD    A,(IX+offset)

DD86    ADD   A,(IX+offset)
DD8E    ADC   A,(IX+offset)
```

Opcode		
DD96	SUB	(IX+offset)
DD9E	SBC	A,(IX+offset)
DDA6	AND	(IX+offset)
DDAE	XOR	(IX+offset)
DDB6	OR	(IX+offset)
DDBE	CP	(IX+offset)
DDCBof06	RLC	(IX+offset)
DDCBof0E	RRC	(IX+offset)
DDCBof16	RL	(IX+offset)
DDCBof1E	RR	(IX+offset)
DDCBof26	SLA	(IX+offset)
DDCBof2E	SRA	(IX+offset)
DDCBof3E	SRL	(IX+offset)
DDCBof46	BIT	0,(IX+offset)
DDCBof4E	BIT	1,(IX+offset)
DDCBof56	BIT	2,(IX+offset)
DDCBof5E	BIT	3,(IX+offset)
DDCBof66	BIT	4,(IX+offset)
DDCBof6E	BIT	5,(IX+offset)
DDCBof76	BIT	6,(IX+offset)
DDCBof7E	BIT	7,(IX+offset)
DDCBof86	RES	0,(IX+offset)
DDCBof8E	RES	1,(IX+offset)
DDCBof96	RES	2,(IX+offset)
DDCBof9E	RES	3,(IX+offset)
DDCBofA6	RES	4,(IX+offset)
DDCBofAE	RES	5,(IX+offset)
DDCBofB6	RES	6,(IX+offset)
DDCBofBE	RES	7,(IX+offset)
DDCBofC6	SET	0,(IX+offset)
DDCBofCE	SET	1,(IX+offset)
DDCBofD6	SET	2,(IX+offset)
DDCBofDE	SET	3,(IX+offset)
DDCBofE6	SET	4,(IX+offset)
DDCBofEE	SET	5,(IX+offset)
DDCBofF6	SET	6,(IX+offset)
DDCBofFE	SET	7,(IX+offset)

Opcode		
DDE1	POP	IX
DDE3	EX	(SP),IX
DDE5	PUSH	IX
DDE9	JP	(IX)
DDF9	LD	SP,IX
ED40	IN	B,(C)
ED41	OUT	(C),B
ED42	SBC	HL,BC
ED43	LD	(dddd),BC
ED44	NEG	
ED45	RETN	
ED46	IM	0
ED47	LD	I,A
ED48	IN	C,(C)
ED49	OUT	(C),C
ED4A	ADC	HL,BC
ED4B	LD	BC,(adr)
ED4D	RETI	
ED50	IN	D,(C)
ED51	OUT	(C),D
ED52	SBC	HL,DE
ED53	LD	(adr),DE
ED56	IM	1
ED57	LD	A,I
ED58	IN	E,(C)
ED59	OUT	(C),E
ED5A	ADC	HL,DE
ED5B	LD	DE,(adr)
ED5E	IM	2
ED60	IN	H,(C)
ED61	OUT	(C),H
ED62	SBC	HL,HL
ED67	RRD	
ED68	IN	L,(C)
ED69	OUT	(C),L
ED6A	ADC	HL,HL
ED6F	RLD	
ED72	SBC	HL,SP
ED73	LD	(adr),SP
ED78	IN	A,(C)
ED79	OUT	(C),A
ED7A	ADC	HL,SP
ED7B	LD	SP,(adr)
EDA0	LDI	
EDA1	CPI	
EDA2	INI	
EDA3	OUTI	
EDA8	LDD	
EDA9	CPD	
EDAA	IND	
EDAB	OUTD	

Opcode		
ED80	LDIR	
ED81	CPIR	
ED82	INIR	
ED83	OTIR	
ED88	LDDR	
ED89	CPDR	
ED8A	INDR	
ED8B	OTDR	
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD21	LD	IY,dddd
FD22	LD	(adr),IY
FD23	INC	IY
FD29	ADD	IY,IY
FD2A	LD	IY,(adr)
FD2B	DEC	IY
FD34	INC	(IY+offset)
FD35	DEC	(IY+offset)
FD36	LD	(IY+offset),dd
FD39	ADD	IY,SP
FD46	LD	B,(IY+offset)
FD4E	LD	C,(IY+offset)
FD56	LD	D,(IY+offset)
FD5E	LD	E,(IY+offset)
FD66	LD	H,(IY+offset)
FD6E	LD	L,(IY+offset)
FD70	LD	(IY+offset),B
FD71	LD	(IY+offset),C
FD72	LD	(IY+offset),D
FD73	LD	(IY+offset),E
FD74	LD	(IY+offset),H
FD75	LD	(IY+offset),L
FD77	LD	(IY+offset),A
FD7E	LD	A,(IY+offset)
FD86	ADD	A,(IY+offset)
FD8E	ADC	A,(IY+offset)
FD96	SUB	(IY+offset)
FD9E	SBC	A,(IY+offset)
FDA6	AND	(IY+offset)
FDAE	XOR	(IY+offset)
FDB6	OR	(IY+offset)
FDBE	CP	(IY+offset)

Opcode		
FDCBof06	RLC	(IY+offset)
FDCBof0E	RRC	(IY+offset)
FDCBof16	RL	(IY+offset)
FDCBof1E	RR	(IY+offset)
FDCBof26	SLA	(IY+offset)
FDCBof2E	SRA	(IY+offset)
FDCBof3E	SRL	(IY+offset)
FDCBof46	BIT	0,(IY+offset)
FDCBof4E	BIT	1,(IY+offset)
FDCBof56	BIT	2,(IY+offset)
FDCBof5E	BIT	3,(IY+offset)
FDCBof66	BIT	4,(IY+offset)
FDCBof6E	BIT	5,(IY+offset)
FDCBof76	BIT	6,(IY+offset)
FDCBof7E	BIT	7,(IY+offset)
FDCBof86	RES	0,(IY+offset)
FDCBof8E	RES	1,(IY+offset)
FDCBof96	RES	2,(IY+offset)
FDCBof9E	RES	3,(IY+offset)
FDCBofA6	RES	4,(IY+offset)
FDCBofAE	RES	5,(IY+offset)
FDCBofB6	RES	6,(IY+offset)
FDCBofBE	RES	7,(IY+offset)
FDCBofC6	SET	0,(IY+offset)
FDCBofCE	SET	1,(IY+offset)
FDCBofD6	SET	2,(IY+offset)
FDCBofDE	SET	3,(IY+offset)
FDCBofE6	SET	4,(IY+offset)
FDCBofEE	SET	5,(IY+offset)
FDCBofF6	SET	6,(IY+offset)
FDCBofFE	SET	7,(IY+offset)
FDE1	POP	IY
FDE3	EX	(SP),IY
FDE5	PUSH	IY
FDE9	JP	(IY)
FDF9	LD	SP,IY

□ Dynamische Kenndaten der Z80A-CPU

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.25	[12]	μsec	
	t_w (ΦH)	Clock Pulse Width, Clock High	110	[E]	nsec	
	t_w (ΦL)	Clock Pulse Width, Clock Low	110	2000	nsec	
	t_r, t_f	Clock Rise and Fall Time		30	nsec	
A_{0-15}	t_D (AD)	Address Output Delay		110	nsec	C_L = 50pF
	t_F (AD)	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from RD, WR, IORQ or MREQ	[3]		nsec	
	t_{caf}	Address Stable From RD or WR During Float	[4]		nsec	
D_{0-7}	t_D (D)	Data Output Delay		150	nsec	C_L = 50pF
	t_F (D)	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi}$ (D)	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\bar{\Phi}}$ (D)	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From WR	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
MREQ	$t_{DL\bar{\Phi}}$ (MR)	MREQ Delay From Falling Edge of Clock, MREQ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi}$ (MR)	MREQ Delay From Rising Edge of Clock, MREQ High		85	nsec	
	$t_{DH\bar{\Phi}}$ (MR)	MREQ Delay From Falling Edge of Clock, MREQ High		85	nsec	
	t_w (MRL)	Pulse Width, MREQ Low	[8]		nsec	
	t_w (MRH)	Pulse Width, MREQ High	[9]		nsec	
IORQ	$t_{DL\Phi}$ (IR)	IORQ Delay From Rising Edge of Clock, IORQ Low		75	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}}$ (IR)	IORQ Delay From Falling Edge of Clock, IORQ Low		85	nsec	
	$t_{DH\Phi}$ (IR)	IORQ Delay From Rising Edge of Clock, IORQ High		85	nsec	
	$t_{DH\bar{\Phi}}$ (IR)	IORQ Delay From Falling Edge of Clock, IORQ High		85	nsec	
RD	$t_{DL\Phi}$ (RD)	RD Delay From Rising Edge of Clock, RD Low		85	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}}$ (RD)	RD Delay From Falling Edge of Clock, RD Low		95	nsec	
	$t_{DH\Phi}$ (RD)	RD Delay From Rising Edge of Clock, RD High		85	nsec	
	$t_{DH\bar{\Phi}}$ (RD)	RD Delay From Falling Edge of Clock, RD High		85	nsec	
WR	$t_{DL\Phi}$ (WR)	WR Delay From Rising Edge of Clock, WR Low		65	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}}$ (WR)	WR Delay From Falling Edge of Clock, WR Low		80	nsec	
	$t_{DH\bar{\Phi}}$ (WR)	WR Delay From Falling Edge of Clock, WR High		80	nsec	
	t_w (WRL)	Pulse Width, WR Low	[10]		nsec	
M1	t_{DL} (M1)	M1 Delay From Rising Edge of Clock, M1 Low		100	nsec	C_L = 50pF
	t_{DH} (M1)	M1 Delay From Rising Edge of Clock, M1 High		100	nsec	
RFSH	t_{DL} (RF)	RFSH Delay From Rising Edge of Clock, RFSH Low		130	nsec	C_L = 50pF
	t_{DH} (RF)	RFSH Delay From Rising Edge of Clock, RFSH High		120	nsec	
WAIT	t_s (WT)	WAIT Setup Time to Falling Edge of Clock	70		nsec	
HALT	t_D (HT)	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
INT	t_s (IT)	INT Setup Time to Rising Edge of Clock	80		nsec	
NMI	t_w (NML)	Pulse Width, NMI Low	80		nsec	
BUSRQ	t_s (BQ)	BUSRQ Setup Time to Rising Edge of Clock	50		nsec	
BUSAK	t_{DL} (BA)	BUSAK Delay From Rising Edge of Clock, BUSAK Low		100	nsec	C_L = 50pF
	t_{DH} (BA)	BUSAK Delay From Falling Edge of Clock, BUSAK High		100	nsec	
RESET	t_s (RS)	RESET Setup Time to Rising Edge of Clock	60		nsec	
	t_F (C)	Delay to Float (MREQ, IORQ, RD and WR)		80	nsec	
	t_{mr}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 70$

[8] $t_{w(\overline{MRL})} = t_c - 30$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 20$

[10] $t_{w(\overline{WRL})} = t_c - 30$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 65$

NOTES:

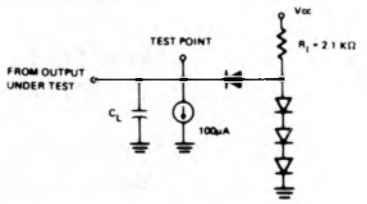
A. Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The RESET signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%
Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum



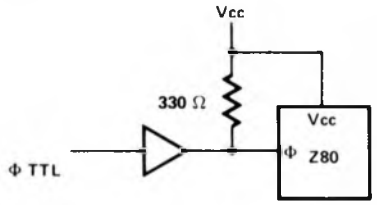
Load circuit for Output

☐ Kapazitäten

$T_A = 25°C$, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_Φ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

[1] Clock Driver



An external clock pull-up resistor of (330Ω) will meet both the A.C. and D.C. clock requirements.

☐ Statische Kenndaten

$T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$ unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC}-.2$		V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH}=-250μA
I_{CC}	Power Supply Current		90	200	mA	t_c=250nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{CC}$

☐ Absolute Grenzdaten*

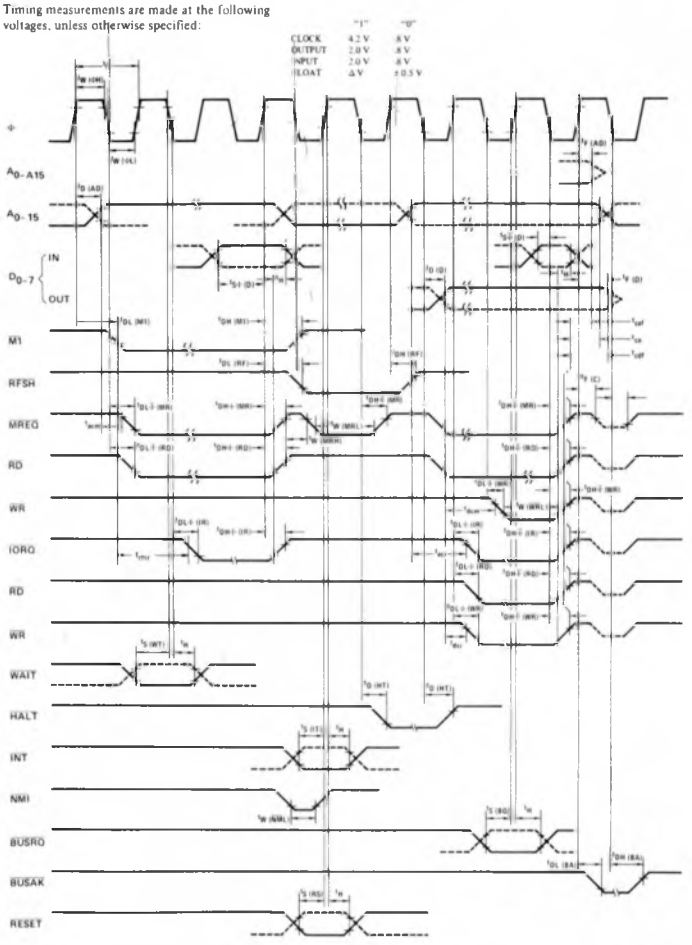
Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

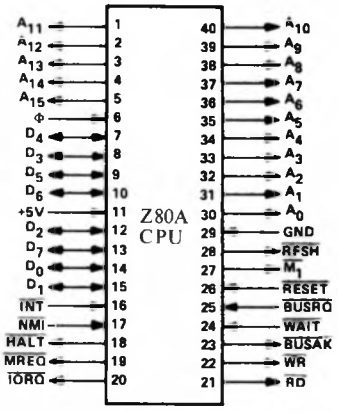
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

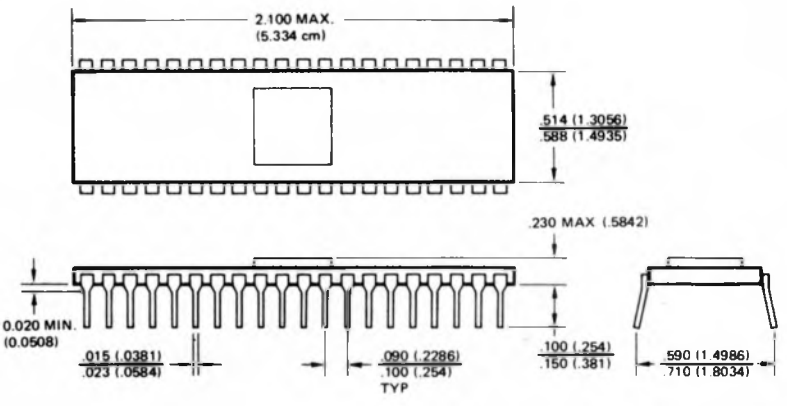
Zeitverhalten der Z80A-CPU

Timing measurements are made at the following voltages, unless otherwise specified:



☐ Pin-Belegung





*Dimensions for metric system are in parenthesis

☐ Bestellbezeichnung

Z80A-CPU/PS Standardversion (0 . . . 70°C) in Plastikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-CPU/CS Standardversion (0 . . . 70°C) im Keramikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-PIO

Zilog

PARALLEL-EIN / AUSGABE-BAUSTEIN

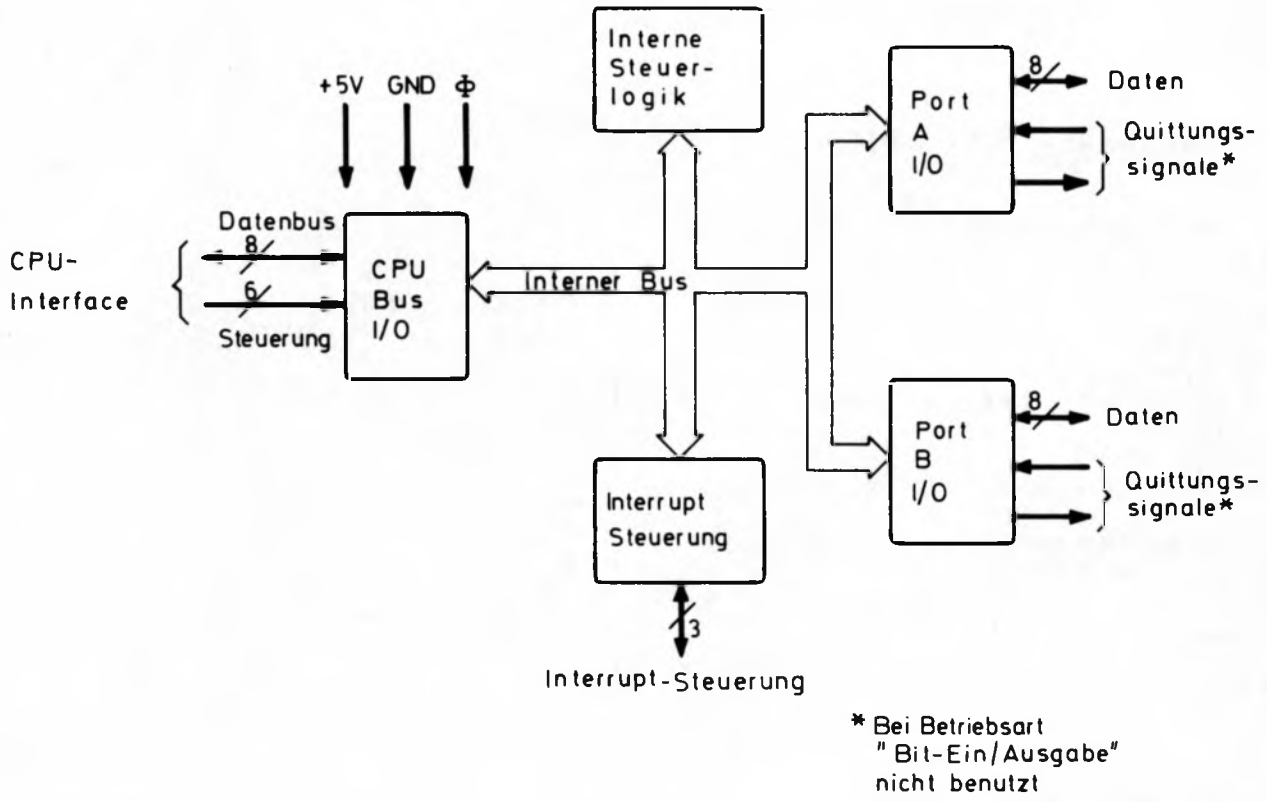


Bild 1: Z80A-PIO Blockschaltbild

1. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Der Software-programmierbare Parallel-I/O (=„PIO")-Interfacebaustein enthält 2 TTL-kompatible Ports, über die der Datenverkehr zwischen dem Mikroprozessor und der Umwelt (=„Peripherie") abgewickelt wird. Er macht jegliche zusätzliche Interrupt-Steuerungs- und Priorisierungslogik überflüssig und erlaubt verschachtelten, priorisierten Interrupt beliebig vieler Ebenen im gesamten Speicherbereich.

☐ Aufbau

- N-Kanal Silicon-Gate-Depletion Load Technologie
- 40 Pin DIP-Gehäuse
- Stromversorgung über eine einzige 5V-Quelle
- 5V-Einphasen-Takt
- Zwei unabhängige 8 bit-bidirektionale Ports mit Einrichtung für Quittungsbetrieb ("Handshaking")

☐ Kenndaten des Z80A-PIO

- Quittungsbetrieb im Interrupt-Mode zur schnellen Anforderungsbearbeitung
- Jedes Port kann in einer der folgenden 4 Betriebsarten arbeiten:
  - ☐ Byte-Ausgabe
  - ☐ Byte-Eingabe
  - ☐ Byte-Ein/Ausgabe (bidirektionaler Betrieb, nur bei Port A)
  - ☐ Bit-Ein/Ausgabe
- Unter Zustandsbedingungen des Peripheren Geräts programmierbare Interruptbearbeitung
- Automatische Interrupt-Vektorerzeugung und Prioritätscodierung ohne zusätzlichen Schaltungsaufwand durch Kaskadierung der Bausteine ("Daisy chain priority interrupt logic"): vgl. Darstellung Bild 4
- 8 Ausgänge für den direkten Anschluß von Darlington-Transistoren ausgelegt (Push-Pull).
- Alle Ein- und Ausgänge voll TTL-kompatibel.

Bild 1 zeigt ein Blockschaltbild des PIO.
Der Baustein umfaßt — ein Interface zur CPU-Anschaltung
— interne Steuerlogik
— Logik für I/O-Port A
— Logik für I/O-Port B
— Interrupt-Steuerlogik

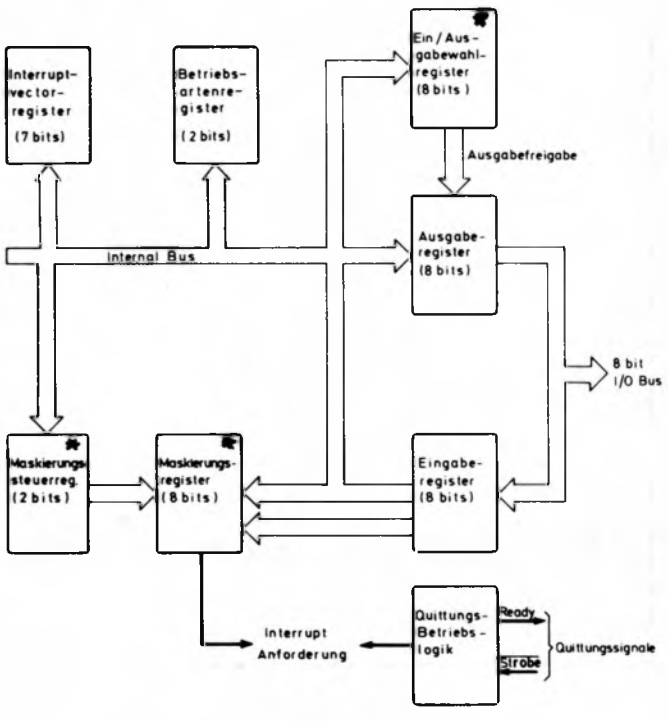
Die I/O-Port-Logik selbst besteht aus 6 Registern mit Quittungsbetrieb-Steuerlogik (siehe Bild 2), und zwar:

- ☐ ein 2 bit-Betriebsarten-Register
- ☐ ein 8 bit-Ausgabe-Register
- ☐ ein 8 bit-Eingabe-Register
- ☐ ein 2 bit-Maskierungs-Steuerungsregister } nur bei Betriebsart Bit-I/O benutzt!
- ☐ ein 8 bit-Maskierungs-Register
- ☐ ein 8 bit-Ein/Ausgabe-Wahl-Register

Beschreibung der einzelnen Register

- 2 bit-Betriebsarten-Register: Wird von der CPU zur Festlegung der Betriebsart (Byte-Ausgabe, Byte-Eingabe, Byte-Ein/Ausgabe, bzw. Bit-Ein/Ausgabe) geladen.
- 8 bit-Ausgabe-Register: Für Datenübertragung von der CPU an periphere Schaltungen
- 8 bit-Eingabe-Register: Für Datenübertragung von peripheren Schaltungen an die CPU
- 2 bit-Maskierungs-Steuerregister: Wird von der CPU zur Definition der aktiven Zustände der Anschlüsse einer peripheren Schaltung (Low oder High) und zur Festlegung der Interruptsignal-Erzeugung (Ausgabe des Interrupt-Signals, wenn **alle** unmaskierten PIO-Anschlüsse aktiv sind (UND-Bedingung), bzw. wenn **irgendeiner** der unmaskierten PIO-Anschlüsse aktiv ist (ODER-Bedingung) geladen
- 8 bit-Maskierungsregister: Wird von der CPU geladen; sein Inhalt legt die Port-Anschlüsse fest, die für die Erzeugung einer Interrupt-Anforderung maßgeblich sind

Bild 2: Blockschaltbild eines Ports



* Wird nur in Betriebsart Bit-Ein/Ausgabe für die Erzeugung einer Interrupt-Anforderung verwendet.

- 8 bit-Ein/Ausgabewahl-Register: Wird bei Betriebsart „Bit-Ein/Ausgabe" von der CPU geladen. Sein Inhalt legt fest, welche Port-Anschlüsse als Eingänge und welche als Ausgänge verwendet werden.

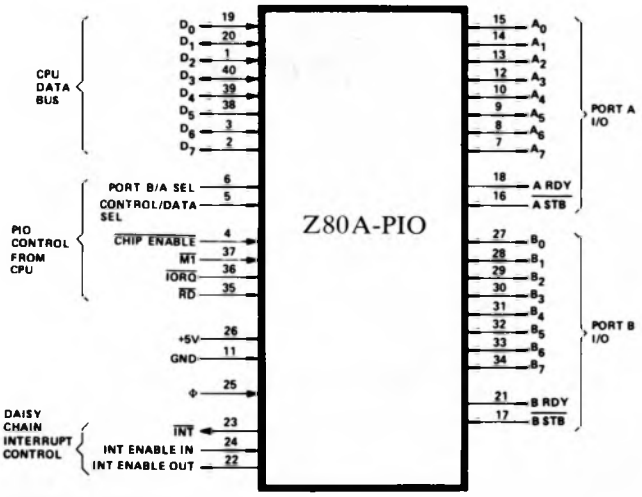


Bild 3: Pinbelegung

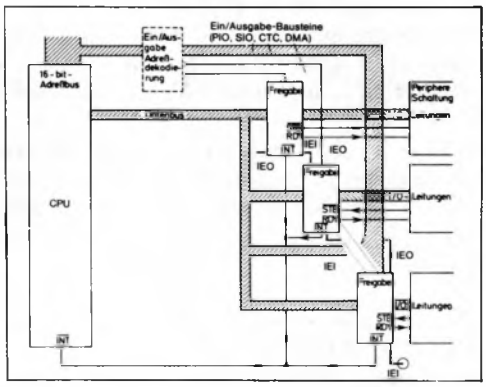


Bild 4: Interrupt-Architektur

□ Z80A-PIO-Pinbelegung

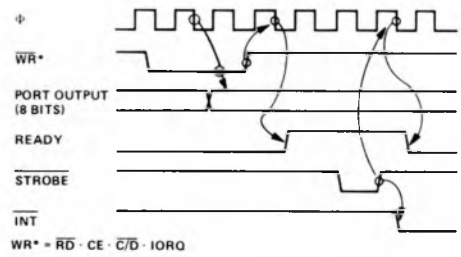
Pin	Funktion	Kommentar
D0...D7	Datenbus	Tri-State Ein/Ausgänge, mit Z80-CPU-Datenbus verbunden
B/A SEL	Portauswahl (= "Port B or A Select")	Eingang, High-aktiv
C/D SEL	Schalteingang Steuerwort/Datenwort (= "Control or Data Select")	Eingang, High-aktiv
$\overline{CE}$	Bausteinfreigabe (= "Chip enable")	Eingang, Low-aktiv
$\overline{\Phi}$	Systemtakt	Eingang, mit Z80-CPU-Systemtakt zu verbinden
$\overline{M1}$	Maschinenzyklus 1 (= "Machine Cycle One Signal")	Eingang, Low-aktiv, mit $\overline{M1}$ der Z80-CPU zu verbinden
$\overline{IORQ}$	Ein/Ausgabe-Anforderung (= "In/Output-Request")	Eingang, Low-aktiv; mit $\overline{IORQ}$ der Z80-CPU zu verbinden
$\overline{RD}$	Lesen (= "Read")	Eingang, Low aktiv, mit $\overline{RD}$ der Z80-CPU zu verbinden
IEI	Interrupt-Freigabe Eingang (= "Interrupt Enable In")	Eingang, High-aktiv
IEO	Interrupt-Freisignal Ausgang (= "Interrupt Enable Out")	Ausgang, High-aktiv
$\overline{INT}$	Interrupt-Anforderung (= "Interrupt Request")	Ausgang, Low-aktiv, Open Drain
A0...A7	A-Ein/Ausgänge	Bidirektional, Tristate, Anschlüsse des A-Ports
$\overline{ASTB}$	A-Trigger (= "Port A Strobe Pulse")	Eingang, Low-aktiv, für Triggerimpuls von der peripheren Elektronik zur Datenübernahme
ARDY	Quittung (= "A-Ready")	Ausgang, High-aktiv Quittierung des ASTB-Empfangs
B0...B7	B-Ein/Ausgänge	Bidirektional, Tristate, Anschlüsse des B-Ports
$\overline{BSTB}$	B-Trigger (= "Port B Strobe Pulse")	Eingang, Low-aktiv, für Triggerimpuls von der peripheren Elektronik zur Datenübernahme
BRDY	Quittung (= "B-Ready")	Ausgang, High-aktiv Quittierung des $\overline{BSTB}$-Empfangs

□ Zeitdiagramme

Betriebsart Ausgabe (= Betriebsart 0)

Bei der Ausführung eines Ausgabebefehls durch die CPU veranlaßt die steigende Flanke des WR-Signals die Übernahme der Daten von der CPU über den Datenbus in das ausgewählte Ausgabe-Register. Der WRITE-Impuls setzt das READY-Bit ("READY-Flag") nach der fallenden Flanke von Φ als Signal dafür, daß nun die Daten im Register abrufbereit sind. Das READY-Bit bleibt aktiv bis zur steigenden Flanke des Taktimpulses ("$\overline{STROBE}$"), die am PIO anliegt, sobald die Daten von der peripheren Schaltung übernommen sind (= „Quittung").

Die steigende Flanke des Taktsignals ($\overline{STROBE}$) erzeugt ein $\overline{INT}$-Signal, falls das Unterbrechungsfreigabe-Flip-Flop ("Interrupt-Enable-Flip-Flop") gesetzt ist und die anfordernde periphere Schaltung die momentan höchste Priorität hat.

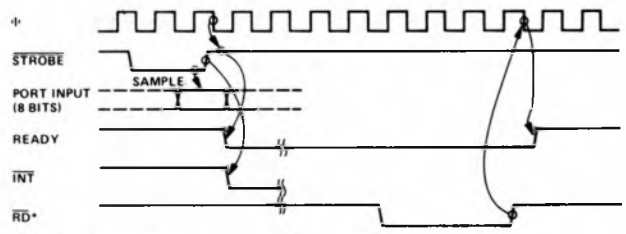


Betriebsart Eingabe (= Betriebsart 1)

Bei der fallenden Flanke von $\overline{\text{STROBE}}$ werden die Daten des ausgewählten Ports in das Eingaberegister gebracht.

Die nächste steigende Flanke von $\overline{\text{STROBE}}$ aktiviert $\overline{\text{INT}}$, falls der Interrupt freigegeben ist und höchste Priorität vorliegt. Die folgende fallende Flanke von Φ bringt READY in den inaktiven Zustand als Zeichen dafür, daß im Eingaberegister Daten anstehen und keine weiteren Daten eingeschrieben werden sollen, bis der Eingabepuffer von der CPU gelesen ist.

Nach dem Lesevorgang bewirkt die steigende Flanke von $\overline{\text{RD}}$ ein Setzen des READY-Signals zum Zeitpunkt der nächsten fallenden Flanke von Φ. Jetzt kann das PIO neue Daten von der angeschlossenen peripheren Schaltung empfangen.



$\overline{RD}^* = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$ MODE 1 (INPUT) TIMING

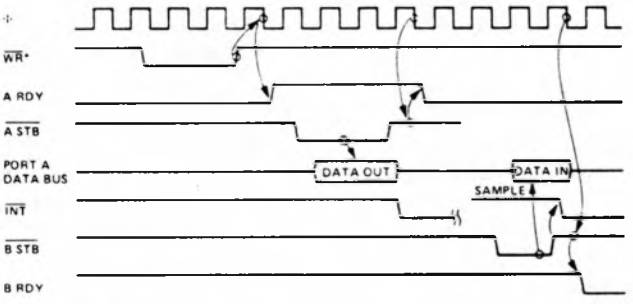
Betriebsart Byte-Ein/Ausgabe (bidirektional) (Betriebsart 2)

ist eine Kombination der Betriebsarten „Byte-Eingabe" und „Byte-Ausgabe", wobei alle 4 Quittungs-(="Handshaking")-Leitungen des Bausteins und die 8 Datenleitungen des Ports A verwendet werden.

Die beiden Quittungsleitungen von Port A werden hier für die Ausgabe-, die von Port B für die Eingabesteuerung benützt.

Eine Datenausgabe an das Port A darf nur während $\overline{\text{ASTB}}$ = Low erfolgen. Seine steigende Flanke kann zur Übergabe der Daten an die periphere Schaltung erfolgen.

Bei Wahl von Betriebsart "Byte-Ein/Ausgabe" für das Port A muß Port B in der Betriebsart „Bit-Ein/Ausgabe" benutzt werden.

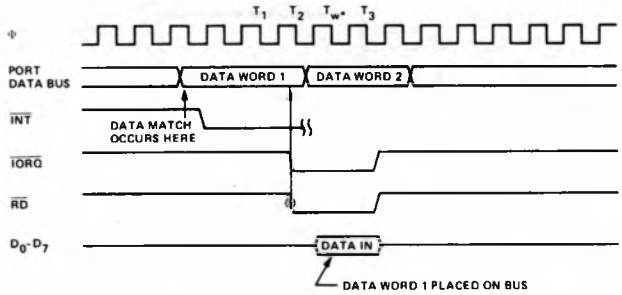


$\overline{WR}^* = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

Betriebsart Bit-Ein/Ausgabe (= Betriebsart 3)

Hier wird nicht mit Quittungssignalen gearbeitet, d.h. eine Eingabe oder eine Ausgabe von Daten kann zu jedem beliebigen Zeitpunkt erfolgen. Beim Ausgeben werden die Daten nach dem gleichen Zeitschema wie bei der Byte-Ausgabe im Ausgabe-Register abgelegt.

Bei der Eingabe sind die der CPU übergebenen Daten zusammengesetzt aus Daten aus dem Eingaberegister (das ist bei den Bits der Fall, die vom Ein/Ausgabewahl-Register durch Einsen als Eingabeleitungen definiert wurden) und aus dem Inhalt des Ausgaberegisters (das ist bei den Bits der Fall, die vom Ein/Ausgabewahlregister als Ausgänge definiert wurden).



* Timing Diagram Refers to Bit Mode Read.

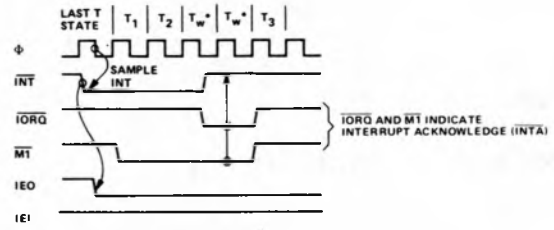
☐ Interrupt-Bearbeitung

Die Steuerlogik einer peripheren Schaltung darf während M1 ihren Interrupt-Freigabezustand nicht ändern, wodurch es möglich wird, das Interruptfreigabe-Signal durch die Prioritätsbestimmende Kaskadierung ("Daisy chain priority interrupt logic") der einzelnen PIO's durchzuschalten.

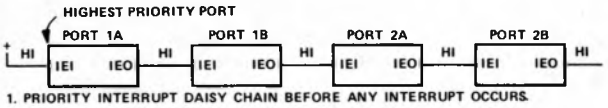
Diejenige periphere Schaltung, die IEI High und IEO Low während INTA liefert, bewirkt, daß zu diesem Zeitpunkt ein vorprogrammierter 8 bit-Interrupt-Vektor auf den Datenbus gelegt wird.

IEO bleibt Low, bis von der CPU eine RETI (= RETurn from Interrupt = Rücksprung aus Interrupt-Anforderungs-Bedienroutine)-Anweisung während IEI = High ausgeführt wird.

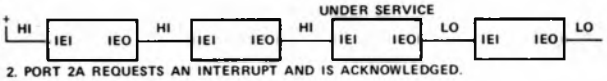
Die 2 Byte-RETI-Anweisung wird hierfür vom PIO intern hardwaremäßig dekodiert.



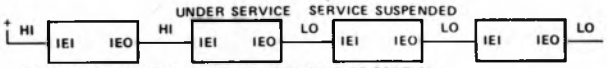
Die Priorisierung erfolgt nach folgendem Schema:



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

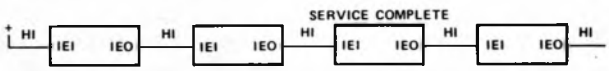


2. PORT 2A REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.



3. PORT 1B INTERRUPTS, SUSPENDS SERVICING OF PORT 2A.

4. PORT 1B SERVICE ROUTINE COMPLETE, "RETI" ISSUED, PORT 2A SERVICE RESUMED.



5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF PORT 2A SERVICE ROUTINE.

Laden des Interrupt-Vektors

Zur Adressierung der richtigen Interrupt-Anforderungs-Bedienroutine muß der CPU vom PIO ein Interrupt-Vektor zugeleitet werden.

Dieser 8 bit-Vektor wird während dem Interrupt-Bestätigungs-Zyklus (="Interrupt Acknowledge Cycle") vom Baustein mit der momentan höchsten Priorität auf den Datenbus gelegt.

Vor Auftreten der ersten Interrupt-Anforderung muß der jeweils gewünschte Vektor von der CPU in den PIO durch Angabe eines Steuerwortes an das betreffende Port mit folgendem Format geschrieben werden:

D7	D6	D5	D4	D3	D2	D1	D0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

identifiziert das Steuerwort als Interrupt-Vektor

Auswahl der gewünschten Betriebsart

Hierfür wird das 2 Bit-Betriebsartauswahl-Register über die höchstwertigen 2 Bits (M_1 und M_0) eines Steuerworts, das von der CPU an das betreffende Port ausgegeben wird, gesetzt.

Format dieses Steuerworts:

D7	D6	D5	D4	D3	D2	D1	D0
M_1	M_0	X	X	1	1	1	1

Betriebsart — nicht benutzt — identifiziert das Steuerwort als Betriebsartenauswahl-Wort

Betriebsart	M_1	M_0
Byte-Ausgabe	0	0
Byte-Eingabe	0	1
Byte-Ein/Ausgabe	1	0
Bit-Ein/Ausgabe	1	1

Sondereinrichtungen bei der Betriebsart Bit-Ein/Ausgabe

In dieser Betriebsart ist nach der Ausgabe des Betriebsart-Auswahl-Steuerwortes von der CPU ein weiteres Steuerwort auszugeben, das die einzelnen Anschlüsse des betreffenden Ports als Eingänge bzw. als Ausgänge definiert.
Eine „1" in diesem Steuerwort bedeutet, daß der zugehörige PIO-Anschluß einen Eingang darstellt, eine 0 bedeutet, daß der Anschluß ab dann als Ausgang benutzt werden kann.

Format:

D7	D6	D5	D4	D3	D2	D1	D0
I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

Interrupt-Steuerung

Bits	Wert	Bedeutung
7	0	Interruptflipflop rückgesetzt, Interrupt-Anforderungen werden nicht angenommen
7	1	Interruptflipflop gesetzt, Interrupt-Anforderungen werden bearbeitet
6, 5, 4	X	Werden nur bei Betriebsart Bit-Ein/Ausgabe benutzt, in allen anderen Betriebsarten ignoriert
3, 2, 1, 0	0111	identifizieren das Steuerwort als Interrupt-Steuerwort

Format

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt Freigabe	UND/ ODER	High/ Low	nächstes Steuerwort ist Maske	0	1	1	1

nur in Betriebsart 3 benutzt — bedeutet Interruptsteuerwort

Falls in diesem Steuerwort das Bit D4 = high war (= „nächstes Steuerwort ist Maske"), muß ein weiteres Steuerwort zur Maskierung an das Port ausgegeben werden.

Sein Format ist:

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Die Maske wirkt so, daß nur Anschlüsse mit Maskierungsbit $MB_n = 0$ zur Erzeugung einer Interrupt-Anforderung herangezogen werden.

Das Interrupt-Freigabe-FlipFlop ist durch folgendes Steuerwort zu beeinflussen:

D7	D6	D5	D4	D3	D2	D1	D0
Int Freigabe	X	X	X	0	0	1	1

Bei „Int Freigabe" = 1 wird die Interrupt-Anforderung einer peripheren Schaltung bearbeitet, bei „Int-Freigabe" = 0 wird eine Interrupt-Anforderung ignoriert.

☐ Dynamische Kenndaten

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	250	[1]	nsec	
	t_W (ΦH)	Clock Pulse Width, Clock High	105	2000	nsec	
	t_W (ΦL)	Clock Pulse Width, Clock Low	105	2000	nsec	
	t_r, t_f	Clock Rise and Fall Times		30	nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
CS, $\overline{CE}$ ETC.	$t_{S\Phi}$ (CS)	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	145		nsec	
D_0-D_7	t_{DR} (D)	Data Output Delay From Falling Edge of $\overline{RD}$		380	nsec	[2]
	$t_{S\Phi}$ (D)	Data Set-Up Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle	50		nsec	C_L = 50 pf
	t_{DI} (D)	Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle		250	nsec	[3]
	t_F (D)	Delay to Floating Bus (Output Buffer Disable Time)		110	nsec	
IEI	t_S (IEI)	IEI Set-Up Time to Falling edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	t_{DH} (IO)	IEO Delay Time from Rising Edge of IEI		160	nsec	[5]
	t_{DL} (IO)	IEO Delay Time from Falling Edge of IEI		130	nsec	[5] C_L = 50 pf
	t_{DM} (IO)	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		190	nsec	[5]
$\overline{IORQ}$	$t_{S\Phi}$ (IR)	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle.	115		nsec	
$\overline{M1}$	$t_{S\Phi}$ (M1)	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle See Note B	90		nsec	
$\overline{RD}$	$t_{S\Phi}$ (RD)	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	115		nsec	
A_0-A_7, B_0-B_7	t_S (PD)	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (Mode 1)	230		nsec	
	t_{DS} (PD)	Port Data Ourput Delay from Falling Edge of $\overline{STROBE}$ (Mode 2)		210	nsec	[5]
	t_F (PD)	Delay to Floating Port Data Bus from Rising Edge of $\overline{STROBE}$ (Mode 2)		180	nsec	C_L = 50 pf
	t_{DI} (PD)	Port Data Stable from Rising Edge of $\overline{IORQ}$ During WR Cycle (Mode 0)		180	nsec	[5]
$\overline{ASTB}$, $\overline{BSTB}$	t_W (ST)	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	t_D (IT)	$\overline{INT}$ Delay time from Rising Edge of $\overline{STROBE}$		440	nsec	
	t_D (IT3)	$\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		380	nsec	
ARDY, BRDY	t_{DH} (RY)	Ready Response Time from Rising Edge of $\overline{IORQ}$		t_c+ 410	nsec	[5] C_L = 50 pf
	t_{DL} (RY)	Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 360	nsec	[5]

A. 2.5 t_c > (N-2) t_{DL} (IO) + t_{DM} (IO) + t_S (IEI) + TTL Buffer Delay, if any

B. $\overline{M1}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] t_c = t_W (ΦH) + t_W (ΦL) + t_r + t_f

[2] Increase t_{DR} (D) by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[3] Increase t_{DI} (D) by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[4] For Mode 2: t_W (ST) > t_S (PD)

[5] Increase these values by 2 nsec for each 10 pf increase in loading up to 100 pf max

☐ Kapazitäten

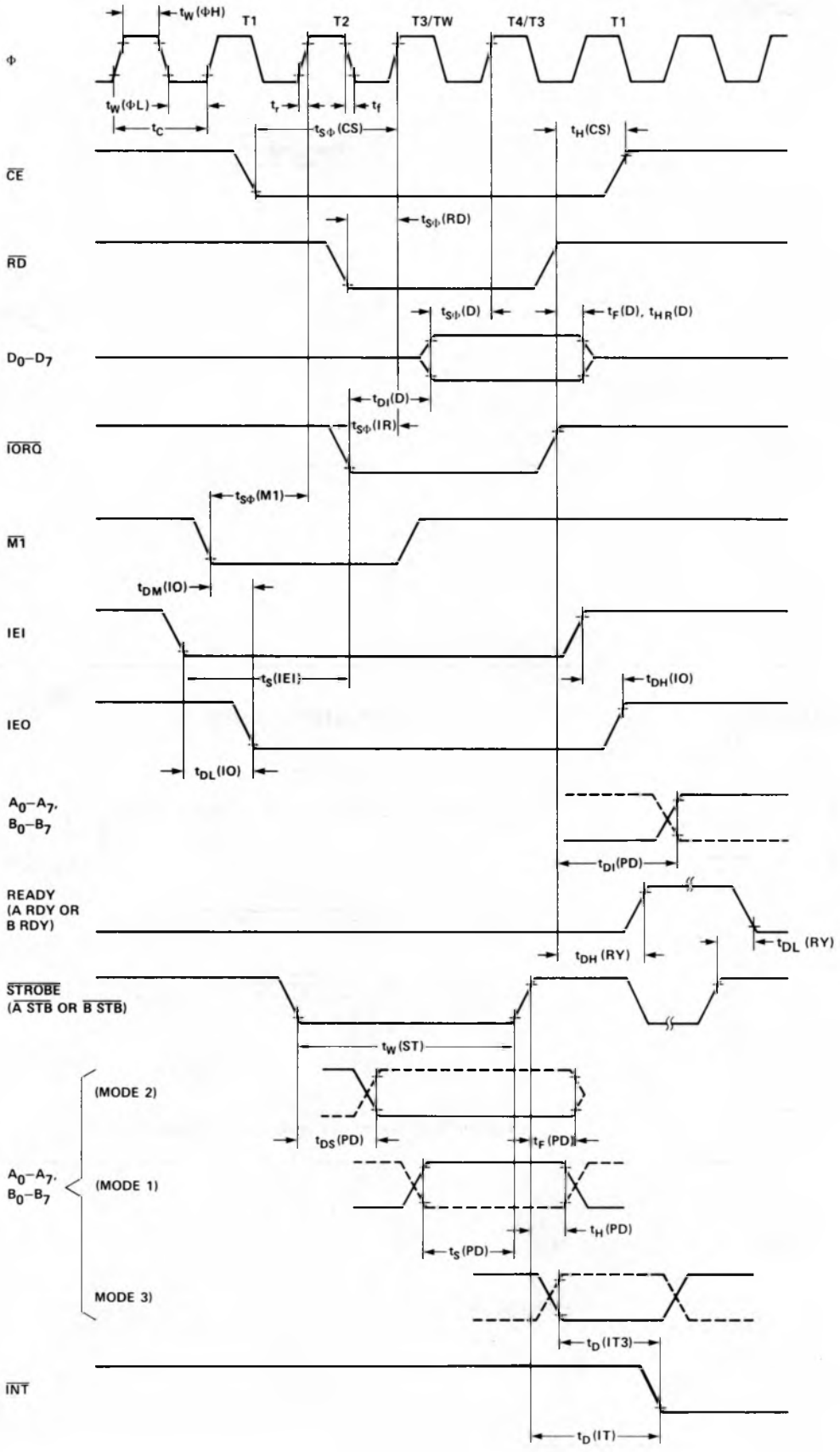
TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	.20	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

Zeitverhalten des Z80A-PIO

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV =	+0.5V



☐ Grenzdaten

Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.6 W

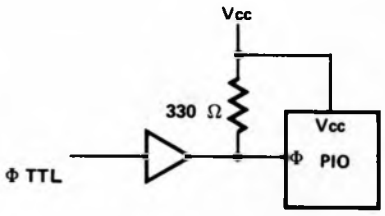
***Comment**
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

☐ Statische Kenndaten

TA = 0° C to 70° C, Vcc = 5 V ± 5% unless otherwise specified

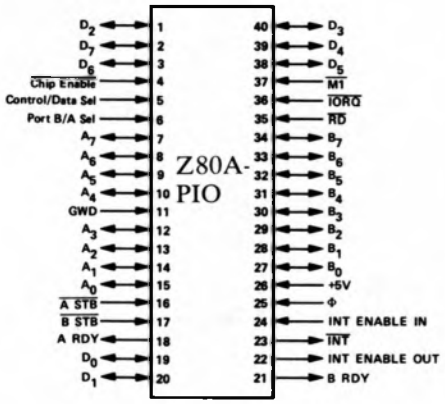
Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6	Vcc+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	Vcc	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} - 250 μA
I_{CC}	Power Supply Current		70	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to Vcc
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT} = 2.4 to Vcc
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	μA	$0 \leq V_{IN} \leq Vcc$
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5 V Port B Only

[1] Clock Driver

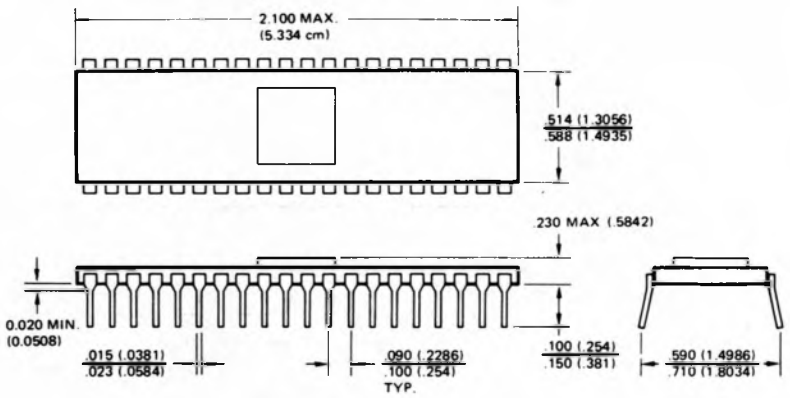


The external pull-up resistor (330 Ω) can satisfy the clock AC and DC requirements.

☐ Pin-Belegung



☐ Gehäuseabmessungen



*Dimensions for metric system are in parenthesis

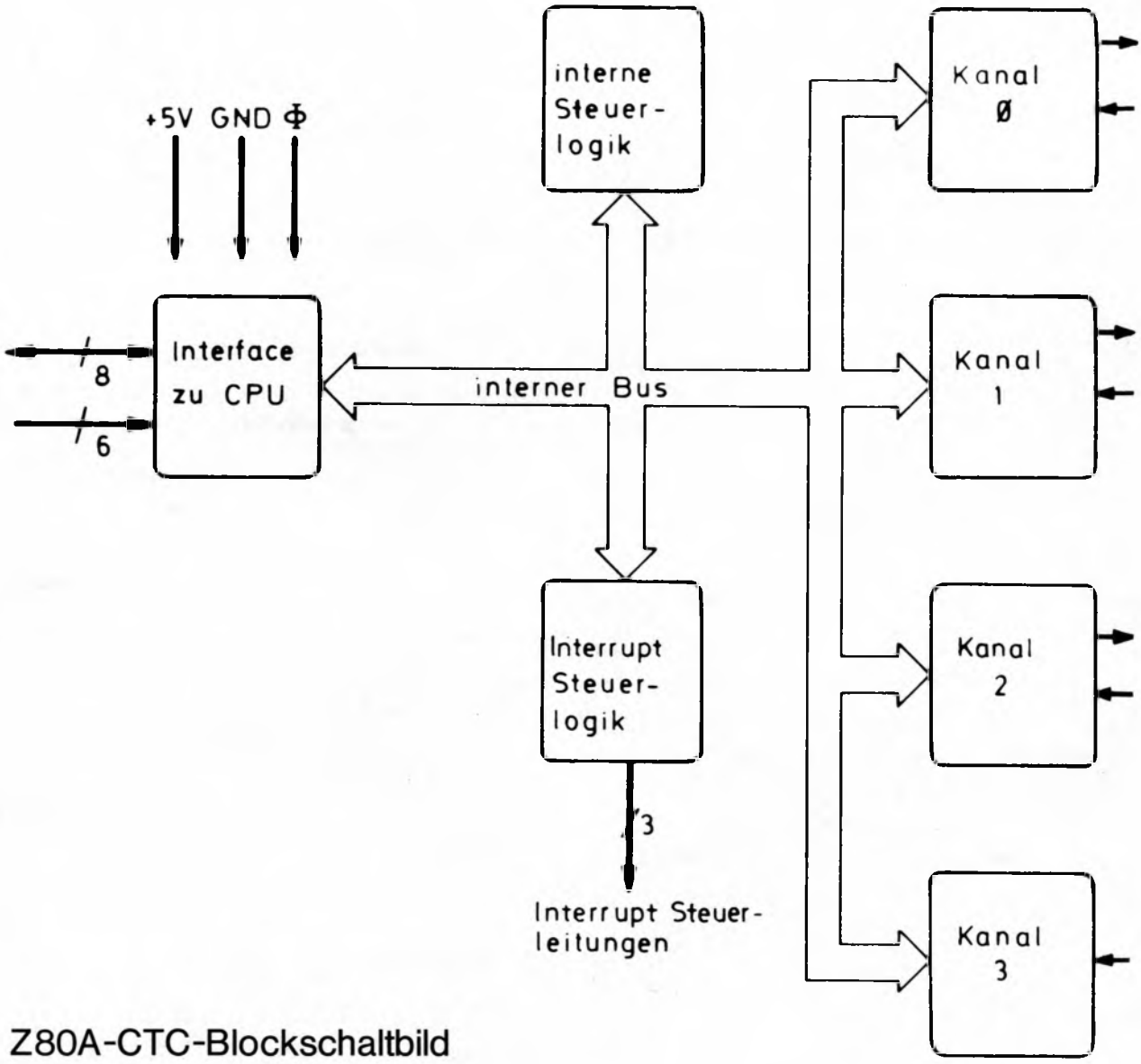
☐ Bestellbezeichnung

Z80A-PIO/PS Standardversion (0 . . . 70℃) in Plastikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-PIO/CS Standardversion (0 . . . 70℃) im Keramikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-CTC ZÄHLER/ZEITGEBER-BAUSTEIN

Zilog



Z80A-CTC-Blockschaltbild

1. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Der Baustein Z80A-CTC ist ein Zähler/Zeitgeber-Baustein (="Counter Timer Circuit") für dieses System, der über 4 voneinander unabhängige Software — programmierbare Zähler/Zeitgeber verfügt. Er macht jegliche zusätzliche Interrupt-Steuerungs- und Priorisierungslogik überflüssig und erlaubt verschachtelten, priorisierten Interrupt beliebig vieler Ebenen im gesamten Speicherbereich.

☐ Technische Einzelheiten

- N-Kanal-Silicon Gate Depletion Load-Technologie
- 28 Pin DIP-Gehäuse
- Stromversorgung über eine einzige +5 V-Versorgungsspannung
- Einphasen -5 V-Takt
- 4 voneinander unabhängige, software-programmierbare 8 bit Zähler/16 bit Zeitgeber-Kanäle
- jeder dieser Kanäle wahlweise als Zähler oder Zeitgeber verwendbar
- programmierbare Interrupts bei Erreichen von programmmäßig festlegbaren Zähler- oder Zeitgeber-Werten.
- Rückwärtszähler hält Anzahl der bis Null auszuführenden Zählschritte auslesebereit
- Vorteiler durch 16 oder 256 für jeden Zeitgeber-Kanal
- Zeitgeber kann wahlweise durch einen positiven oder negativen Triggerimpuls gestartet werden.
- Die Ausleseausgänge von drei Kanälen sind zum direkten Anschluß von Darlington-Transistoren ausgelegt (Push-Pull).
- Automatische Interrupt-Vektorerzeugung und Prioritätscodierung ohne zusätzlichen Schaltungsaufwand durch Kaskadierung der Bausteine (="Daisy chain priority interrupt logic").
- Alle Ein- und Ausgänge voll TTL-kompatibel
- Maximale Zählfrequenz in Betriebsart „Zähler" =f Φ/2

☐ Architektur des Z80A-CTC

Ein Blockdiagramm Z80A-CTC ist im Bild 1 wiedergegeben: Die Funktionseinheiten des Bausteins sind

— 4 Zähler/Zeitgeber-Kanäle
— Interface zu Daten- und Steuerbus der Z80-CPU
— interne Steuerlogik und
— Interrupt-Steuerlogik

Jedem Zähler/Zeitgeber-Kanal ist ein Interruptvektor zugeordnet, wobei der Interrupt-Prioritätsrang durch die Kanalnummer mit Nr. 0 als höchste Priorität bestimmt ist.

Jeder der Kanäle besteht aus 2 Registern, (ein 8 bit Zeitkonstantenregister und ein 8 bit Kanal-Steuerregister), 2 Zählern (ein 8 bit Rückwärtszähler und ein auf den Wert „16" oder „256" einstellbaren 8 bit Vorteiler) und wiederum einer eigenen Steuerlogik (Bild 2).

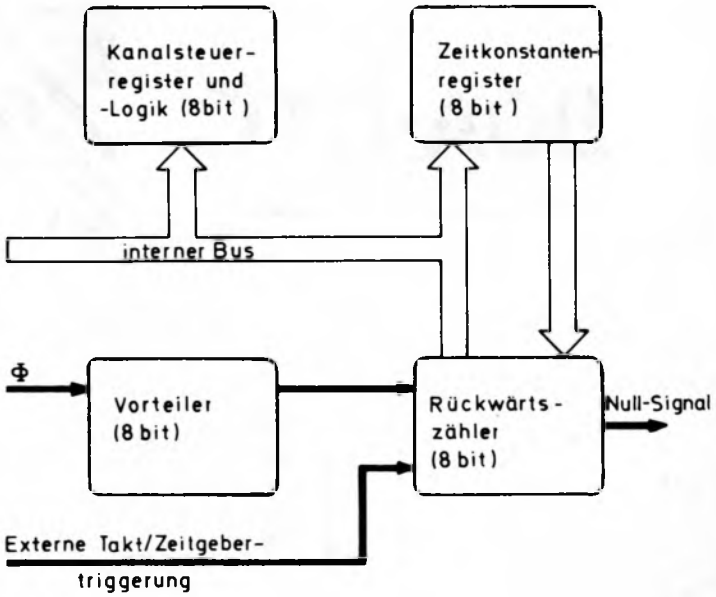
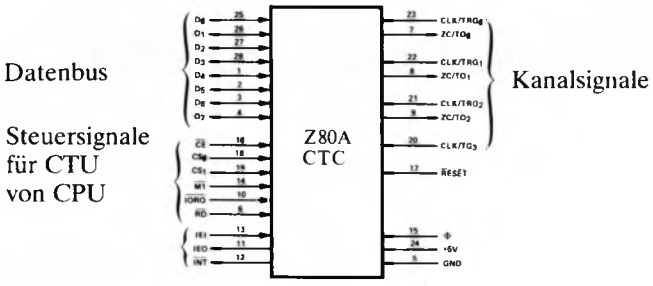


Bild 2: Architektur eines Kanals

Die Funktionen der einzelnen Einheiten sind folgende:

— **Das Zeitkonstantenregister** (8 bit) wird vom Mikroprozessor zum Initialisieren und Wiedersetzen des Rückwärtszählers beim Erreichen des Zählerstandes Null geladen.
— **Das Kanalsteuerregister** (8 bit) wird vom Mikroprozessor zur Bestimmung der Kanalbetriebsart geladen.
— **Der Rückwärtszähler** (8 bit) wird entweder auf Veranlassung des Anwenderprogramms oder automatisch beim Erreichen des Zählerstandes Null auf den Wert des Zeitkonstantenregisters gesetzt. Der Zähler wird in der Kanal-Betriebsart „Zeitgeber" über den Vorteiler, in der Kanal-Betriebsart „Zähler" durch ein Signal am Eingang CLK/TRIG dekrementiert. Der momentane Wert des Rückwärtszählers kann zu jedem beliebigen Zeitpunkt vom Mikroprozessor aus gelesen werden.
— **Der Vorteiler** (8 bit) wird nur in der Betriebsart „Zeitgeber" benützt; er teilt hier den Systemtakt-Puls um den softwaremäßig festlegbaren Wert 16 oder 256.

☐ Pinbeschreibung



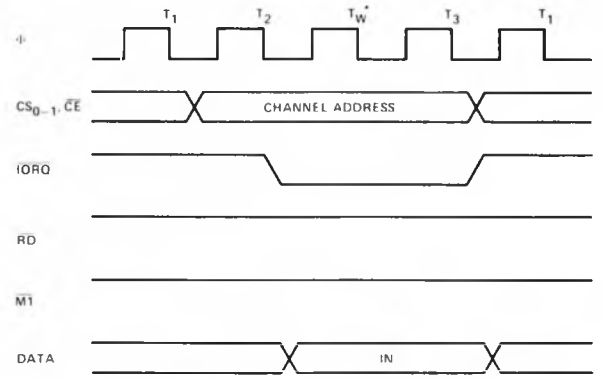
Pin	Funktion	Kommentar
CLK/ TRG 0	Takt/Trigger 0 (="Clock/ Trigger 0")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 0
CLK/ TRG 1	Takt/Trigger 1 (="Clock/ Trigger 1")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 1

Pin	Funktion	Kommentar
CLK/ TRG 2	Takt/Trigger 2 (="Clock/ Trigger 2")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 2
CLK/ TRG 3	Takt/Trigger 3 (="Clock/ Trigger 3")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 3
ZC/TO 0	Nulldurchgang/ Zeitgebermeldung 0 (="Zero Count/ Timeout 0")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 0
ZC/TO 1	Nulldurchgang/ Zeitgebermeldung 1 (="Zero Count/ Timeout 1")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 1
ZC/TO 2	Nulldurchgang/ Zeitgebermeldung 2 (="Zero Count/ Timeout 2")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 2
CS 1 und CS 0	Kanalauswahl (="Channel Select")	(Eingang, high-aktiv): Eingabe der 2 Bit-Adresse des des vom Mikroprozessor angesprochenen Kanals
D 7 bis D 0	Datenbus (="Data Bus")	(Bidirektional, Tristate) 8 bit-Datenbus
$\overline{CE}$	Bausteinauswahl (="Chip Enable")	Eingang, Low-aktiv
Φ	Systemtakt (="System Clock")	Eingang
$\overline{M1}$	Maschinenzyklus 1 (="Machine Cycle One")	Eingang, Low aktiv Hier wird das Signal $\overline{M1}$ des Mikroprozessors Z80-CPU angelegt
$\overline{IORQ}$	Ein/Ausgabe-Anforderung (="Input/Output Request")	Eingang, Low-aktiv Hier wird das Signal $\overline{IORQ}$ des Mikroprozessors Z80-CPU angelegt
$\overline{RD}$	Lesen (="Read")	Eingang, Low-aktiv Hier wird das Signal $\overline{RD}$ des Mikroprozessors Z80-CPU angelegt
IEI	Interrupt-Freigabe-Eingang (="Interrupt Enable Input")	Eingang, High-aktiv bildet zusammen mit IEO des vorigen Bausteins die Interrupt-Prioritätskette
IEO	Interrupt-Freigabe-Ausgang (="Interrupt Enable Output")	Ausgang, High-aktiv für Interrupt-Prioritätskette
$\overline{INT}$	Interrupt-Anforderung	Ausgang, Open-Drain, Low-aktiv. Meldet Interruptanforderung an den Mikroprozessor.
$\overline{RESET}$	Rückstelleingang (="Reset")	Eingang, Low-aktiv $\overline{RESET}$ unterbricht den Zählvorgang aller Kanäle und setzt die Interrupt-Freigabebits der Steuerregister aller 4 Kanäle zurück, bringt ZC/TO 0 bis 2 und $\overline{INT}$ in den inaktiven Zustand, setzt Ausgang IEO gleich Wert am Eingang IEI und bringt alle Ausgänge in den hochohmigen Zustand.

☐ CTC-Schreibzyklus

Hier werden Kanalsteuerwort, Zeitkonstante und Interrupt-Vektor eingeschrieben.

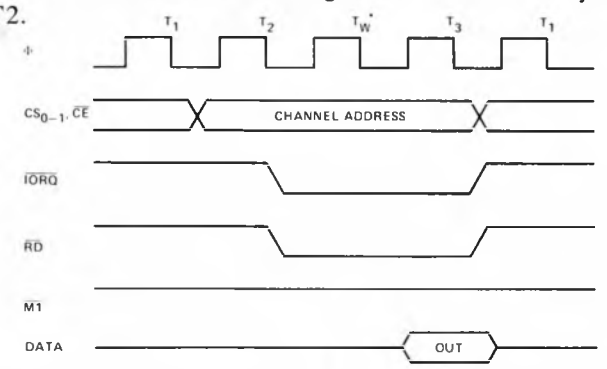
Zur Beachtung: Über den automatisch erzeugten Wartezyklus Tw* dürfen keine Wartezyklen beim Schreiben in die CTC-



Register eingefügt werden. Da der CTC kein spezifisches Schreibsignal geliefert bekommt, erzeugt er es sich intern aus dem RD-Signal.

☐ CTC-Lesezyklus

In der Betriebsart „Zähler" kann der Momentanwert jedes Kanalrückwärtszählers ausgelesen werden. Der auf dem Datenbus ausgelesene Wert repräsentiert die Anzahl der steigenden Taktflanken **vor** der steigenden Taktflanke des Zyklus T2.

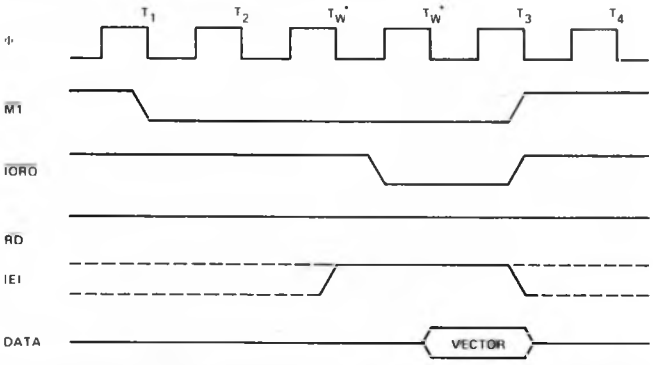


Außer dem automatisch erzeugten Wartezyklus Tw* darf auch beim CTC-Lesen kein Wartezyklus eingefügt werden.

☐ Interrupt-Quittungs-Zyklus

Der Mikroprozessor quittiert nach einer gewissen Zeit den Empfang einer Interrupt-Anforderung vom CTC durch die Signale $\overline{M1}$ und $\overline{IORQ}$.

In der Zwischenzeit ermittelt der CTC intern den anfordernden Kanal mit der höchsten Priorität.



Zum sicheren Einschwingen der Interrupt-Kaskadensignale werden die Unterbrechungsanforderungszustände aller Kanäle „eingefroren", solange $\overline{M1}$ aktiv ist.

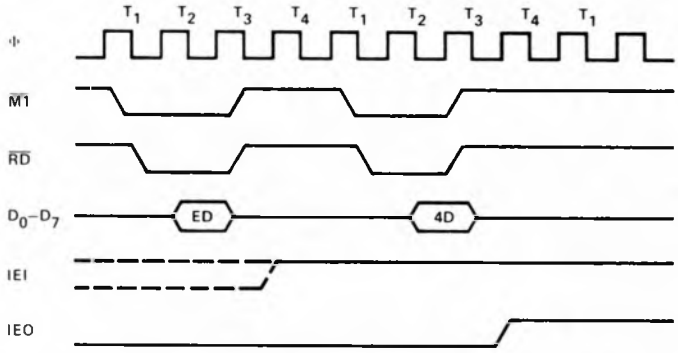
Ist dann der IEI-Eingang des CTC aktiv, bringt der anfordernde Kanal höchster Priorität den Inhalt seines Interrupt-Vektorregisters auf den Datenbus, sobald $\overline{IORQ}$ aktiv wird.

Das Einfügen zusätzlicher WAIT-Zyklen ist hier zulässig.

□ Rückkehr von Interrupt

Wenn an einem Z80A bzw. Z80-Mikrocomputerbaustein kein Interruptanforderungs-Signal ansteht und gerade keine Interrupt-Service-Routine ausgeführt wird, ist sein Signal IEO = IEI.

Befindet sich eine Interrupt-Service-Routine in Bearbeitung (d.h. der Baustein hat eine Interrupt-Anforderung ausgestellt und eine Interrupt-Bestätigung (= „Interrupt Acknowledge") von der CPU empfangen), ist sein IEO-Ausgang in jedem Fall auf LOW, wodurch niederpriorisierte Bausteine an der Ausstellung einer Interruptanforderung gehindert werden.



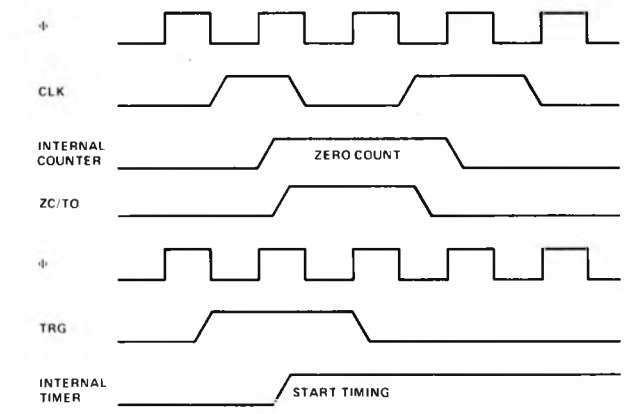
Steht an dem Baustein ein Interruptanforderungs-Signal der ihm zugeordneten Peripherie an, das noch nicht von der CPU quittiert wurde, liegt an IEO LOW-Signal, bis der Hexadezimalwert ED als das erste Byte eines 2 Byte-Opcodes auftritt. Daraufhin wird IEO HIGH bis zum Auftreten des nächsten Opcode-Bytes, wo IEO wieder auf LOW geht. War das zweite Byte dieses Opcodes eine Hexadezimalzahl 4 D, so hat es sich bei dieser Anweisung um einen RETI-Befehl gehandelt.

Nach dem Empfang des Opcodes ED weist ausschließlich **das** Port an IEI High-Signal und an IEO LOW-Signal auf, das die letzte Interruptanforderung an die CPU geschickt hat und nun mit der Interrupt-Behandlung beschäftigt ist; es muß zwangsläufigerweise von den Ports, die zu diesem Zeitpunkt von der CPU eine Interruptquittung bekommen haben, das momentan höchstpriorisierte sein, und bei allen übrigen Ports ist nun IEI = IEO. Beim nächsten OP-Code 4 D verläßt das zuletzt aktive Port seinen Interrupt-Bedienzustand.

In den M1-Zyklen ist an dieser Stelle das Einfügen von WAIT-Zyklen zulässig.

□ Zähl- und Zeitgeber-Vorgang

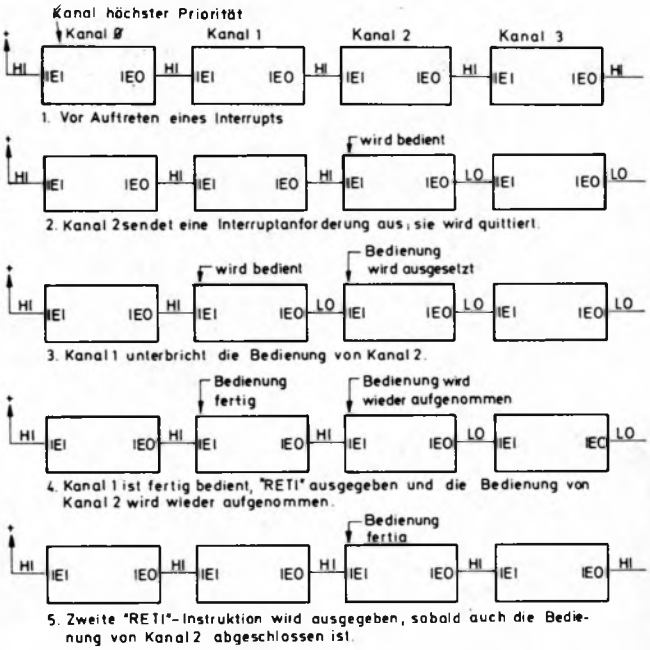
In der Betriebsart „Zähler" veranlaßt die steigende Flanke des CLK-Eingang jeweils ein Dekrementieren des Rückwärtszählers. Dieses Signal wird völlig asynchron empfangen, muß jedoch eine gewisse Mindestdauer haben.



Der Zähler selbst arbeitet jedoch synchron mit Φ, wobei eine gewisse Schaltzeit zur Verfügung stehen muß, wenn der Zähler bereits bei der folgenden steigenden Flanke von Φ nach Aktivierung des CLK-Eingangs dekrementiert werden soll. In der Betriebsart „Zeitgeber" kann der Vorteiler von einer steigenden oder einer fallenden Flanke aus dem TRG-Eingang angesprochen werden.

Wie in der Betriebsart „Zähler" wird diese Flanke völlig asynchron empfangen und muß eine Mindestdauer aufweisen. Entsprechend muß auch wieder für eine bestimmte Schaltzeit bis zur nächsten steigenden -Flanke gesorgt sein, wenn der Zeitgeberstart zu diesem Zeitpunkt erfolgen soll.

Der Vorteiler zählt steigende Flanken von Φ.



□ Vorgang der Prioritäts-Kaskadierung

In der Skizze ist eine typische Interruptabfolge als Beispiel dargestellt.

Darin fordert Kanal 2 einen Interrupt an und wird bedient. Währenddessen fordert der Kanal 1 ebenfalls einen Interrupt an und wird wegen seiner höheren Priorität bedient, wobei die Bedienung von Kanal 2 unterbrochen wird. Sobald nun die Bedienung von Kanal 1 abgeschlossen und die RETI-Anweisung ausgeführt ist, wird die niedriger priorisierte Behandlung von Kanal 2 fortgesetzt und zuende geführt.

□ Laden des Interrupt-Vektors

Dem Mikroprozessor muß vom unterbrechenden Kanal ein Interrupt-Vektor geliefert werden, aufgrund dessen der Mikroprozessor die Startadresse der zugehörigen Interrupt-Bedienroutine ermittelt.

Während des Interrupt-Quittungs-Zyklus (= "Interrupt Acknowledge Cycle") wird dieser Vektor vom anfordernden Kanal höchster Priorität auf den Datenbus gelegt; vorher muß der Vektor dem CTC über Kanal Nr. 0 und Datenbit D 0 gleich Null vorgegeben werden.

D 7 ... D 3 enthalten dabei den Vektor; D 2 und D 1 sind beim Vektorladen unbenützt. Sobald der CTC auf eine Interrupt-Quittung reagiert, enthalten D 2 und D 1 den Binärcode der Nummer des höchstpriorisierten anfordernden Kanals, und D 0 steht auf Null, da die Interrupt-Bedienroutine immer auf einer geraden Adresse beginnt.

Kanal 0 ist hardwaremäßig der höchstpriorisierte Kanal.

Format des zugehörigen Steuerworts:

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
V 7	V 6	V 5	V 4	V 3	x	x	0

x = beliebiger Binärwert

☐ Laden des Zeitkonstanten-Registers

Nach einem Steuerwort mit Bit 2 = 1 wird das nächstfolgende Steuerwort für den betreffenden Kanal als Zeitkonstante interpretiert und ins Zeitkonstantenregister des Kanals geladen.
Eine Steuerwortkonstante = 0 wird als Zeitkonstante = 256 interpretiert.

Format:

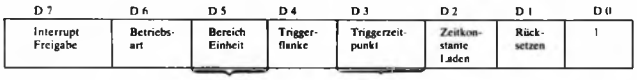
D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
ZK 7	ZK 6	ZK 5	ZK 4	ZK 3	ZK 2	ZK 1	ZK 0

☐ Programmierung des CTC

Festlegen der Betriebsart

Hat bit 0 den Wert 1, so wird das aktuelle Datenwort in das Kanalsteuerregister übernommen.

Format:



Nur für Betriebsart „Zeitgeber"

Bemerkung:
Werte von D 0 ... D 6 sind beim Einschalten nicht definiert

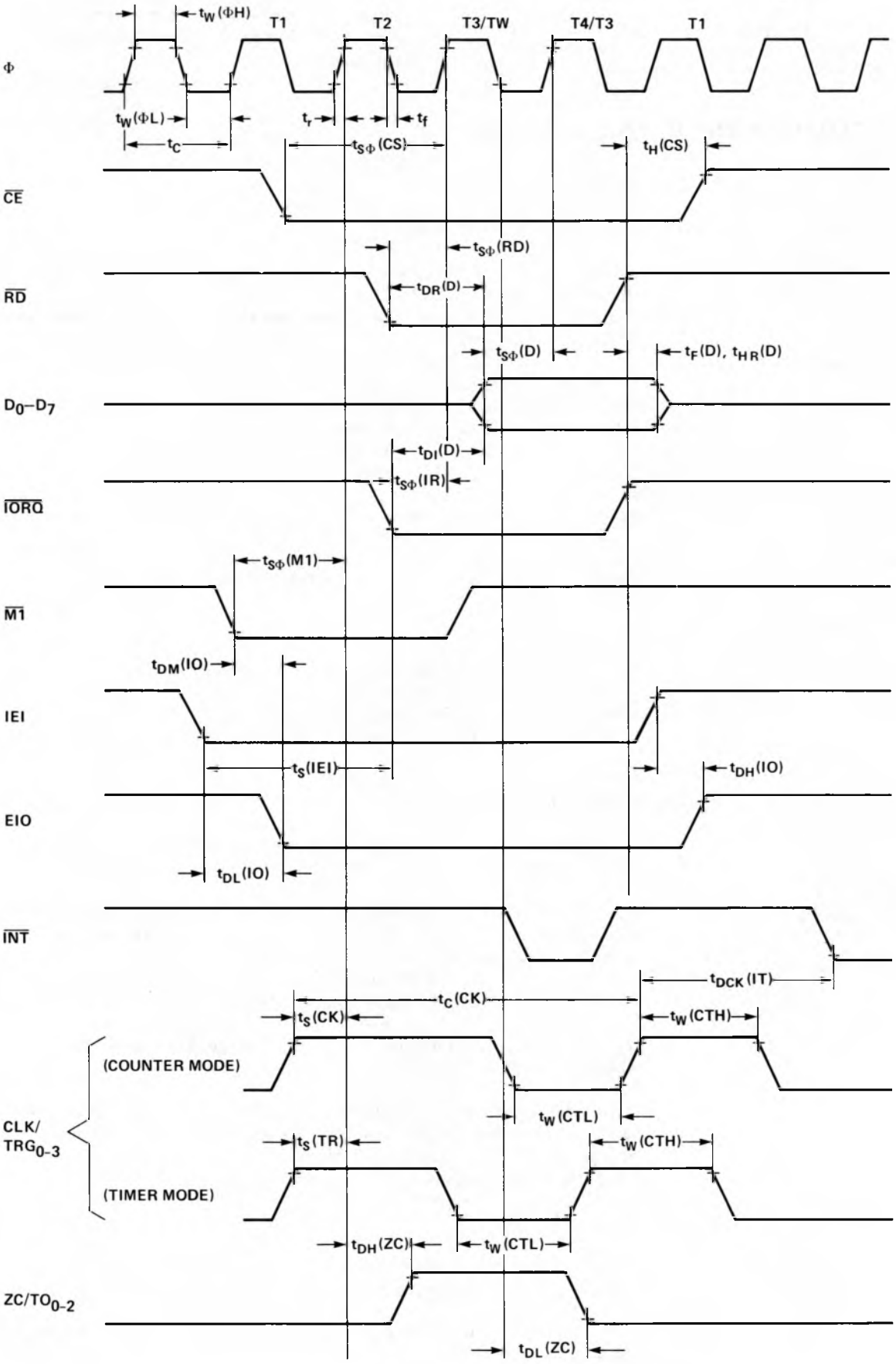
☐ Erläuterungen zur Programmierung

Bit.-Nr.	Wert	Erklärung
7	0	Interrupt des Kanals gesperrt
	1	Interrupt des Kanals freigegeben; Interruptanforderung erfolgt jedesmal, wenn der Rückwärtszähler den Wert Null erreicht. Durch ein Setzen des Bit 7 wird **kein** Interrupt durch ein **vorangegangenes** Zähler = Null-Ergebnis veranlaßt.
6	0	Betriebsart „Zeitgeber". Der Rückwärtszähler wird vom Verteiler gestartet. Die Periode des Zählers ist dabei $c = t_c \cdot P \cdot TC$ mit t_c = Systemtakt-Periode P = Vorteiler (= 16 oder = 256) TC = 8 bit programmierbare Zeitkonstante (max. = 256)
	1	Betriebsart „Zähler". Der Rückwärtszähler wird von externen Signalen getriggert, der Vorteiler bleibt unbenützt.
5 (nur bei Betriebsart „Zeitgeber")	0	Systemtakt wird vom Vorteiler um den Faktor 16 heruntergeteilt.
	1	Systemtakt wird vom Vorteiler um den Faktor 256 heruntergeteilt.
4	0	Betriebsart Zeitgeber: Zeitmessung wird von negativer Flanke gestartet Betriebsart Zähler: Die negative Flanke dekrementiert den Rückwärtszähler
	1	Betriebsart Zeitgeber: Zeitmessung wird von positiver Flanke gestartet Betriebsart Zähler: Die positive Flanke dekrementiert den Rückwärtszähler.
3 (nur bei Betriebsart „Zeitgeber")	0	Die Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, der auf das Laden der Zeitkonstante folgt, und zwar mit dessen steigender Flanke von T_2
	1	Der externe Triggereingang wird zur Veranlassung des Beginns des Zeitgebervorgangs freigegeben, und zwar nach der steigenden Flanke von T_2 **des** Maschinenzyklus', der auf das Laden der Zeitkonstante folgt. Der Vorteiler wird, soweit die nötige Vorbereitungszeit zur Verfügung stand, nach 2 weiteren Taktzyklen dekrementiert, andernfalls nach 3 weiteren Taktzyklen.
2	0	Auf das Kanal-Steuerwort folgt keine Zeitkonstante. Die Zeitkonstante ist zum Anlaufen lassen des Zeitmeßvorgangs noch einzugeben.
	1	Das nächste Kontrollwort für den betreffenden Kanal stellt die Zeitkonstante für den Rückwärtszähler dar. Wird eine Zeitkonstante während eines Zeitmeßvorgangs eingegeben, wird zuerst die laufende Messung zuende geführt, bevor die neue Zeitkonstante in den Rückwärtszähler geschrieben wird.
1	0	Kanal zählt weiter
	1	Abbruch der momentanen Operation. Falls Bit 2 = 1 ist, wird die Operation nach dem Laden einer neuen Zeitkonstante fortgesetzt. Ist Bit 2 = 0, muß hierfür ein neues Steuerwort an den CTC übermittelt werden.

□ Zeitverhalten des Z80A-CTC

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{CC} - .6V	.45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	±0.5V



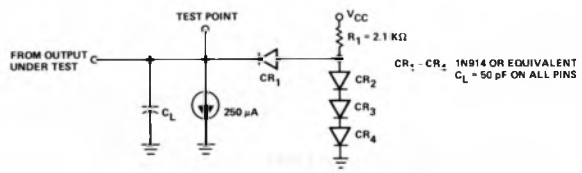
☐ Dynamische Kenndaten

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	250	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	60		ns	
D_0–D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ During Read Cycle		380	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORG During INTA Cycle		160	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		190	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	115		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	90		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	115		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 140$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 140$		Timer Mode
CLK/TRG_0–$_3$	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		50		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for enabling of Prescaler on Following Rising Edge of Φ	210			Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200			Counter and Timer Modes
ZC/TO_0–$_2$	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190		Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		190		Counter and Timer Modes

Notes: [1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.
[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.
[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum.
[4] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

OUTPUT LOAD CIRCUIT



☐ Absolute Grenzdaten

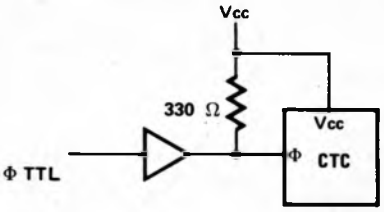
Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.8W

***Comment**
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

☐ Statische Kenndaten

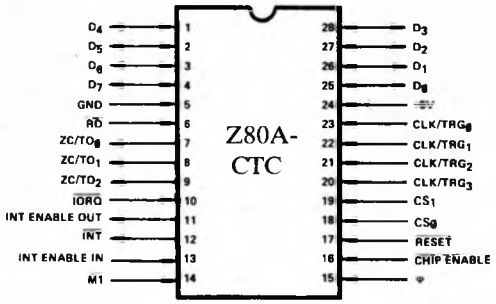
Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage [1]	V_{CC} - .6	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
I_{CC}	Power Supply Current		120	mA	T_C = 250 nsec
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5V R_{EXT} = 390Ω

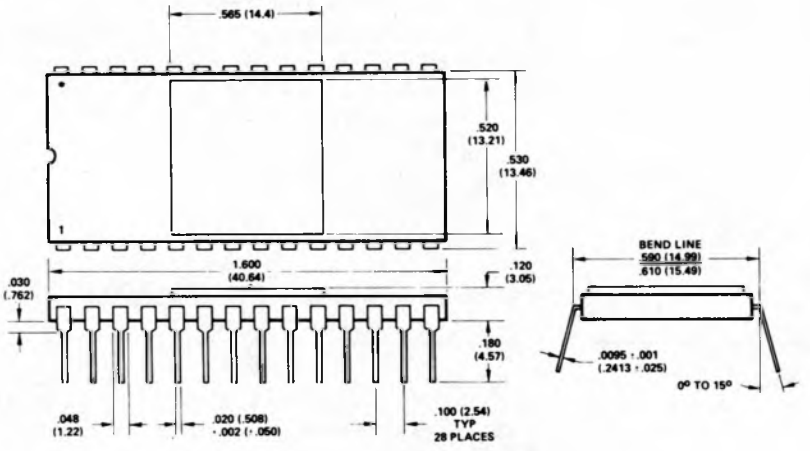
[1] Clock Driver



The external pull-up resistor (330 Ω) can satisfy the clock AC and DC requirements.

☐ Pin-Belegung





* DIMENSIONS FOR METRIC SYSTEM IN PARENTHESES (mm)

☐ Bestellbezeichnung

Z80A-CTC/PS Standardversion (0 . . . 70°C) in Plastikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-CTC/CS Standardversion (0 . . . 70°C) im Keramikgehäuse $U_B = 5\ V \pm 5\%$

Z80A-DMA

Zilog

BAUSTEIN FÜR DIREKTEN SPEICHERZUGRIFF

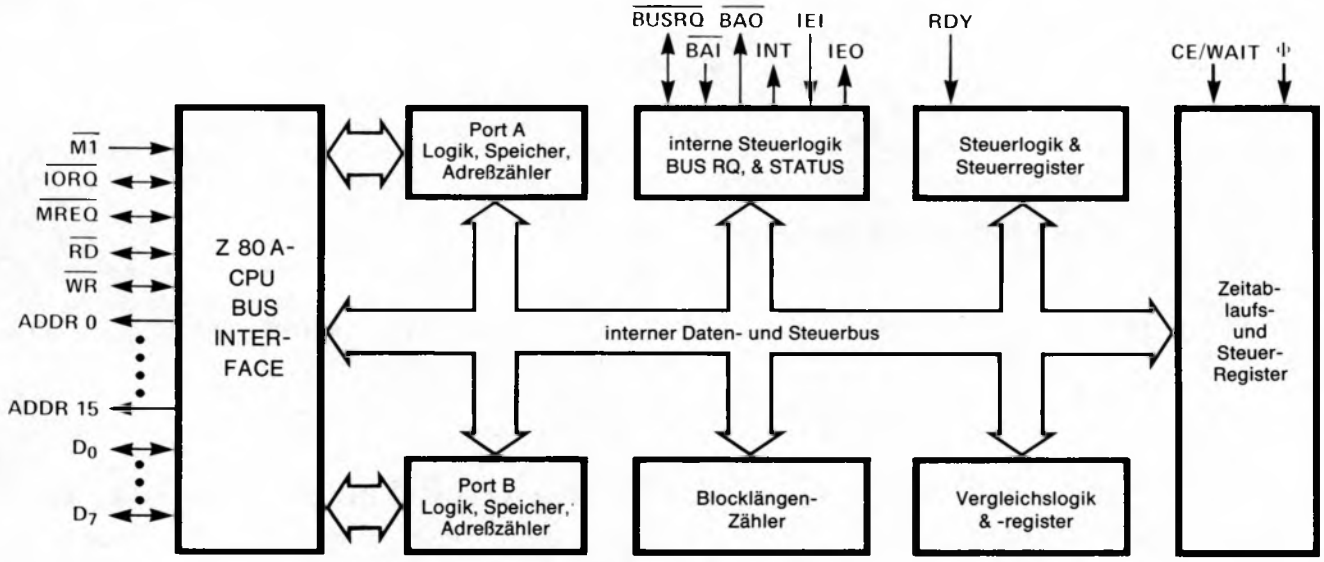


Bild 1: Blockschaltbild des Z80A DMA

1. ZILOG-Z80A MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Das Computersystem Z80A umfaßt ein komplettes Angebot an MC-Halbleiterbausteinen, fertig aufgebauten MC-Platinen, betriebsbereiten OEM-Computersystemen, verschiedenen Typen höchstleistungsfähiger Mikrocomputer-Entwicklungssystemen mit der zugehörigen Betriebssoftware (auch höhere Computer-Sprachen!) und Standard-Anwendersoftware.

Dabei ist das Halbleiterbausteinangebot so geartet, daß auch komplexere MC-Systeme mit minimaler MC-Bausteinanzahl zu realisieren sind. Als Speicherbausteine sind sämtliche Standardchips verwendbar, zusätzliche Logik ist im ganzen System praktisch nicht nötig.

Der Befehlssatz der Z80A-CPU verringert durch seine hohe Effizienz Programmentwicklungszeit und Speicherbedarf; außerdem wird eine besonders hohe effektive Verarbeitungsgeschwindigkeit erzielt, was für Anwendungen in der Prozeßkontrolle bzw. Steuer- und Regeltechnik ausschlaggebend ist. Befehle zur Einzel-Bit-, 4 bit-, 8 bit- und 16 bit-Verarbeitung und zum Transfer ganzer Datenblöcke machen das System zum idealen Universalstandard für alle Aufgabengebiete, wie Maschinensteuerung, Textverarbeitung, Automatisierung sowie allgemeine Geräterealisierung.

Beim Baustein DMA (Direct Memory Access) handelt es sich um einen hochintegrierten, programmierbaren Baustein zur Implementierung „direkter Speicherzugriffe"

- ☐ für die besonders schnelle Übertragung von Daten zwischen zwei Funktionsblöcken (= „Ports") innerhalb eines Z80-Systems. Ports können hierbei sowohl Arbeitsspeicher als auch Ein-/Ausgabe-Schnittstellen peripherer Geräte sein.
- ☐ zur schnellen Suche nach einem bestimmten byte oder einer 8-bit Kombination (bit maskierbar) in einem Datenblock. Der Anwender kann zusätzlich festlegen, ob das gesuchte Zeichen gleichzeitig übertragen werden soll oder nicht.

Der Baustein enthält einen solchen DMA-Kanal. Die hierfür notwendigen Zeit- und Steuersignale einschließlich der Adressierung erzeugt die Logik des DMA-Bausteins.

☐ Aufbau

- N-Kanal Silicon-Gate-Depletion Load Technologie
- 40 pin DIP-Gehäuse
- eine einzige 5 V-Spannungsversorgung
- 5 V-Einphasen-Takt
- 1 DMA-Kanal = 2 DMA-Ports

☐ Kenndaten des DMA

- Drei Funktionstypen sind möglich:
  - ☐ nur die Übertragung
  - ☐ nur die Suche
  - ☐ Suche und Übertragung
- Bei einem Übertragungsvorgang werden 2 Adressen gebildet (eine für den Lese-Port, die andere für den Schreib-Port).
- Bei der Übertragung und Suche wird die Portadresse, beginnend von einer vorprogrammierbaren Anfangsadresse, automatisch inkrementiert bzw. dekrementiert (oder bleibt unverändert).
- Seine vier Betriebsarten sind per Programm festzulegen:
  - ☐ Ein-Byte-Übertragung: Pro Bus-request-Anforderung wird 1 byte übertragen.
  - ☐ Peripheriegesteuerte Operation (burst): Operation läuft bis zu einem „Ende" Signal der Peripherie an die DMA-Ports.
  - ☐ Programmgesteuerte Operation (continuous): Operation läuft, bis ein Block mit per Programm festgelegter Länge abgearbeitet ist.
  - ☐ transparente: Operation: („≙ refresh cycles stealing") Die Operation findet während der Speicherrefresh-Zyklen statt.
- Das Zeitverhalten jedes Ports ist programmierbar.
- Beim Auffinden eines gesuchten bytes beim Übertragungsende eines Blocks oder Ready können Interrupts vorprogrammiert werden.
- Eine Operation kann automatisch oder auf Kommando vollständig wiederholt werden (Auto restart oder load).
- Sobald eine bestimmte Anzahl von bytes übertragen worden ist, kann vom DMA-Baustein ein Strobe-Signal ohne Unterbrechung der Übertragung abgegeben werden.
- Mehrfach-DMA-Strukturen mit unterschiedlicher Priorität können in einfacher Weise realisiert werden.
- Der DMA-Kanal kann per Programm freigegeben, gesperrt oder zurückgesetzt werden.
- Per Software kann auch der Status des DMA-Kanals vollständig abgefragt werden.
- Übertragungsraten bis 1,25 Megabyte sind möglich.
- Automatische Interrupt-Vektorerzeugung ohne zusätzlichen Schaltungsaufwand mit Prioritätskaskadierung der Bausteine (daisy chain priority interrupt logic). Die Bestätigung der Busanforderung („BUSRQ") erfolgt hierbei ebenfalls automatisch.
- Alle Ein- und Ausgänge sind TTL-kompatibel.
- Von der CPU können die momentanen Werte von Portadreß-, Bytezählern oder Status-Registern gelesen werden. Die Auswahl des zu lesenden Registers erfolgt durch ein Maskierungs-Steuerwort.

☐ Die Architektur des DMA-Bausteins

Bild 1 zeigt das Blockschaltbild des DMA. Er besteht aus folgenden Funktionseinheiten:

- Bus interfaces
- Steuerlogik und zugehörige Register: Sämtliche Steuerparameter (wie z. B. Betriebsart oder Funktionstyp) werden in dieser Schaltung erkannt und zwischengespeichert.
- Adreß-, Byte-Zähler und Pulsschaltung: Hier werden die Portadressen für Lese- und Schreiboperationen erzeugt und in- bzw. dekrementiert. Ferner setzt die Byte-Zähl-Schaltung im Statusregister ein Flag, wenn das „nullte" Byte transferiert wird. Außerdem erzeugt die Pulsschaltung jedes Mal, wenn die niederwertigen 8 bit des byte-Zählers mit dem Inhalt des „Puls"-Registers übereinstimmen, einen Impuls.
- Schaltung für die Steuerung des DMA-Zeitverhaltens: Das Zeitverhalten bei Lese- und Schreibzugriffen wird für beide Ports per Programm festgelegt.
- Vergleicher: In diesem Block werden durch Vorgabe einer Maske (mask byte) bestimmte bits des aktuellen Worts mit einem gesuchten byte (match byte) verglichen. Sobald das gesuchte byte während eines Übertragungsvorganges gefunden wurde, wird im Statusregister ein Flag gesetzt.

- Interrupt- und BUSRQ-Schaltung: Im Interrupt Steuerregister wird festgelegt, unter welchen Bedingungen der DMA-Baustein einen Interrupt erzeugen darf. Ob dann zu diesem Zeitpunkt ein INT- oder BUSRQ-Signal ausgegeben wird, hängt von der jeweiligen Prioritätscodierungs-Logik ab. Ferner ist in diesem Schaltungsteil ein Interrupt-Vektor-Register enthalten, dessen Inhalt einen Teil des Zeigers einer Interrupt-Bedienroutine darstellt.
- Zustandsregister: In diesem Register sind die laufenden und für den Fortgang bestimmenden DMA-Status-Bedingungen gespeichert.

☐ Beschreibung der DMA-Register

Auf folgende interne DMA-Register hat der Anwender Zugriff:

Register	Registerlänge (bit)	L*1	S*2	Erläuterung
● Steuer-Register			X	die für den DMA notwendigen Steuerinformationen sind hier zwischengespeichert, wie z. B. ob ein Interrupt oder Puls erzeugt werden soll, in welcher Betriebsart gearbeitet werden soll, usw.
● Zeitablauf Register	8		X	bestimmt das zeitliche Schreib-/Leseverhalten für die beiden Ports
● Interrupt-Vektor-Register	8	X	X	Bei Interrupt-Anforderung wird der in diesem Register zwischengespeicherte 8 bit-Vektor auf den Datenbus ausgegeben. Der Registerinhalt kann nur während eines Interrupt-Quittungszyklus gelesen werden.
● Blocklängenzähler	16		X	Enthält die gesamte gesuchte und/oder zu übertragende Blocklänge.
● Byte-Zähler	16	X		Die übertragenen oder gesuchten Bytes werden solange aufgezählt bis der Inhalt mit dem Blocklängenregister übereinstimmt. Im Statusregister wird dann das bit „Blocklängenende" gesetzt. Der jeweilige Vorgang kann je nach Programmierung abgebrochen oder einen Interrupt erzeugen. Mit Load oder Continue wird der Byte-Zähler auf Null zurückgesetzt.
● Vergleichs-Register	8		X	gespeichert wird hier ein byte, zu dem ein gleichartiges byte (match byte) während eines Suchvorganges gefunden werden soll.
● Maskierungs Register	8		X	durch eine 8-bit Maske wird festgelegt, welche bits im Vergleichsregister für das Auffinden einer gleichartigen bit-Kombination herangezogen werden.
● Startadressenregister (Port A und Port B)	16		X	gespeichert werden hier die Startadressen (höher- und niederwertige 8-bit) für beide Ports (z. B. Arbeitsspeicher). Bei Suchvorgängen muß nur ein Port spezifiziert und adressiert werden; bei peripheren Ports werden nur die niederwertigen 8-bit als feste Adresse verwendet.
● Adreß-Zähler (Port A und Port B)	16	X		die jeweiligen Startadressen werden in die Adreßzähler mit Load oder Continue geladen. Durch die Programmierung wird festgelegt ob die Adressen fest bleiben oder in- oder dekrementiert werden.
● Pulssteuer-Register	8		X	sobald die hier zwischengespeicherten Sektor- oder Blocklängen (angegeben in bytes) abgearbeitet sind, wird als Rückmeldesignal für ein peripheres Gerät am pin $\overline{INT}$ ein Impuls ausgegeben. Der CPU-Baustein interpretiert dieses Puls-Signal nicht als eine Interrupt-Anforderung, da die Leitung $\overline{BUSRQ}$ vom DMA zuvor aktiviert wurde.
● Status Register	8	X		Enthält die Statusinformationen (siehe Programmierung des DMA-Bausteins).

*1: diese Informationen können nur gelesen werden.
*2: diese Informationen können nur eingeschrieben werden.
*1 und *2: diese Informationen können sowohl gelesen als auch eingeschrieben werden.

☐ Funktionstypen des DMA-Bausteins

Drei Funktionstypen sind möglich:
☐ nur die Übertragung
☐ nur die Suche und
☐ kombinierte Übertragung und Suche

Bei einem Übertragungsvorgang werden die Daten von einem Port gelesen und byteweise in das andere Port geschrieben. (Die beiden Ports werden als Port A und Port B bezeichnet). Die Ports können so programmiert werden, daß sie entweder dem Arbeitsspeicher oder peripheren Ein-/Ausgabe-Schnittstellen zugeordnet sind. Folgende Datenübertragungen sind möglich:

- von einer peripheren Einheit zu einer anderen
- von einem Bereich im Arbeitsspeicher zu einem anderen
- von einer peripheren Einheit zum Arbeitsspeicher

Bei einem Suchvorgang werden die Daten nur gelesen und byteweise mit zwei Register im DMA verglichen. In einem

Register steht das für den Vergleich vorgegebene byte (match byte), im anderen (falls erforderlich) ein Masken-byte. Dadurch wird das Aufsuchen einzelner bits oder beliebiger Kombinationen möglich. Sobald das gesuchte 8bit-Wort gefunden ist, wird im DMA-Baustein ein Statusbit gesetzt. Je nachdem, wie der DMA-Baustein programmiert wurde, wird dann der Suchvorgang abgebrochen und/oder ein Interrupt erzeugt.
Bei kombinierten Übertragungs- und Suchvorgängen wird solange ein Datenblock übertragen, bis das gesuchte byte bzw. die gesuchte bit-Kombination gefunden wurde. Bei Übertragung kann dann ähnlich wie beim Suchvorgang abgebrochen und/oder ein Interrupt erzeugt werden. Der Funktionstyp wird mit dem Kommandobyte 1A programmiert.

☐ Betriebsart

Der DMA-Baustein wird vom Anwenderprogramm auf eine seiner 4 möglichen Betriebsarten eingestellt:

- byteweise: die CPU übernimmt wieder die Kontrolle nach jeder Einzelbyte-Operation
- andauernd: (burst) solange der RDY-Eingang aktiv ist, gilt der entsprechende Port als sende- oder empfangsbereit und vom DMA werden Operationen durchgeführt bis das Blockende oder eine vorprogrammierte Vergleichsbedingung erreicht ist
- fortlaufend: (continuous) Die gesamte Suche und/oder Übertragung von Datenblöcken ist abgeschlossen bevor die CPU wieder die Kontrolle übernimmt
- transparent: die DMA-Operation wird während Refreshzyklen durchgeführt. Diese Form von DMA läuft ohne jeden Zeitverlust ab

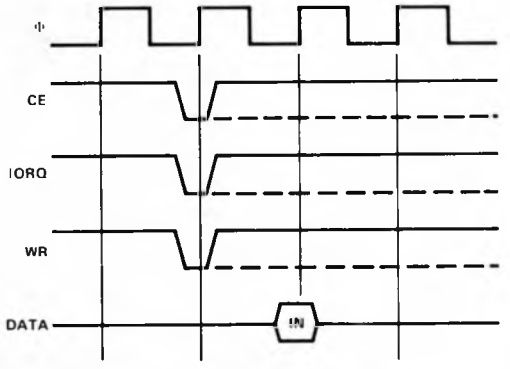
☐ Adressierung

Die Adressierung der DMA-Ports ist entweder fest oder aber ab einer vorgegebenen Startadresse sequentiell inkrementierend bzw. dekrementierend. Die Länge des zu übertragenden Blocks ist durch das vom Anwenderprogramm programmierte Blocklängenregister festgelegt. Bei einem Übertragungsvorgang werden 2 Adressen gebildet (eine für den Lese-Port, die andere für den Schreib-Port).
Der Z80-DMA kann Blocklängen bis zu 64 Kbytes adressieren.

☐ Pin-Belegung

Pin	Funktion	Erläuterung
A_0—A_{15}	Adreßbus	Tristate Ausgänge, high-aktiv; Adressierung des Arbeitsspeichers oder eines I/0 Ports
D_0—D_7	Datenbus	Tristate Ein-/Ausgänge (bedirektional) high-aktiv; über den Datenbus erfolgt der Datenaustausch zwischen CPU, Speicher und E/A-Bausteinen
+5 V	Spannungsversorgung	
GND	Bezugspotential	
Φ	Systemtakt	Eingang
$\overline{M1}$	Maschinenzyklus 1	Eingang für M1 Signal von der Z80-CPU, low-aktiv
$\overline{IORQ}$	Ein/Ausgabe-Anforderung („In/Output-Request")	Ein/Ausgabe-Anforderung vom und an den Systembus.
$\overline{MREQ}$	Speicheranforderung (memory-request)	Tristate Ein-/Ausgang, low-aktiv; Speicheranforderung vom und an den Systembus
$\overline{RD}$	Lesen (Read)	Tristate Ein-/Ausgang; Lesesignal vom und für den Systembus
$\overline{WR}$	Schreiben (Write)	Tristate Ein-/Ausgang; Schreibsignal vom und für den Systembus
$\overline{CE/WAIT}$	Bausteinfreigabe/Warten (chip enable/wait)	Eingang, low-aktiv, das Bausteinfreigabesignal kann auch als $\overline{WAIT}$-Eingang programmiert werden, falls $\overline{BAI}$ low ist.
$\overline{BUSRQ}$	Busanforderung (Bus request)	Ein-/Ausgang, low-aktiv, angefordert wird die Kontrolle über Adreß-, Daten und Status/Steuer-Bus
$\overline{BAI}$	Busanforderungsbestätigung-Eingang (Bus acknowledge input)	Zeigt mit seinem Aktiv-(= low) werden an, daß die Buskontrolle an den DMA-Baustein abgegeben wird
$\overline{BAO}$	Busanforderungsbestätigung-Ausgang (Bus acknowledge output)	Ausgang, low-aktiv. $\overline{BAI}$ und $\overline{BAO}$ sind kaskadierbare Verbindungen zur BUS-Prioritätssteuerung z. B. bei Verwendung mehrerer DMA-Kanäle
$\overline{INT}$	Interruptanforderung (interrupt request)	Ausgang, low-aktiv
IEI	Interrupt-Freigabe-Eingang (interrupt enable in)	Eingang, high-aktiv
IEO	Interrupt-Freigabe-Ausgang (Interrupt enable output)	Ausgang, high-aktiv, IEI u. IEO sind kaskadierbare Verbindungen zur Interrupts-Prioritätssteuerung
RDY	Quittierung (Ready)	Eingang, aktiv high oder low je nach Programmierung; hierdurch wird dem DMA mitgeteilt, ob die entsprechende periphere Einheit zum Lesen oder Schreiben bereit ist

☐ Zeitdiagramme des DMA

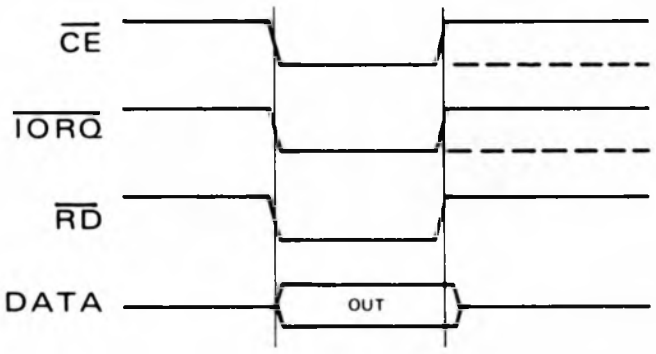


1. Zyklus zum Schreiben eines Kommandobytes

Ein Kommando- oder Steuerbyte wird in ein internes Z80-DMA-Register geschrieben. Das gezeigte Zeitverhalten ist durch die CPU Ausgabebefehle bestimmt.

2. Zyklus zum Lesen eines Registers

Das Lesen des Inhalts des Statusregisters, des Adreßzählers oder anderer lesbarer Register/Zähler geschieht nach folgendem Zeitverhalten, daß durch die CPU-Eingabebefehle bestimmt ist.



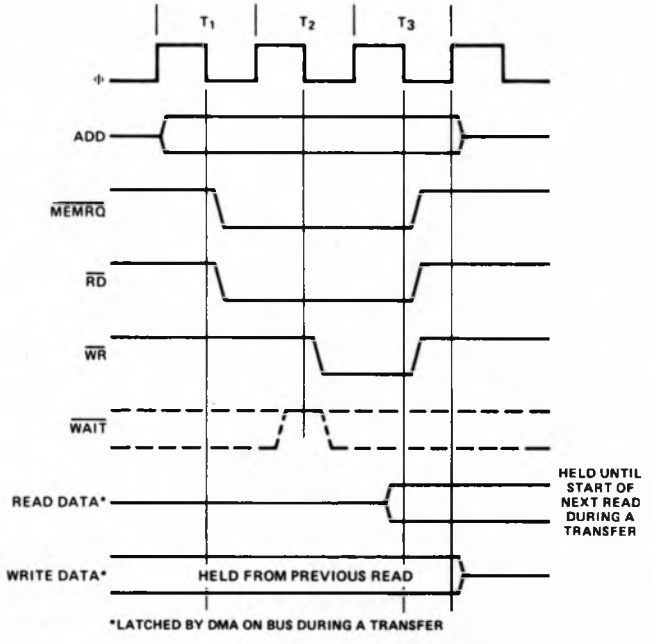
3. Speicherzugriff

Das DMA-Zeitverhalten beim Arbeitsspeicherzugriff entspricht sowohl beim Lese- als auch Schreibvorgang dem der CPU.

Dieses Zeitverhalten wird auch automatisch nach jedem RESET-Signal, beim Einschalten der Speisespannung und nach einem RESET-Kommando programmiert, und wird falls kein anderes Kommandowort gesendet wird, bei allen Such- oder Übertragungsaktivitäten gegenüber dem Arbeitsspeicher beibehalten. Während der Speicherlesephase eines Übertragungszyklus werden die Daten im DMA mit der fallenden Taktflanke während T3 bis zu einem darauffolgenden Schreibzyklus zwischengespeichert. Während der Speicherschreibphase eines Übertragungszyklus werden die Daten vom vorausgegangenen Lesezyklus bis zum Ende des momentanen Zyklus zwischengespeichert.

Zur Beachtung:

Der DMA-Baustein ist für Speicherzugriffe mit 3 Maschinen (T-) Zyklen ausgelegt. Es ist jedoch das Einfügen von „WAIT"-Zyklen möglich, da mit der abfallenden Taktflanke von T2 der log. Zustand des WAIT-Signals abgefragt wird. Ist dieser low, so wird ein weiterer T-Zyklus angehängt und das WAIT-Signal nochmals abgefragt. Auf diese Weise kann die Dauer von Speicherzugriffen beliebig ausgedehnt und ein Anpassen an langsamere Speicher problemfrei vorgenommen werden.

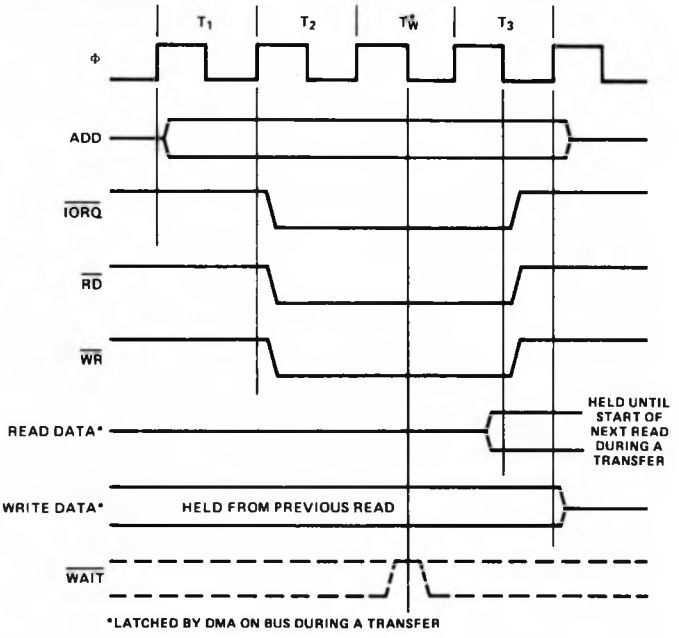


4. Verkehr mit Ein-/Ausgabeeinheiten

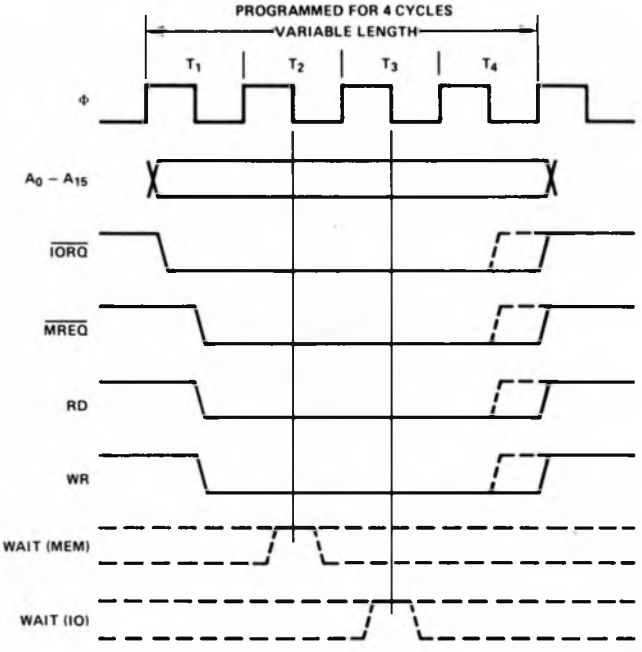
Das Zeitverhalten ist in diesem Fall identisch mit dem der CPU für Lese-Schreiboperationen mit Ein-/Ausgabe-Bausteinen, und wird dem Baustein DMA beim Anlegen der Speisespannung, einem Signal auf der RESET-Leitung oder nach einem Reset-Kommandowort automatisch vorgegeben. Es wird, falls dem DMA keine anderen Kommandowörter zugesendet werden, bei allen Such- und Übertragungsvorgängen mit peripheren Ein-/Ausgabe Schnittstellen beibehalten. Während der Ein-/Ausgabe-Lesephase eines Übertragungszyklus werden die Daten mit der abfallenden Taktflanke des T3-Taktimpulses bis zum nächstfolgenden Schreibzyklus zwischengespeichert. Während der Ein-/Ausgabe-Schreibphase eines Übertragungszyklus werden die Daten vom vorangegangenen Lesezyklus bis zum Ende des momentanen Zyklus zwischengespeichert.

Zur Beachtung:

Ist das WAIT-Signal während der fallenden Taktflanke von T_W^*, gleich low, dann wird ein neuer T-Zyklus angehängt und das WAIT-Signal nochmals abgefragt. Auf diese Weise kann die Dauer von peripheren Ein-/Ausgabezugriffen beliebig ausgedehnt werden.



5. Variabler Zyklus



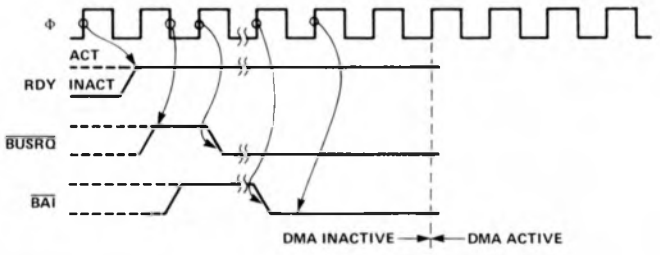
Eine besondere Eigenschaft des Z80-DMA ist seine Möglichkeit, das Zeitverhalten von Speicher oder Ein/Ausgabe-Vorgängen frei zu programmieren. Damit kann

a) das Zeitverhalten an die jeweiligen Anforderungen der gewählten Systemkomponenten angepaßt und
b) die Datentransferrate optimiert werden,

ohne daß hierfür zusätzliche Schaltungen oder Programmteile nötig werden. Die Zykluslänge kann auf ein bis 4 Maschinen-(T-)Zyklen (und mehr bei Verwendung von $\overline{WAIT}$-zyklen) vorprogrammiert werden. Die Signallängen können entsprechend nebenstehendem Impulsdiagramm variiert werden. Die Datenzwischenspeicherung während des Übertragungsvorganges erfolgt nach der Taktflanke, die eine steigende Flanke von $\overline{RD}$- zur Folge hat. Bis zum Ende des folgenden Schreibzyklus werden diese Daten gespeichert.

6. Busverwaltung

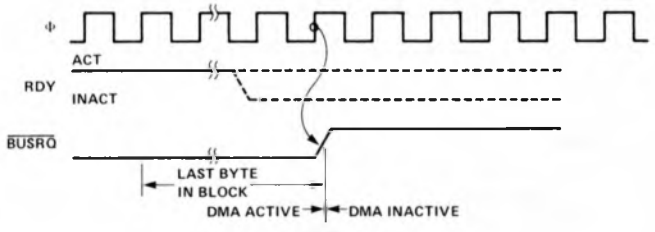
6.1. Busanforderungen bei den Betriebsarten:
byteweise, andauernd und fortlaufend
Das Ready-Signal wird mit jeder ansteigenden Flanke des Taktes Φ abgefragt. Sobald die Ready-Leitung als aktiv (=„1") erkannt wird, erzeugt die nächstfolgende ansteigende Takt-Flanke das $\overline{BUSRQ}$-Signal.



Die CPU aktiviert dann die $\overline{BUSAK}$-Leitung, die entweder direkt mit der BAI-Leitung verbunden wird oder über die kaskadierbare Busprioritätssteuerung.
Sobald $\overline{BAI}$ = „0" (wiederum abgefragt mit jeder steigenden Taktflanke) wird mit der nächstfolgenden ansteigenden Taktflanke ein DMA-Zyklus eingeleitet.

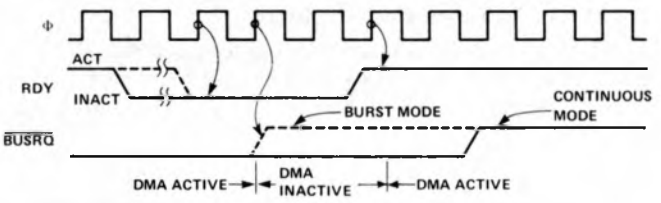
6.2. Busfreigaben beim DMA
Die Busfreigabe (also Rückgabe der Kontrolle über die Busse an die CPU oder andere DMA's) erfolgt je nach Betriebsart oder Sonderbedingung auf unterschiedliche Weise bei

a) Blockende:
Betriebsart andauernd und fortlaufend, falls eine „automatisches Wiederholen" nicht programmiert wurde.

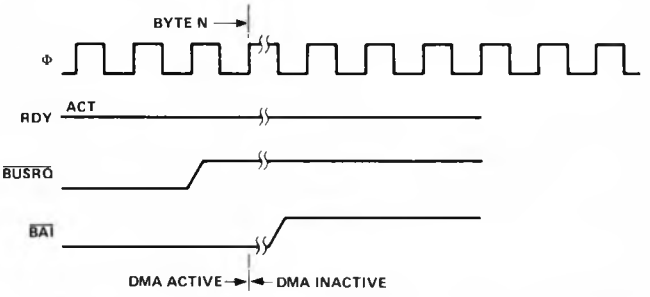


b) der Betriebsart andauernd (burst):
Busfreigabe erfolgt, nachdem die RDY-Leitung inaktiv (=„0") wird (Zeitdiagramm siehe c).

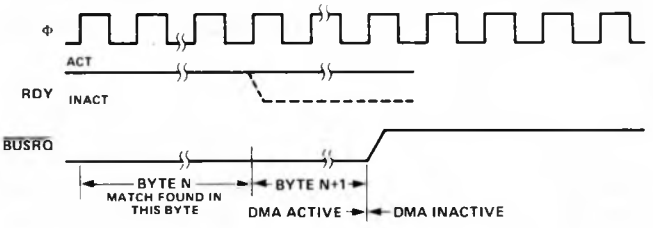
c) der Betriebsart fortlaufend:
Busfreigabe erfolgt, nachdem Blockende oder eine (byte) Vergleichbedingung gefunden wurde.
Die Kontrolle über die Busse wird auch mit RDY = „0" solange ausgeübt ($\overline{BUSRQ}$ = „0") bis der Zyklus bei RDY abgeschlossen werden kann.



d) der Betriebsart byteweise:
Die Busfreigabe erfolgt gleichgültig vom RDY-Zustand nach der steigenden Taktflanke Φ vor dem Ende jeder Lesezyklen bei Suchvorgängen und jeder Schreibzyklen bei Übertragungsvorgängen.
Erst wenn $\overline{BUSRQ}$ = „1" und $\overline{BAI}$ = „1" kann der DMA erneut die Kontrolle über die Busse übernehmen.

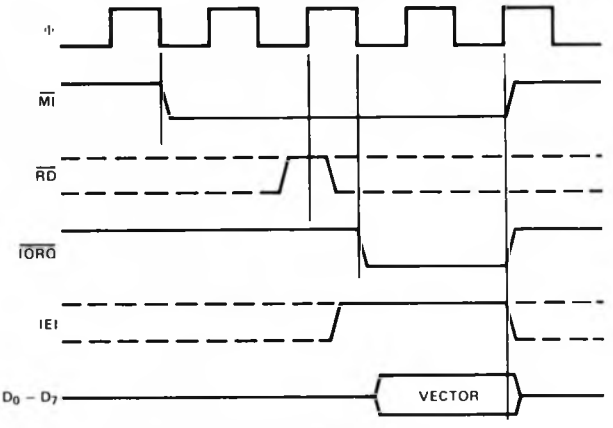


e) Vergleichen bei den Betriebsarten andauernd und fortlaufend:
Sobald eine (byte) Vergleichsbedingung gefunden worden ist, wird vom DMA eine weitere byte-Operation eingeleite und danach die Busse freigegeben. Das „stop on compare"-bit (siehe Commandowort 2A, D2) muß zuvor natürlich aktiviert sein.



7. ($\overline{M1}$ ∧ $\overline{IQ}$)-Interrupt-Quittungszyklus

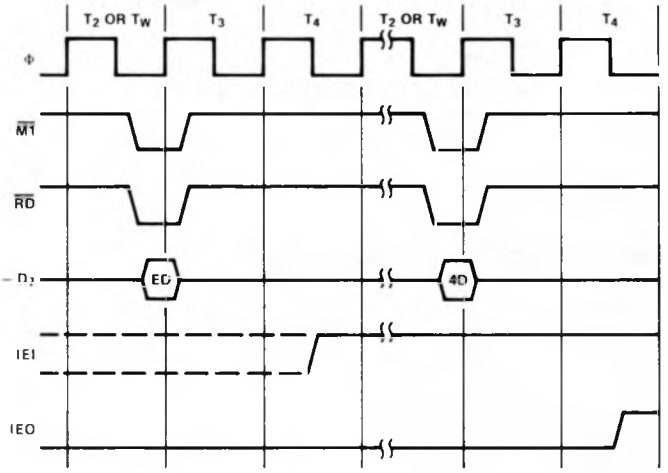
Nachdem vom DMA ein Interrupt angefordert wurde, wird von der CPU eine Interrupt-Quittung ausgegeben ($\overline{M1}$ ∧ $\overline{IORQ}$). In der Zwischenzeit wird durch die Interruptlogik festgestellt, ob der DMA-Kanal die höchste Priorität hat.



(Die Interruptkaskade muß dazu erst einschwingen. Es dürfen sich deshalb die Zustände der Interruptanforderungen nicht ändern, solange $\overline{M1}$ aktiv ist). Ein DMA-Vektor wird dann auf den Datenbus ausgegeben, sobald $\overline{IORQ}$ aktiv wird.

8. Rücksprung vom Interruptzyklus

Durch eine RETI-Anweisung am Ende einer Interrupt-Service-Routine wird der eindeutige Ablauf bei priorisierten, verschachtelten Interrupts gewährleistet. Die RETI-Anweisung wird im DMA-Baustein zu Identifizierungszwecken dekodiert.

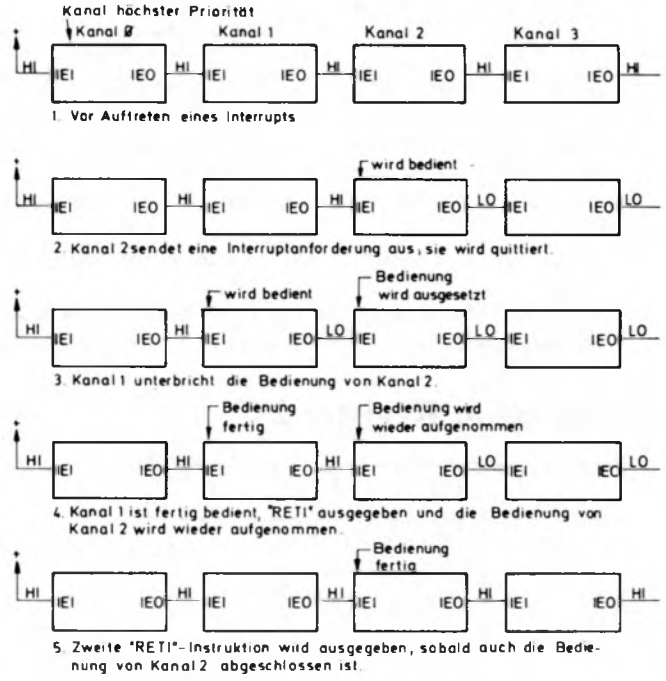


Bei Erkennen des Codeworts EDH wird das IEI-Signal des jeweils angesprochenen Bausteines aktiv. Wird 4 DH empfangen, so wird der momentan bediente Kanal neu initialisiert und sein IEO-Ausgang aktiviert.

☐ Vorgang der Prioritäts-Kaskadierung

In der Skizze ist eine typische Interruptabfolge als Beispiel dargestellt.

Darin fordert Kanal 2 einen Interrupt an und wird bedient. Währenddessen fordert der Kanal 1 ebenfalls einen Interrupt an und wird wegen seiner höheren Priorität bedient, wobei die Bedienung von Kanal 2 unterbrochen wird. Sobald nun die Bedienung von Kanal 1 abgeschlossen und die RETI-Anweisung ausgeführt ist, wird die niedriger priorisierte Behandlung von Kanal 2 fortgesetzt und zuende geführt.



☐ Lesen der DMA-Register

Der DMA-Baustein enthält sieben Register, die sequentiell in der unten aufgezählten Reihenfolge gelesen werden können:

- Status-Register
- Blocklängenregister (niederwertiger Teil)
- Blocklängenregister (höherwertiger Teil)
- Portadresse A (niederwertiger Teil)
- Portadresse A (höherwertiger Teil)
- Portadresse B (niederwertiger Teil)
- Portadresse B (höherwertiger Teil)

Das Weiterschalten von Register zu Register erfolgt durch ein internes Zeigerregister, das bei jedem READ-Signal fortgeschrieben wird. Soll ein Register nicht gelesen werden, kann es durch Setzen einer „0" an der entsprechenden Stelle der Lese-Maske übersprungen werden.

Nach jedem Reset Chip. Reset timing, RESTART, Continue, Enable Chip, Disable Chip oder Reset RD wird der interne Zeiger auf das erste durch die Lesemarke festgelegte Register gesetzt. Nach jedem RD-Status-Signal weist der Zeiger immer auf das Status Register (unabhängig von der Lese-(„Response")-Maske).

☐ Programmierung des DMA-Bausteines

Im aktiven Zustand übernimmt der DMA-Baustein die Kontrolle über die Systembusse, den Datentransfer und die Interruptverarbeitung.

Ein Kommandowort kann dem DMA-Baustein sowohl im aktiven als auch im inaktiven Zustand übermittelt werden; es bewirkt automatisch einen Übergang in den nicht aktiven Zustand. In diesem Zustand befindet sich der Baustein auch, wenn die Stromversorgung eingeschaltet oder der DMA-Baustein auf eine andere Weise zurückgesetzt wird.

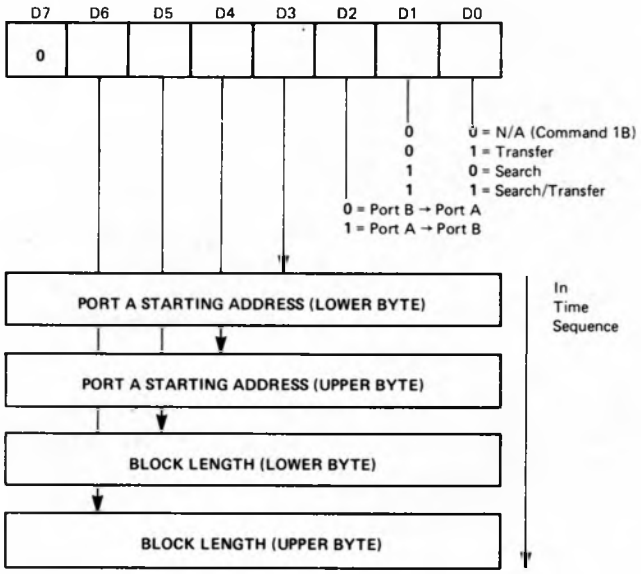
Bevor der DMA-Baustein seine eigentliche Aufgabe erfüllen kann, muß er durch das Anwenderprogramm über die CPU wie ein Ein-/Ausgabe Port initialisiert werden. D.h. durch Ausgabebefehle wird eine Folge von Kommandoworte über den Datenbus in die verschiedenen DMA-Register geladen. Die Kommandoworte enthalten 8 bit-Steuerinformationen, die den Zustand und die Arbeitsweise des DMA-Bausteins bestimmen und beeinflussen wie z.B. das Setzen eines Interruptfreigabebits oder zusätzlich benötigte Informationen, die nach dem jeweiligen Kommandobyte geladen werden wie z.B. die Startadresse eines Ports.

Die Funktion jedes einzelnen bits der 6 verschiedenen Kommandobytes wird anhand folgender Zusammenstellung klar:

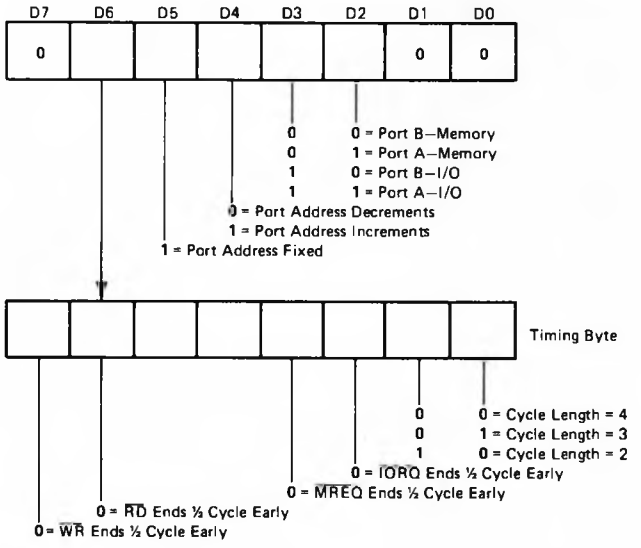
Die Kommandobytes sind in 2 Gruppen eingeteilt
Gruppe 1 mit den bytes 1 A, 1 B
Gruppe 2 mit den bytes 2 A, 2 B, 2 C, 2 D
und sind durch die bits D_0, D_1 und D_7 gekennzeichnet.

Eine Übersicht über die Formate der Kommandoworte ist in den folgenden Skizzen gezeigt:

Kommando Register 1A



Kommando Register 1B

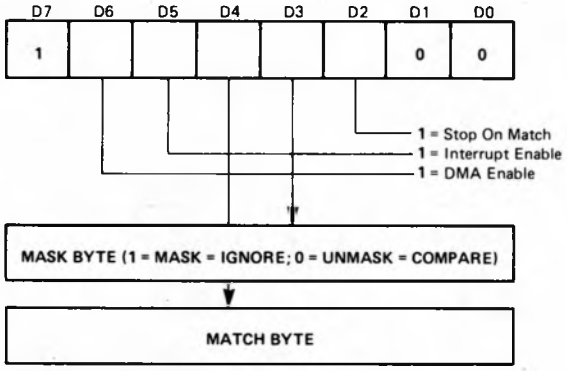


Für Übertragungen wird dieses Byte normalerweise zweimal geschrieben, nämlich einmal für Port A und dann für Port B.

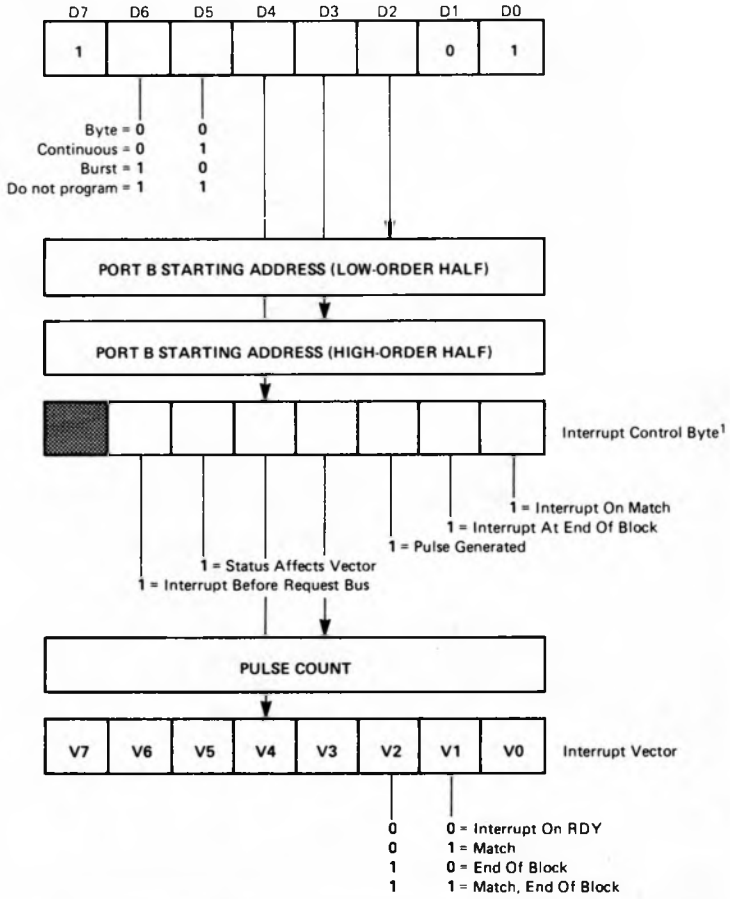
Eine „1" in Bit D_3 bis D_6 bedeutet, daß das betreffende Byte im Folgenden an den DMA gesendet wird. Beachten Sie, daß die Reihenfolge der Bytes zwingend ist.

Der DMA-Baustein überträgt oder sucht grundsätzlich ein Byte mehr als die Zahl im Blocklängenregister angibt. Eine „0" im Blocklängenregister veranlaßt das Übertragen oder Suchen von $(2^{16}+1)$ Bytes. Die kürzeste programmierbare Blocklänge ist daher 2 Byte (durch eine „1" im Blocklängenregister festgelegt).

Kommando Register 2A



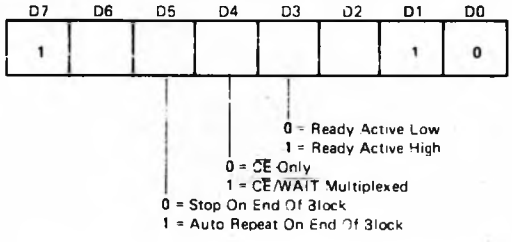
Kommando Register 2B



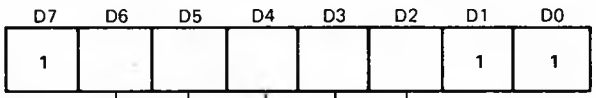
Bei „Interrupt before Requesting Bus" (eine 1 in Bit 6 des Interrupt-Control-Bytes) stellt der DMA-Baustein seine Interrupt-Anforderung erst dann aus, wenn der DMA-Baustein vorher die folgenden Kommandos empfangen hat:

- ☐ „Freigabe nach RETI"-Kommando (B7 in Kommando Byte 2D).
- ☐ „DMA-Freigabe"-Kommando (87 in Kommando-Byte 2D).
- ☐ Eine RETI-Anweisung, die das IUS-Flipflop des DMA-Bausteins rücksetzt (IUS = Interrupt under Service-Latch)

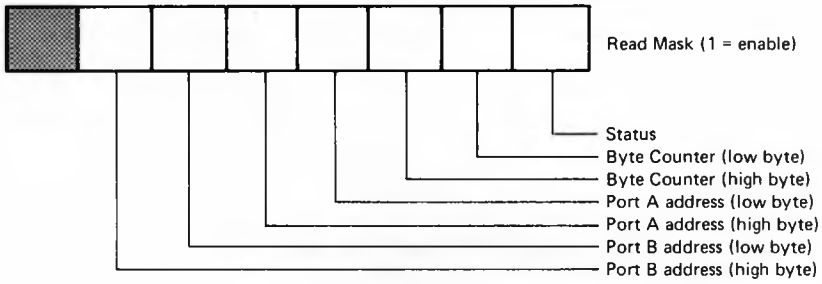
Kommando Register 2C



Kommando Register 2D



HEX					
C3	1	0	0	0	0 = **Reset** Interrupt circuitry, disable interrupt and bus request logic, unforce internal ready condition, disable "MUXCE" and stop auto repeat.
C7	1	0	0	0	1 = **Reset Port A Timing** to standard Z-80 CPU timing.
CB	1	0	0	1	0 = **Reset Port B Timing** to standard Z-80 CPU timing.
CF	1	0	0	1	1 = **Load** starting address for both ports, clear byte counter.*
D3	1	0	1	0	0 = Addresses **continue** from present locations, clear byte counter.
AB	0	1	0	1	0 = **Enable interrupts**
AF	0	1	0	1	1 = **Disable interrupts**
A3	0	1	0	0	0 = **Reset and disable interrupt** circuits (like RETI) and unforce the internal ready condition
87	0	0	0	0	1 = **Enable DMA** ⎫ Both affect all operations except interrupts, but do not
83	0	0	0	0	0 = **Disable DMA** ⎭ reset any functions.
A7	0	1	0	0	1 = **Initiate read** sequence to the first register designated as readable by the Read Mask register.
BF	0	1	1	1	1 = Set **read status** so next read is from status register.
B3	0	1	1	0	0 = **Force** an internal **ready** condition independent of the RDY input. Used for memory-to-memory operations where no RDY signal is needed. This command does not function in the "byte-at-a-time" mode.
8B	0	0	0	1	0 = **Clear** Match and End of Block **status** bits.
B7	0	1	1	0	1 = **Enable after RETI** so DMA will request bus only after receiving a RETI. Must be followed by an Enable DMA command.
BB	0	1	1	1	0 = **Read mask** is the following byte.



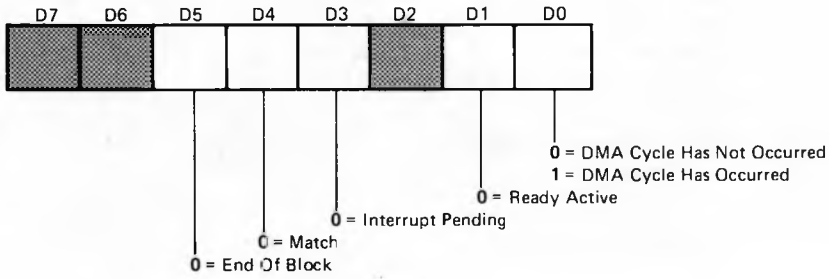
* Loading Port Addresses. The "Load" command (CF in Command Register 2D) loads a fixed address only into a port selected as the source, not into a port selected as the destination. Therefore, the destination address must be loaded by temporarily mislabeling the destination as the source.

The following example is a set-up procedure for a transfer from Port A to Port B:

1. Command byte 1A with B as source port
2. Command byte 2D with CF = load
3. Command byte 1A with A as source port
4. Command byte 2D with CF = load
5. Command byte 2D with 87 = Enable DMA

This manipulation is required only when the destination has a fixed address.

Status Register



□ Erläuterung zur Programmierung

Funktionstypenschlüssel

C_1	C_0	Funktion
0	0	nicht erlaubt
0	1	nur die Übertragung
1	0	nur die Suche
1	1	Suche und Übertragung

Betriebsartenschlüssel

M_1	M_0	Ablaufweise
0	0	byteweise
0	1	fortlaufend
1	0	burst
1	1	transparent

Zykluslängenschlüssel

T_1	T_2	Zykluslänge
0	0	4
0	1	3
1	0	2
1	1	1

(1) Eine „0" in D_2, D_3, D_6, oder D_7 hat zur Folge, daß das entsprechende Steuersignal eine halbe Taktzeit zuvor beendet ist. Für den Übertragungs- (Lesen u. Schreiben) und Suchvorgang (Lesen) sind mindestens 2 Taktzyklen erforderlich.

Interruptstatusschlüssel

(falls bit 5 des Interruptsteuerbytes gesetzt („1") wurde)

D_2	D_1	Funktion
0	0	Interruptanforderungen nach
0	1	Vergleichsbyte gefunden, Blocklängenende nicht gefunden
1	0	Vergleichsbyte nicht gefunden, Blocklängenende gefunden
1	1	Vergleichsbyte und Blocklängenende gefunden

Zustandsschlüssel

Hex	f_4	f_3	f_2	f_1	f_0	Bedeutung	Kurzbeschreibung
C3	1	0	0	0	0	Chip zurücksetzen	Setzt alle Interruptbedingungen zurück, sperrt Interrupts und Busse
C7	1	0	0	0	1	Setzt Zeitverhalten der Ports A/B zurück	gleicht das Zeitverhalten der Ports A/B an das Zeitverhalten der Z80-CPU an
CB	1	0	0	1	0		
CF	1	0	0	1	1	Load	setzt den Bytezähler auf Null zurück und lädt die Startadressen für beide Ports
D3	1	0	1	0	0	Restart bei momentanem Adreßstand	lädt nochmals die Blocklänge. Die Ausführung wird beim momentanen Adreßstand begonnen.
AB	0	1	0	1	0	Interruptfreigabe	Interrupts werden erlaubt
AF	0	1	0	1	1	Interruptsperre	Interrupts werden nicht erlaubt
A3	0	1	0	0	0	Interrupt-rücksetzung	Alle Interrupts werden zurückgesetzt (vergleiche mit RETI)
87	0	0	0	0	1	Chip Freigabe	Alle Operationen mit Ausnahme des Interrupts werden freigegeben oder
83	0	0	0	0	0	Chip Sperre	blockiert; es wird nichts zurückgesetzt
BB	0	1	1	1	0	Lesemarke folgt	welche Register ausgegeben werden, steht im nächst folgenden byte
A7	0	1	0	0	1	Zurücksetzen von Read	Das erste auslesbare, durch die Lesemarke bestimmte Register wird als nächstes gelesen.
BF	0	1	1	1	1	Lesen des Status (RD Status)	Als nächstes wird das Status Register ausgelesen
B3	0	1	1	0	0	Erzwungenes Ready-Signal (Force Ready)	Bei gewissen Operationen wird kein Ready-Signal benötigt (z.B. bei der Speicher zu Speicherübertragung). Das Ready-Signal wird hier als aktiv betrachtet, unabhängig vom tatsächlichen log. Zustand.
B7	0	1	1	0	1	Freigabe der Busanforderung nach einem RETI-Signal	Vom DMA-Baustein kommt keine Busanforderung, solange des RETI-Signal nicht empfangen wurde.
8B	0	0	0	1	0	Zurücksetzen des Status (RST Status)	Das Match und Blockende-Statusbit wird zurückgesetzt.

Masken-Byte

Ein Vergleich zwischen dem gelesenen und dem vorgegebenen Bit in einem Datenwort findet statt, falls an der entsprechenden Stelle eine log „0" im Masken byte vorgegeben wurde.

Match-Byte

Wörter bis zu 8-bit (D_0-D_7) können während des Lesevorganges verglichen werden. Näheres siehe Masken-Byte.

☐ Interrupt-Vektor

Während der Interrupt-Quittungsphase steht der CPU dieser 8-bit Vektor zur Erzeugung der Tabellenadresse für die Bedienroutinenstartadresse zur Verfügung (falls der DMA-Baustein die höchste Priorität hat).

☐ Anzahl der Operationen

Mit diesem 8-bit Wort wird der Puls-Zähler vorgesetzt. Nach jeder Operation wird der Zähler dekrementiert. Sobald der Zählerstand Null erreicht, erscheint ein Impuls auf der INT-Leitung. Anschließend wird der Zähler wieder vorgesetzt. Eine Interruptanforderung wird hierbei nicht erzeugt.

☐ Programmbeispiel

In dem unten gezeigten Beispiel sollen Daten (Blocklänge 1001 H Bytes) aus dem Arbeitsspeicher (Port A) auf ein peripheres Gerät (Port B). In diesem Beispiel ist die Port A-Anfangsadresse 1050 H und die feste Peripherieadresse 05 H.

	D7	D6	D5	D4	D3	D2	D1	D0	HEX
1) Command Register 1A sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0 Group One	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Addr Follows	1 Port A Lower Addr Follows	0 B →A Temporary For Loading B Address	0	1 Command Byte 1A Transfer, No Search	79
2) Port A address (lower)	0	1	0	1	0	0	0	0	50
3) Port A address (upper)	0	0	0	1	0	0	0	0	10
4) Block length (lower)	0	0	0	0	0	0	0	0	00
5) Block length (upper)	0	0	0	1	0	0	0	0	10
6) Command Register 1B defines Port A as memory with incrementing address.	0 Group One	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port Is Memory	1 This Is Port A	0	0 Byte 1B	14
7) Command Register 1B defines Port B as peripheral with fixed address.	0 Group One	0 No Timing Follows	1 Fixed Address	0 Not Used	1 Port Is I/O	0 This Is Port B	0	0 Byte 1B	28
8) Command Register 2B sets mode to Burst, sets DMA to expect Port B address.	1 Group Two	1	0 Burst Mode	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Addr Follows	0	1 Byte 2B	C5
9) Port B address (lower)	0	0	0	0	0	1	0	1	05
10) Command Register 2C sets Ready active High.	1 Group Two	0 Not Used	0 No Auto Restart	0 No Wait States	1 RDY Active HIGH	0 Not Used	1	0 Byte 2C	8A
11) Command Register 2D loads Port B address and resets block counter.	1 Group Two	1	0	0 Load	1	1	1	1 Byte 2D	CF
12) Command Register 1A sets Port A as source. *	0 Group One	0	0	0 No Addr Or Block Length Bytes	0	1 A →B	0	1 Byte 1A, Transfer No Search	03
13) Command Register 2D loads Port A address and resets block counter. *	1 Group Two	1	0	0 Load	1	1	1	1 Byte 2D	CF
14) Command byte 2D enables DMA to start operation.	1 Group Two	0	0	0 Enable DMA		1	1	1 Byte 2D	87

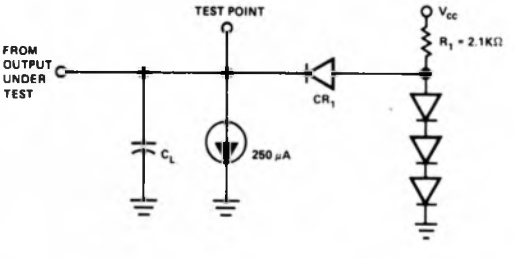
NOTE: The actual number of bytes transferred is one more than specified by the block length.
* These commands are necessary only in the case of a fixed destination address.

☐ Kapazitäten

$T_A = 25°C, f = 1\ MHz$

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	35	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

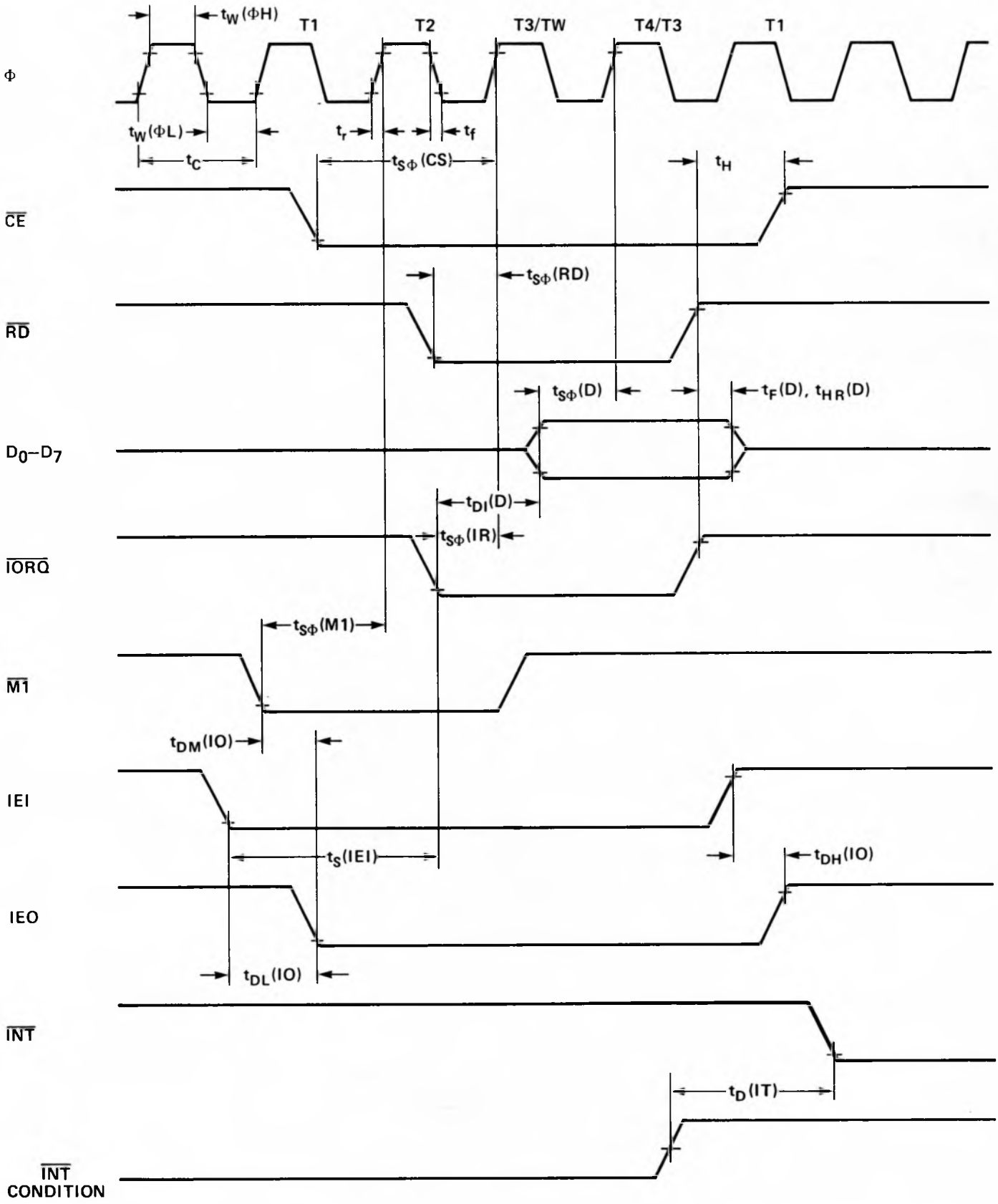
Load Circuit for Output



□ Zeitverhalten des Z80A-DMA als periphere (inaktive) Einheit

Timing measurements are made at the following voltages, unless otherwise specified:

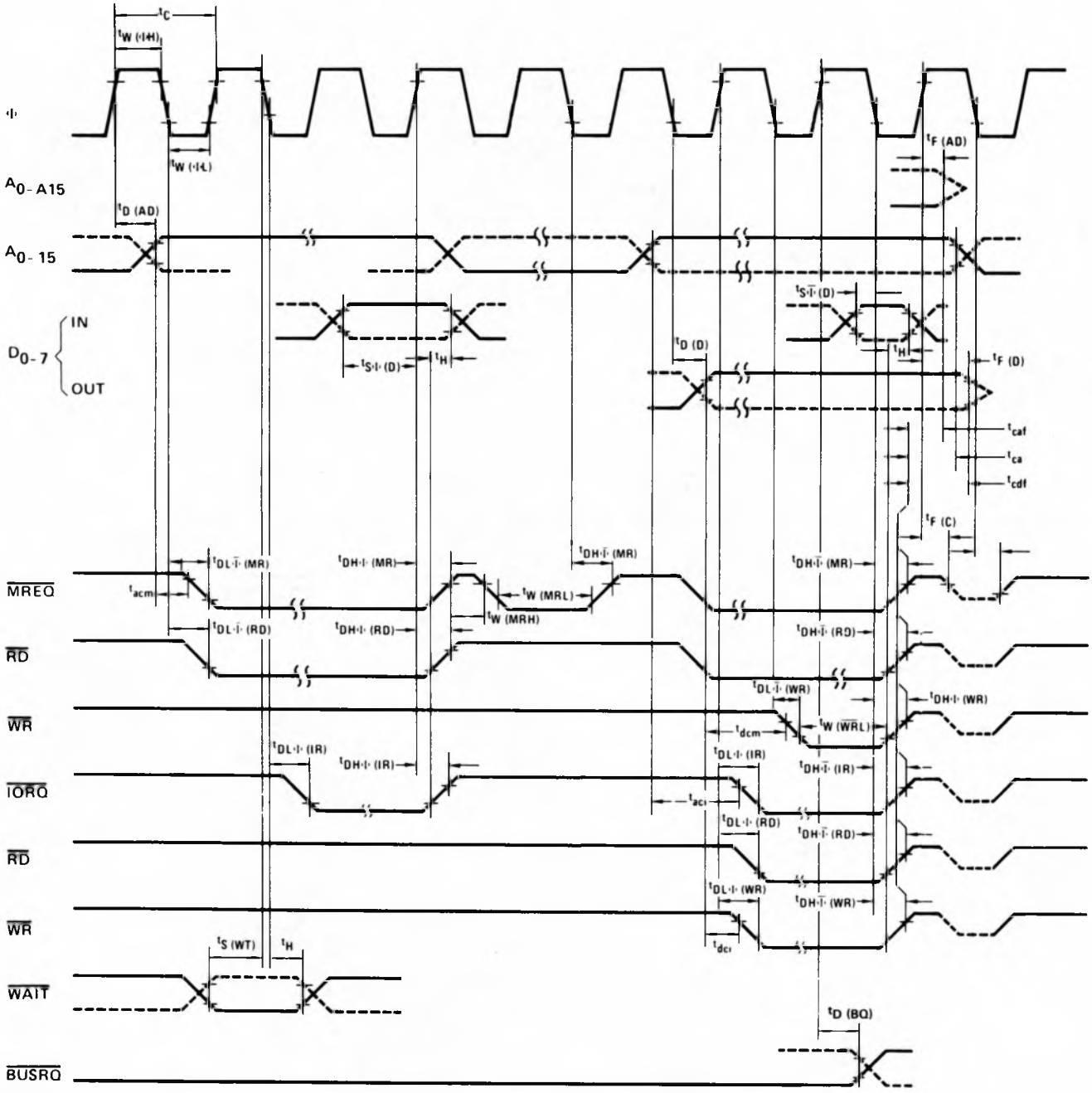
	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV =	+0.5V



☐ Zeitverhalten des Z80A-DMA als aktive Einheit (= Bus Controller)

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV =	+0.5V



☐ Dynamische Kenndaten des Z80A-DMA als periphere (inaktive) Einheit

$T_A = 0°C$ to $70°C$, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{w(\Phi H)}$ $t_{w(\Phi L)}$ t_r, t_f	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	250 105 105	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_H	Any Hold Time for Specified Setup Time	0		nsec	
$\overline{CE}$, WR	$t_{S\Phi(CS)}$	Control Signal Setup Time to Rising Edge of Φ During Write Cycle	145		nsec	
D_{0-7}	$t_{DR(D)}$ $t_{S\Phi(D)}$ $t_{DI(D)}$ $t_{F(D)}$	Data Output Delay from Falling Edge of $\overline{RD}$ Data Setup Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle Delay to Floating Bus (Output Buffer Disable Time)	50	380 250 110	nsec nsec nsec nsec	[2] C_L = 50pF [3]
IEI	$t_{S(IEI)}$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH(IO)}$ $t_{DL(IO)}$ $t_{DM(IO)}$	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		160 130 190	nsec nsec nsec	C_L = 50pF
$\overline{IORQ}$	$t_{S\Phi(IR)}$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Write Cycle	145		nsec	
$\overline{M1}$	$t_{S\Phi(M1)}$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B.	90		nsec	
$\overline{RD}$	$t_{S\Phi(RD)}$	$\overline{RD}$ Setup Time to Rising Edge of Φ During $\overline{M1}$ Cycle	115		nsec	
$\overline{INT}$	$t_{D(IT)}$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		500	nsec	
BAO	$t_{DH(BO)}$ $t_{DL(BO)}$	BAO Delay from Rising Edge of BAI BAO Delay from Falling Edge of BAI	150 150	200 200	nsec nsec	

[1] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

A. $2.5\,t_c > (N-2)\,t_{DL(IO)} + t_{DM(IO)} + t_{S(IEI)}$ + TTL Buffer Delay, if any

☐ Dynamische Kenndaten des Z80A-DMA als aktive Einheit (= Bus-Controller)

T_A = 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted.

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS	
Φ	t_c	Clock Period	.25	[12]	μsec		[12] $t_c = t_{w(\Phi H)}+t_{w(\Phi L)}+t_r+t_f$
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	2000	nsec		
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec		
	t_r, t_f	Clock Rise and Fall Time		30	nsec		
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec		
	$t_{F(AD)}$	Delay to Float		90	nsec	C_L = 50pF	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	D	[1] $t_{acm} = t_{w(\Phi H)}+t_f-65$
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	D	[2] $t_{aci} = t_c-70$
	t_{ca}	Address Stable from $\overline{RD}$ or $\overline{WR}$	[3]		nsec	D	[3] $t_{ca} = t_{w(\Phi L)}+t_r-50$
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	D	[4] $t_{caf} = t_{w(\Phi L)}+t_r-45$
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec		
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec		
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During Read When Rising Edge Ends RD			nsec	C_L = 200pF	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During Read When Falling Edge Ends $\overline{RD}$			nsec		
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	D	[5] $t_{dcm} = t_c-170$
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	D	[6] $t_{dci} = t_{w(\Phi L)}+t_r-170$
	t_{cdf}	Data Stable From $\overline{WR}$	[7]		nsec	D	[7] $t_{cdf} = t_{w(\Phi L)}+t_r-70$
	t_H	Any Hold Time for Setup Time		0	nsec		
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec		
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay from Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec		
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec		
	$t_{DL\Phi(MR)}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	C_L = 50pF	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	D	[8] $t_{w(MRL)} = t_c-30$
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	D	[9] $t_{w(MRH)} = t_{w(\Phi H)}+t_f-20$
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec		
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec		
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	C_L = 50pF	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec		
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ Low		85	nsec		
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ Low		95	nsec		
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ High		85	nsec	C_L = 50pF	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ High		85	nsec		
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ Low		65	nsec		
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ Low		80	nsec		
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ High		80	nsec	C_L = 50pF	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ High		80	nsec		
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec		[10] $t_{w(WR)} = t_c-30$
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec		
$\overline{BUSRQ}$	$t_{D(BQ)}$	$\overline{BUSRQ}$ Delay Time from Rising Edge of Clock		100	nsec		
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec		

NOTES:

A. Data should be enabled onto the DMA data bus when $\overline{RD}$ is active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V±5%
(1) ΔC_L = +100pF(A_0-A_{15} and Control Signals), add 30 nsec to timing shown.

D. During Standard CPU Timing

☐ Absolute Grenzdaten

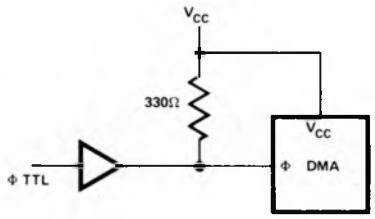
Temperature Under Bias	Specified operating range.
Storage Temperature	−65°C to +150°C
Voltage On Any Pin with Respect to Ground	−0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

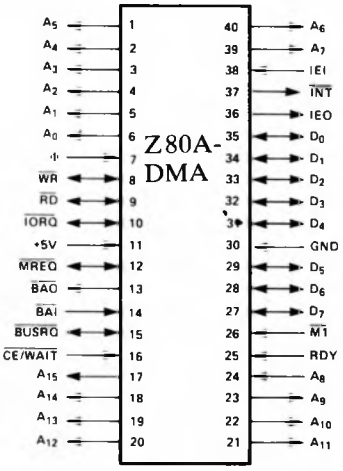
☐ Statische Kenndaten

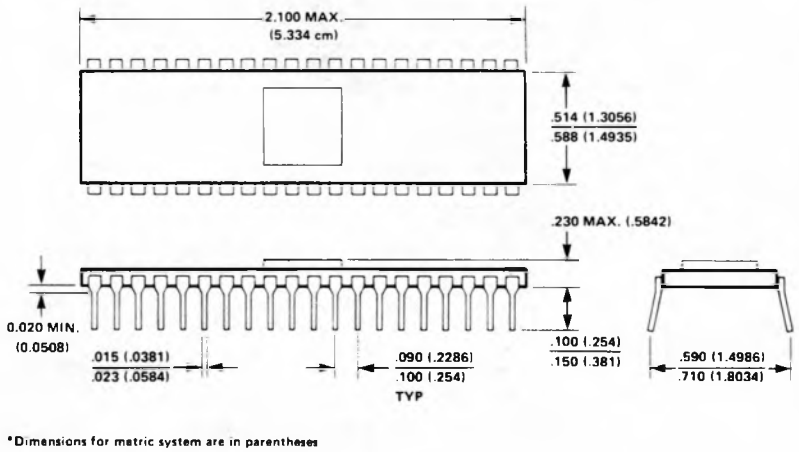
Clock Driver



An external clock pull-up resistor of (330Ω) will meet both the AC and DC clock requirements.

☐ Pin-Belegung





*Dimensions for metric system are in parentheses

☐ Bestellbezeichnung

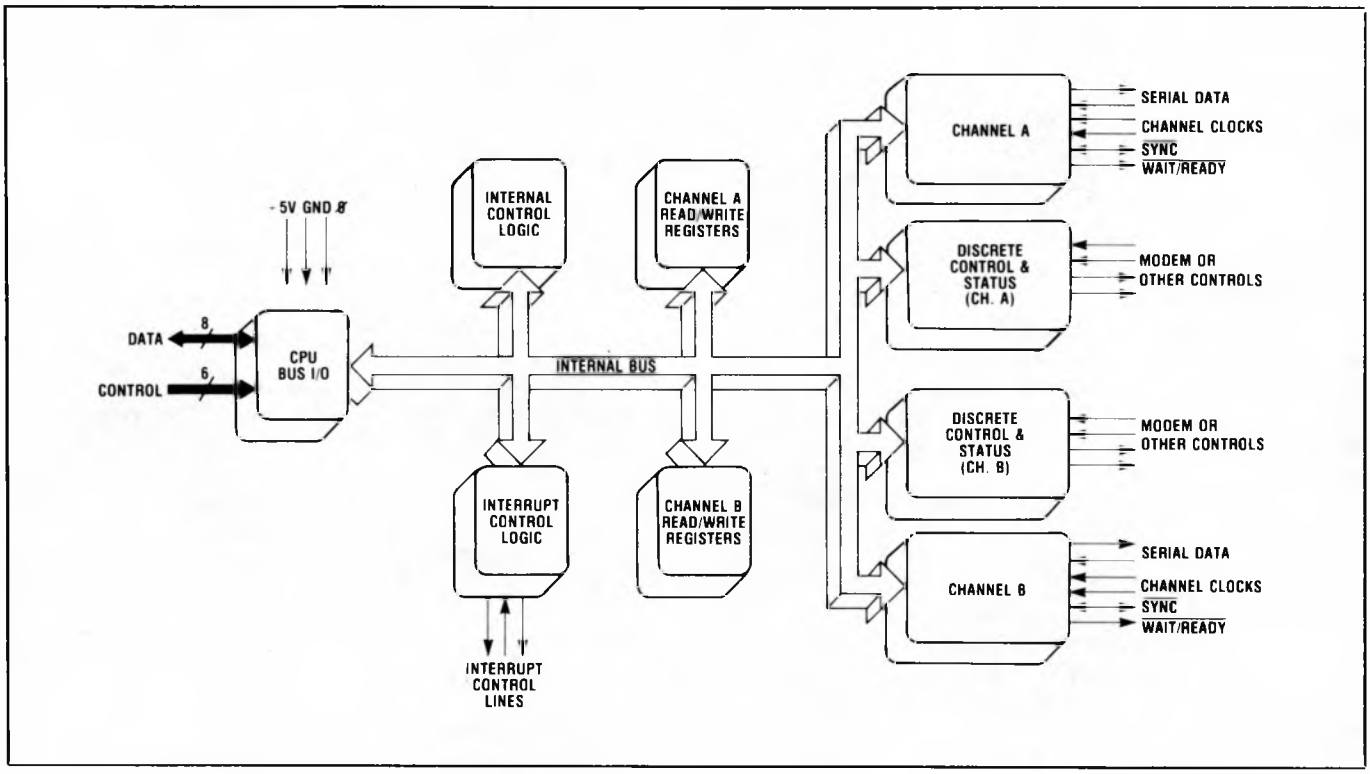
Z80A-DMA/PS Standardversion (0...70°C) in Plastikgehäuse

Z80A-DMA/CS Standardversion (0...70°C) in Kerami kgehäuse

Z80A-SIO

Zilog

UNIVERSELLER-SERIEN-EIN/AUSGABE-BAUSTEIN



1. ZILOG-Z80A MIKROCOMPUTER-BAUSTEINE

Der Baustein Z80A-SIO (**S**erial **I**nput/**O**utput) ist ein komplexer, programmierbarer, unsiverseller Baustein zur Abwicklung von seriellen Ein/Ausgabe-Aktivitäten in 8bit-Mikrocomputer-Systemen. Er beinhaltet 2 vollständige Duplex-Serien-Ein/Ausgabekanäle, also insgesamt 2 Eingabe- und 2 Ausgabekanäle.
Für Anwendungen, bei denen ein einziger Duplex-Serienkanal ausreicht, ist die Version SIO/9 verfügbar, bei der lediglich die Anschlüsse des Kanals A herausgeführt sind.
Er führt die Umwandlung von paralleler in serielle Information und umgekehrt durch, wobei er für asynchrone, synchrone und bitweise synchrone Datenübertragungen geeignet ist. Dadurch ist er u.a. in der Lage, Datenübertragungsprotokolle wie das IBM, BiSync, HDLC, SDLC und serielle Übertragungen nach anderen Verfahren abzuwickeln.
Bei jeder synchronen Datenübertragung kann der Baustein zyklische Redundanzprüfungssignale (CRC) erzeugen.
In der asynchronen Betriebsart ist er über Kommandowörter der CPU jedem asynchronen Übertragungsformat anzupassen.
Der Baustein eignet sich auch besonders als universeller Magnetplatten- (z.B. Floppy-Disk-) Controller.

☐ Aufbau

- N-Kanal Silocon Gate Depletion Load Technologie
- 40 pin-DIP-Gehäuse
- Eine einzige +5 V Speisespannungsversorgung
- Einphasen 5 V-Taktanschluß
- 2 volle Duplexkanäle.

☐ Eigenschaften

- Beide Duplexkanäle voneinander völlig unabhängig
- Übertragungsrate 0...880 kBit/sec bei 4 MHz Systemtakt
- 0...880 kBit/sec bei 4 MHz Systemtakt
- Datenempfangsregister: 4fache Pufferung
- Datensenderregister: 2fache Pufferung
- Asynchrone Betriebsweise:
  - ☐ 5, 6, 7 oder 8 Datenbits / übertragenem Zeichen
  - ☐ 1, 1½ oder 2 Stop-Bits
  - ☐ Gerade, ungerade oder keine Parität
  - ☐ x1, x16, x32 und x64 Taktzyklen-Betriebsart
  - ☐ Erzeugung und Prüfung von Break (= „Unterbrechungs"-) Bedingungen
  - ☐ Parity-Overrun und Wortformatfehler (= „Framing Error")-Prüfung
- Binär synchrone Betriebsweise:
  - ☐ Interne oder externe Zeichen-Synchronisation
  - ☐ 1 oder 2 Sync-Zeichen in getrennten Registern
  - ☐ automatisches Einfügen/Ausblenden von Sync-Zeichen
  - ☐ CRC-Erzeugung und Prüfung
- HDLC- oder IBM-SDLC-Betriebsart:
  - ☐ Null-Einfügung und -Ausblendung
  - ☐ Automatisches Flag-Einfügen
  - ☐ Adreßfeld-Erkennung
  - ☐ I-Feld-Residuum-Behandlung
  - ☐ Gültige Empfangs-Messages Overrun-geschützt
  - ☐ CRC-Erzeugung und -Prüfung
- getrennte Modem-Control-Eingänge und Ausgänge für beide Duplexkanäle
- Sowohl CRC 16 als auch CRC-CCITT (−0 und −1) sind implementiert.
- Modem-Status ist abfragbar.

☐ Architektur des SIO

Ein Blockschaltbild des Bausteins finden Sie auf der vorhergehenden Seite. Daraus sind die Hauptfunktionselemente des Bausteins ersichtlich:

- Zwei Vollduplex-Datenübertragungskanäle mit Parallel/Serien- und Serien/Parallel-Wandlern, bzw. ein solcher Vollduplex-Kanal bei SIO/9.
- Steuerlogik, Interruptlogik und das Interface zum Anschluß des Bausteins an die Mikrocomputer-Systembusse.

Die beiden Datenübertragungskanäle verfügen jeweils über fünf 8bit-Steuerregister, zwei 8bit-Statusregister und zwei 8bit Sync-Zeichen-Register.
Jeder Empfänger verfügt über drei 8bit-Pufferregister, die nach dem FIFO-Prinzip (= "first in; first out") organisiert sind. Jeder Sender hat über das 8bit-Ausgabeschieberegister hinaus 1 zusätzliches Puffer-Register. Die 16bit CRC-Generatoren/Prüfer sind in geeigneter Weise intern rückgekoppelte 16bit-Schieberegister, die über Kommandowörter per Programm für zwei verschiedene CRC-Codes programmiert werden können.
Die Interrupt-Steuerlogik bestimmt, welchem Baustein und welchem Kanal innerhalb des Bausteins die momentan höchste Priorität zugeordnet ist; dadurch ist automatische priorisierte Vektorinterrupt-Behandlung ohne jegliche zusätzliche Priorisierungs- oder Interruptcontroller-Bausteine möglich.
SIO-Intern hat Kanal A die höhere Priorität.
Innerhalb beider Kanäle gilt folgende Prioritätshierarchie: Empfänger, Sender, Extern/Status.
Der Interrupt-Vektor wird in ein zusätzliches Register des Kanal B geschrieben und kann über diesen Kanal auch von der CPU gelesen werden.

☐ Anschlußbeschreibung

Bezeichnung	Funktion	Kommentar
D_0—D_7	Datenbus	Bidirektionale Tri-State-Datenleitungen zum Anschluß an den System-Datenbus
B/$\overline{A}$	Kanal „B/A-Select"	Eingang zur Kanalauswahl (High bedeutet „Kanal B selektiert")
C/$\overline{D}$	„Control/ Data-Select"	Eingang Steuerwort/Datenübertragung (High bedeutet „Steuerwort übertragen)
$\overline{CE}$	„Chip enable"	Eingang, low-aktiv: Chip-Freigabe
$\overline{M1}$	„Machine-cycle 1"	Eingang, low-aktiv: Maschinenzyklus M1 oder CPU (Steuerbus-Signal)
$\overline{IORQ}$	„I/O-Request"	Eingang, low-aktiv: Ein/Ausgabe-Anforderung von der CPU
$\overline{RD}$	„Read"	Eingang, low-aktiv: Lesezyklus der CPU (Steuerbus-Signal)
Φ	„Clock"	Eingang: Systemtakt
$\overline{INT}$	„Interrupt-request"	Ausgang: low-aktiv: Interrupt-Anforderung von der Peripherie
IEI	„Interrupt enable In"	Eingang, high-aktiv
IEO	„Interrupt enable Out"	Ausgang, high-aktiv Mit diesen beiden Signalen wird die Interrupt-Priorisierungskette (Daisy Chain) gebildet.
$\overline{RESET}$	„Reset"	Eingang, low-aktiv: Sperrt sowohl Sender als auch Empfänger T×DA und T×DB werden in den aktiven Zustand gebracht, Modem Control in den High-Zustand. Nach dem Rücksetzen müssen die Steuerregisterinformationen neu eingeschrieben werden, bevor irgendeine Datenübertragung stattfindet. Sämtliche Interrupts werden gesperrt.

Bezeichnung	Funktion	Kommentar
W/RDYA W/RDYB*	„Wait/Ready"	1 Ausgang pro Kanal, der als „Ready-Leitung für den Anschluß von DMA-Controllern oder als „Wait"-Leitung zur Synchronisierung der CPU mit der Datenübertragungsrate programmiert werden kann; low-aktiv.
CTSA CTSB*	„Clear to Send"	1 Anschluß pro Kanal, Schmitt-Trigger-Eingänge, low-aktiv: In der Betriebsart „Auto-Enable" sperrt dieses Signal den Sender seines Kanals. Wird der Anschluß nicht zur Senderfreigabe verwendet, steht er als allgemeiner Eingang zur freien Verfügung. Die beiden Leitungen haben Schmitt-Trigger-Eingänge, sodaß auch Eingangssignale geringer Flankensteilheit einwandfrei verarbeitet werden.
DCDA DCDB*	„Data Carrier Detect"	1 Anschluß pro Kanal, Eingänge, low-aktiv: Die Funktion ist ähnlich der von CTS, jedoch wird von DCD der Empfänger des zugehörigen Kanals gesperrt.
RxDA RxDB*	„Receive Data"	1 Anschluß pro Kanal, Eingänge, high-aktiv: Serielle Dateneingänge
TxDA TxDB*	„Transmit Data"	1 Anschluß pro Kanal, Ausgänge, high-aktiv: Serielle Datenausgänge
RxCA RxCB*	„Receiver Clock"	1 Kontaktfleck pro Kanal, Schmitt-Trigger-Eingänge, low-aktiv: (bitte beachten Sie den untenstehenden Kontaktierungshinweis!) Empfängertakteingang; als Taktgeschwindigkeit in der asynchronen Betriebsart können ×1, ×16, ×32 oder ×64 programmiert werden.
TxCA TxCB*	„Transmitter Clock"	1 Kontaktfleck pro Kanal, Schmitt-Trigger-Eingänge, high-aktiv: (bitte beachten Sie den untenstehenden Kontaktierungshinweis!) Sendertakteingang; als Taktgeschwindigkeit sind hier ebenfalls ×1, ×16, ×32 oder ×64 möglich, jedoch müssen Sender- und Empfänger-Taktmultiplikator gleich sein.
RTSA RTSB*	„Request to Send"	1 Anschluß pro Kanal, Ausgänge low-aktiv: Sobald das RTS-Bit eines Kanals gesetzt ist, geht die zugehörige RTS-Leitung in den low-Zustand über. Wird das RTS-Bit in der asynchronen Betriebsart rückgesetzt, geht die zugehörige RTS-Leitung in den high-Zustand, sobald das Sender-Register leer ist. In der synchronen Betriebsart fungiert der Anschluß einfach als Ausgang, an dem dauernd der Wert des RTS-Bits liegt.
DTRA DTRB*	„Data Terminal Ready"	1 Kontaktfleck pro Kanal, Ausgänge, low-aktiv: (bitte beachten Sie untenstehenden Kontaktierungshinweis!) Ausgang, an dem der Wert des DTR-Bits liegt.
SYNCA SYNCB*	„External Character Synchronisation"	2 Ein/Ausgabe-Anschlüsse, low-aktiv: In der Betriebsart Externe Synchronisation beginnt das Aufbauen des Zeichens mit der steigenden Flanke von R×C, die direkt auf die fallende Flanke des Sync-Signals folgt. In der Betriebsart Interne Synchronisation sind diese Anschlüsse Ausgänge, die in dem Teil eines Taktzyklus, aktiv sind, in dem ein Sync-Zeichen erkannt wird. Die Sync-Bedingung ist nicht zwischengespeichert, sodaß der Anschluß zu jeder Zeit unabhängig von der Wortlänge bis Erkennen eines Sync-Bitmusters aktiv ist. In der asynchronen Betriebsart sind diese Anschlüsse einfach direkte Eingänge für die Hunt/Sync-Bits im Statusregister 0 und können zu beliebigen Eingabezwecken herangezogen werden. Bitte beachten Sie: Wird der Anschluß zur externen Synchronisation benützt, darf er frühestens 2 Taktzyklen nach der steigenden Flanke des R×C-Signals aktiv werden, bei dem das letzte Bit des Sync-Zeichens empfangen wurde. Diese Bedingung läßt sich im allgemeinen einfach erfüllen, indem man eine Änderung des SYNC-Signals ausschließlich bei der fallenden Flanke von R×C zuläßt.

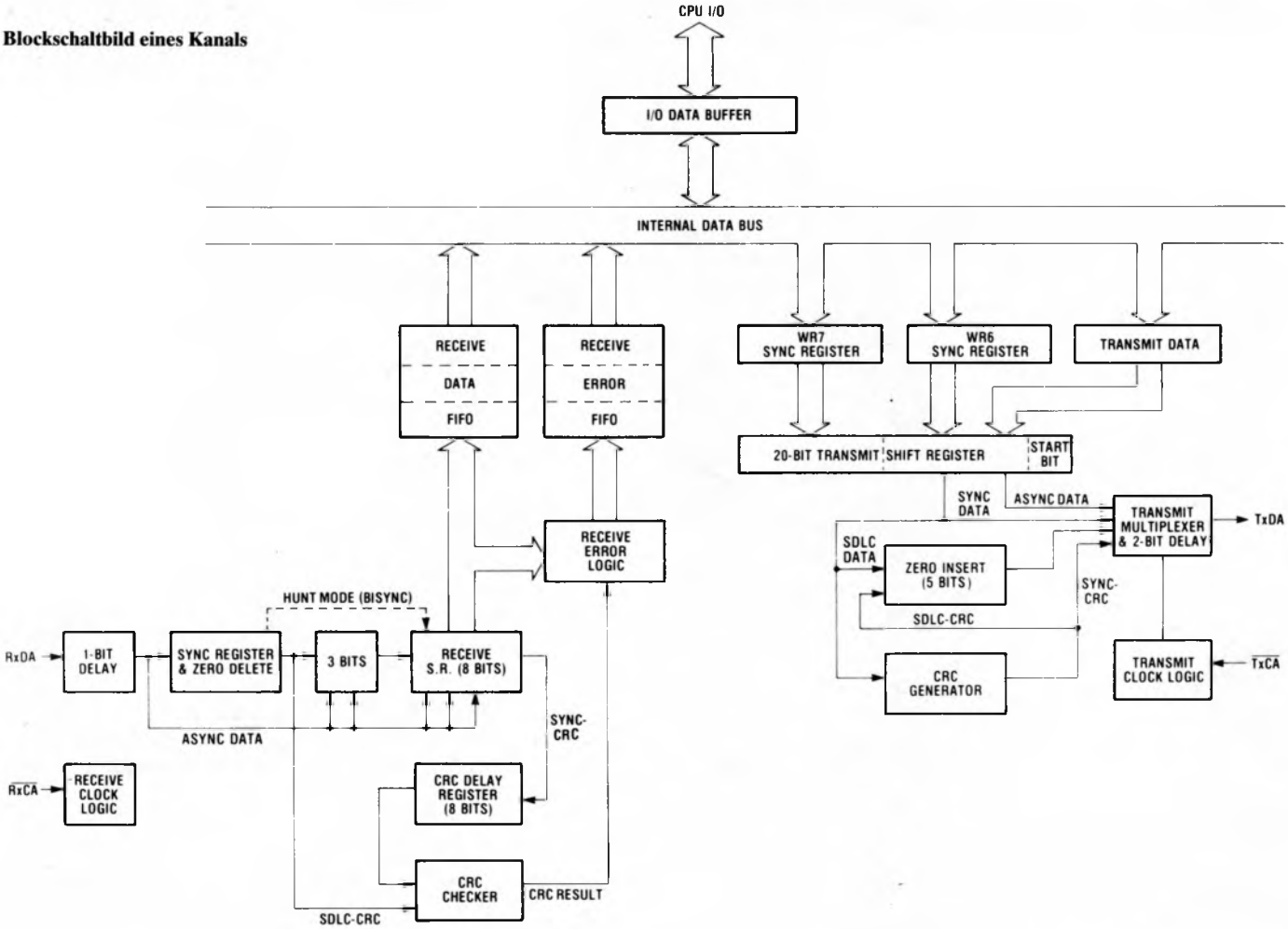
*: Nicht bei SIO/9

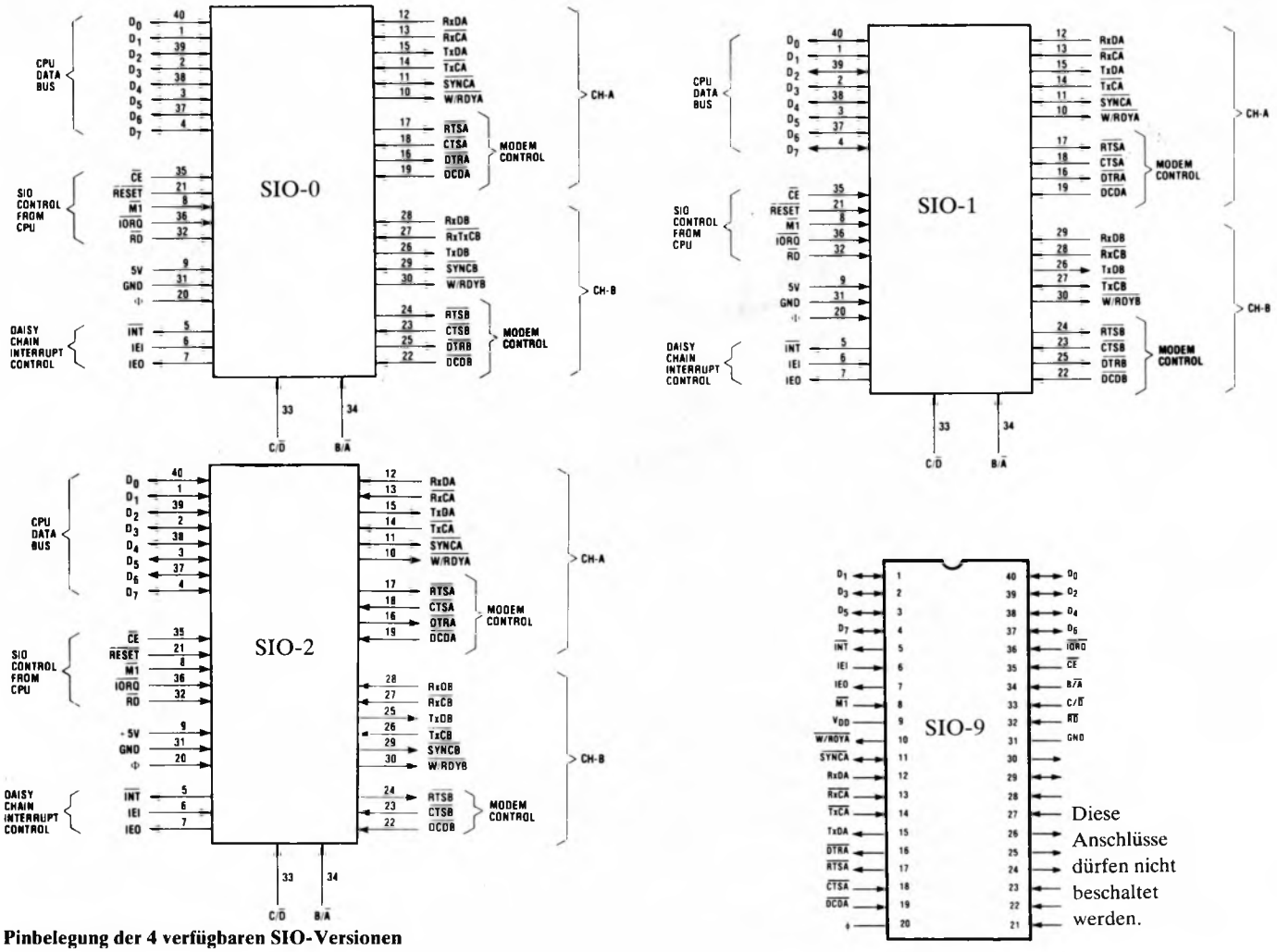
☐ Zur Beachtung*:

Aufgrund der begrenzten Anschlußzahl (40) stehen für die 4 Signale TxCB, RxCB Sync. und DTRB nur 3 pins zur Verfügung. Aus diesem Grund werden folgende 3 Kontaktierversionen des SIO angeboten:

- ☐ SIO/0: TxCB und RxCB sind miteinander verbunden
- ☐ SIO/1: DTRB ist nicht herausgeführt
- ☐ SIO/2: SYNCB ist nicht herausgeführt

Blockschaltbild eines Kanals



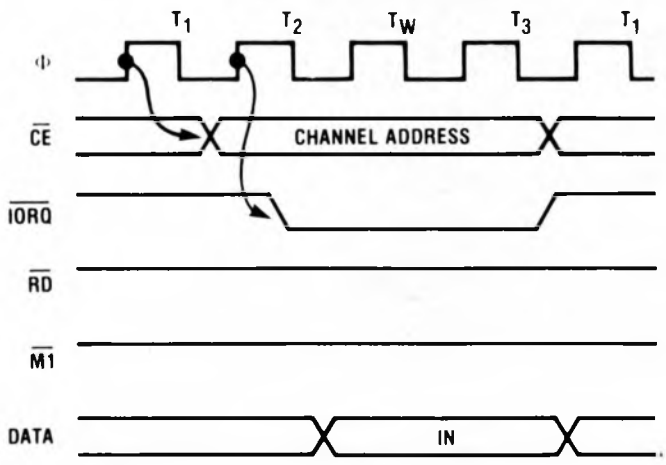


Pinbelegung der 4 verfügbaren SIO-Versionen

SIO-Zeitverhalten

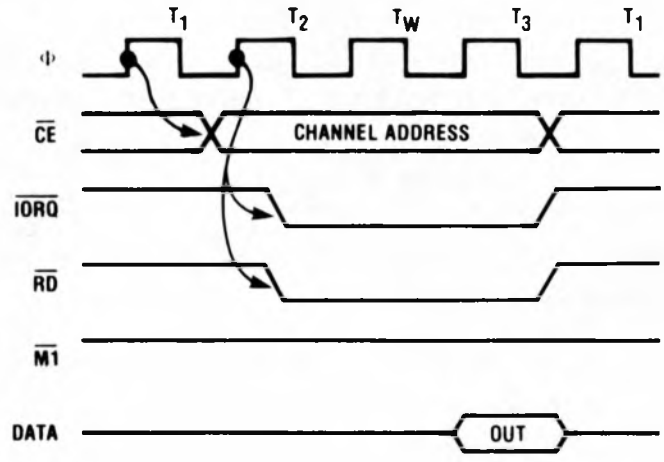
Schreib-Zyklus

Der nebenstehende Zyklus zeigt den Zeitablauf beim Schreiben eines Daten- oder Kontrollbytes in den SIO. Sämtliche Z80 Ausgabebefehle entsprechen diesem Zeitablauf.



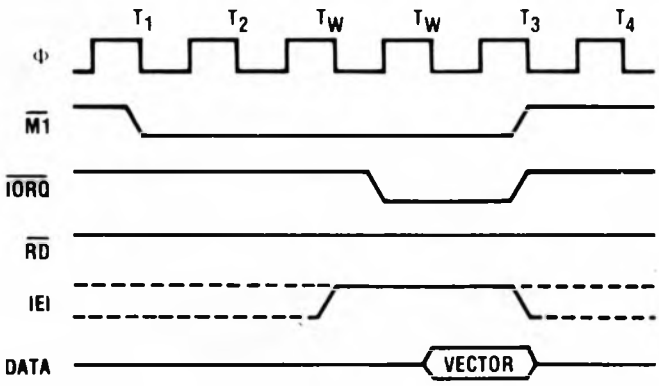
Lese-Zyklus

Der nebenstehende Zeitablauf zeigt das Lesen von Daten oder Statusinformationen aus dem SIO.
Sämtliche Z80 Eingabebefehle entsprechen diesem Zeitablauf.



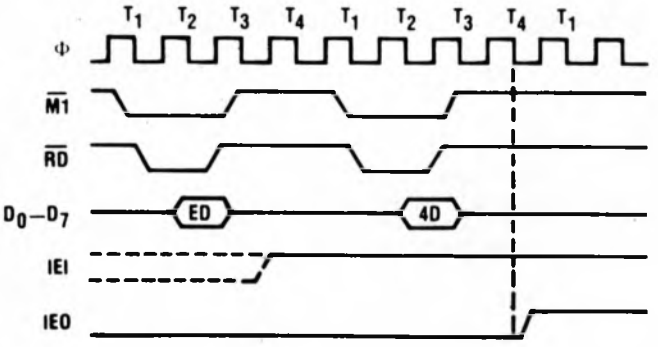
Interrupt-Quittungs-Zyklus

Bei der Annahme eines Interrupts vom SIO durch die CPU sendet diese eine Interrupt-Annahme-Bestätigung (M1 und IORQ). Die Interrupt-Logik des SIO bestimmt darauf die höchstpriorisierte Funktion, die einen Interrupt anmeldet. Um die Stabilisierung der Daisy-Chain-Leitungen zu garantieren, sind die Kanäle während des M1-Signals blockiert und können ihren momentanen Zustand nicht ändern. Der SIO plaziert den entsprechenden Interrupt-Vektor während des Signals IORQ auf den Datenbus.



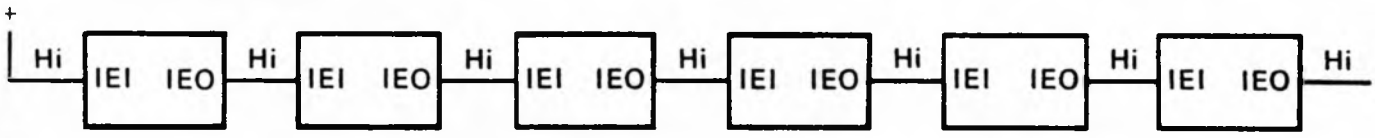
Rücksprung vom Interrupt-Zyklus

Falls ein Z80 Peripherie-Baustein weder einen Interrupt angemeldet hat noch seine Service-Routine laufen hat, gilt IEO = IEI. Bei der momentanen Abarbeitung einer Interrupt-Service-Routine liegt der Ausgang IEO auf null und blockiert die nachfolgenden Peripheriebausteine. Falls ein Interrupt angemeldet, aber noch nicht bestätigt ist, liegt IEO auf null bis der Code „ED" als erstes Byte eines 2-Byte-Befehls decodiert wird. Zu diesem Zeitpunkt liegt IEO auf 1 bis zur Decodierung des nächsten Befehls-Bytes. Anschließend geht IEO auf null. War das 2. Byte des Befehls „4D", so entspricht es einem RETI-Befehl. Nach der Decodierung des Befehlsbytes „ED" hat nur der Peripheriebaustein IEI auf eins und IEO auf null, dessen Interrupt-Anmeldung akzeptiert wurde. War das nächste Byte ein „4D" geht IEO wieder auf eins.



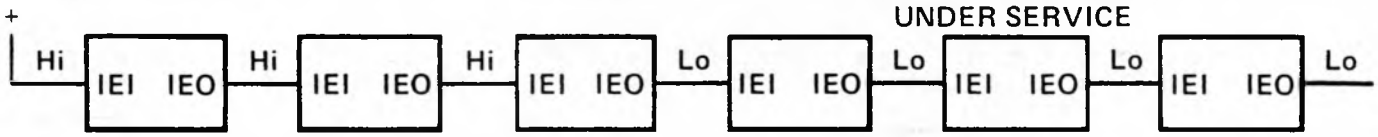
Das folgende Bild erläutert die Priorisierung der einzelnen interruptanfordernden Funktionaleinheiten innerhalb des SIO. Selbstverständlich kann jederzeit eine Kaskadierung mit weiteren SIO's oder anderen Peripherie-Bausteinen der Z80-Familie erfolgen.

Kanal A Empfänger Port	Kanal A Sender Port	Kanal A Extern/ Status Port	Kanal B Empfänger Port	Kanal B Sender Port	Kanal B Extern/ Status Port

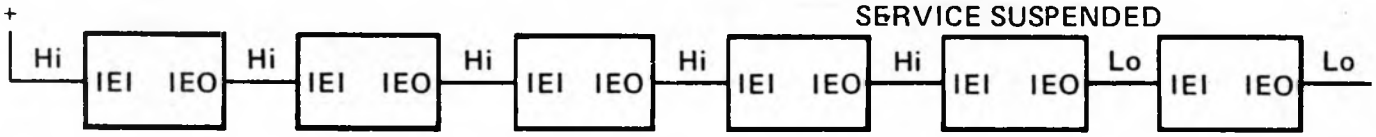


1. Daisy Chain vor irgendeinem Interrupt

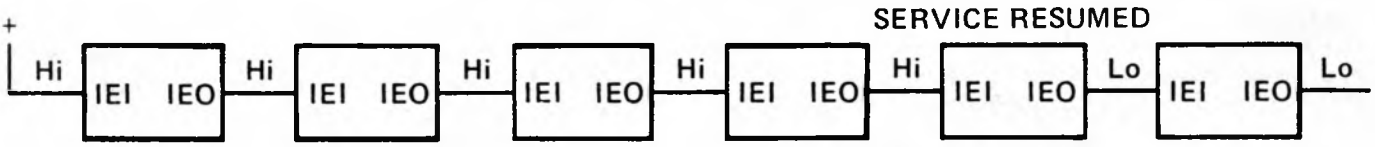
Kanal A Empfänger Port	Kanal A Sender Port	Kanal A Extern/ Status Port	Kanal B Empfänger Port	Kanal B Sender Port	Kanal B Extern/ Status Port



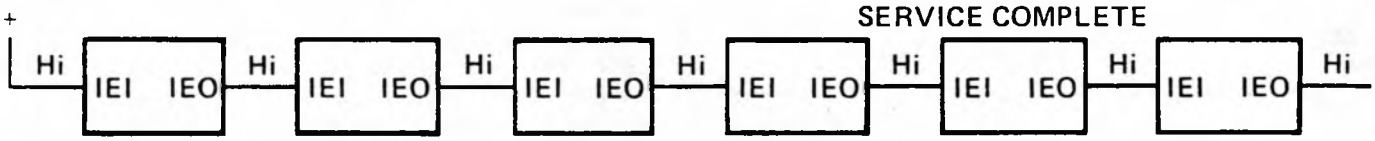
2. Sender port von Kanal B fordert Interrupt an und erhält Bestätigung.



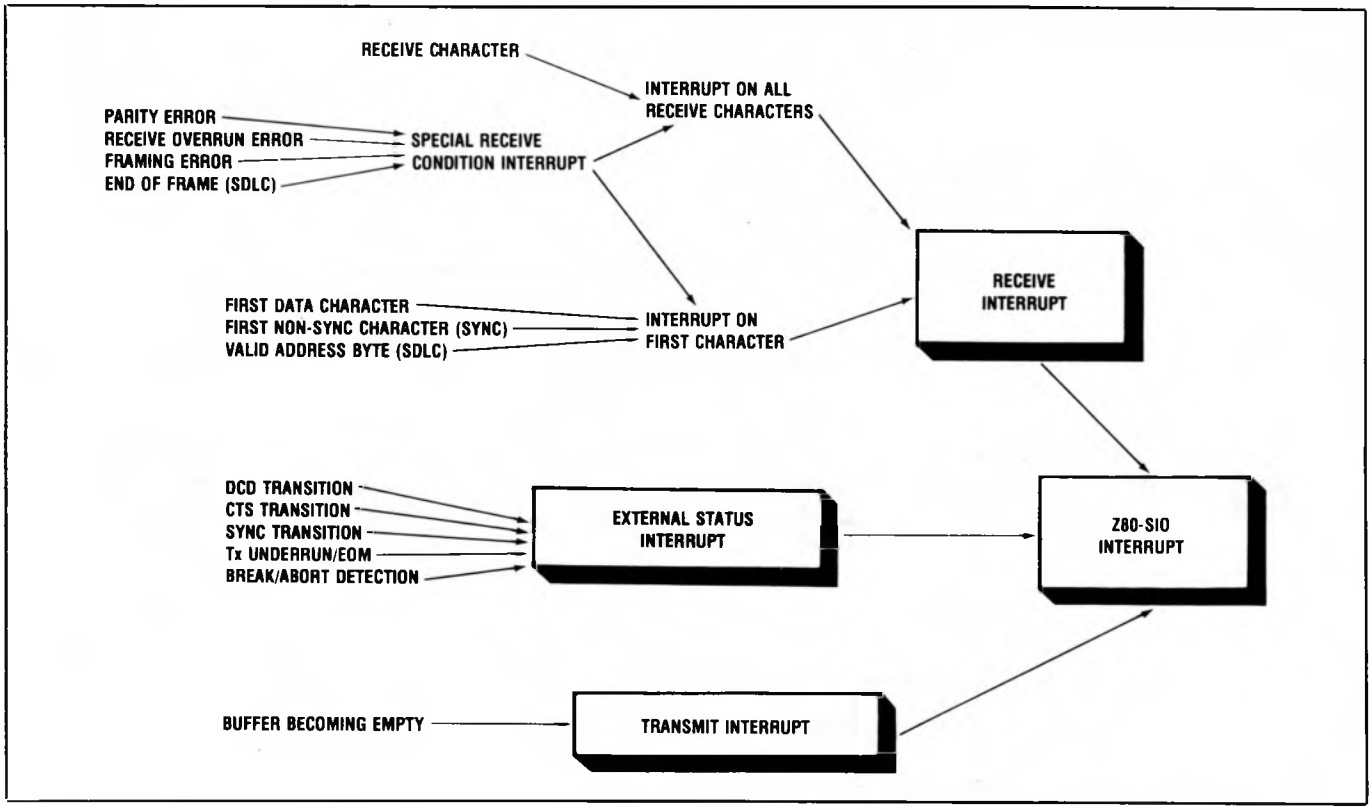
3. Extern/Status-Port von Kanal A fordert Interrupt an und unterbricht die Bedienung von Kanal B.



4. Bedienroutine für Extern/Status-Port von Kanal A ist fertig. Die RETI-Anweisung wird auf den Bus gelegt und die Bearbeitung der Service-Routine für das Senderport von Kanal B wieder aufgenommen.



5. Service-Routine für das Sender-Port von Kanal B ist fertig, zweite RETI-Instruktion liegt auf dem Bus.



Arbeitsweise des SIO

Das Verhalten des SIO wird durch den Inhalt seiner Steuerregister bestimmt. Bevor der SIO einen Datentransfer durchführen kann, müssen seine sämtlichen Steuerregister programmiert werden, wobei einige Kommandos und Verhaltensweisen des SIO auch während des Programmablaufes modifiziert werden können. Die Statusregister sind zu jedem Zeitpunkt auslesbar.

Asynchronbetrieb
Die Eingangskanäle sind 4-fach gepuffert, d.h. es existieren 3 Speicherregister zusätzlich zum Eingabeschieberegister. Die CPU erhält damit Zeit für den Start einer Interrupt-Service-Routine, auch wenn der zu übernehmende Datenblock mit hoher Geschwindigkeit eintrifft. Fehlerbits sind ebenfalls 4-fach gepuffert und werden zur selben Zeit wie die Zeichen geladen. Die Flags-, Empfängerüberlauf- und Paritätsfehler (Receiver Overrun und Parity-Error) werden nur durch einen Error Reset-Befehl (Kommando 6) rückgesetzt. Rahmenfehler und CRC-Fehler zeigen den Status des momentan im Puffer befindlichen Zeichens an. Beim Auslesen des Fehlerstatus zeigt dieser Fehler des momentan im Empfangsregister befindlichen Zeichens sowie irgendwelche Paritäts- oder Überlauffehler, die seit den letzten Fehlerrücksetzkommandos aufgetreten sind, an. Um die Übereinstimmung zwischen dem Zustand des Fehlerregisters und dem Inhalt des Empfangsregisters aufrechtzuerhalten, sollte das Statusregister vor den Daten gelesen werden. Bei vektorisiertem Interrupt wird wegen der Erzeugung eines speziellen Interruptvektors bei Auftreten von Fehlern diese Bedingung leicht eingehalten.
Eine Ausnahme tritt auf bei der Auswahl des Arbeitsmodus „Empfänger-Interrupt nur beim 1. Zeichen". Bei diesem speziellen Interrupt werden sowohl der Fehler als auch das Zeichen so lange gespeichert, bis das Fehlerrücksetzkommando ausgeführt wird. Dadurch wird die Übernahme von neuen Daten bis zur Übernahme der Fehlermeldung verhindert.
Wird der Modus „Interrupt bei jedem Zeichen" gewählt, ändert sich der Interrupt-Vektor abhängig von auftretenden Fehlern. Bei Überlauf wird das letzte eintreffende Zeichen übernommen und das vorhergehende Zeichen geht verloren.
Beim Lesen dieses Zeichens, welches über vorhergehende Zeichen geschrieben wurde, ist das Überlauf-Bit gesetzt und der Interrupt-Vektor „Special Receive Condition" wird geliefert („Status Affects Vector" muß aktiviert sein). Es ist möglich, den SIO in einem Polling-System zu verwenden. Hier muß das Bit „Receive Character Available" als Empfangsmeldung geprüft werden. Sind alle Empfangspufferregister leer, wird dieses Bit automatisch rückgesetzt. Das Bit „Transmit Buffer Empty" muß bei Polling-Systemen geprüft werden, bevor neue Daten in den Sender geschrieben werden können.

Senden
Im Asynchron-Betrieb sendet der SIO ein Zeichen in folgender Form; Im Ruhezustand liegt die Ausgangsleitung auf 1, vorausgesetzt, daß kein „BREAK" programmiert wurde. In diesem Fall liegt die Leitung auf 0 bis das „Send-Break"-Kommando gelöscht und der Baustein rückgesetzt wurde.
Senden kann erst nach dem Setzen des Bits „Transmit Enable" beginnen. Bei der Wahl der Option „Auto Enables" muß der Eingang CTS auf null liegen. Werden die Zeichen mit 5 Bit codiert, müssen die nicht verwendeten Bits D_5, D_6 und D_7 in den eingeschriebenen Datenbytes null sein.

Empfang
Asynchroner Empfang wird nach dem Setzen des Bits „Receiver Enable" möglich. Bei der Wahl der Option „Auto Enables" muß der Eingang DCD auf null liegen. Ein Übergang von 1 auf 0 am RxD-Eingang markiert ein Startbit. Beim Anliegen von mindestens einer halben Bit-Länge wird dieses Startbit als gültig betrachtet und die eintreffenden Datenbits werden so lange zu ihrem mittleren Zeitpunkt abgetastet, bis das vollständige Zeichen übernommen ist. Diese Form der Erkennung eines Startbits verbessert die Fehlererkennung beim Auftreten von Störspitzen auf der Empfangsleitung.

Synchronbetrieb
Alle Formen der synchronen Datenübertragung benötigen einen ungeteilten Takt (x1) für Senden und Empfang. Daten werden zum Zeitpunkt der steigenden Flanke von RxC abgetastet. Sendedatenänderungen erfolgen während der fallenden Flanke von TxC. In allen Fällen befindet sich der Empfangsteil in einem Suchlauf (nach einem internen oder externen Reset). Der Suchlauf startet nur nach der Aktivierung des Empfängers und der eigentliche Datentransfer benötigt eine vorhergehende Zeichensynchronisierung. Nach einem Synchronisierungsverlust kann der Suchlauf durch Schreiben eines Steuerwortes neu gestartet werden, wenn das „Enter Hunt Mode"-Bit gesetzt war. Die Unterschiede der Übertragungsformen, Monosync, Bisync und externe Synchronisierung, ergeben sich nur durch die Form der Erstsynchronisation.

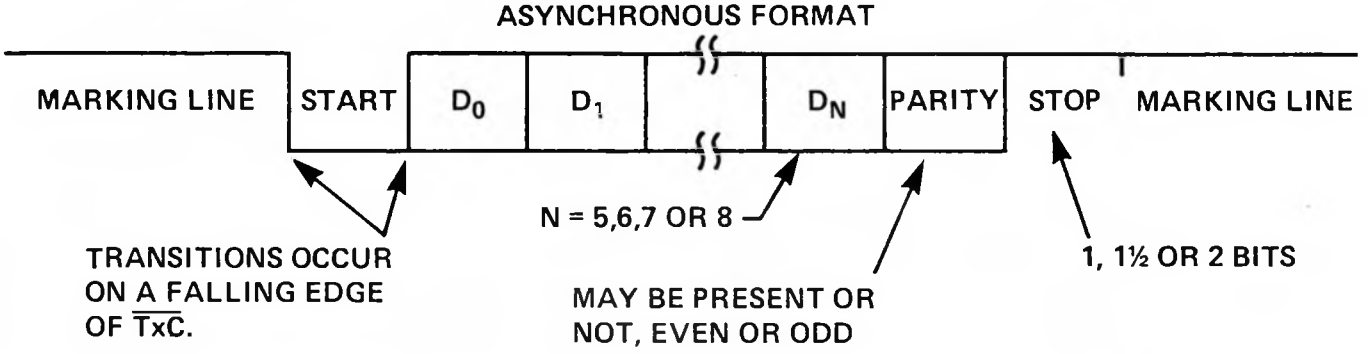
Monosync (8 Bit-Synchronisierung)
Beim Erkennen eines einfachen Synchronzeichens (programmiert in Steuerregister 7) beginnt der Datentransfer.

Bisync (16 Bit-Synchronisierung)
Beim Erkennen von zwei aufeinanderfolgenden Synchronzeichen (programmiert in Steuerregister 6 und 7) beginnt der Datentransfer.
In beiden Fällen (monosync und bisync) geht die SYNC-Leitung beim Erkennen einer Synchronzeichenfolge auf null. Sie bleibt null während des gesamten folgenden Taktimpulses.

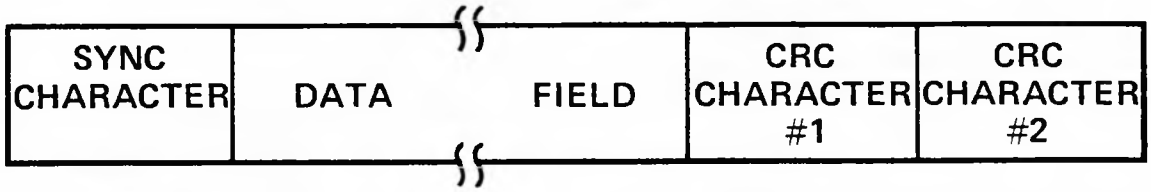
Externe Synchronisierung
Datentransfer beginnt mit der ersten ansteigenden Flanke von RxC nach Aktivierung des Sync-Einganges (aktiv = 0). Der Sync-Eingang sollte während eines kompletten Taktes aktiv bleiben.
Bei allen drei Varianten wird die Übernahme von Daten bis zum Auftreten folgender Ereignisse durchgeführt:

- Externe oder interne Rücksetzung des SIO
- Blockierung des Empfangskanals (per Kommando oder DCD)
- Setzen des Bits „Enter Hunt Mode"

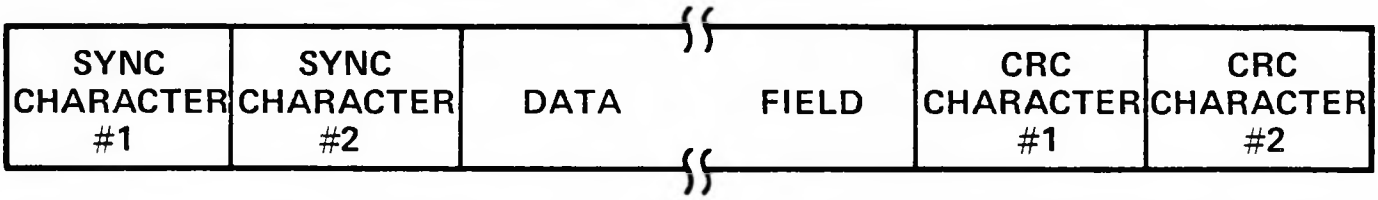
Nach der Erstsynchronisierung unterscheiden sich die drei Varianten nur mehr durch die nachfolgenden Formate:



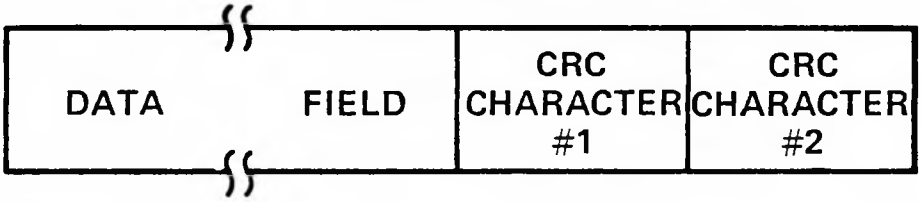
MONOSYNC MESSAGE FORMAT (Internal Sync Detect)



BISYNC MESSAGE FORMAT (Internal Sync Detect)



EXTERNAL SYNC DETECT FORMAT



Senden in Synchronbetrieb (ohne SDLC)

a) Ruhezustand
Im Grundzustand ist die Leitung aktiv (nach Reset oder blockiertem Sendekanal). Break kann zur Erzeugung des Zustandes null auf der Sendeleitung programmiert werden. Der Zustand beginnt im Moment der Programmierung unabhängig vom Inhalt des Senderegisters. Nach der Aktivierung des Sendekanals und der Auswahl der Synchronübertragungsform werden automatisch 8 oder 16 Bit-Synchronzeichen ausgesandt.

b) Verschiedene Interrupt-Aufrufe sind möglich:
- Sendeinterrupt aktiv
  Falls das „Sendeinterrupt Aktiv"-Bit gesetzt ist, wird nach Ausgabe jedes Zeichens (leeres Senderegister) ein Interrupt erzeugt. Dieser Interrupt kann durch Schreiben eines neuen Zeichens in den Sender oder durch Kommando 5 (Reset Transmitter Interrupt Pending) abgearbeitet werden. Bleibt das Senderegister leer, werden keine weiteren Interrupts erzeugt. Nach Schreiben eines neuen Zeichens wird das Senderegister erneut leer und erzeugt den nächsten Interrupt.
- Extern/Statusinterrupt aktiv
  Nach Setzen des Bits „Extern/Statusinterrupt aktiv" können verschiedene Sendebedingungen (Senden von CRC-Zeichen, Senden von SYNC-Zeichen, Änderung von CTS) einen Interrupt erzeugen.
- Alle Interrupts können bei Anwendung in einem Polling-System verhindert werden.

c) Falls CRC inaktiv ist, werden Sync-Zeichen automatisch vom Sender dann ausgeschickt, wenn keine Daten für den Transfer zur Verfügung stehen. Bei aktiviertem CRC wird in der ersten Datenpause das 16 Bit-CRC ausgesandt, gefolgt von Sync-Zeichen.
Während des Aussendens des CRC ist das Bit „Sending CRC/SYNCS" gesetzt und das Bit „Transmit Buffer Empty" zeigt ein volles Senderegister an. CRC wird nicht während des automatischen Aussendens von Sync-Zeichen berechnet.
Kontrolle des CRC-Generators:
Rücksetzen erfolgt mit dem Kommando „Reset Transmit CRC Generator". Nach dem Aktivieren des CRC und des Sendeteils können Daten geladen werden. Vor dem Aussenden von CRC (aber nach dem Laden der ersten Daten) muß das „CRC/SYNC SENT/SENDING" Flag mit dem Kommando „RESET CRC/SYNC SENT/SENDING" rückgesetzt werden.

d) Wird der Sender während der Ausgabe eines Zeichens deaktiviert, wird zwar das Zeichen vollständig ausgegeben, anschließend folgen aber weder CRC noch Sync-Zeichen. Ein im Puffer vorhandenes Zeichen bleibt dort erhalten. Wird während des Aussendens eines Zeichens ein BREAK aktiviert, geht das Zeichen verloren.

e) Grundsätzlich erfolgt die Reihenfolge der Aussendung der Bits vom untersten zum obersten. Daher müssen alle Daten (bei einer kleineren Wortlänge als 8 Bit) nach rechts justiert werden. Bei einer Wortlänge von 5 Bit und darunter muß eine Spezialtechnik (Sektion „Transmit Bits/Char") angewandt werden.

Empfang (außer SDLC)

a) Nach der Programmierung der Übertragungsart und der Sync-Zeichen (in dieser Reihenfolge) kann der Empfänger aktiviert werden. Er geht dann in den Suchlauf und bleibt dort so lange bis einer der folgenden Zustände eintritt:

1) Empfang eines einfachen Sync-Zeichens (Monosync)
2) Empfang eines doppelten Sync-Zeichens (Bisync)
3) Der externe Sync-Eingang geht auf null

In den Fällen 1 und 2 ist die Sync-Leitung ein Ausgang, der die Zeichensynchronisierung meldet. Im Fall 3 ist Sync ein Eingang.

b) Zeichenübernahme beginnt nach Erkennung eines entsprechenden Sync-Zeichens. 4 Formen des Interrupts sind möglich:

1) Interrupt deaktiviert für ein Polling-System oder im „Off-line"-Betrieb

2) Interrupt nur beim ersten Zeichen
Diese Form startet normalerweise eine Abfrageschleife oder einen Block-Transferbefehl, wobei mit Hilfe des „Wait/Ready"-Ausgangs die CPU auf die ankommende Datentransfergeschwindigkeit synchronisiert werden kann. Ebenso kann der SIO mit einem DMA-Baustein verbunden werden, wobei der SIO einen Interrupt beim ersten Zeichen aber anschließend nur mehr bei Auftreten von Fehlern liefert. Diese Interrupt-Form wird mit dem Kommando 4 (Reset Receive Interrupt On First Character) rückgesetzt. Das erste Zeichen nach diesem Kommando erzeugt noch einen Interrupt. Sind externe Statusinterrupts aktiviert, können sie jederzeit auftreten. Paritätsfehler erzeugen keinen Interrupt, jedoch Rahmenfehler und Empfängerüberlauf.

3) Interrupt bei jedem Zeichen
Bei jedem Empfang eines Zeichens wird ein Interrupt erzeugt. Fehler und spezielle Bedingungen erzeugen einen Spezialvektor falls die Form „Status Affects Vektor" aktiv wird. Die Erzeugung eines speziellen Vektors beim Auftreten eines Paritätsfehlers kann optional unterdrückt werden.

c) CRC-Erzeugung und -Kontrolle

1) Die Berechnung eines CRC eines bestimmten Zeichens beginnt 8 Bit-Zeiten bevor das Wort in den Empfänger transferiert wird. Wird der CRC vor dem Transfer des nächsten Zeichens aktiviert, berechnet sich der CRC aufgrund dieses Zeichens. Wird der CRC vor dem nächsten Transfer desaktiviert, erfolgt die Berechnung noch für das momentan verarbeitete Zeichen, aber nicht mehr für das nachfolgende.

2) Der CRC kann so oft wie nötig für eine vorgegebene Berechnung aktiviert und desaktiviert werden.

3) CRC-Codes werden während des Auswahlvorganges der Übertragungsart programmiert. Entweder das CRC 16 Polynom $x^{16}+x^{15}+x^2+1$ oder das SDLC-Polynom $x^{16}+x^{12}+x^5+1$ kann gewählt werden. In allen Fällen außer SDLC wird der CRC-Generator und Kontroller auf null rückgesetzt. Sender und Empfänger müssen dasselbe Polynom verwenden.

4) In Monosync. Bisync und externer Synchronisation enthält das CRC/FRAMING ERROR Bit das Ergebnis der CRC-Kontrolle. Sie erfolgt 16 Bit-Zeiten nach dem Transfer des Zeichens vom Empfangsschieberegister zum Puffer. Eine fehlerfreie Übertragung ist durch null gekennzeichnet. Der Vergleich wird bei jedem Transfer durchgeführt und ist nur gültig, so lange das Zeichen im Empfangs-FIFO enthalten ist.

SDLC-Senden

a) Üblicherweise wird der CRC-Generator vor der Übertragung von Datenblocks rückgesetzt (mit dem Kommando „Reset Transmit CRC Generator"). Dieses Kommando kann jederzeit nach dem Aussenden des CRC der vorhergehenden Meldung erfolgen. Während des Aussendens des CRC ist das Bit „CRC/SYNC SENT/SENDING" gesetzt, jedoch das Bit „Trans Buffer Empty" ist nicht gesetzt. Nach dem Aussenden des CRC wird das „Trans Buffer Empty"-Bit gesetzt. Falls Sendeinterrupts aktiviert sind, zeigt ein stehender Interrupt die vollständige Aussendung des CRC an.

b) Im Ruhezustand werden kontinuierliche Flags gesendet (Sender aktiv), ansonsten wird eine Marke ausgesandt (Log. 1).

c) Eine Abbruchsequenz kann durch das Kommando 1 (Send Abort) erzeugt werden. Dadurch werden 8 bis 14 1-Bits ausgesandt, dann folgen Flags. Alle Sendedaten im Senderegister und Puffer gehen verloren.

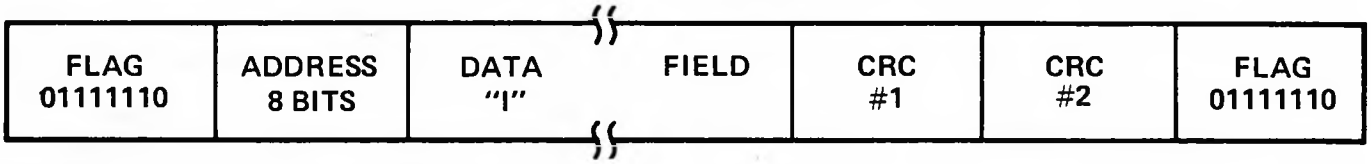
d) Ein bis 8 Bit pro Zeichen können gesendet werden (siehe Beschreibung des Steuerregister 5). Da die Anzahl der Bits pro Zeichen jederzeit geändert werden kann, ist das Füllen eines Datenfeldes mit einer beliebigen Anzahl von Bits möglich. In Verbindung mit den Receiver-Residue-Codes ist der SIO imstande, eine Meldung beliebiger Bit-Anzahl zu empfangen und sie in exakt gleicher Form rückzusenden ohne zusätzliche Information über die Zeichenstruktur des Datenfeldes (I-Feld). Ein Wechsel in der Zahl der Bits pro Zeichen beeinflußt nicht das momentan gesendete Zeichen. Zeichen werden mit der Anzahl von Bits ausgesandt, die im Moment der Übertragung des Zeichens vom Puffer zum Senderegister programmiert sind.

e) Sind keine Daten zum Senden verfügbar, wird automatisch die 2 Byte CRC-Sequenz ausgesandt. Beim Aussenden des CRC wird das Bit CRC/SYNCS SENT/SENDING gesetzt und ein Interrupt (Statuswechsel) erzeugt, sofern Extern/Status Interrupt aktiviert ist. Diese Information kann zur Anzeige des leeren Senders verwendet werden. Nach dem Aussenden des CRC werden kontinuierlich Flags gesendet. Die Kontrolle des CRC-Generators kann in folgender Weise stattfinden:
1) Auswahl der Übertragungsart
2) Rücksetzen des CRC-Generators
3) Schreiben der ersten zwei Datenbytes
4) Rücksetzen des Bits „CRC/SYNCS SENT/SENDING"
5) Schreiben der restlichen Daten
6) Nach der vollständigen Aussendung der Daten werden CRC und Flags automatisch gesendet und der Ablauf kann ab Punkt 2 wiederholt werden.

f) Null-Bits können automatisch in den Datenstrom eingefügt werden, um die Bedingung von max. 5 Eins-Bits in einer Reihe zu erfüllen (gilt nicht für Flags oder Datentransfer-Abbruch).

g) Bei der Auswahl der SDLC-Datenübertragung ergibt das Rücksetzen des CRC-Generators tatsächlich Eins-Bits. Der SDLC/CRC-Code muß eigens gewählt werden.

SDLC/HDLC Message Format



SDLC-Empfang

a) Der Datentransfer beginnt mit dem ersten Zeichen nach mindestens einem Flagzeichen (01111110), sofern Adress-Search-Mode nicht aktiv ist. Bei Aktivierung muß dem Flag entweder die Standard-Adresse (11111111) oder eine programmierte folgen, bevor Datentransfer möglich ist.

1) Falls Interrupts inaktiv sind, kann das Vorhandensein eines Zeichens durch Prüfen des Bits „Receive Character Available" erkannt werden.

2) Falls „Interrupt On First Character Only" gewählt ist, wird damit meist ein Block-Transfer aktiviert. Bei unbekannter Datenblock-Länge kann der „Special Condition"-Interrupt (End of Frame) zum Ausstieg aus der Software-Schleife verwendet werden.
Vor Einlangen des nächsten Datenblocks muß das Kommando 4 „Reset Interrupt On First Character" erfolgen bevor ein neuerlicher Interrupt mit dem ersten Zeichen des nachfolgenden Blocks durchgeführt werden kann.

3) Flags werden nicht übernommen. Die Nullen, die während der Übertragung eingeschaltet wurden, werden automatisch gelöscht.
4) Abbrüche werden als 7 oder mehr Einsen entdeckt und erzeugen einen Statusinterrupt (falls aktiviert) und setzen das „Break Abort"-Bit in Statusregister 0. Nach Ausführung des Kommandos 2 „Reset External/Status Interrupts" wird ein zweiter Interrupt nach Beendigung der Sendung von kontinuierlichen Einsen durchgeführt.

b) Im SDLC-Format ist die Kontrolle des Empfangs-CRC-Generators automatisch. Er wird mit der führenden Flag rückgesetzt und CRC wird berechnet bis zur letzten Flag. Das Byte, welches das „End of Frame"-Bit setzt, ist gleichzeitig das Byte, welches das Ergebnis der CRC-Kontrolle enthält. Ist das Bit „CRC/Framing Error" nicht gesetzt, zeigt der CRC eine gültige Meldung an. Bei der SDLC-Kontrolle muß das Endergebnis 0001110100001111 sein.

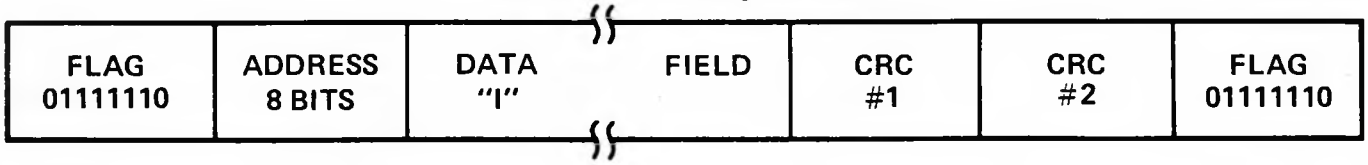
c) Zeichenbitlänge kann während des Ablaufs geändert werden. Falls Adreß- und Steuerbit als 8 Bit-Zeiten transferiert werden, kann der Empfänger während der Übernahme des ersten Informationszeichens auf eine kürzere Zeichenbit-Länge geändert werden.

d) Falls Adreß-Suchlauf nicht gewünscht wird oder Sendungen mit Mehr-Byte-Adressen eintreffen, muß die ungewünschte Meldung nicht komplett übernommen werden. Das Setzen des Bis „Enter Hunt Mode" verhindert die Übernahme von Meldungen bis eine neue Meldung (mit einem Flag am Anfang) empfangen wird.

e) Beim Empfang des führenden Flags wird ein Interrupt mit einem Spezialvektor erzeugt (falls aktiviert). Damit wird der Empfang des Bytes mit einem gesetzten Bit „End of Frame" signalisiert (zuzüglich zum Ergebnis der CRC-Kontrolle). Statusregister 1 enthält 3 Bits des Residue-Codes, der im Moment gültig ist. Im Falle, daß die Anzahl der Bits im I-Feld kein ganzzahliges Vielfaches der Zeichenlänge ist, geben diese Bits den Zwischenraum zwischen den CRC-Steuerbits und den I-Feld-Bits an (siehe Statusregister 1).

f) Paritätskontrolle kann bei Daten im I-Feld nur bei 5 bis 7 Bit-Zeichen und nur bei Halb-Duplex-Protokollen durchgeführt werden.

SDLC/HDLC Message Format



Programmieren des SIO

Allgemeines

Der Z80-SIO ist eine Multifunktions-Peripheriekomponente, die ein weites Feld von seriellen Datenübertragungsprotokollen überstreicht. Seine grundlegende Funktion ist die eines seriell/parallel, parallel/seriell Umsetzers, aber innerhalb dieser Funktion kann man mit Hilfe von Programmbefehlen ihn auf die jeweilige Anwendung festlegen. Zur Programmierung benötigt der SIO eine Serie von Kommandos, die die gewünschte Grundfunktion einstellen und andere Kommandos, die innerhalb der gewählten Betriebsart bestimmte Bedingungen auswählen. Jeder der 2 Kanäle des SIO enthält 8 Steuerregister, die vom System vor der Durchführung eines gewünschten Datentransfers programmiert werden müssen. Der Kanalauswahleingang (B/$\overline{A}$) und der Steuerdateneingang (C/$\overline{D}$) werden über den Adreßbus der Z80-CPU gesteuert.

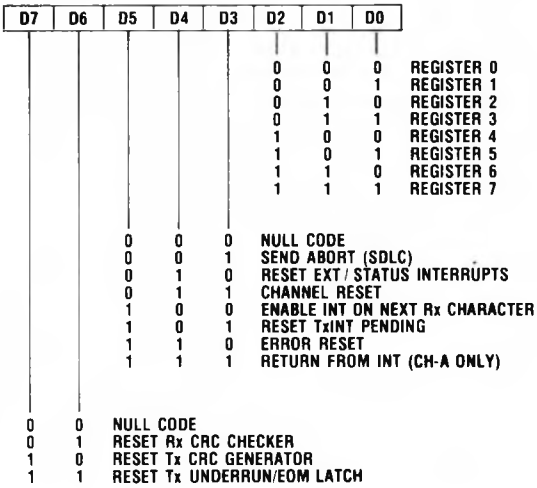
C/$\overline{D}$	B/$\overline{A}$	Funktion
0	0	Kanal A Daten
0	1	Kanal B Daten
1	0	Kanal A Kommando/Status
1	1	Kanal B Kommando/Status

Steuerregister

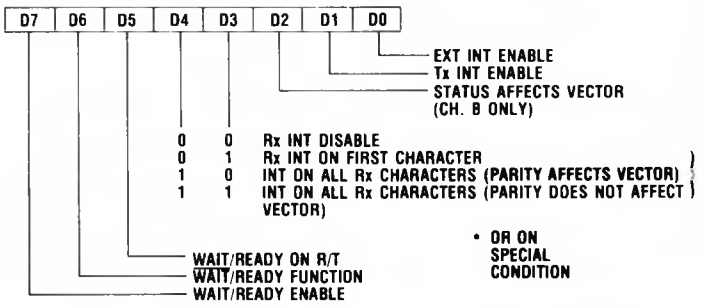
Der Z80-SIO enthält 8 Steuerregister, in die die Codewörter für die gesamten Funktionen eines Kanals eingeschrieben werden. Alle Steuerregister außer Steuerregister 0 benötigen 2 Bytes zur funktionsfähigen Programmierung. Das erste Byte enthält 3 Bits, die das ausgewählte Register markieren (D0 - D2). Das zweite Byte ist das tatsächliche Steuerwort.

Das Steuerregister 0 ist ein Spezialfall. RESET (entweder internen Befehl oder externer Eingang) initialisiert den SIO auf das Steuerregister 0. Alle grundlegenden Kommandos (CMD 2-CMD 0) und CRC-Befehle (CRC 0, CRC 1) können durch ein einziges Steuerbyte im Steuerregister 0 aktiviert werden.

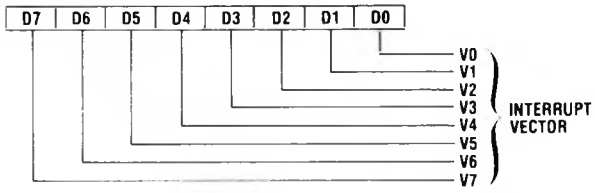
WRITE REGISTER 0



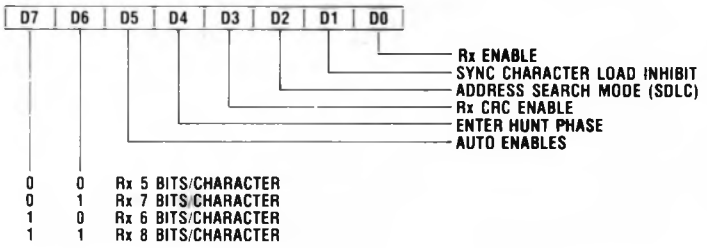
WRITE REGISTER 1



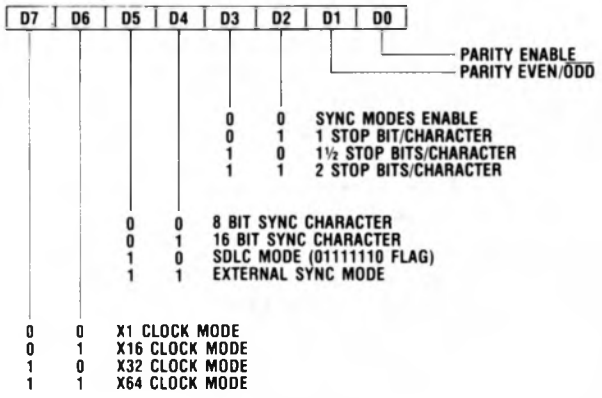
WRITE REGISTER 2 (CHANNEL B ONLY)



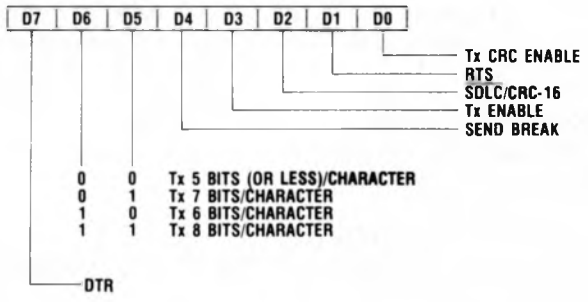
WRITE REGISTER 3



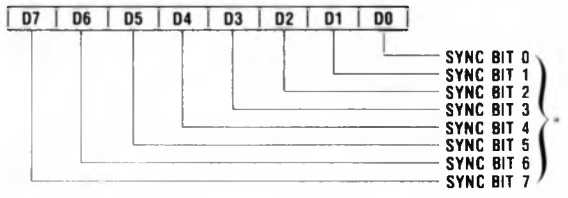
WRITE REGISTER 4



WRITE REGISTER 5

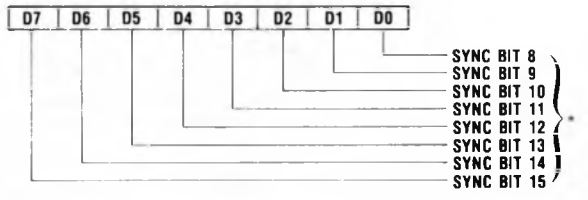


WRITE REGISTER 6



*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7

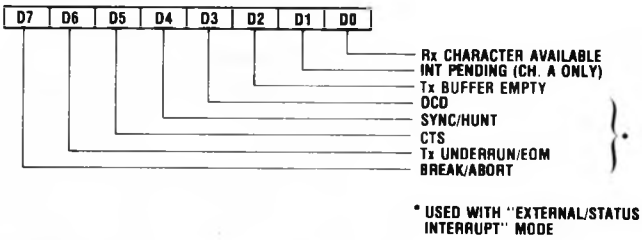


*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

Statusregister

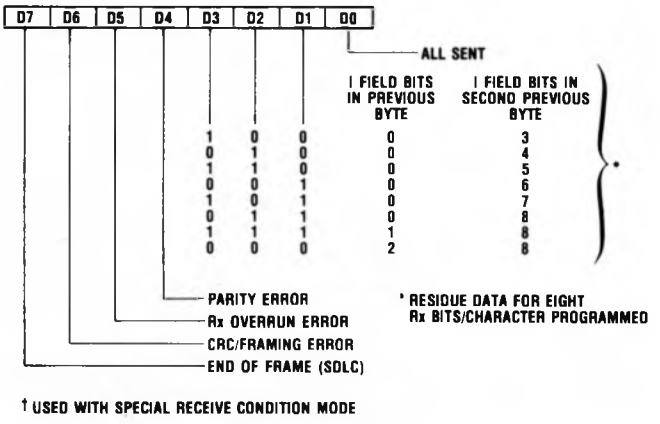
Der Z80-SIO enthält 3 Statusregister, die von der CPU gelesen werden können und Informationen über den Status jedes Kanals liefern. Die Statusinformation enthält Fehlermeldungen, den Interrupt-Vektor und Datentransfersignale. Um den Inhalt eines ausgewählten Statusregisters lesen zu können, muß dem SIO ein Steuerbyte eingeschrieben werden, welches die selbe Zeigerinformation (D0-D2) enthält wie bei Zugriffen zu Steuerregistern. Mit einem Lesezugriff zum SIO kann anschließend der Inhalt des adressierten Statusregisters in die CPU transferiert werden. Beim Entwurf der zugehörigen Programmiersoftware zur Initialisierung des SIO bietet sich die Verwendung der BLOCK I/O-Befehle an.

READ REGISTER 0



* USED WITH "EXTERNAL/STATUS INTERRUPT" MODE

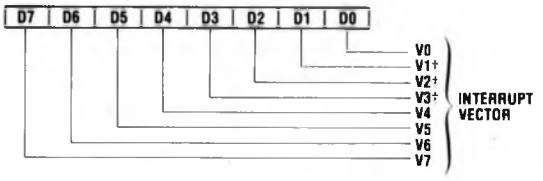
READ REGISTER 1†



* RESIDUE DATA FOR EIGHT Rx BITS/CHARACTER PROGRAMMED

† USED WITH SPECIAL RECEIVE CONDITION MODE

READ REGISTER 2



†VARIABLE IF "STATUS AFFECTS VECTOR" IS PROGRAMMED

Registerbeschreibung

Jeder Kanal enthält die nachfolgenden Steuerregister, die als Kommando adressiert werden.

Steuerregister 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code	CRC Reset Code	CMD	CMD	CMD	PNT	PNT	PNT
1	0	2	1	0	2	1	0

$PNT_0 - PNT_2$ (D_0-D_2)

Diese Bits sind Zeigerbits, die dem SIO die Adresse jenes Registers liefern, in welches das folgende Byte geschrieben wird. Das erste in einen Kanal eingeschriebene Byte nach einem Reset wird grundsätzlich ins Steuerregister 0 transferiert. Jedes Byte, welches einem Lese- oder Schreibbefehl zu irgendeinem Register (nicht zur Register 0) folgt, wird in Register 0 geschrieben.

CMD_0 to CMD_2 (D_3-D_5)

Command	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no affect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Reset Receive Interrupt on First Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A only)

Kommando 0 — Es wird üblicherweise zum Setzen des Zeigers für ein nachfolgendes Byte verwendet (Adressierung eines Registers).

Kommando 1 (Send Abort) — wird nur in SDLC-Format verwendet und erzeugt eine Sequenz von 8 bis 13 Einsen.

Kommando 2 (Reset External/Status Interrupts) — nach einem externen oder Status-Interrupt (z.B. Anzeige der Änderung auf einer Modem-Leitung oder BREAK) werden die Statusbits vom Statusregister 0 gespeichert. Dieses Kommando aktiviert sie erneut und erlaubt einen neuerlichen Interrupt. Die Zwischenspeicherung ermöglicht die Erkennung von kurzen Impulsen an den Eingängen und gibt der CPU Zeit für einen Lesevorgang.

Kommando 3 (Channel Reset) — Dieses Kommando wirkt wie ein externer Reset, aber nur auf einem Kanal. Der Reset am Kanal A setzt außerdem die Interrupt-Prioritäts-Logik zurück. Nach diesem Kommando müssen alle Steuerregister neu beschrieben werden. 4 zusätzliche Systemtakte sollten anschließend abgewartet werden, bevor neuerliche Kommandos in den Kanal eingeschrieben werden.

Kommando 4 (Reset Receive Interrupt on First Receive Character) — Bei der Programmierung des Formates „Interrupt nur beim 1. Empfangszeichen" ist es nicht nötig, das Kommando nach jeder vollständigen Meldung neu zu aktivieren.

Kommando 5 (Reset Transmitter Interrupt Pending) — Beim Format „Interrupt bei jedem Zeichen" liefert der Sender einen Interrupt, wenn seine Senderegister leer sind. In Fällen, in denen keine weiteren Zeichen gesendet werden, verhindert dieses Kommando weitere Sende-Interrupts (Interrupts erfolgen erst wieder, nachdem ein neuerliches Zeichen in den Sendeteil geladen wurde).

Kommando 6 (Error Reset, Latches) — Paritäts- und Überlauffehler bleiben im Statusregister 1 so lange gespeichert, bis sie mit diesem Befehl rückgesetzt werden. Dadurch braucht die Fehlerkontrolle des Blockdatentransfers erst am Ende des Blocks zu erfolgen.

Kommando 7 (Return from Interrupt) — Dieses Kommando (nur für Kanal A) wird vom SIO wie ein RETI-Befehl auf dem Datenbus interpretiert, d.h. das Interrupt Under Service-Latch der internen Baugruppe (Sender, Empfänger) wird rückgesetzt. Damit wird Baugruppen niederer Priorität Interrupt erlaubt. Die interne Daisy-Chain kann auch in Systemen, die keine externe Daisy-Chain enthalten, verwendet werden.

CRC RESET CODE 0 (D_6) und CRC RESET CODE 1 (D_7)

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (keine Wirkung)
0	1	Empfangsseitigen CRC-Prüfer rücksetzen
1	0	Senderseitigen CRC-Generator rücksetzen
1	1	CRC/SYNCS Sende-Flip Flop rücksetzen

Steuerregister 1

Steuerregister 1 enthält die Steuerbits für die verschiedenen Arten des Interrupts und WAIT/READY.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Wait/ Ready Enable	Ready/ Wait	FN/ FN	W/Ready On R/T	Empfang Interrupt Mode 1	Empfang Interrupt Mode 0	Status Affects Vector	Trans Interrupt Enable	Ext Interrupts Enable

EXT INT ENABLE (D_0)

Das Ext Int Enable-Bit erlaubt Interrupts als Folge von Zustandsänderungen auf den DCD, CTS oder SYNC-Leitungen, weiters als Folge eines BREAKS oder beim Senden eines CRC oder Sync-Zeichens.

TRANS INT ENABLE (D_1)

Dieses Bit erlaubt Interrupts jedes Mal beim Leerzustand des Sendepuffers.

STATUS AFFECTS VECTOR (D_2)

Bei Auswahl dieses Formats wird der entstehende Interrupt-Vektor variabel:

	V_3	V_2	V_1	
	0	0	0	Ch B Senderpuffer leer
	0	0	1	Ch B External/Status Änderung
Ch B	0	1	0	Ch B Empfangenes Zeichen steht an
	1	0	0	Ch A Senderpuffer leer
	1	0	1	Ch A External/Status Änderung
Ch A	1	1	0	Ch A Empfangenes Zeichen steht an
	1	1	1	Ch A Besonderer Zustand des Empfängers

Ist Bit D_2 gleich null, wird der Grundvektor ausgegeben.

REC INT MODE 0 (D_3), REC INT MODE 1 (D_4)

Die beiden Bits gemeinsam spezifizieren die 4 variablen Meldebedingungen:

MODE	D_4 REC INT MODE 1	D_3 REC INT MODE 0	
0	0	0	Receiver interrupts disabled
1	0	0	Empfänger-Interrupt bei Fehler zum ersten Zeichen
2	1	0	Interrupt on all Receive Characters-Parity/affects Vector
3	1	1	Interrupt on all Receive Characters-Parity error does not affect Vector.

W/READY on R/T (D_5)

Wenn die W/R-Leitung aktiviert ist, wählt dieses Bit Meldung bei: Empfänger ist leer (Bit = 1) oder Senderpuffer ist voll (Bit = 0).

READY FN/WAIT FN (D_6)
Soll die W/R-Leitung mit der CPU in Wait arbeiten, sollte dieses Bit null sein. In Zusammenarbeit mit einem DMA als READY muß es 1 sein. Die Ready-Funktion kann jederzeit auftreten unabhängig davon, ob der SIO adressiert ist. Die Wait-Funktion wird nur dann aktiv, wenn die CPU versucht, Daten aus dem SIO zu lesen, die noch nicht vollständig empfangen sind bzw. versucht, Daten zu schreiben, während der Senderpuffer noch immer voll ist. In der Wait-Funktion ist der Ausgang als Open Drain geschaltet und folgt der fallenden Flanke des Systemtaktes. Bei der Ready-Funktion ist der Ausgang aktiv und folgt der steigenden Flanke des Systemtaktes.

WAIT/READY ENABLE (D_7)
Die W/R-Leitung bleibt 1 (ready) oder tristate (wait) bis dieses Bit auf 1 programmiert wird.

Steuerregister 2
Register 2 ist das Interrupt-Vektor-Register und existiert nur in Kanal B. Beim Lesen werden $V_4 - V_7$ und V_0 wie beim Schreiben geliefert. V_1 bis V_3 folgen „Status Affects Vektor" (falls gewählt).

Steuerregister 3
Register 3 enthält Steuerbits für einen Teil der Empfangslogik.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
RCVR Bits/ Char 0	RCVR Bits/ Char 1	Auto Enables	Enter Hunt Mode	RECVR CRC Enabl	Address Search Mode	Sync Char Load Inhibit	Receiver Enabl

RECEIVER ENABLE (D_0)
„Eins" erlaubt grundsätzlich Empfangsfunktionen.

SYNC CHAR LOAD INHIBIT (D_1)
Bei Auswahl dieser Option werden die einer Meldung vorangehenden Sync-Zeichen nicht in die Empfangspuffer-Register geladen. Die CRC-Berechnung wird während der Entfernung der SYNC-Zeichen nicht gestoppt.

ADDRESS SEARCH MODE (D_2)
Im SDLC-Format verhindert das Setzen dieses Bits Interrupts, sofern die Adressen der Meldungen nicht mit der programmierten Adresse oder der Global-Adresse (IIIIIIII) übereinstimmen.

RECVR CRC ENABLE (D_3)
Ist dieses Bit gesetzt, beginnt die Berechnung des CRC beim Start des letzten Zeichens vom Empfangsregister zum Puffer.

ENTER HUNT MODE (D_4)
Geht die Zeichensynchronisierung verloren oder wird bestimmt, daß der Inhalt einer ankommenden Meldung nicht benötigt wird (SDLC), kann mit dem Schreiben einer 1 in dieses Bit der Sync-Zeichen-Suchlauf neu beginnen.

AUTO ENABLES (D_5)
Bei gesetztem Bit fungieren die DCD und CTS-Eingänge als Empfangs- und Senderaktivierleitungen. Bei rückgesetztem Bit wirken die beiden Eingänge nur auf ihre entsprechenden Bits im Statusregister 0.

RCVR BITS/CHAR 1 (D_6), RCVR BITS/CHAR 0 (D_7)
Diese Bits bestimmen gemeinsam, wie viele Empfangene Bits gemeinsam einem Zeichen entsprechen. Diese Bits können während der Zusammenstellung eines Zeichens verändert werden, falls dies geschieht, bevor die Anzahl der programmierten Bits erreicht wurde.

D_6 Receiver Bits/Character 1	D_7 Receiver Bits/Character 0	Bits/Character
0	0	5
0	1	6
1	0	7
1	1	8

Steuerregister 4
Dieses Register enthält Steuerbits zur Steuerung von Sender und Empfänger.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate 1	Clock Rate 0	Sync Modes 1	Sync Modes 0	Stop Bits 1	Stop Bits 0	Parity Even/ Odd	Parity

PARITY (D_0)
Falls D_0 gesetzt, bewirkt es ein zugefügtes Paritätsbit beim Senden und Erwartung eines Paritätsbits beim Empfangen.

PARITY EVEN/ODD (D_1)
Auswahl gerader (Bit = 0) oder ungerader (Bit = 1) Parität.

STOP BITS 0 (D_2), STOP BITS 1 (D_3)
Bestimmung der Anzahl der Stopbits beim Senden, der Empfänger erwartet immer ein Stopbit (Asynchron).
Bei 00 wird Synchronbetrieb markiert

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync Modes
0	1	1 Stop Bit pro Zeichen
1	0	$1^1/_2$ Stop Bits pro Zeichen
1	1	2 Stop Bits pro Zeichen

SYNC MODES 0 (D_4), SYNC MODES (D_5)
Auswahl der verschiedenen SYNC-Arten

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDCL Mode (01111110 sync pattern)
1	1	External Sync Mode

CLOCK RATE 0 (D_6), CLOCK RATE 1 (D_7)
Bestimmung des Multiplikationsfaktors zwischen Takt und Datentransferrate (im Synchronbetrieb ist „x 1" Bedingung), wobei der Faktor für Senden und Empfang gilt.
Grundsätzlich muß der Systemtakt (0) mindestens 4,5-fach der Datenrate sein. Beim Faktor x 1 muß die Bit-Synchronisierung extern erfolgen.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate X 1 = Clock Rate
0	1	Data Rate X16 = Clock Rate
1	0	Data Rate X32 = Clock Rate
1	1	Data Rate X64 = Clock Rate

Steuerregister 5
Die hier enthaltenen Bits steuern meist den Sender.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Transmit Bits/ Char 0	Transmit Bits/ Char 1	Send Break	Transmit Enable	SDLC/ CRC16	RTS	Transmit CRC Enable

TRANSMIT CRC ENABLE (D_0)
Bei Setzen dieses Bits wird das momentan ausgesandte Zeichen mit CRC berechnet. Der CRC wird nicht automatisch bei leerem Senderkanal ausgesandt.

RTS (D_1)
Steuerbit für RTS-Leitung. Bit = 1: RTS geht auf 0 (aktiv), wird das Bit = 0 geht RTS erst dann auf 1 (inaktiv), wenn der Sender leer ist.

SDLC/CRC/16 (D_2)
Auswahl des CRC-Codes. Bit = 1 bewirkt das SDLC-Polynom

$$X^{16} + X^{12} + X^5 + 1,$$

Bit = 0 das CRC-16-Polynom

$$X^{16} + X^{15} + X^2 + 1.$$

TRANSMIT ENABLE (D_3)
Datenausgabe bleibt blockiert und TxD auf „1" (Markting) bis das Bit = 1 wird. Bei Rücksetzen wird das momentane Zeichen voll ausgesandt (jedoch: CRC wird unterbrochen!)

SEND BREAK (D_4)
Gesetzt, bewirkt dieses Bit das Aussenden von „0" (Spacing) auf TxD, unabhängig von der momentanen Ausgabe.

TRANSMIT BITS/CHAR 0 (D_5), TRANSMIT BITS/CHAR 1 (D_6)
Diese Bits kontrollieren die Bitzahl/Zeichen, die vom eingeschriebenem Datenbyte tatsächlich ausgegeben werden.

D_5 Transmit Bits/Character 1	D_6 Transmit Bits/Character 0	Bits/Character
0	0	5 or less
0	1	6
1	0	7
1	1	8

Alle Bits sind rechtsorientiert, D_0 wird zuerst gesendet. Bei „5 or less" können 1 – 5 Bits transferiert werden, wobei die Daten laut Tafel formatiert sein müssen.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	sendet 1 Bit
1	1	1	0	0	0	D	D	sendet 2 Bits
1	1	0	0	0	D	D	D	sendet 3 Bits
1	0	0	0	D	D	D	D	sendet 4 Bits
0	0	0	D	D	D	D	D	sendet 5 Bits

DTR (D_7)
Dieses Bit kontrolliert die DTR-Leitung. Bit = 1 bewirkt DTR = 0 (aktiv)

Steuerregister 6
Dieses Register enthält die ersten 8 Bit einer BiSync-Sequenz, muß in SDLC-Format mit der Prüfadresse programmiert sein und das SYNC-Zeichen bei 8 Bit-Synchronbetrieb enthalten. Bei EXT/SYNC-MODE nicht verwendet.

Steuerregister 7
Enthält das 2. Byte in BiSync-Transfer oder das 8 Bit-SYNC-Zeichen. In SDLC-Format muß es mit 01111110 programmiert sein.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYN15	SYN14	SYN13	SYN12	SYN11	SYN10	SYN9	SYN8

Statusregister 0
Wird mit Registerzeiger 000 ausgewählt.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Sending CRC/ Syncs	CTS	Sync/ Hunt	DCD	Transmit Buffer Empty	Interrupt Pending	Receive Character Available

RECEIVE CHARACTER AVAILABLE (D_0)
Gesetzt, falls mindestens ein Zeichen im Puffer vorhanden.

INTERRUPT PENDING (D_1)
Bei Anmeldung irgendeines INTERRUPTS im SIO wird dieses Bit gesetzt, allerdings nur im Kanal A vorhanden, in B immer 0.

TRANSMITTER BUFFER EMPTY (D_2)
Gesetzt, wenn Sende-Puffer leer, außer bei Senden eines CTC.

DCD (D_3)
Zeigt den Zustand der DCD-Leitung zum Zeitpunkt der letzten Änderung eines der fünf Status-Bits (DCD, CTS, SYNC/ HUNT, BREAK/ABORT, SENDING CRC/SYNCS). Zum Lesen des momentanen Status muß dieses Bit unmittelbar nach dem Kommando 2 gelesen werden.

SYNC/HUNT (D_4)
Im Asynchronbetrieb zeigt dieses Bit den Zustand der SYNC-Leitung. Bei Synchronbetrieb wird dieses Bit nach erfolgter Zeichensynchronisierung rückgesetzt und kann nur durch Schreiben des „Enter Hunt Mode"-Bits wieder gesetzt werden.

CTS (D_5)
Zeigt den Zustand der CTS-Leitung (invertiert)

BREAK/ABORT (D_6)
Im Asynchronbetrieb wird dieses Bit durch Erkennen eines „BREAK" gesetzt. Nach Neuaktivierung der Eingänge mit Kommando 2 wird dieses Bit beim Ende des „BREAK" rückgesetzt. Falls „EXTERNAL/STATUS"-Interrupts erlaubt sind, erzeugt dieses Bit einen Interrupt.
Im SDLC-Format wird dieses Bit beim Erkennen einer Abbruch-Sequenz gesetzt (sieben oder mehr Einsen).

SENDING CRC/SYNCS (D_7)
Im Synchronbetrieb wird der CRC beim ersten Auftreten eines leeren Senderegisters innerhalb einer Meldung gesendet. Interrupts werden nur mit einem gesetzten Bit erlaubt. Ist dieses Bit gesetzt und das „TRANSMITTER BUFFER EMPTY"-Bit rückgesetzt, wird der CRC ausgegeben. Sind beide Bits gesetzt, werden automatisch SYNC-Zeichen ausgegeben.

Statusregister 1
Dieses Register wird mit dem auf 001 gesetzten Zeiger ausgesprochen. Nach dem Lesen steht der Zeiger automatisch wieder auf 000.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End Of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

ALL SENT (D_0)
Im Asynchronbetrieb ist dieses Bit nach Aussenden aller im Sendekanal enthaltenen Zeichen gesetzt.

RESIDUE CODE 0 (D_1) – RESIDUE CODE 2 (D_3)
Diese Bits markieren die Länge des I-Feldes im SDLC-Format, in Fällen, in denen das I-Feld kein ganzzahliges Vielfaches der Zeichenlänge ist. Nur bei Transfers mit gesetztem „END OF FRAME" (SDLC)-Bit sinnvoll. Bei einem auf 8 Bit/Zeichen gesetzten Empfänger zeigen die Bits folgendes:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Felder sind rechts justiert.
Bei anderer Zeichenlänge können ähnliche Code-Tafeln konstruiert werden. Falls kein Rest bleibt, gilt immer:

Residue Code 2	Residue Code 1	Residue Code 0
1	0	1

PARITY ERROR (D_4)
Bei aktivierter Paritätskontrolle wird dieses Bit bei falscher Parität gesetzt. Es bleibt bis zu einem Kommando 6 gespeichert.

RECEIVER OVERRUN ERROR (D_5)
Zeigt Empfänger-Überlauf an (mehr als 4 Zeichen), Rücksetzung nur durch Kommando 6. Ist „Status Affects Vector" aktic, liefert das Überlauf-Zeichen einen Interrupt mit dem „Special Receive Condition"-Vektor.

CRC/FRAMING ERROR (D_6)
Bei Auftreten eines Rahmenfehlers (im Asynchronbetrieb) ist diese Bit nur während des jeweiligen falschen Zeichens gesetzt. Bei der Erkennung eines Rahmenfehlers wird automatisch eine halbe Bitzeit addiert, um eine Interpretation als Startbit zu vermeiden.
Im Synchronbetrieb zeigt das Bit den CRC-Vergleich an.

END OF FRAME (D_7)
Zeigt in SDLC-Format eine gültige Ende-FLAG nebst gültigen CRC und RESIDUE-Codes an.

Statusregister 2
Enthält den Interrupt-Vektor wie das Steuerregister 2, falls das „Status Affects Vector"-Bit nicht gesetzt ist.
Bei gesetztem Bit liefert es den Interrupt-Vektor, der momentan verarbeitet wird. Falls kein Interrupt anliegt, sind $V_3 = 0$ und $V_1 = 1$, der Rest wie programmiert.
Lesen nur aus Kanal B möglich.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0
				Variable if „Status Affects Vector" is enabled			

Programmierbeispiele:
Eine typische Kaltstartroutine (nach einem internen oder externen RESET) würde folgendermaßen aussehen:

B/A	C/D	RD	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Erläuterung
1	1	1	0	0	0	0	0	0	1	0	Pointer set to Register 2B
1	1	1	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0	Interrupt Vector loaded
1	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4B
1	1	1	0	1	X	X	0	1	1	1	Even parity, 1 stop bit, X16 clock, asynchronous mode selected
1	1	1	0	0	0	0	0	1	0	1	Pointer set to Write Register 5B
1	1	1	0	0	1	0	1	0	1	0	7 bits/transmit character, transmitter enabled
1	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3B
1	1	1	0	1	1	0	0	0	0	1	7 bits/receive character, DCD and CTS enable Receiver and Transmitter, Receiver enabled
1	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1B
1	1	1	0	0	0	1	0	1	1	1	Interrupt on every character, status affects Vector external/status interrupts enabled

Nun wird Kanal B zum asynchronen Senden und Empfang vorbereitet. Als nächstes wird eine entsprechende Vorbereitung für Kanal A durchgeführt.

0	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4A
0	1	1	0	0	1	0	0	0	0	0	SDLC mode and X1 clock selected, no parity

Programmierbeispiel:

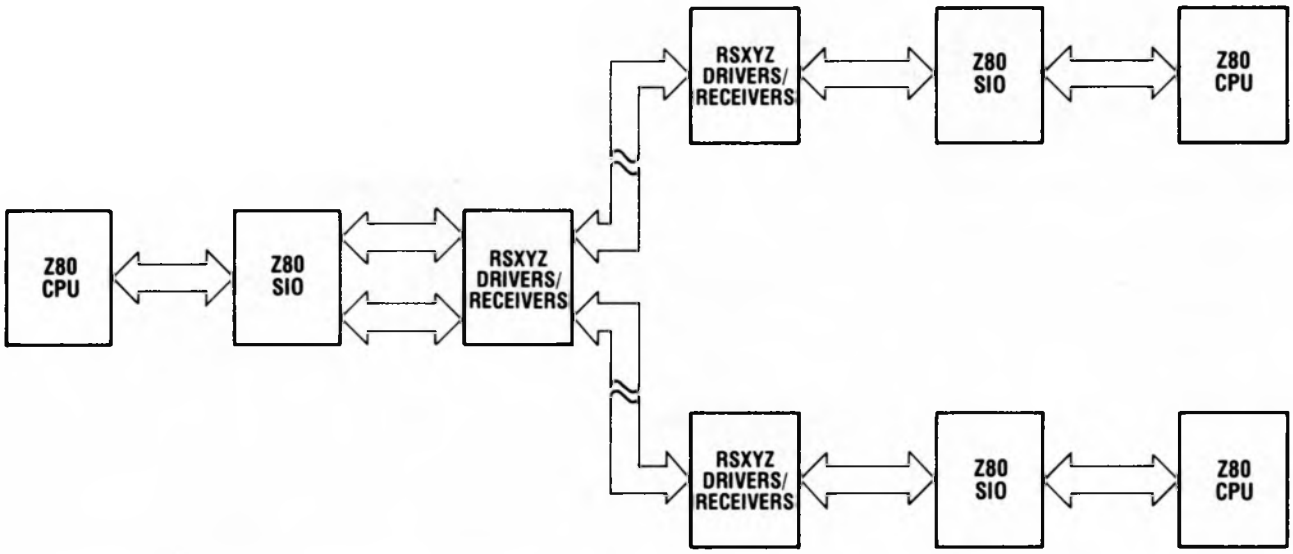
B/A	C/D	RD	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Erläuterung
0	1	1	0	1	0	0	0	1	1	0	Pointer set to Write Register 6A, Reset Receive CRC Checker
0	1	1	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	AD_1	AD_0	SDLC message address entered
0	1	1	1	0	0	0	0	1	1	1	Pointer set to Write Register 7A, Reset Transmit CRC generator
0	1	1	0	1	1	1	1	1	1	0	SDLC Flag entered
0	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1A
0	1	1	0	0	0	1	0	1	1	1	Interrupt every character, status affects vector, external/status interrupts enabled
0	1	1	0	0	0	1	0	1	0	1	Pointer set to Write Register 5A, Reset External/Status Interrupts
0	1	1	1	1	1	0	1	0	0	0	SDLC CRC Code selected, 8 bits/transmit character, CRC and transmitter enabled
0	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3A
0	1	1	1	1	1	0	1	1	0	1	8 bits/receive character, DCD and CTS enable receiver and transmitter, receiver is enabled, SIO searches for programmed address

Nun wird Kanal A für SDCL-Übertragung vorbereitet.

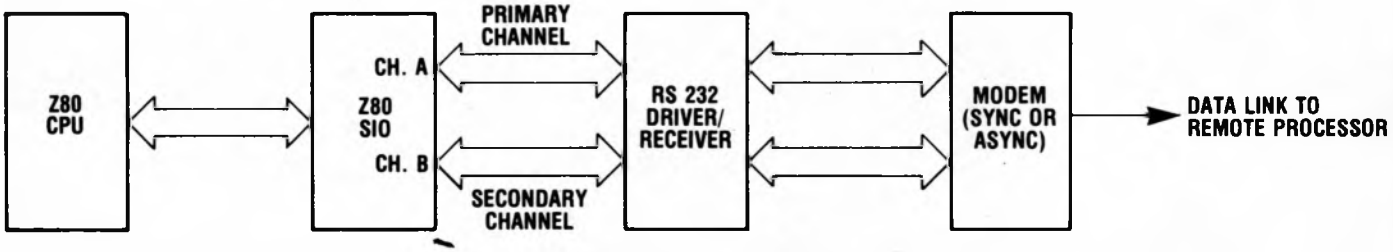
0	0	1	D	D	D	D	D	D	D	D	Address byte to be sent Ch. A
0	1	1	1	1	0	0	0	0	0	0	Reset CRC/SYNCS SENT/SENDING, pointer to register 0, so CRC can be automatically sent at end of message

Anwendungshinweise:

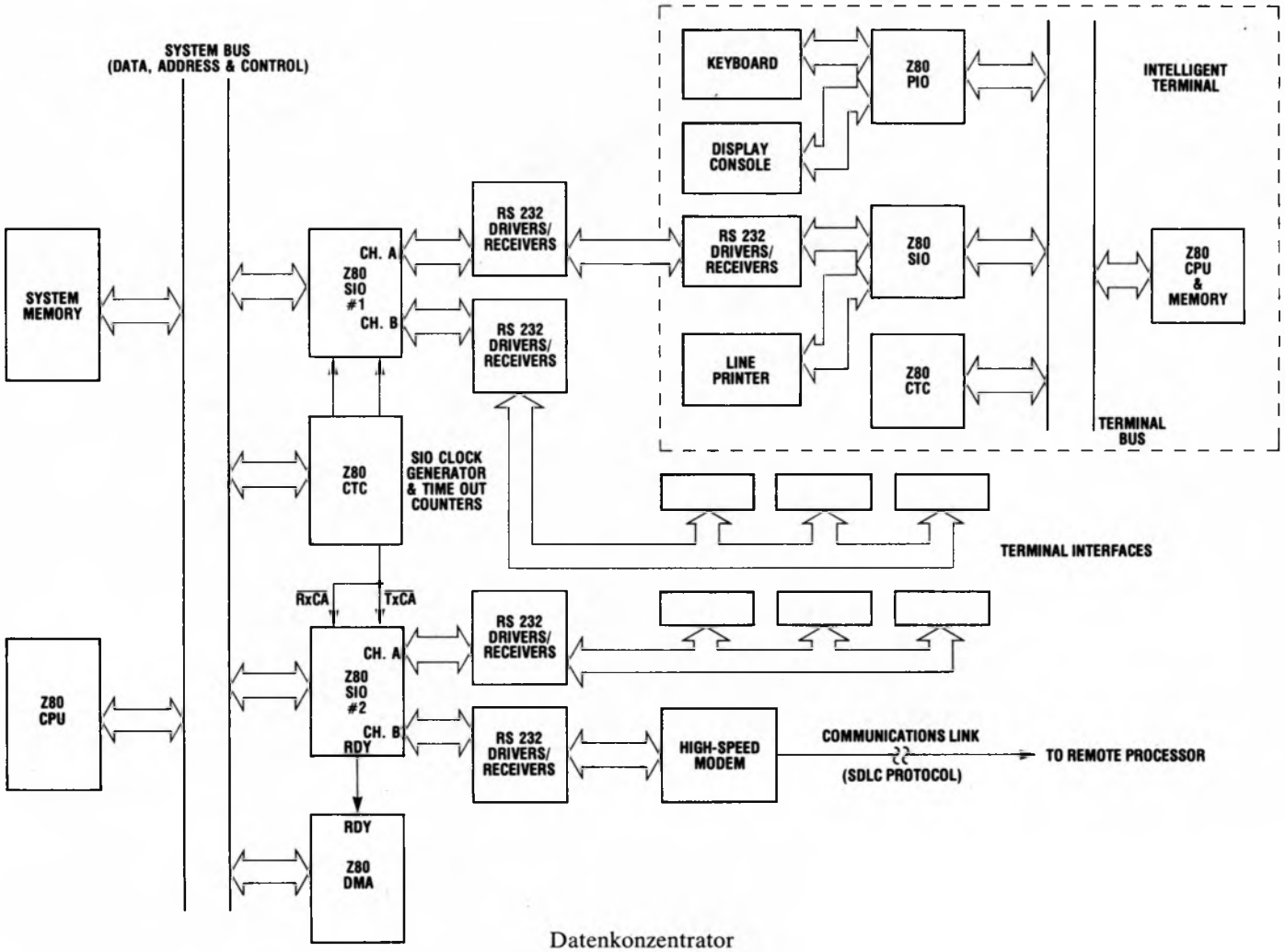
Die folgenden Abbildungen illustrieren typische Anwendungen der SIO-Bausteine bzw. ihrer verschiedenen Ausführungen. Die Zeichnungen bedürfen kaum einer Erläuterung und stellen nur einen kleinen Ausschnitt aus dem Anwendungsbereich dieses extrem universellen Bausteins dar. Fertige Standardhardware dieser Art ist im Rahmen der Z80-Baugruppenserie ECB unter der Bezeichnung ECB/F und ECB/C8 verfügbar.



Synchrone/Asynchrone Rechner/Rechner-Serienkommunikation für einen Haupt- und zwei Sub-Computer

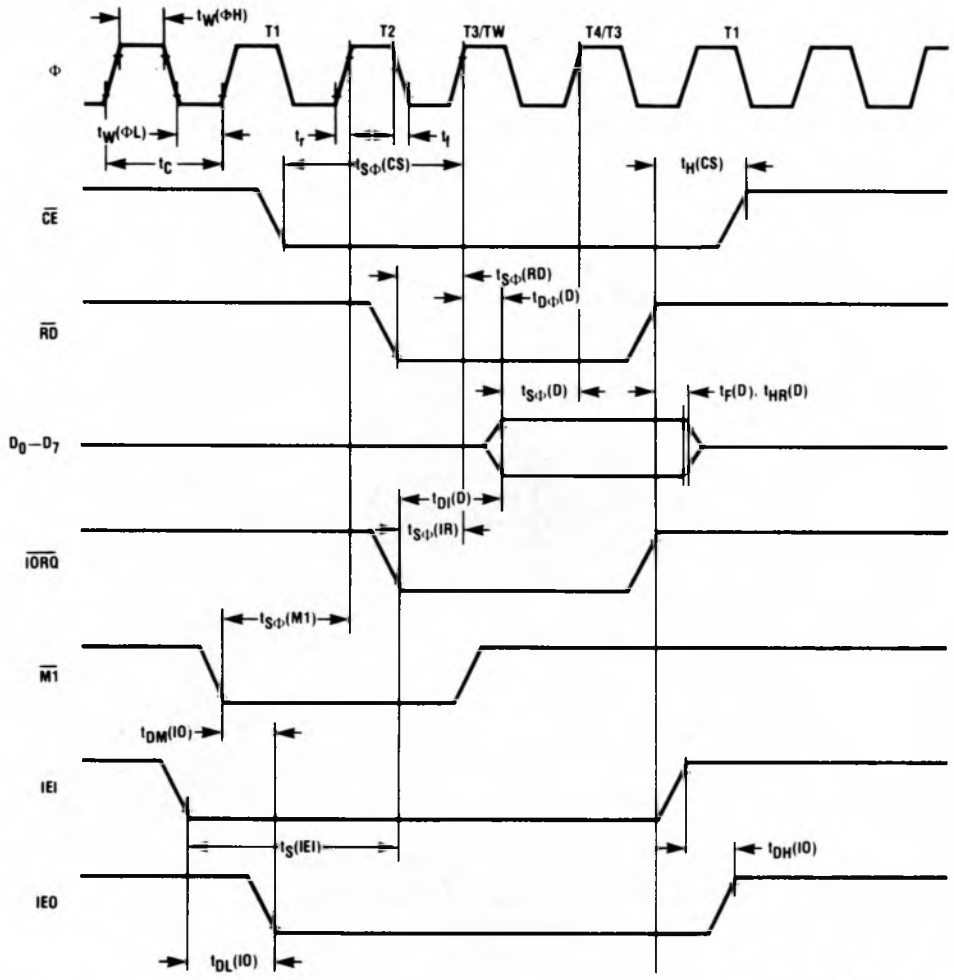


Sychrone/Asynchrone Rechner/Rechner-Kopplung über Telefonnetz



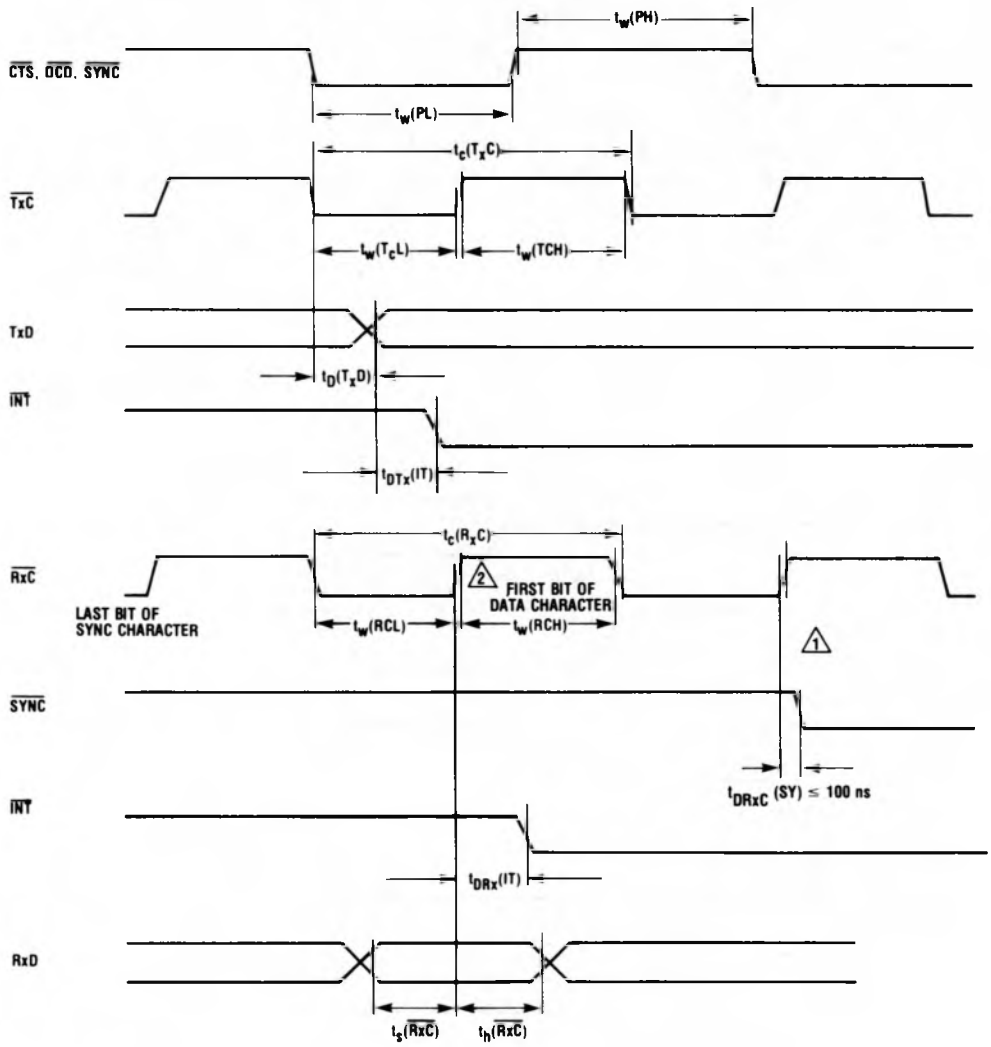
Datenkonzentrator

$T_A = 0°C$, $V_{CC} = +5V$, $\pm 5\%$



Signal	Symbol	Parameter	Z80A-SIO Min	Max	Unit
	$t_C(\phi)$	Clock Period	250	4000	ns
	$t_w(\phi H)$	Clock Pulse Width, clock HIGH	105	2000	ns
ϕ	$t_w(\phi L)$	Clock Pulse Width, clock LOW	105	2000	ns
	t_r, t_f	Clock Rise and Fall Times	0	30	ns
	t_H	Any Unspecified Hold Time for setup times specified below	0		ns
$\overline{CE}$, B/$\overline{A}$ C/$\overline{D}$, $\overline{IORQ}$	$t_{S\phi}$(CS)	Control Signal Setup Time to rising edge of ϕ during Read or Write Cycle	145		ns
	$t_{D\phi}$(D)	Data Output Delay from rising edge of ϕ during Read Cycle		220	ns
	$t_{S\phi}$(D)	Data Setup Time to rising edge of ϕ during Write or M1 Cycle	50		ns
D_0-D_7	t_{DI}(D)	Data Output Delay from falling edge of $\overline{IORQ}$ during INTA Cycle		160	ns
	t_F(D)	Delay to Floating Bus (output buffer disable time)		110	ns
IEI	t_S(IEI)	IEI Setup Time to falling edge of $\overline{IORQ}$ during INTA Cycle	140		ns
	t_{DH}(IO)	IEO Delay Time from rising edge of IEI (after 'ED' decode)		100	ns
IEO	t_{DL}(IO)	IEO Delay Time from falling edge of IEI		100	ns
	t_{DM}(IO)	IEO Delay Time from falling edge of $\overline{M1}$ (interrupt occurring just prior to M1)		190	ns
M1	$t_{S\phi}$(M1)	$\overline{M1}$ Setup Time to rising edge of ϕ during INTA or M1 Cycle	90		ns
$\overline{RD}$	$t_{S\phi}$(RD)	$\overline{RD}$ Setup Time to rising edge of ϕ during Read or M1 Cycle	115		ns

***If WAIT from the SIO is to be used, $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ and $\overline{M1}$ must be valid for as long as the Wait condition is to persist.**



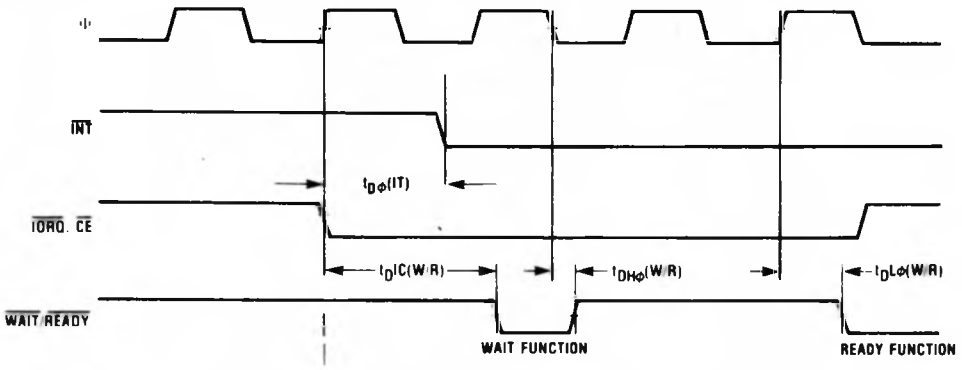
NOTES:

1. The SYNC input must be driven Low on the rising edge of RxC delayed two complete clock cycles from the last bit of the sync character.
2. Data character assembly begins on the next Receive Clock cycle after the last bit of the sync character is received.

Signal	Symbol	Parameter	Z80A-SIO Min	Max	Unit
INT	$t_{DRx}(IT)$	INT Delay Time from rising edge of RxC	10	13	ϕ periods
	$t_{DTx}(IT)$	INT Delay Time from transition of Xmit Data Bit	5	9	ϕ periods
CTSA, CTSB, DCDA, DCDB, SYNCA, SYNCB	$t_w(PH)$	Minimum HIGH Pulse Width for latching state into register and generating interrupt	200		ns
	$t_w(PL)$	Minimum LOW Pulse Width for latching state into register and generating interrupt	200		ns
SYNCA, SYNCB	$t_{DL}(SY)$	Sync Pulse Delay Time from rising edge of RxC, Output Modes	4	7	ϕ periods
	$t_{DRxC}(SY)$	Sync Pulse Delay Time from rising edge of RxC External Sync Mode		100	ns
TxCA, TxCB	$t_c(TxC)$	Transmit Clock Period	400	∞	ns
	$t_w(TCH)$	Transmit Clock Pulse Width, clock HIGH	180	∞	ns
	$t_w(TCL)$	Transmit Clock Pulse Width, clock LOW	180	∞	ns
TxDA, TxDB†	$t_D(TxD)$	TxD Output Delay from falling Edge of TxC (x1 Clock Mode)		300	ns
RxCA, RxCB	$t_c(RxC)$	Receive Clock Period	400	∞	ns
	$t_w(RCH)$	Receive Clock Pulse Width, clock HIGH	180	∞	ns
	$t_w(RCL)$	Receive Clock Pulse Width, clock LOW	180	∞	ns
RxDA, RxDB†	$t_s(RxC)$	Setup Time to rising edge of RxC, x1 mode	0		ns
	$t_h(RxC)$	Hold Time from rising edge of RxC, x1 mode	140		ns

†In all modes, the system clock (ϕ) rate must be at least 4.5 times the maximum data rate. RESET must be active a minimum of one complete ϕ cycle.

Dynamische Kenndaten



Signal	Symbol	Parameter	Z80A-SIO Min	Max	Unit
$\overline{INT}$	$t_{D\phi(IT)}$	$\overline{INT}$ Delay Time from rising edge of ϕ		200	ns
	$t_{DIC(W/R)}$	$\overline{WAIT/READY}$ Delay Time from $\overline{IORQ}$ or $\overline{CE}$ in Wait Mode		130	ns
	$t_{DH\phi(W/R)}$	$\overline{WAIT/READY}$ Delay Time from falling edge of ϕ, $\overline{WAIT/READY}$ HIGH, Wait Mode		130	ns
$\overline{WAIT/READY}$	$t_{DRx(W/R)}$	$\overline{WAIT/READY}$ Delay Time from rising edge of $\overline{RxC}$ Data Bit, Ready Mode	10	13	ϕ periods
	$t_{DTx(W/R)}$	$\overline{WAIT/READY}$ Delay Time from center of Transmit Data Bit, Ready Mode	5	9	ϕ periods
	$t_{DL\phi(W/R)}$	$\overline{WAIT/READY}$ Delay Time from rising edge of ϕ, $\overline{WAIT/READY}$ LOW, Ready Mode		120	ns

Statische Kenndaten

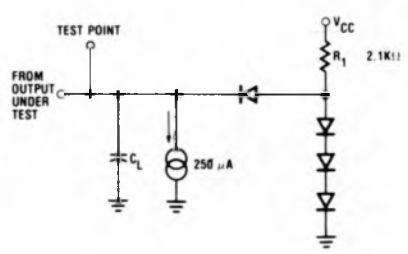
$T_A = 0\,°C$ to $70\,°C$, $V_{CC} = +5V, \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{CC}-0.6$	+5.5	V	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	+2.4		V	$I_{OH} = -250\,\mu A$
I_{LI}	Input Leakage Current	−10	+10	µA	$0 \leq V_{IN} \leq V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	−10	+10	µA	$0 \leq V_{IN} \leq V_{CC}$
$I_{LI(SY)}$	$\overline{SYNC}$ Pin Leakage Current	−40	+10	µA	
I_{CC}	Power Supply Current		100	mA	

Kapazitäten

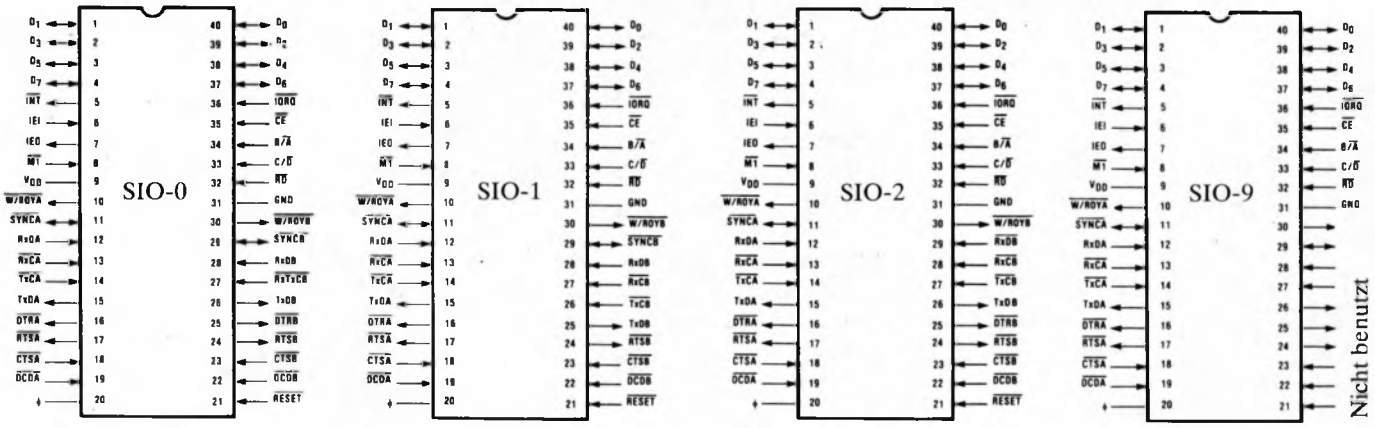
$T_A = 25\,°C$, $f = 1$ MHz

Symbol	Parameter	Min.	Max.	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

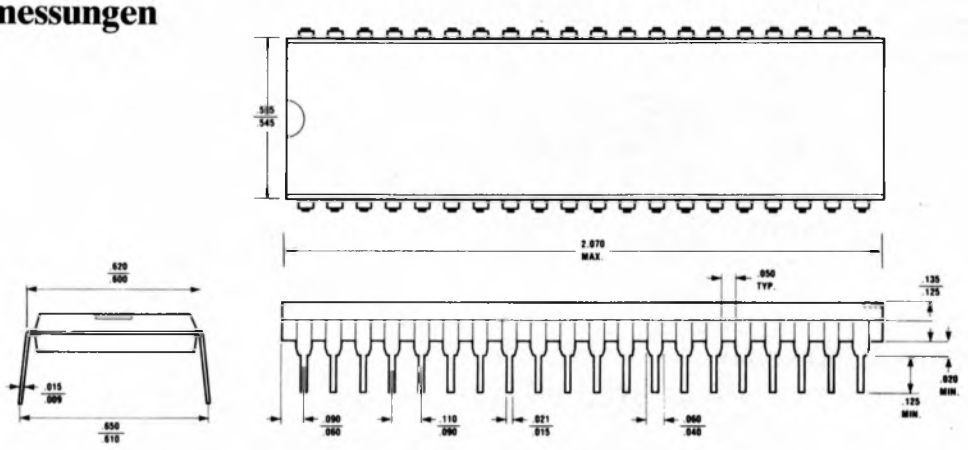


$C_L = 50$ pF. Increase delay by 10 ns for each 50 pF increase in C_L, up to 200 pF maximum.

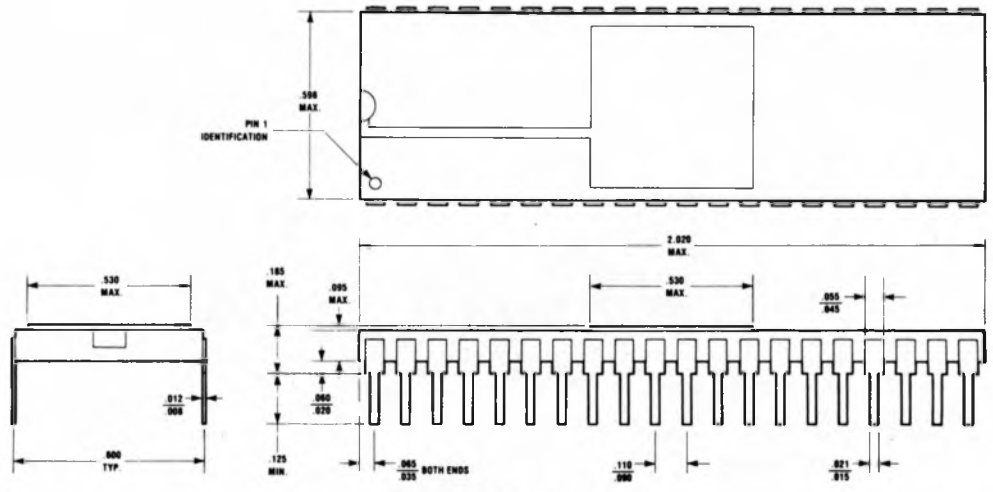
Pin-Belegung



Gehäuseabmessungen



40-Pin Plastik



40-Pin Keramik

□ Bestellbezeichnung

Z80A-SIO/PS	Standardversion (0 . . . 70°C) in Plastikgehäuse $U_B = 5\ V \pm 5\%$
Z80A-SIO/CS	Standardversion (0 . . . 70°C im Keramikgehäuse $U_B = 5\ V \pm 5\%$

Die Zusätze nach den oben angegebenen Bestellbezeichnungen -0, -1, -2 und -9 bezeichnen die verschiedenen Kontaktierungsformen, in denen der SIO geliefert werden kann. Bei der 1-Kanal-SIO-Version Z80-SIO/...-9 ist Kanal B nicht benutzt; die bezeichneten Eingänge dürfen nicht verwendet werden.

Zilog

Z 80-CPU Blockschaltbild

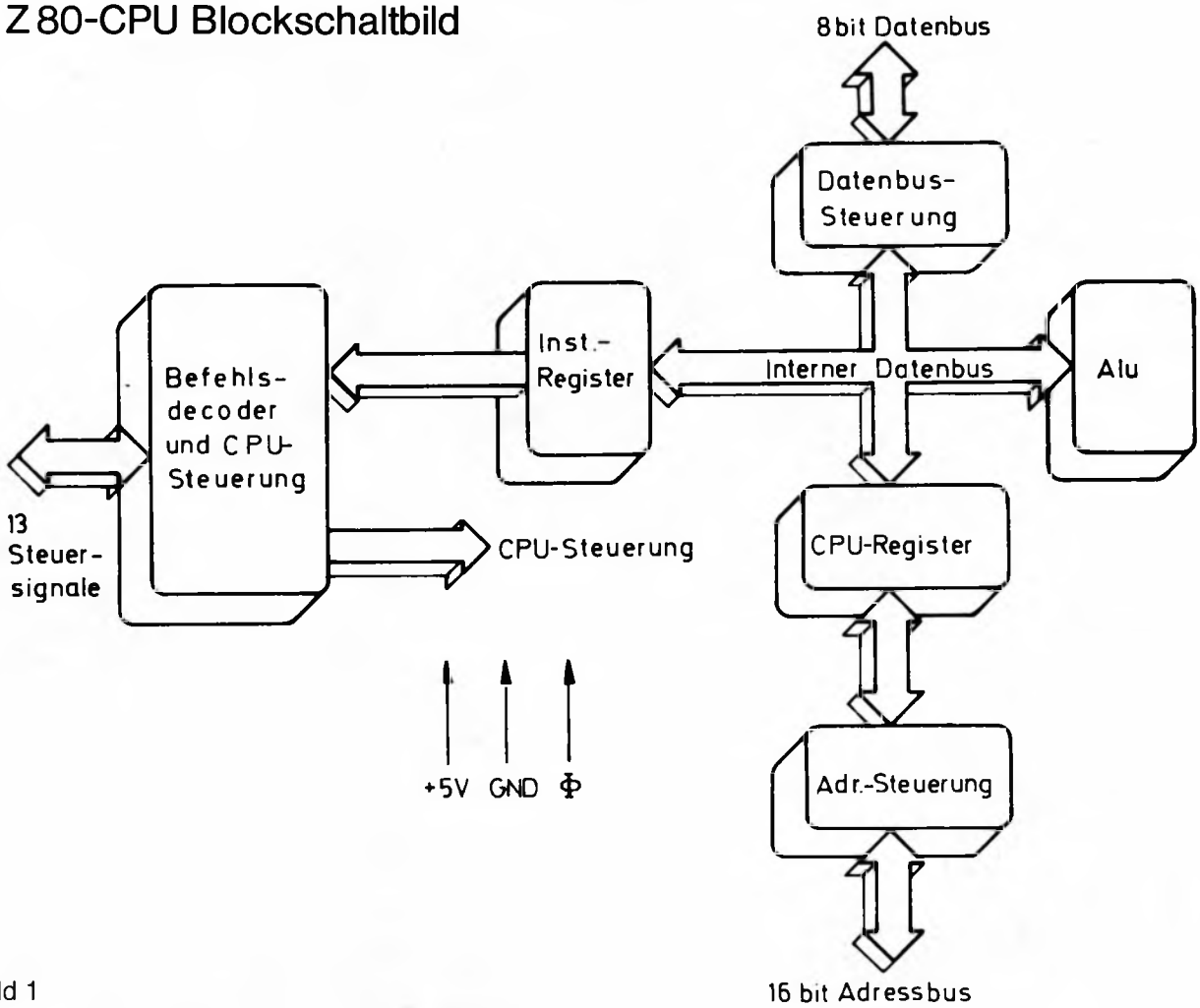


Bild 1

2. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Das Computersystem Z80 umfaßt ein komplettes Angebot an MC-Halbleiterbausteinen, fertig aufgebaute MC-Platinen, fertige OEM-Computersysteme und ein Entwicklungshilfssystem mit der dazugehörigen Betriebssoftware.

Dabei ist das Halbleiterbausteinangebot so geartet, daß auch komplexere MC-Systeme mit minimaler MC-Bausteinanzahl zu realisieren sind. Als Speicherbausteine sind sämtliche Standardchips verwendbar, zusätzliche Logik ist im ganzen System praktisch nicht nötig.

Z80-CPU-Register

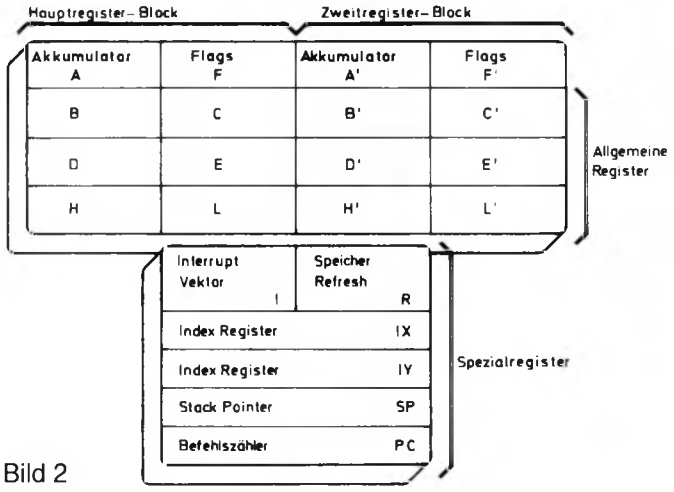


Bild 2

☐ Aufbau

Die Z80-CPU (Ionenimplantierter n-Kanal-Silicon-Gate-Schaltkreis) ist als Mikroprozessor der dritten Generation ein Bauelement auf dem neuesten Stand der Technik mit bisher unerreichter Leistungsfähigkeit.

Der gegenüber bisher verfügbaren Mikroprozessoren höhere Informations-Durchsatz ("throughput") und die effizientere Programmspeicherausnutzung erlauben die Realisierung von Mikroprozessoren-Systemen mit höheren Anforderungen als bei Systemen mit Mikroprozessoren der 2. Generation.

Darüberhinaus vereinfacht Z 80 durch die Notwendigkeit nur einer einzigen Speisespannung und der direkten Anschlußmöglichkeit preiswerter Standardspeicherbausteine den nötigen Schaltungsaufwand.

Bild 1 zeigt ein Blockschaltbild der CPU. In Bild 2 ist die Registerorganisation (= insgesamt 208 Bit dem Anwender zugänglicher Schreib/Lesespeicher) dargestellt.

Hiervon sind zwei gleichaufgebaute Blöcke mit je 6 allgemeinen Registern, die jeweils wahlweise 8- oder 16-Bit-Operationen erlauben.

Hinzu kommen zwei Akkumulatoren und Status(„Flag"-) Register.

Die Operationen der CPU laufen im Haupt-Register/Akku-Block ab; der Zugriff zum zweiten Register/Akku-Block erfolgt durch Übergangs-(„Exchange")-Anweisungen.

Diese alternative Arbeitsweise ermöglicht wechselweises Arbeiten in Haupt- und Hintergrundprogramm ohne Auslagern von Registerinhalten in den Arbeitsspeicher und dadurch besonders schnelle und effiziente Interruptbehandlung.

Der 16-Bit-Stack-Pointer der CPU dient zur Bearbeitung von Mehrebenen („Multilevel-") Interrupts, praktisch unbegrenzte Unterprogramm-Verschachtelung ("Subroutine Nesting") und Zwischenspeicherung von Datenblöcken.

Zwei 16-Bit-Indexregister erlauben die Bearbeitung von Tabellen und relokativen (d.h. im Adreßraum des Arbeitsspeichers verschiebbaren) Informationen.

Ein eigenes Refresh-Register wurde für ein direktes, übersichtliches Arbeiten mit externen dynamischen Speichern ohne Software-Aufwand implementiert.

Das Unterbrechungsregister („I-Register") schließlich liefert zur Realisierung einer besonders leistungsfähigen Art der Interrupt-Behandlung die höherwertigen 8 bits eines Zeigers, der über eine Tabelle den Sprung in Interrupt-Behandlungsprogramme ermöglicht; die niederwertigen 8 Bit der Anfangsadressen werden in üblicher Weise von den anfordernden Schaltung geliefert.

Hervorstechendste Eigenschaft ist die fortschrittliche Architektur, die eine Reduktion der zur Lösung eines bestimmten Problems nötigen Befehle um typisch 50% ermöglicht. Hierdurch werden Programmentwicklungs- und Testkosten eingespart, was besonders bei in kleinen Stückzahlen gefertigten Geräten ausschlaggebend ist; gleichzeitig wird die Anzahl der im System benötigten Programmspeicherbausteine reduziert (wichtig für Systeme in großen Stückzahlen) und die Verarbeitungsgeschwindigkeit in hohem Maß gesteigert. Erzielt wird diese Wirkung durch die bisher genannten und folgenden Eigenschaften der CPU:

- Echte Speicher — indirekte Interruptbearbeitungstechnik möglich (auf Interruptmode 2); dadurch ist eine in der Mikrocomputertechnik bisher unerreichte Flexibilität in der Interruptbehandlung möglich, wie sie bisher nur bei Prozeßrechnern bekannt war.
- In der CPU eingebauter Refresh Controller zum direkten Anschluß von dynamischen Speichern an die CPU ohne zusätzliche Hard- oder Software. Zu den genannten architektonischen Eigenschaften kommen folgende Softwarebesonderheiten:
  - ☐ Befehle zur Behandlung von 4 bit, 8 bit und 16 bit Datenwörter
  - ☐ Befehle zum Register rotieren **und** schieben
- Als einziger Prozessor ist die Z80-CPU in der Lage, sowohl Einzelbits als auch ganze Dateien mit einem einzigen Befehl zu bearbeiten (Blocktransfers mit einer Blockgröße bis zu 64 kByte bei einer Übertragungsrate von 5,3 μsec/Byte, Blocksuchbefehl, Block-Ein/Ausgaben mit einer Ein/Ausgaberate von 200 kByte/sec, Setzen, Testen oder Rücksetzen irgend eines einzelnen Bits in einem der CPU Register oder einer Speicherstelle).

☐ Kenndaten der Z80-CPU

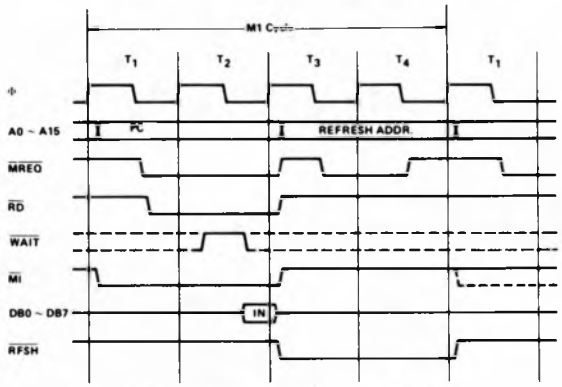
- 1 Chip-Mikroprozessor in n-Kanal-Silicon-Gate-Technologie.
- Befehlssatz umfaßt 158 Befehle, darunter sämtliche 78 8080-Instruktionen (voll Software-kompatibel!).
  Über den 8080-Befehlssatz hinaus verfügt der Z 80 über umfassende 16-, 8-, 4- und Einzel-Bit-Instruktionen und zusätzliche Adressierweisen (indizierte, relative und Bit-Adressierung).
- 17 interne Register.
- 3 schnelle Interrupt-Behandlungsarten und ein zusätzlicher, nichtmaskierbarer Interrupt.
- Direkter Anschluß von dynamischen oder statischen Standardspeicherchips ohne zusätzlichen Bauteileaufwand.
- Eingebaute dynamische Refresh-Hardware.
- Minimale Befehlsausführungsdauer: 1,0 Mikrosekunde.
- Stromversorgung über eine einzige 5-V-Versorgungsspannung.
- 5-V-Einphasen-Takt.
- Alle Anschlüsse TTL-kompatibel.

□ Z80-CPU-Pinbelegung

Pin	Funktion	Kommentar
A_0-A_{15}	Adreßbus	Tri-State Ausgänge, High-aktiv. Liefern die Adressen für den Speicher (bis zu 64 kByte) und die Ein/Ausgabebausteine.
D_0-D_7	Datenbus	Tri-State Ein/Ausgänge (Bidirektional), High-aktiv. Über den Datenbus erfolgt der Datenaustausch zwischen CPU und Speicher bzw. CPU und E/A-Bausteinen.
$\overline{M1}$	Maschinenzyklus Eins	Ausgang, Low-aktiv. M 1 = aktiv bedeutet, daß der momentane Maschinenzyklus der Operationscode-Lesezyklus (Fetch-Zyklus) der momentan auszuführenden Anweisung ist.
$\overline{MREQ}$	Speicheranforderung (= „Memory-Request“)	Tri-State Ausgang, Low-aktiv. MREQ = aktiv bedeutet, daß auf dem Adreßbus die Adresse für einen Speicherzugriff (Lesen oder Schreiben) ansteht.
$\overline{IOREQ}$	Ein/Ausgabe-Anforderung (= „Input/Output-Request“)	Tri-State-Ausgang, Low-aktiv. IOREQ = aktiv bedeutet, daß an den niederwertigen 8 bits des Adreßbus eine Adresse zur I/O-Portauswahl (Eingabe oder Ausgabe) ansteht. Ein IOREQ-Signal wird auch dann erzeugt, wenn eine Interruptanforderung akzeptiert wurde; in diesem Fall kann dann der zugehörige Interrupt-Vektor auf den Datenbus gelegt werden.
$\overline{RD}$	Lesen (“Read”)	Tri-State-Ausgang, Low-aktiv. RD = aktiv bedeutet, daß die CPU Daten vom Speicher oder von einem I/O-Port lesen soll. Der angesprochene Speicher oder I/O-Baustein interpretiert das Signal als Aufforderung, Daten auf den Datenbus zu legen.
$\overline{WR}$	Schreiben (“Write”)	Tri-State-Ausgang, Low-aktiv. WR = aktiv bedeutet, daß die CPU Daten für den Speicher oder einen I/O-Baustein auf dem Datenbus bereithält.
$\overline{RFSH}$	Refresh	Ausgang, Low-aktiv. RFSH = aktiv bedeutet, daß die niederwertigen 7 bit des Adreßbus eine Refreshadresse für dynamische Speicher führen und das laufende MREQ-Signal zur Einleitung eines Refresh-Zyklus für alle angeschlossenen dynamischen Speicher zu benutzen ist.
$\overline{HALT}$	Halt-Zustand	Ausgang, Low-aktiv. HALT = aktiv bedeutet, daß die CPU einen (Software-)HALT-Befehl ausgeführt hat und zur weiteren Abarbeitung des Programms auf ein Interrupt-Signal wartet (einen nicht-maskierbaren Interrupt oder aber einen freigegebenen maskierbaren Interrupt). Im Halt-Zustand führt die CPU zur Sicherstellung des Refresh-Vorgangs Leerbefehle (NOP's) aus.
$\overline{WAIT}$	Warte-Signal (“Wait”)	Eingang, Low-aktiv. Low-Signal am WAIT-Eingang zeigt der CPU, daß die angesprochenen Speicher- oder I/O-Bausteine noch nicht zur Datenübertragung bereit sind. Die CPU beginnt mit Wait-Zyklen, solange der Eingang aktiviert wird.
$\overline{INT}$	Interrupt-Eingang (“Interrupt Request”)	Eingang, Low-aktiv. Das Interrupt-Anforderungssignal wird von einer peripheren Schaltung erzeugt. die Anforderung wird nach Abarbeitung des in Ausführung befindlichen Befehls berücksichtigt, soweit das interne Software-gesteuerte Interrupt-Freigabe-Flip Flop gesetzt und das BUSRQ-Signal nicht aktiv ist.
$\overline{NMI}$	Nicht maskierbarer Interrupt (“nonmaskable interrupt”)	Eingang, Low-aktiv. Eine Interrupt-Anforderung auf diesem Eingang hat höhere Priorität als Interruptanforderungen auf dem Eingang INT für maskierbare Interrupts und wird durch das interne Interrupt-Freigabe-Flip Flop nicht beeinflußt. Gelangt ein Low-Signal an den NMI-Eingang, so wird entsprechend einer RESTART-Instruktion die Programmbehandlung bei Speicheradresse 0066H fortgesetzt.
$\overline{RESET}$	Rückstellen (“Reset”)	Eingang, Low-aktiv. Bewirkt Rücksetzen (= 0) von Interrupt-Freigabe-Flip-Flop, Befehlszähler, Register I und R und bringt die CPU in die 8080-Interrupt-Betriebsart. Während des Rückstellungsvorgangs befinden sich Daten- und Adreßbus im hochohmigen, sämtliche übrigen Ausgänge im inaktiven Zustand.
$\overline{BUSRQ}$	Bus-Anforderung (“Bus Request”)	Eingang, Low-aktiv. Bringt Adreß-, Daten- und Steuerbus-Signal in den hochohmigen Zustand, sodaß externe Schaltungen diese Leitungen benutzen können.
$\overline{BUSAK}$	Bus-Anforderungsbestätigung (“Bus Acknowledgement”)	Ausgang, Low-aktiv. Bestätigt, daß Adreß-, Daten- und Steuerbus in den hochohmigen Zustand gebracht wurden.

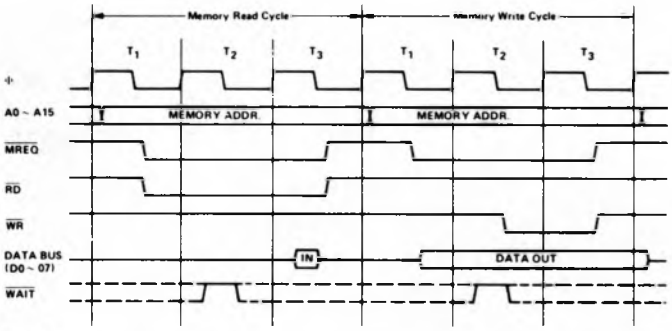
☐ Befehlslesezyklus ("Instruction Op-Code-Fetch")

Der Inhalt des Befehlszählers wird bei Beginn des Zyklus auf den Adreßbus gebracht; 1/2 Taktimpuls später wird $\overline{MREQ}$ aktiv, sodaß die fallende Flanke des $\overline{MREQ}$ direkt als Freigabe-Signal ("Chip Enable") für dynamische Speicherbausteine benutzt werden kann. Während $\overline{RD}$ = aktiv müssen die Daten aus dem Speicher auf den Datenbus gebracht werden; die CPU holt diese Daten mit der steigenden Flanke des Taktimpulses T_3 ab. Taktimpulse T_3 und T_4 eines Lesezyklus' werden benutzt, um dynamischen Speichern während des Befehlsdekodier- und Ausführungsvorgangs in der CPU das Refresh-Signal zu liefern.
$\overline{RFSH}$ gibt diese Refresh-Signale frei.



☐ Speicherzugriffszyklen ("Memory-Read-or-Write")

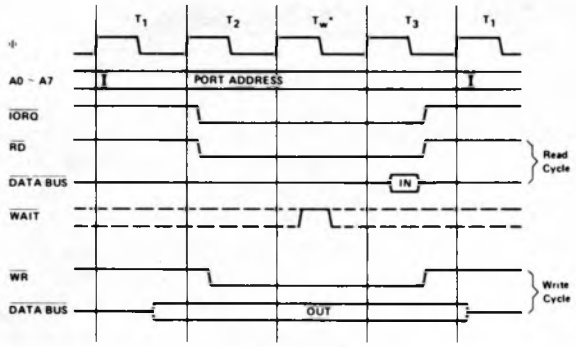
Bei Lese- oder Schreibzyklen verhalten sich die Signale $\overline{MREQ}$ und $\overline{RD}$ genauso wie beim Befehlslesezyklus. Beim Schreibzyklus wird $\overline{MREQ}$ aktiv, sobald die Information auf dem Adreßbus stabil ist, sodaß dieses Signal direkt als Bausteinauswahlsignal ("Chip-Enable") für dynamische Speicher verwendet werden kann.



$\overline{WR}$ ist aktiv, sobald die Informationen auf dem Datenbus stabil sind, sodaß er für praktisch alle Halbleiterspeicher als R/W-Signal zu verwenden ist.

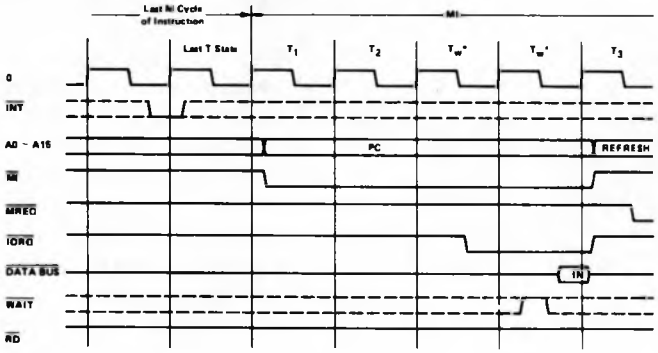
☐ Ein/Ausgabezyklus ("Input or Output Cycles")

Während Ein/Ausgabe Operationen wird automatisch ein Wartezyklus (T_W*) eingefügt, um dem I/O Port genügend Zeit zur Adreßdekodierung und eventueller Aktivierung des WAIT-Signals zu geben.



☐ Unterbrechungsbehandlungs-Zyklen ("Interrupt Request/Acknowledge Cycle")

Der Interrupt-Eingang wird von der CPU bei der steigenden Flanke des letzten Taktimpulses jeder ausgeführten Anweisung abgefragt. Im Interrupt-Fall wird dann ein besonderer M1-Zyklus erzeugt, bei dem $\overline{IORQ}$ statt $\overline{MREQ}$ aktiv wird als Signal für die unterbrechende Schaltung, daß nun ein 8-Bit-Vektor auf den Datenbus zu legen ist. Zwei Wartezyklen T_W* werden hierbei automatisch eingefügt, um der Prioritätsschaltung genügend Zeit zum Durchschalten der Prioritätssignale zu geben.



☐ Z 80-Befehlssatz

Im folgenden wird lediglich ein großer Überblick über den Z 80-Befehlssatz gegeben, Einzelheiten finden Sie im Z-80-Manual. Zu unterscheiden sind folgende Befehlsgruppen:

- 8 Bit-Ladebefehle (="8 bit loads")
- 16 Bit-Ladebefehle (="16 bit loads")
- Austauschbefehle (="Exchanges")
- Blocktransfers im Speicher ("Memory Block Moves")
- Blocksuchbefehle ("Memory Block Searches")
- 8 Bit arthmetische und logische Befehle (="8 bit arithmetic and logic")
- 16 Bit arithmetische Befehle (= 16 bit arithmetic)
- Allgemeine Akkumulator- und Status-Anweisungen (="General purpose Accu and Flag Operations")
- Akku-Rotieren und -Schieben (="Rotate and Shift")
- Bit Setzen, Rücksetzen und Testen (="Bit Set, Reset and Test")
- Ein/Ausgabe (="Input and Output")
- Sprünge (="Jumps")
- Unterprogrammaufrufe ("Calls")
- Restarts ("Restarts")
- Rücksprünge (="Returns")
- Sonstige Befehle ("Miscellaneous Group")

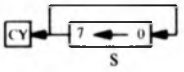
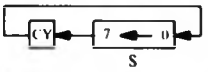
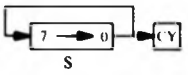
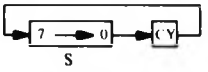
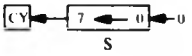
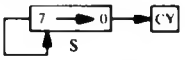
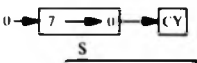
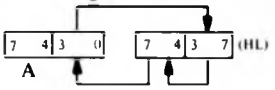
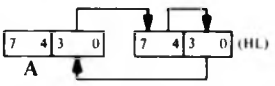
ZEICHENERKLÄRUNG

b	bezeichnet eine Bit-Position in einem Register oder einer Speicherstelle
cc	Statusbedingungscode ("Flag condition") Erlaubte Bedingungen: NZ: ungleich Null (= "Nonzero") Z: gleich Null (= "Zero") NC: Kein Übertrag (= "Non carry") C: Übertrag (= "Carry") PO: Ungerade oder kein Überlauf ("Parity Odd") PE: Gerade oder Überlauf ("Parity Even") P: Positiv N: Negativ
d	Zielregister (8 bit)
dd	16 bit-Zielregister oder Zieladresse im Speicher
e	8 bit vorzeichenbehaftetes Zweierkomplement der Distanz bei relativen Sprüngen oder indizierter Adressierung
L	bezeichnet die 8 speziellen Zieladressen in Seite 0 (dezimal 0, 8, 16, 32, 40, 48 und 56).
n	8 bit-Binärzahl.
nn	16 bit-Binärzahl.
r	allgemeines 8 bit-Register (A, B, C, D, E, H oder L)
s	8 bit Senderregister oder Speicherstelle.
s_b	ein Bit in einem bestimmten 8 bit-Register oder Speicherstelle.
ss	16 bit Senderegister oder Speicherstelle.
Index „L"	Niederwertige (= "Low order") 8 bits eines 16-bit-Registers
Index „H"	Höherwertige (= "High order") 8 bits eines 16-bit-Registers
( )	„Inhalt von . . . ." Zeichen zwischen den Klammern stellen einen Zeiger auf eine Speicherstelle oder ein I/O Port dar.

8 bit Register sind: A, B, C, D, E, H, L, I und R.
16 bit „Paare": AF, BC, DE und HL.
16 bit Register: SP, PC, IX und IY.

Als Adressierweisen kommen (auch Kombinationen) in Frage:
Direkt ("immediate")
Erweitert direkt ("immediate extended")
Modifizierte Seite Null ("Modified Page Zero")
Relativ ("relative")
Erweitert ("extended")
Indiziert ("indexed")
über Register:
impliziert ("implied")
indirekt über Register ("register indirect")
Adressierung eines Bits.

	Mnemo	Durchgeführte Operation	Bemerkungen
8 bit-Ladebefehle	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16 bit-Ladebefehle	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H$; $(SP-2) \leftarrow ss_L$	ss = BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP)$; $dd_H \leftarrow (SP+1)$	dd = BC, DE, HL, AF, IX, IY
Registeraustausch	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix} BC \\ DE \\ HL \end{pmatrix} \leftrightarrow \begin{pmatrix} BC' \\ DE' \\ HL' \end{pmatrix}$	
	EX (SP), ss	$(SP) \leftrightarrow ss_L$, $(SP+1) \leftrightarrow ss_H$	ss ≡ HL, IX, IY
Block-Suchbefehle	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
	CPI	A-(HL), HL ← HL+1 BC ← BC-1	
	CPIR	A-(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	A-(HL) sets the flags only. A is not affected
	CPD	A-(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1 BC ← BC-1, Repeat until BC= 0 or A = (HL)	
8 bit-arithmetische und logische Operationen	ADD s	A ← A + s	
	ADC s	A ← A + s + CY	CY is the carry flag
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ⊕ s	

	Mnemo	Durchgeführte Operation	Bemerkungen
16 bit-arithmetische Operationen	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d - 1	
	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
BCD-, Akku- und Flag-Operationen	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	A ← 00 – A	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
Verschiedene	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
	RLC s		
	RL s		
	RRC s		
	RR s		
	SLA s		s ≡ r, (HL) (IX+e), (IY+e)
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemo	Durchgeführte Operation	Bemerkungen
Bit Setzen, Rücksetzen, Testen	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag
	SET b, s	$s_b \leftarrow 1$	s ≡ r, (HL) (IX+e), (IY+e)
	RES b, s	$s_b \leftarrow 0$	
Ein/Ausgabe	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
Springbefehle	JP nn	PC ← nn	cc { NZ PO; Z PE; NC P; C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	
	JR kk, e	If condition kk is true PC ← PC + e, else continue	kk { NZ NC; Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
Unterprogramm-aufruf	CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L, PC ← nn	cc { NZ PO; Z PE; NC P; C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
Restarts	RST L	(SP-1) ← PC_H (SP-2) ← PC_L, PC_H ← 0 PC_L ← L	
Rücksprünge	RET	PC_L ← (SP), PC_H ← (SP+1)	cc { NZ PO; Z PE; NC P; C M
	RET cc	If condition cc is false continue, else same as RET	
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

Z80-CPU-Befehlssatz, alphabetisch sortiert

ALPHABETICAL ASSEMBLY MNEMONIC	OPERATION
ADC HL,ss	Add with Carry Reg. pair ss to HL
ADC A,s	Add with carry operand s to Acc.
ADD A,n	Add value n to Acc.
ADD A,r	Add Reg. r to Acc.
ADD A,(HL)	Add location (HL) to Acc.
ADD A,(IX+d)	Add location (IX+d) to Acc.
ADD A,(IY+d)	Add location (IY+d) to Acc.
ADD HL,ss	Add Reg. pair ss to HL
ADD IX,pp	Add Reg. pair pp to IX
ADD IY,rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b,(HL)	Test BIT b of location (HL)
BIT b,(IX+d)	Test BIT b of location (IX+d)
BIT b,(IY+d)	Test BIT b of location (IY+d)
BIT b,r	Test BIT b of Reg. r
CALL cc,nn	Call subroutine at location nn if condition cc is true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B≠0
EI	Enable interrupts
EX (SP),HL	Exchange the location (SP) and HL
EX (SP),IX	Exchange the location (SP) and IX
EX (SP),IY	Exchange the location (SP) and IY
EX AF,AF'	Exchange the contents of AF and AF'
EX DE,HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC,DE,HL with contents of BC',DE',HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A,(n)	Load the Acc. with input from device n
IN r,(C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditional Jump to (IY)
JP cc,nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JR C,e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JR NC,e	Jump relative to PC+e if carry=0
JR NZ,e	Jump relative to PC+e if non zero (Z=0)
JR Z,e	Jump relative to PC+e if zero (Z=1)
LD A,(BC)	Load Acc. with location (BC)
LD A,(DE)	Load Acc. with location (DE)
LD A,I	Load Acc. with I
LD A,(nn)	Load Acc. with location nn
LD A,R	Load Acc. with Reg. R
LD (BC),A	Load location (BC) with Acc.
LD (DE),A	Load location (DE) with Acc.
LD (HL),n	Load location (HL) with value n
LD dd,nn	Load Reg. pair dd with value nn
LD dd,(nn)	Load Reg. pair dd with location (nn)
LD HL,(nn)	Load HL with location (nn)
LD (HL),r	Load location (HL) with Reg. r
LD I,A	Load I with Acc.
LF IX,nn	Load IX with value nn
LD IX,(nn)	Load IX with location (nn)
LD (IX+d),n	Load location (IX+d) with value n
LD (IX+d),r	Load location (IX+d) with Reg. r
LD IY,nn	Load IY with value nn
LD IY,(nn)	Load IY with location (nn)
LD (IY+d),n	Load location (IY+d) with value n
LD (IY+d),r	Load location (IY+d) with Reg. r
LD (nn),A	Load location (nn) with Acc.
LD (nn),dd	Load location (nn) with Reg. pair dd
LD (nn),HL	Load location (nn) with HL
LD (nn),IX	Load location (nn) with IX
LD (nn),IY	Load location (nn) with IY
LD R,A	Load R with Acc.
LD r,(HL)	Load Reg. r with location (HL)
LD r,(IX+d)	Load Reg. r with location (IX+d)
LD r,(IY+d)	Load Reg. r with location (IY+d)
LD r,n	Load Reg. r with value n
LD r,r'	Load Reg. r with Reg. r'
LD SP,HL	Load SP with HL
LD SP,IX	Load SP with IX
LD SP,IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE,HL and BC
LDDR	Load location (DE) with location (HL), decrement DE,HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE,HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE,HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' of operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C),r	Load output port (C) with Reg. r
OUT (n),A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B
POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b,m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circclar
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A,s	Subtract operand s from Acc. with carry
SBC HL,ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b,(HL)	Set Bit b of location (HL)
SET b,(IX+d)	Set Bit b of location (IX+d)
SET b,(IY+d)	Set Bit b of location (IY+d)
SET b,r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmatic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

Befehlsvergleichsliste der Systeme Z80 und 8080 A

Opcode	8080A		Z80		Opcode	8080A		Z80	
00	NOP		NOP		30	----		JR	NC,disp
01	LXI	B,dddd	LD	BC,dddd	31	LXI	SP,dddd	LD	SP,dddd
02	STAX	B	LD	(BC),A	32	STA	adr	LD	(adr),A
03	INX	B	INC	BC	33	INX	SP	INC	SP
04	INR	B	INC	B	34	INR	M	INC	(HL)
05	DCR	B	DEC	B	35	DCR	M	DEC	(HL)
06	MVI	B,dd	LD	B,dd	36	MVI	M,dd	LD	(HL),dd
07	RLC		RLCA		37	STC		SCF	
08			EX	AF,AF'	38	----		JR	C,disp
09	DAD	B	ADD	HL,BC	39	DAD	SP	ADD	HL,SP
0A	LDAX	B	LD	A,(BC)	3A	LDA	adr	LD	A,(adr)
0B	DCX	B	DEC	BC	3B	DCX	SP	DEC	SP
0C	INR	C	INC	C	3C	INR	A	INC	A
0D	DCR	C	DEC	C	3D	DCR	A	DEC	A
0E	MVI	C,dd	LD	C,dd	3E	MVI	A,dd	LD	A,dd
0F	RRC		RRCA		3F	CMC		CCF	
10	----		DJNZ	disp	40	MOV	B,B	LD	B,B
11	LXI	D,dddd	LD	DE,dddd	41	MOV	B,C	LD	B,C
12	STAX	D	LD	(DE),A	42	MOV	B,D	LD	B,D
13	INX	D	INC	DE	43	MOV	B,E	LD	B,E
14	INR	D	INC	D	44	MOV	B,H	LD	B,H
15	DCR	D	DEC	D	45	MOV	B,L	LD	B,L
16	MVI	D,dd	LD	D,dd	46	MOV	B,M	LD	B,(HL)
17	RAL		RLA		47	MOV	B,A	LD	B,A
18	----		JR	disp	48	MOV	C,B	LD	C,B
19	DAD	D	ADD	HL,DE	49	MOV	C,C	LD	C,C
1A	LDAX	D	LD	A,(DE)	4A	MOV	C,D	LD	C,D
1B	DCX	D	DEC	DE	4B	MOV	C,E	LD	C,E
1C	INR	E	INC	E	4C	MOV	C,H	LD	C,H
1D	DCR	E	DEC	E	4D	MOV	C,L	LD	C,L
1E	MVI	E,dd	LD	E,dd	4E	MOV	C,M	LD	C,(HL)
1F	RAR		RRA		4F	MOV	C,A	LD	C,A
20	----		JR	NZ,disp	50	MOV	D,B	LD	D,B
21	LXI	H,dddd	LD	HL,dddd	51	MOV	D,C	LD	D,C
22	SHLD	adr	LD	(adr),HL	52	MOV	D,D	LD	D,D
23	INX	H	INC	HL	53	MOV	D,E	LD	D,E
24	INR	H	INC	H	54	MOV	D,H	LD	D,H
25	DCR	H	DEC	H	55	MOV	D,L	LD	D,L
26	MVI	H,dd	LD	H,dd	56	MOV	D,M	LD	D,(HL)
27	DAA		DAA		57	MOV	D,A	LD	D,A
28	----		JR	Z,disp	58	MOV	E,B	LD	E,B
29	DAD	H	ADD	HL,HL	59	MOV	E,C	LD	E,C
2A	LHLD	adr	LD	HL,(adr)	5A	MOV	E,D	LD	E,D
2B	DCX	H	DEC	HL	5B	MOV	E,E	LD	E,E
2C	INR	L	INC	L	5C	MOV	E,H	LD	E,H
2D	DCR	L	DEC	L	5D	MOV	E,L	LD	E,L
2E	MVI	L,dd	LD	L,dd	5E	MOV	E,M	LD	E,(HL)
2F	CMA		CPL		5F	MOV	E,A	LD	E,A

Opcode	8080A		Z80		Opcode	8080A		Z80	
60	MOV	H,B	LD	H,B	90	SUB	B	SUB	B
61	MOV	H,C	LD	H,C	91	SUB	C	SUB	C
62	MOV	H,D	LD	H,D	92	SUB	D	SUB	D
63	MOV	H,E	LD	H,E	93	SUB	E	SUB	E
64	MOV	H,H	LD	H,H	94	SUB	H	SUB	H
65	MOV	H,L	LD	H,L	95	SUB	L	SUB	L
66	MOV	H,M	LD	H,(HL)	96	SUB	M	SUB	(HL)
67	MOV	H,A	LD	H,A	97	SUB	A	SUB	A
68	MOV	L,B	LD	L,B	98	SBB	B	SBC	A,B
69	MOV	L,C	LD	L,C	99	SBB	C	SBC	A,C
6A	MOV	L,D	LD	L,D	9A	SBB	D	SBC	A,D
6B	MOV	L,E	LD	L,E	9B	SBB	E	SBC	A,E
6C	MOV	L,H	LD	L,H	9C	SBB	H	SBC	A,H
6D	MOV	L,L	LD	L,L	9D	SBB	L	SBC	A,L
6E	MOV	L,M	LD	L,(HL)	9E	SBB	M	SBC	A,(HL)
6F	MOV	L,A	LD	L,A	9F	SBB	A	SBC	A,A
70	MOV	M,B	LD	(HL),B	A0	ANA	B	AND	B
71	MOV	M,C	LD	(HL),C	A1	ANA	C	AND	C
72	MOV	M,D	LD	(HL),D	A2	ANA	D	AND	D
73	MOV	M,E	LD	(HL),E	A3	ANA	E	AND	E
74	MOV	M,H	LD	(HL),H	A4	ANA	H	AND	H
75	MOV	M,L	LD	(HL),L	A5	ANA	L	AND	L
76	HLT		HALT		A6	ANA	M	AND	(HL)
77	MOV	M,A	LD	(HL),A	A7	ANA	A	AND	A
78	MOV	A,B	LD	A,B	A8	XRA	B	XOR	B
79	MOV	A,C	LD	A,C	A9	XRA	C	XOR	C
7A	MOV	A,D	LD	A,D	AA	XRA	D	XOR	D
7B	MOV	A,E	LD	A,E	AB	XRA	E	XOR	E
7C	MOV	A,H	LD	A,H	AC	XRA	H	XOR	H
7D	MOV	A,L	LD	A,L	AD	XRA	L	XOR	L
7E	MOV	A,M	LD	A,(HL)	AE	XRA	M	XOR	(HL)
7F	MOV	A,A	LD	A,A	AF	XRA	A	XOR	A
80	ADD	B	ADD	A,B	B0	ORA	B	OR	B
81	ADD	C	ADD	A,C	B1	ORA	C	OR	C
82	ADD	D	ADD	A,D	B2	ORA	D	OR	D
83	ADD	E	ADD	A,E	B3	ORA	E	OR	E
84	ADD	H	ADD	A,H	B4	ORA	H	OR	H
85	ADD	L	ADD	A,L	B5	ORA	L	OR	L
86	ADD	M	ADD	A,(HL)	B6	ORA	M	OR	(HL)
87	ADD	A	ADD	A,A	B7	ORA	A	OR	A
88	ADC	B	ADC	A,B	B8	CMP	B	CP	B
89	ADC	C	ADC	A,C	B9	CMP	C	CP	C
8A	ADC	D	ADC	A,D	BA	CMP	D	CP	D
8B	ADC	E	ADC	A,E	BB	CMP	E	CP	E
8C	ADC	H	ADC	A,H	BC	CMP	H	CP	H
8D	ADC	L	ADC	A,L	BD	CMP	L	CP	L
8E	ADC	M	ADC	A,(HL)	BE	CMP	M	CP	(HL)
8F	ADC	A	ADC	A,A	BF	CMP	A	CP	A

Opcode	8080A		Z80	
C0	RNZ		RET	NZ
C1	POP	B	POP	BC
C2	JNZ	adr	JP	NZ,adr
C3	JMP	adr	JP	adr
C4	CNZ	adr	CALL	NZ,adr
C5	PUSH	B	PUSH	BC
C6	ADI	dd	ADD	A,dd
C7	RST	0	RST	0
C8	RZ		RET	Z
C9	RET		RET	
CA	JZ	adr	JP	Z,adr
CB	———		see below	
CC	CZ	adr	CALL	Z,adr
CD	CALL	adr	CALL	adr
CE	ACI	dd	ADC	A,dd
CF	RST	1	RST	8
D0	RNC		RET	NC
D1	POP	D	POP	DE
D2	JNC	adr	JP	NC,adr
D3	OUT	port	OUT	port,A
D4	CNC	adr	CALL	NC,adr
D5	PUSH	D	PUSH	DE
D6	SUI	dd	SUB	dd
D7	RST	2	RST	10H
D8	RC		RET	C
D9	———		EXX	
DA	JC	adr	JP	C,adr
DB	IN	port	IN	A,port
DC	CC	adr	CALL	C,adr
DD	———		see below	
DE	SBI	dd	SBC	A,dd
DF	RST	3	RST	18H
E0	RPO		RET	PO
E1	POP	H	POP	HL
E2	JPO	adr	JP	PO,adr
E3	XTHL		EX	(SP),HL
E4	CPO	adr	CALL	PO,adr
E5	PUSH	H	PUSH	HL
E6	ANI	dd	AND	dd
E7	RST	4	RST	20H
E8	RPE		RET	PE
E9	PCHL		JP	(HL)
EA	JPE	adr	JP	PE,adr
EB	XCHG		EX	DE,HL
EC	CPE	adr	CALL	PE,adr
ED	———		see below	
EE	XRI	dd	XOR	dd
EF	RST	5	RST	28H
F0	RP		RET	P
F1	POP	PSW	POP	AF
F2	JP	adr	JP	P,adr
F3	DI		DI	
F4	CP	adr	CALL	P,adr
F5	PUSH	PSW	PUSH	AF
F6	ORI	dd	OR	dd
F7	RST	6	RST	30H
F8	RM		RET	M
F9	SPHL		LD	SP,HL
FA	JM	adr	JP	M,adr
FB	EI		EI	
FC	CM	adr	CALL	M,adr
FD	———		see below	
FE	CPI	dd	CP	dd
FF	RST	7	RST	38H

Folgende Anweisungen sind nur im Befehlssatz des System Z80 implementiert

Opcode		
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
CB06	RLC	(HL)
CB07	RLC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
CB0E	RRC	(HL)
CB0F	RRC	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
CB16	RL	(HL)
CB17	RL	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
CB1E	RR	(HL)
CB1F	RR	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB26	SLA	(HL)
CB27	SLA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB2E	SRA	(HL)
CB2F	SRA	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
CB3E	SRL	(HL)
CB3F	SRL	A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB46	BIT	0,(HL)
CB47	BIT	0,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB4E	BIT	1,(HL)
CB4F	BIT	1,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB56	BIT	2,(HL)
CB57	BIT	2,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB5E	BIT	3,(HL)
CB5F	BIT	3,A

Opcode		
CBC0	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBC6	SET	0,(HL)
CBC7	SET	0,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBCE	SET	1,(HL)
CBCF	SET	1,A
CBD0	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBD6	SET	2,(HL)
CBD7	SET	2,A
CBD8	SET	3,B
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBDE	SET	3,(HL)
CBDF	SET	3,A
CBE0	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBE6	SET	4,(HL)
CBE7	SET	4,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBEE	SET	5,(HL)
CBEF	SET	5,A

Opcode		
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBF6	SET	6,(HL)
CBF7	SET	6,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CBFE	SET	7,(HL)
CBFF	SET	7,A
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD21	LD	IX,dddd
DD22	LD	(adr),IX
DD23	INC	IX
DD29	ADD	IX,IX
DD2A	LD	IX,(adr)
DD2B	DEC	IX
DD34	INC	(IX+offset)
DD35	DEC	(IX+offset)
DD36	LD	(IX+offset),dd
DD39	ADD	IX,SP
DD46	LD	B,(IX+offset)
DD4E	LD	C,(IX+offset)
DD56	LD	D,(IX+offset)
DD5E	LD	E,(IX+offset)
DD66	LD	H,(IX+offset)
DD6E	LD	L,(IX+offset)
DD70	LD	(IX+offset),B
DD71	LD	(IX+offset),C
DD72	LD	(IX+offset),D
DD73	LD	(IX+offset),E
DD74	LD	(IX+offset),H
DD75	LD	(IX+offset),L
DD77	LD	(IX+offset),A
DD7E	LD	A,(IX+offset)
DD86	ADD	A,(IX+offset)
DD8E	ADC	A,(IX+offset)

Opcode			Opcode		
CB60	BIT	4,B	CB90	RES	2,B
CB61	BIT	4,C	CB91	RES	2,C
CB62	BIT	4,D	CB92	RES	2,D
CB63	BIT	4,E	CB93	RES	2,E
CB64	BIT	4,H	CB94	RES	2,H
CB65	BIT	4,L	CB95	RES	2,L
CB66	BIT	4,(HL)	CB96	RES	2,(HL)
CB67	BIT	4,A	CB97	RES	2,A
CB68	BIT	5,B	CB98	RES	3,B
CB69	BIT	5,C	CB99	RES	3,C
CB6A	BIT	5,D	CB9A	RES	3,D
CB6B	BIT	5,E	CB9B	RES	3,E
CB6C	BIT	5,H	CB9C	RES	3,H
CB6D	BIT	5,L	CB9D	RES	3,L
CB6E	BIT	5,(HL)	CB9E	RES	3,(HL)
CB6F	BIT	5,A	CB9F	RES	3,A
CB70	BIT	6,B	CBA0	RES	4,B
CB71	BIT	6,C	CBA1	RES	4,C
CB72	BIT	6,D	CBA2	RES	4,D
CB73	BIT	6,E	CBA3	RES	4,E
CB74	BIT	6,H	CBA4	RES	4,H
CB75	BIT	6,L	CBA5	RES	4,L
CB76	BIT	6,(HL)	CBA6	RES	4,(HL)
CB77	BIT	6,A	CBA7	RES	4,A
CB78	BIT	7,B	CBA8	RES	5,B
CB79	BIT	7,C	CBA9	RES	5,C
CB7A	BIT	7,D	CBAA	RES	5,D
CB7B	BIT	7,E	CBAB	RES	5,E
CB7C	BIT	7,H	CBAC	RES	5,H
CB7D	BIT	7,L	CBAD	RES	5,L
CB7E	BIT	7,(HL)	CBAE	RES	5,(HL)
CB7F	BIT	7,A	CBAF	RES	5,A
CB80	RES	0,B	CBB0	RES	6,B
CB81	RES	0,C	CBB1	RES	6,C
CB82	RES	0,D	CBB2	RES	6,D
CB83	RES	0,E	CBB3	RES	6,E
CB84	RES	0,H	CBB4	RES	6,H
CB85	RES	0,L	CBB5	RES	6,L
CB86	RES	0,(HL)	CBB6	RES	6,(HL)
CB87	RES	0,A	CBB7	RES	6,A
CB88	RES	1,B	CBB8	RES	7,B
CB89	RES	1,C	CBB9	RES	7,C
CB8A	RES	1,D	CBBA	RES	7,D
CB8B	RES	1,E	CBBB	RES	7,E
CB8C	RES	1,H	CBBC	RES	7,H
CB8D	RES	1,L	CBBD	RES	7,L
CB8E	RES	1,(HL)	CBBE	RES	7,(HL)
CB8F	RES	1,A	CBBF	RES	7,A

Opcode		
DD96	SUB	(IX+offset)
DD9E	SBC	A,(IX+offset)
DDA6	AND	(IX+offset)
DDAE	XOR	(IX+offset)
DDB6	OR	(IX+offset)
DDBE	CP	(IX+offset)
DDCBof06	RLC	(IX+offset)
DDCBof0E	RRC	(IX+offset)
DDCBof16	RL	(IX+offset)
DDCBof1E	RR	(IX+offset)
DDCBof26	SLA	(IX+offset)
DDCBof2E	SRA	(IX+offset)
DDCBof3E	SRL	(IX+offset)
DDCBof46	BIT	0,(IX+offset)
DDCBof4E	BIT	1,(IX+offset)
DDCBof56	BIT	2,(IX+offset)
DDCBof5E	BIT	3,(IX+offset)
DDCBof66	BIT	4,(IX+offset)
DDCBof6E	BIT	5,(IX+offset)
DDCBof76	BIT	6,(IX+offset)
DDCBof7E	BIT	7,(IX+offset)
DDCBof86	RES	0,(IX+offset)
DDCBof8E	RES	1,(IX+offset)
DDCBof96	RES	2,(IX+offset)
DDCBof9E	RES	3,(IX+offset)
DDCBofA6	RES	4,(IX+offset)
DDCBofAE	RES	5,(IX+offset)
DDCBofB6	RES	6,(IX+offset)
DDCBofBE	RES	7,(IX+offset)
DDCBofC6	SET	0,(IX+offset)
DDCBofCE	SET	1,(IX+offset)
DDCBofD6	SET	2,(IX+offset)
DDCBofDE	SET	3,(IX+offset)
DDCBofE6	SET	4,(IX+offset)
DDCBofEE	SET	5,(IX+offset)
DDCBofF6	SET	6,(IX+offset)
DDCBofFE	SET	7,(IX+offset)

Opcode		
DDE1	POP	IX
DDE3	EX	(SP),IX
DDE5	PUSH	IX
DDE9	JP	(IX)
DDF9	LD	SP,IX
ED40	IN	B,(C)
ED41	OUT	(C),B
ED42	SBC	HL,BC
ED43	LD	(dddd),BC
ED44	NEG	
ED45	RETN	
ED46	IM	0
ED47	LD	I,A
ED48	IN	C,(C)
ED49	OUT	(C),C
ED4A	ADC	HL,BC
ED4B	LD	BC,(adr)
ED4D	RETI	
ED50	IN	D,(C)
ED51	OUT	(C),D
ED52	SBC	HL,DE
ED53	LD	(adr),DE
ED56	IM	1
ED57	LD	A,I
ED58	IN	E,(C)
ED59	OUT	(C),E
ED5A	ADC	HL,DE
ED5B	LD	DE,(adr)
ED5E	IM	2
ED60	IN	H,(C)
ED61	OUT	(C),H
ED62	SBC	HL,HL
ED67	RRD	
ED68	IN	L,(C)
ED69	OUT	(C),L
ED6A	ADC	HL,HL
ED6F	RLD	
ED72	SBC	HL,SP
ED73	LD	(adr),SP
ED78	IN	A,(C)
ED79	OUT	(C),A
ED7A	ADC	HL,SP
ED7B	LD	SP,(adr)
EDA0	LDI	
EDA1	CPI	
EDA2	INI	
EDA3	OUTI	
EDA8	LDD	
EDA9	CPD	
EDAA	IND	
EDAB	OUTD	

Opcode		
ED80	LDIR	
ED81	CPIR	
ED82	INIR	
ED83	OTIR	
ED88	LDDR	
ED89	CPDR	
EDBA	INDR	
EDBB	OTDR	
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD21	LD	IY,dddd
FD22	LD	(adr),IY
FD23	INC	IY
FD29	ADD	IY,IY
FD2A	LD	IY,(adr)
FD2B	DEC	IY
FD34	INC	(IY+offset)
FD35	DEC	(IY+offset)
FD36	LD	(IY+offset),dd
FD39	ADD	IY,SP
FD46	LD	B,(IY+offset)
FD4E	LD	C,(IY+offset)
FD56	LD	D,(IY+offset)
FD5E	LD	E,(IY+offset)
FD66	LD	H,(IY+offset)
FD6E	LD	L,(IY+offset)
FD70	LD	(IY+offset),B
FD71	LD	(IY+offset),C
FD72	LD	(IY+offset),D
FD73	LD	(IY+offset),E
FD74	LD	(IY+offset),H
FD75	LD	(IY+offset),L
FD77	LD	(IY+offset),A
FD7E	LD	A,(IY+offset)
FD86	ADD	A,(IY+offset)
FD8E	ADC	A,(IY+offset)
FD96	SUB	(IY+offset)
FD9E	SBC	A,(IY+offset)
FDA6	AND	(IY+offset)
FDAE	XOR	(IY+offset)
FDB6	OR	(IY+offset)
FDBE	CP	(IY+offset)

Opcode		
FDCBof06	RLC	(IY+offset)
FDCBof0E	RRC	(IY+offset)
FDCBof16	RL	(IY+offset)
FDCBof1E	RR	(IY+offset)
FDCBof26	SLA	(IY+offset)
FDCBof2E	SRA	(IY+offset)
FDCBof3E	SRL	(IY+offset)
FDCBof46	BIT	0,(IY+offset)
FDCBof4E	BIT	1,(IY+offset)
FDCBof56	BIT	2,(IY+offset)
FDCBof5E	BIT	3,(IY+offset)
FDCBof66	BIT	4,(IY+offset)
FDCBof6E	BIT	5,(IY+offset)
FDCBof76	BIT	6,(IY+offset)
FDCBof7E	BIT	7,(IY+offset)
FDCBof86	RES	0,(IY+offset)
FDCBof8E	RES	1,(IY+offset)
FDCBof96	RES	2,(IY+offset)
FDCBof9E	RES	3,(IY+offset)
FDCBofA6	RES	4,(IY+offset)
FDCBofAE	RES	5,(IY+offset)
FDCBofB6	RES	6,(IY+offset)
FDCBofBE	RES	7,(IY+offset)
FDCBofC6	SET	0,(IY+offset)
FDCBofCE	SET	1,(IY+offset)
FDCBofD6	SET	2,(IY+offset)
FDCBofDE	SET	3,(IY+offset)
FDCBofE6	SET	4,(IY+offset)
FDCBofEE	SET	5,(IY+offset)
FDCBofF6	SET	6,(IY+offset)
FDCBofFE	SET	7,(IY+offset)
FDE1	POP	IY
FDE3	EX	(SP),IY
FDE5	PUSH	IY
FDE9	JP	(IY)
FDF9	LD	SP,IY

□ Dynamische Kenndaten der Z80-CPU

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to MREQ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to IORQ, RD or WR (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from RD or WR	[3]		nsec	
	t_{caf}	Address Stable From RD or WR During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		260	nsec	C_L = 200pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\bar{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to WR (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to WR (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From WR	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
MREQ	$t_{DL\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ Low		100	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	MREQ Delay From Rising Edge of Clock, MREQ High		100	nsec	
	$t_{DH\bar{\Phi}(MR)}$	MREQ Delay From Falling Edge of Clock, MREQ High		100	nsec	
	$t_{w(MRL)}$	Pulse Width, MREQ Low	[8]		nsec	
	$t_{w(MRH)}$	Pulse Width, MREQ High	[9]		nsec	
IORQ	$t_{DL\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ Low		90	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ Low		110	nsec	
	$t_{DH\Phi(IR)}$	IORQ Delay From Rising Edge of Clock, IORQ High		100	nsec	
	$t_{DH\bar{\Phi}(IR)}$	IORQ Delay From Falling Edge of Clock, IORQ High		110	nsec	
RD	$t_{DL\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD Low		100	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD Low		130	nsec	
	$t_{DH\Phi(RD)}$	RD Delay From Rising Edge of Clock, RD High		100	nsec	
	$t_{DH\bar{\Phi}(RD)}$	RD Delay From Falling Edge of Clock, RD High		110	nsec	
WR	$t_{DL\Phi(WR)}$	WR Delay From Rising Edge of Clock, WR Low		80	nsec	C_L = 50pF
	$t_{DL\bar{\Phi}(WR)}$	WR Delay From Falling Edge of Clock, WR Low		90	nsec	
	$t_{DH\Phi(WR)}$	WR Delay From Falling Edge of Clock, WR High		100	nsec	
	$t_{w(WRL)}$	Pulse Width, WR Low	[10]		nsec	
M1	$t_{DL(M1)}$	M1 Delay From Rising Edge of Clock, M1 Low		130	nsec	C_L = 30pF
	$t_{DH(M1)}$	M1 Delay From Rising Edge of Clock, M1 High		130	nsec	
RFSH	$t_{DL(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH Low		180	nsec	C_L = 30pF
	$t_{DH(RF)}$	RFSH Delay From Rising Edge of Clock, RFSH High		150	nsec	
WAIT	$t_{s(WT)}$	WAIT Setup Time to Falling Edge of Clock	70		nsec	
HALT	$t_{D(HT)}$	HALT Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
INT	$t_{s(IT)}$	INT Setup Time to Rising Edge of Clock	80		nsec	
NMI	$t_{w(NML)}$	Pulse Width, NMI Low	80		nsec	
BUSRQ	$t_{s(BQ)}$	BUSRQ Setup Time to Rising Edge of Clock	80		nsec	
BUSAK	$t_{DL(BA)}$	BUSAK Delay From Rising Edge of Clock, BUSAK Low		120	nsec	C_L = 50pF
	$t_{DH(BA)}$	BUSAK Delay From Falling Edge of Clock, BUSAK High		110	nsec	
RESET	$t_{s(RS)}$	RESET Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float (MREQ, IORQ, RD and WR)		100	nsec	
	t_{mr}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 180$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 180$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 50$

[8] $t_{w(MRL)} = t_c - 40$

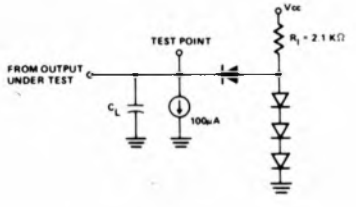
[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(WR)} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

NOTES:

A. Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.
B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
C. The RESET signal must be active for a minimum of 3 clock cycles.
D. Output Delay vs. Loaded Capacitance
   TA = 70°C Vcc = +5V ±5%
   (1) ΔC_L = +100pF ($A_0 - A_{15}$ and Control Signals), add 30 ns to timing shown.
E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum



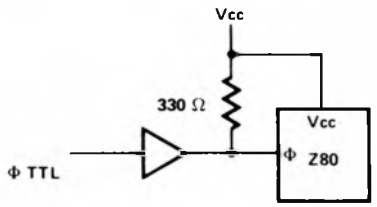
Load circuit for Output

☐ Kapazitäten

$T_A = 25°C$, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_Φ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

[1] Clock Driver



An external clock pull-up resistor of (330Ω) will meet both the A.C. and D.C. clock requirements.

☐ Statische Kenndaten

$T_A = 0°C$ to $70°C$, $V_{cc} = 5V \pm 5\%$ unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	$V_{cc}-0.6$ [1]		V_{cc}	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current			150	mA	t_c=400nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leqslant V_{IN} \leqslant V_{cc}$

☐ Absolute Grenzdaten

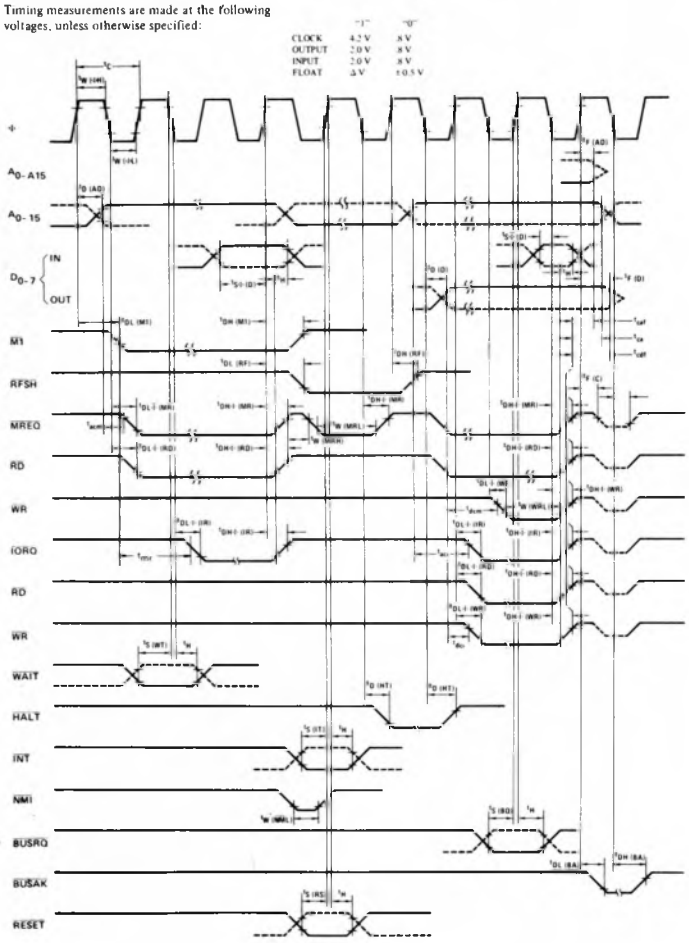
Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

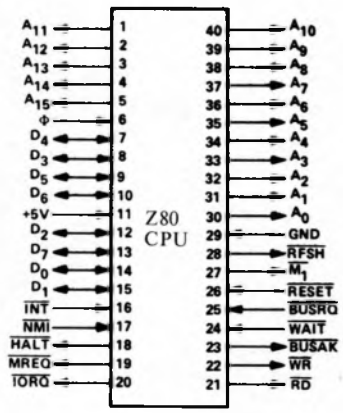
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

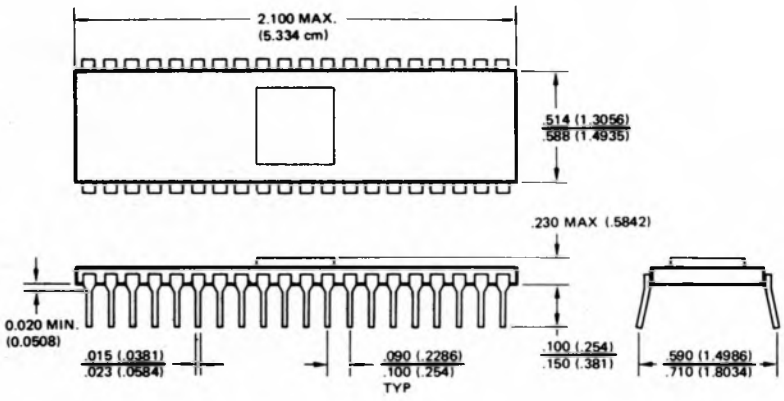
☐ Zeitverhaltèn der Z80-CPU

Timing measurements are made at the following voltages, unless otherwise specified:



☐ Pin-Belegung





*Dimensions for metric system are in parenthesis

☐ Bestellbezeichnung

Z80-CPU/PS Standardversion (0...70°C) in Plastikgehäuse U_B = 5 V ±5%

Z80-CPU/CS Standardversion (0...70°C) im Keramikgehäuse U_B = 5 V ±5%

Z80-CPU/CE Industrieversion mit erweitertem Temperaturbereich (—40° +85°C) im Keramikgehäuse U_B = 5 V ±5%

Z80-CPU/CM Militärische Version (—55°C...+125°C) im Keramikgehäuse U_B = 5 V ±10%

Z80-CPU/CM-B Militärische Version (—55°C...+125°C im Keramikgehäuse nach JAN 883-B

Z80-PIO

Zilog

PARALLEL-EIN-/ AUSGABE-BAUSTEIN

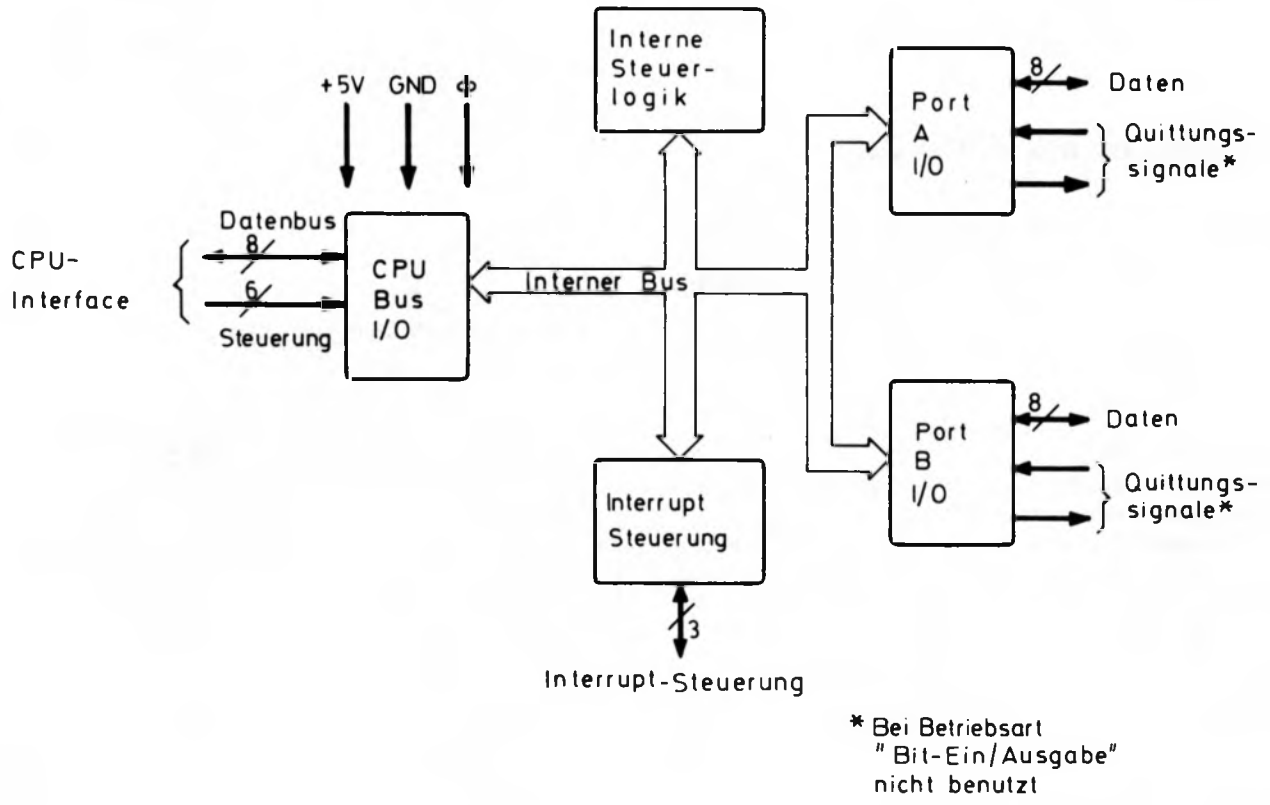


Bild 1: Z 80-PIO-Blockschaltbild

2. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Der Software-programmierbare Parallel-I/O (= „PIO")-Interfacebaustein enthält 2 TTL-kompatible Ports, über die der Datenverkehr zwischen dem Mikroprozessor und der Umwelt (= „Peripherie") abgewickelt wird. Er macht jegliche zusätzliche Interrupt-Steuerungs- und Priorisierungslogik überflüssig und erlaubt verschachtelten, priorisierten Interrupt beliebig vieler Ebenen im gesamten Speicherbereich.

☐ Aufbau

- N-Kanal Silicon-Gate-Depletion Load Technologie
- 40 Pin DIP-Gehäuse
- Stromversorgung über eine einzige 5V-Quelle
- 5V-Einphasen-Takt
- Zwei unabhängige 8 bit-bidirektionale Ports mit Einrichtung für Quittungsbetrieb ("Handshaking")

☐ Kenndaten des Z80-PIO

- Quittungsbetrieb im Interrupt-Mode zur schnellen Anforderungsbearbeitung
- Jedes Port kann in einer der folgenden 4 Betriebsarten arbeiten:
  - ☐ Byte-Ausgabe
  - ☐ Byte-Eingabe
  - ☐ Byte-Ein/Ausgabe (bidirektionaler Betrieb, nur bei Port A)
  - ☐ Bit-Ein/Ausgabe
- Unter Zustandsbedingungen des Peripheren Geräts programmierbare Interruptbearbeitung
- Automatische Interrupt-Vektorerzeugung und Prioritätscodierung ohne zusätzlichen Schaltungsaufwand durch Kaskadierung der Bausteine ("Daisy chain priority interrupt logic"); vgl. Darstellung Bild 4
- 8 Ausgänge für den direkten Anschluß von Darlington-Transistoren ausgelegt (Push-Pull).
- Alle Ein- und Ausgänge voll TTL-kompatibel.

Bild 1 zeigt ein Blockschaltbild des PIO.
Der Baustein umfaßt — ein Interface zur CPU-Anschaltung
— interne Steuerlogik
— Logik für I/O-Port A
— Logik für I/O-Port B
— Interrupt-Steuerlogik

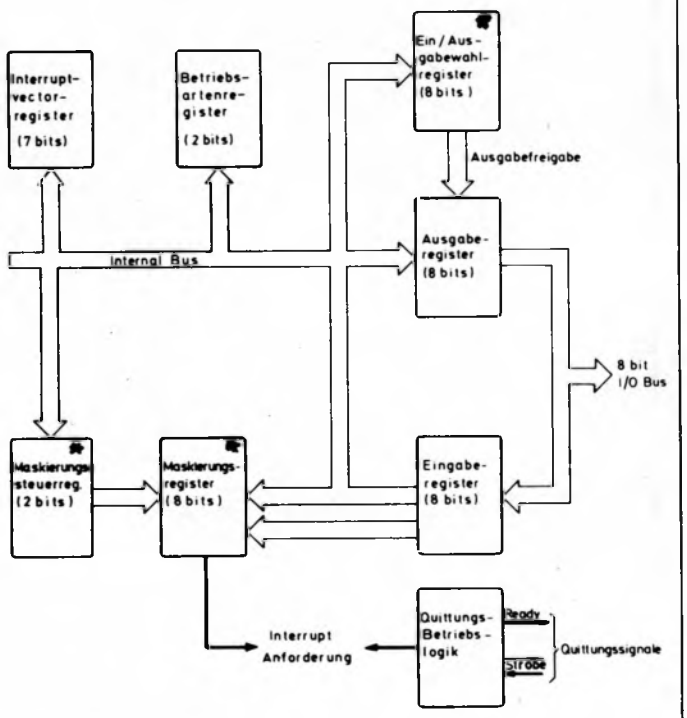
Die I/O-Port-Logik selbst besteht aus 6 Registern mit Quittungsbetrieb-Steuerlogik (siehe Bild 2), und zwar:

- ☐ ein 2 bit-Betriebsarten-Register
- ☐ ein 8 bit-Ausgabe-Register
- ☐ ein 8 bit-Eingabe-Register
- ☐ ein 2 bit-Maskierungs-Steuerungsregister } nur bei Betriebsart Bit-I/O benutzt!
- ☐ ein 8 bit-Maskierungs-Register
- ☐ ein 8 bit-Ein/Ausgabe-Wahl-Register

Beschreibung der einzelnen Register

- 2 bit-Betriebsarten-Register: Wird von der CPU zur Festlegung der Betriebsart (Byte-Ausgabe, Byte-Eingabe, Byte-Ein/Ausgabe, bzw. Bit-Ein/Ausgabe) geladen.
- 8 bit-Ausgabe-Register: Für Datenübertragung von der CPU an periphere Schaltungen
- 8 bit-Eingabe-Register: Für Datenübertragung von peripheren Schaltungen an die CPU
- 2 bit-Maskierungs-Steuerregister: Wird von der CPU zur Definition der aktiven Zustände der Anschlüsse einer peripheren Schaltung (Low oder High) und zur Festlegung der Interruptsignal-Erzeugung (Ausgabe des Interrupt-Signals, wenn **alle** unmaskierten PIO-Anschlüsse aktiv sind (UND-Bedingung), bzw. wenn **irgendeiner** der unmaskierten PIO-Anschlüsse aktiv ist (ODER-Bedingung) geladen
- 8 bit-Maskierungsregister: Wird von der CPU geladen; sein Inhalt legt die Port-Anschlüsse fest, die für die Erzeugung einer Interrupt-Anforderung maßgeblich sind

Bild 2: Blockschaltbild eines Ports



* Wird nur in Betriebsart Bit-Ein/Ausgabe für die Erzeugung einer Interrupt-Anforderung verwendet.

- 8 bit-Ein/Ausgabewahl-Register: Wird bei Betriebsart „Bit-Ein/Ausgabe" von der CPU geladen. Sein Inhalt legt fest, welche Port-Anschlüsse als Eingänge und welche als Ausgänge verwendet werden.

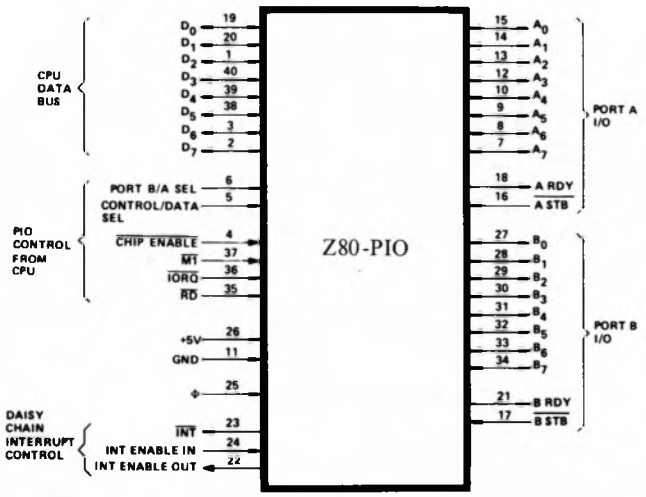


Bild 3: Pinbelegung

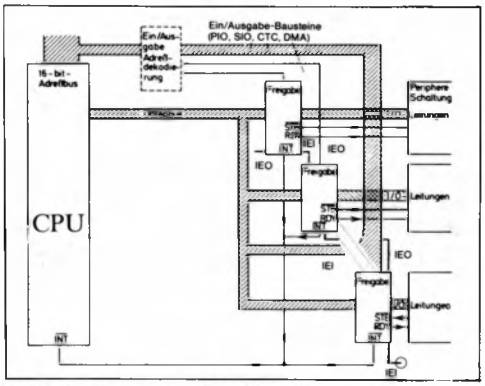


Bild 4: Interrupt-Architektur

☐ Z80-PIO-Pinbelegung

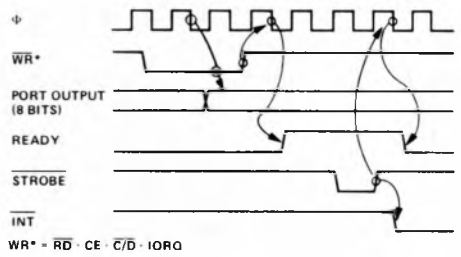
Pin	Funktion	Kommentar
D0...D7	Datenbus	Tri-State Ein/Ausgänge, mit Z80-CPU-Datenbus verbunden
B/A SEL	Portauswahl (= "Port B or A Select")	Eingang, High-aktiv
C/D SEL	Schalteingang Steuerwort/Datenwort (= "Control or Data Select")	Eingang, High-aktiv
$\overline{CE}$	Bausteinfreigabe (= "Chip enable")	Eingang, Low-aktiv
$\overline{\Phi}$	Systemtakt	Eingang, mit Z80-CPU-Systemtakt zu verbinden
$\overline{M1}$	Maschinenzyklus 1 (= "Machine Cycle One Signal")	Eingang, Low-aktiv, mit $\overline{M1}$ der Z80-CPU zu verbinden
$\overline{IORQ}$	Ein/Ausgabe-Anforderung (= "In/Output-Request")	Eingang, Low-aktiv; mit $\overline{IORQ}$ der Z80-CPU zu verbinden
$\overline{RD}$	Lesen (= "Read")	Eingang, Low aktiv, mit $\overline{RD}$ der Z80-CPU zu verbinden
IEI	Interrupt-Freigabe Eingang (= "Interrupt Enable In")	Eingang, High-aktiv
IEO	Interrupt-Freisignal Ausgang (= "Interrupt Enable Out")	Ausgang, High-aktiv
$\overline{INT}$	Interrupt-Anforderung (= "Interrupt Request")	Ausgang, Low-aktiv, Open Drain
A0...A7	A-Ein/Ausgänge	Bidirektional, Tristate, Anschlüsse des A-Ports
$\overline{ASTB}$	A-Trigger (= "Port A Strobe Pulse")	Eingang, Low-aktiv, für Triggerimpuls von der peripheren Elektronik zur Datenübernahme
ARDY	Quittung (= "A-Ready")	Ausgang, High-aktiv Quittierung des ASTB-Empfangs
B0...B7	B-Ein/Ausgänge	Bidirektional, Tristate, Anschlüsse des B-Ports
$\overline{BSTB}$	B-Trigger (= "Port B Strobe Pulse")	Eingang, Low-aktiv, für Triggerimpuls von der peripheren Elektronik zur Datenübernahme
BRDY	Quittung (= "B-Ready")	Ausgang, High-aktiv Quittierung des $\overline{BSTB}$-Empfangs

☐ Zeitdiagramme

Betriebsart Ausgabe (= Betriebsart 0)

Bei der Ausführung eines Ausgabebefehls durch die CPU veranlaßt die steigende Flanke des WR-Signals die Übernahme der Daten von der CPU über den Datenbus in das ausgewählte Ausgabe-Register. Der WRITE-Impuls setzt das READY-Bit ("READY-Flag") nach der fallenden Flanke von Φ als Signal dafür, daß nun die Daten im Register abrufbereit sind. Das READY-Bit bleibt aktiv bis zur steigenden Flanke des Taktimpulses ("$\overline{STROBE}$"), die am PIO anliegt, sobald die Daten von der peripheren Schaltung übernommen sind (= „Quittung").

Die steigende Flanke des Taktsignals ($\overline{STROBE}$) erzeugt ein $\overline{INT}$-Signal, falls das Unterbrechungsfreigabe-Flip-Flop ("Interrupt-Enable-Flip-Flop") gesetzt ist und die anfordernde periphere Schaltung die momentan höchste Priorität hat.

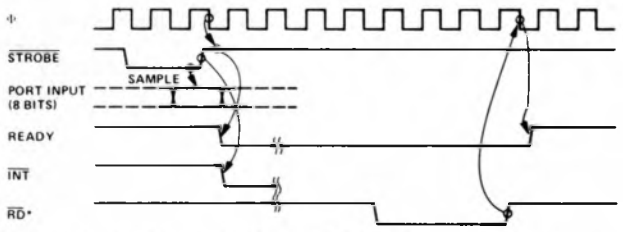


Betriebsart Eingabe (= Betriebsart 1)

Bei der fallenden Flanke von $\overline{\text{STROBE}}$ werden die Daten des ausgewählten Ports in das Eingaberegister gebracht.

Die nächste steigende Flanke von $\overline{\text{STROBE}}$ aktiviert $\overline{\text{INT}}$, falls der Interrupt freigegeben ist und höchste Priorität vorliegt. Die folgende fallende Flanke von Φ bringt READY in den inaktiven Zustand als Zeichen dafür, daß im Eingaberegister Daten anstehen und keine weiteren Daten eingeschrieben werden sollen, bis der Eingabepuffer von der CPU gelesen ist.

Nach dem Lesevorgang bewirkt die steigende Flanke von $\overline{\text{RD}}$ ein Setzen des READY-Signals zum Zeitpunkt der nächsten fallenden Flanke von Φ. Jetzt kann das PIO neue Daten von der angeschlossenen peripheren Schaltung empfangen.



$\overline{RD}^* = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$ MODE 1 (INPUT) TIMING

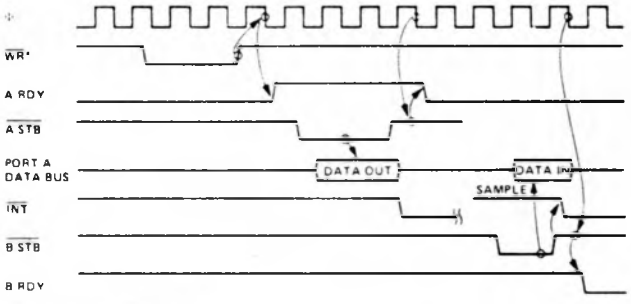
Betriebsart Byte-Ein/Ausgabe (bidirektional) (Betriebsart 2)

ist eine Kombination der Betriebsarten „Byte-Eingabe" und „Byte-Ausgabe", wobei alle 4 Quittungs-(="Handshaking")-Leitungen des Bausteins und die 8 Datenleitungen des Ports A verwendet werden.

Die beiden Quittungsleitungen von Port A werden hier für die Ausgabe-, die von Port B für die Eingabesteuerung benützt.

Eine Datenausgabe an das Port A darf nur während $\overline{\text{ASTB}}$ = Low erfolgen. Seine steigende Flanke kann zur Übergabe der Daten an die periphere Schaltung erfolgen.

Bei Wahl von Betriebsart "Byte-Ein/Ausgabe" für das Port A muß Port B in der Betriebsart „Bit-Ein/Ausgabe" benutzt werden.

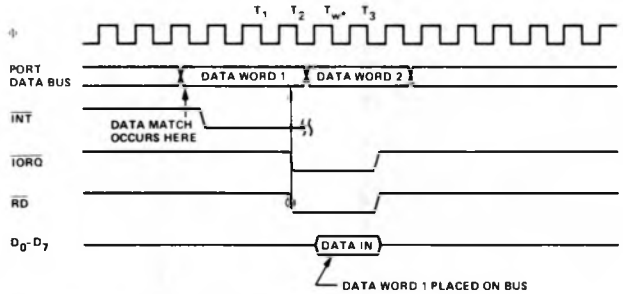


$\overline{WR}^* = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

Betriebsart Bit-Ein/Ausgabe (= Betriebsart 3)

Hier wird nicht mit Quittungssignalen gearbeitet, d.h. eine Eingabe oder eine Ausgabe von Daten kann zu jedem beliebigen Zeitpunkt erfolgen. Beim Ausgeben werden die Daten nach dem gleichen Zeitschema wie bei der Byte-Ausgabe im Ausgabe-Register abgelegt.

Bei der Eingabe sind die der CPU übergebenen Daten zusammengesetzt aus Daten aus dem Eingaberegister (das ist bei den Bits der Fall, die vom Ein/Ausgabewahl-Register durch Einsen als Eingabeleitungen definiert wurden) und aus dem Inhalt des Ausgaberegisters (das ist bei den Bits der Fall, die vom Ein/Ausgabewahlregister als Ausgänge definiert wurden).



* Timing Diagram Refers to Bit Mode Read.

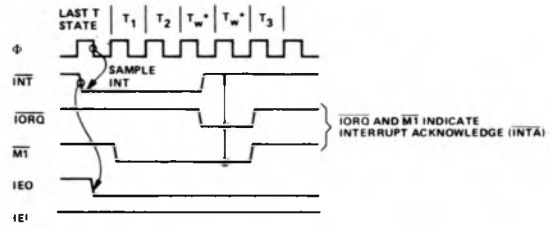
☐ Interrupt-Bearbeitung

Die Steuerlogik einer peripheren Schaltung darf während M1 ihren Interrupt-Freigabezustand nicht ändern, wodurch es möglich wird, das Interruptfreigabe-Signal durch die Prioritätsbestimmende Kaskadierung ("Daisy chain priority interrupt logic") der einzelnen PIO's durchzuschalten.

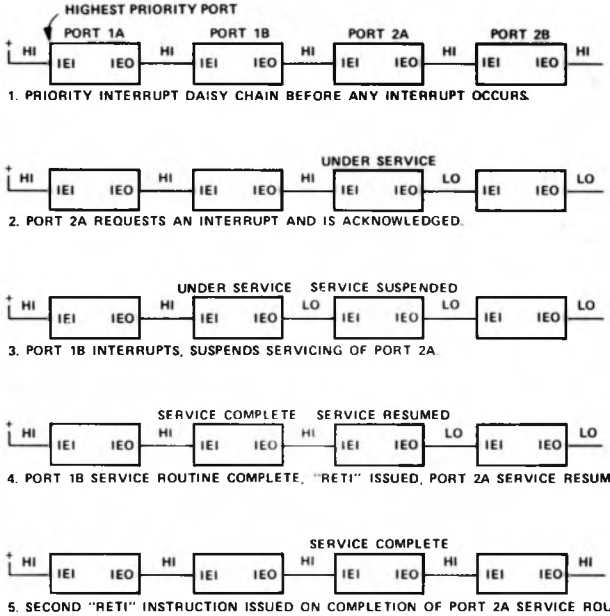
Diejenige periphere Schaltung, die IEI High und IEO Low während INTA liefert, bewirkt, daß zu diesem Zeitpunkt ein vorprogrammierter 8 bit-Interrupt-Vektor auf den Datenbus gelegt wird.

IEO bleibt Low, bis von der CPU eine RETI (= RETurn from Interrupt = Rücksprung aus Interrupt-Anforderungs-Bedienroutine)-Anweisung während IEI = High ausgeführt wird.

Die 2 Byte-RETI-Anweisung wird hierfür vom PIO intern hardwaremäßig dekodiert.



Die Priorisierung erfolgt nach folgendem Schema:



Laden des Interrupt-Vektors

Zur Adressierung der richtigen Interrupt-Anforderungs-Bedienroutine muß der CPU vom PIO ein Interrupt-Vektor zugeleitet werden.

Dieser 8 bit-Vektor wird während dem Interrupt-Bestätigungs-Zyklus (="Interrupt Acknowledge Cycle") vom Baustein mit der momentan höchsten Priorität auf den Datenbus gelegt.

Vor Auftreten der ersten Interrupt-Anforderung muß der jeweils gewünschte Vektor von der CPU in den PIO durch Angabe eines Steuerwortes an das betreffende Port mit folgendem Format geschrieben werden:

D7	D6	D5	D4	D3	D2	D1	D0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

identifiziert das Steuerwort als Interrupt-Vektor

Auswahl der gewünschten Betriebsart

Hierfür wird das 2 Bit-Betriebsartauswahl-Register über die höchstwertigen 2 Bits (M_1 und M_0) eines Steuerworts, das von der CPU an das betreffende Port ausgegeben wird, gesetzt.

Format dieses Steuerworts:

D7	D6	D5	D4	D3	D2	D1	D0
M_1	M_0	X	X	1	1	1	1

Betriebsart | nicht benutzt | identifiziert das Steuerwort als Betriebsartenauswahl-Wort

Betriebsart	M_1	M_0
Byte-Ausgabe	0	0
Byte-Eingabe	0	1
Byte-Ein/Ausgabe	1	0
Bit-Ein/Ausgabe	1	1

Sondereinrichtungen bei der Betriebsart Bit-Ein/Ausgabe

In dieser Betriebsart ist nach der Ausgabe des Betriebsart-Auswahl-Steuerwortes von der CPU ein weiteres Steuerwort auszugeben, das die einzelnen Anschlüsse des betreffenden Ports als Eingänge bzw. als Ausgänge definiert.
Eine „1" in diesem Steuerwort bedeutet, daß der zugehörige PIO-Anschluß einen Eingang darstellt, eine 0 bedeutet, daß der Anschluß ab dann als Ausgang benutzt werden kann.

Format:

D7	D6	D5	D4	D3	D2	D1	D0
I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

Interrupt-Steuerung

Bits	Wert	Bedeutung
7	0	Interruptflipflop rückgesetzt, Interrupt-Anforderungen werden nicht angenommen
7	1	Interruptflipflop gesetzt, Interrupt-Anforderungen werden bearbeitet
6, 5, 4	X	Werden nur bei Betriebsart Bit-Ein/Ausgabe benutzt, in allen anderen Betriebsarten ignoriert
3, 2, 1, 0	0111	identifizieren das Steuerwort als Interrupt-Steuerwort

Format

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt Freigabe	UND/ ODER	High/ Low	nächstes Steuerwort ist Maske	0	1	1	1

nur in Betriebsart 3 benutzt | bedeutet Interruptsteuerwort

Falls in diesem Steuerwort das Bit D4 = high war (= „nächstes Steuerwort ist Maske"), muß ein weiteres Steuerwort zur Maskierung an das Port ausgegeben werden.

Sein Format ist:

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Die Maske wirkt so, daß nur Anschlüsse mit Maskierungsbit $MB_n = 0$ zur Erzeugung einer Interrupt-Anforderung herangezogen werden.

Das Interrupt-Freigabe-FlipFlop ist durch folgendes Steuerwort zu beeinflussen:

D7	D6	D5	D4	D3	D2	D1	D0
Int Freigabe	X	X	X	0	0	1	1

Bei „Int Freigabe" = 1 wird die Interrupt-Anforderung einer peripheren Schaltung bearbeitet, bei „Int-Freigabe" = 0 wird eine Interrupt-Anforderung ignoriert.

☐ Dynamische Kenndaten

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{W(\Phi H)}$ $t_{W(\Phi L)}$ t_r, t_f	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	400 170 170	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
CS, $\overline{CE}$ ETC.	$t_{S\Phi}$ (CS)	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	280		nsec	
D_0-D_7	t_{DR} (D) $t_{S\Phi}$ (D) t_{DI} (D) t_F (D)	Data Output Delay from Falling Edge of $\overline{RD}$ Data Set-Up Time to Rising Edge of ΦDuring Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle. Delay to Floating Bus (Output Buffer Disable Time)	50	430 340 160	nsec nsec nsec nsec	[2] C_L = 50 pf [3]
IEI	t_S (IEI)	IEI Set-Up Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	t_{DH} (IO) t_{DL} (IO) t_{DM} (IO)	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		210 190 300	nsec nsec nsec	[5] [5] C_L = 50 pf [5]
$\overline{IORQ}$	$t_{S\Phi}$ (IR)	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	250		nsec	
$\overline{M1}$	$t_{S\Phi}$ (M1)	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B.	210		nsec	
$\overline{RD}$	$t_{S\Phi}$ (RD)	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	240		nsec	
A_0-A_7, B_0-B_7	t_S (PD) t_{DS} (PD) t_F (PD) t_{DI} (PD)	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (Mode 1) Port Data Output Delay from Falling Edge of $\overline{STROBE}$ (Mode 2) Delay to Floating Port Data Bus from Rising Edge of $\overline{STROBE}$ (Mode 2) Port Data Stable from Rising Edge of $\overline{IORQ}$ During WR Cycle (Mode 0)	260	 230 200 200	nsec nsec nsec nsec	 [5] C_L = 50 pf [5]
$\overline{ASTB}$, $\overline{BSTB}$	t_W (ST)	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	t_D (IT) t_D (IT3)	$\overline{INT}$ Delay Time from Rising Edge of $\overline{STROBE}$ $\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		490 420	nsec nsec	
ARDY, BRDY	t_{DH} (RY) t_{DL} (RY)	Ready Response Time from Rising Edge of $\overline{IORQ}$ Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 460 t_c+ 400	nsec nsec	[5] C_L = 50 pf [5]

A. 2.5 t_c > (N-2) t_{DL} (IO) + t_{DM} (IO) + t_S (IEI) + TTL Buffer Delay, if any

B. $\overline{M1}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_{W(\Phi H)} + t_{W(\Phi L)} + t_r + t_f$

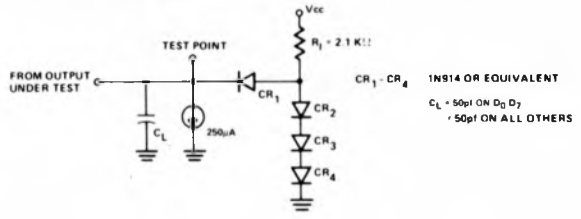
[2] Increase t_{DR} (D) by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[3] Increase t_{DI} (D) by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[4] For Mode 2: $t_{W(ST)} > t_{S(PD)}$

[5] Increase these values by 2 nsec for each 10 pf increase in loading up to 100 pf max.

Output load circuit.



☐ Kapazitäten

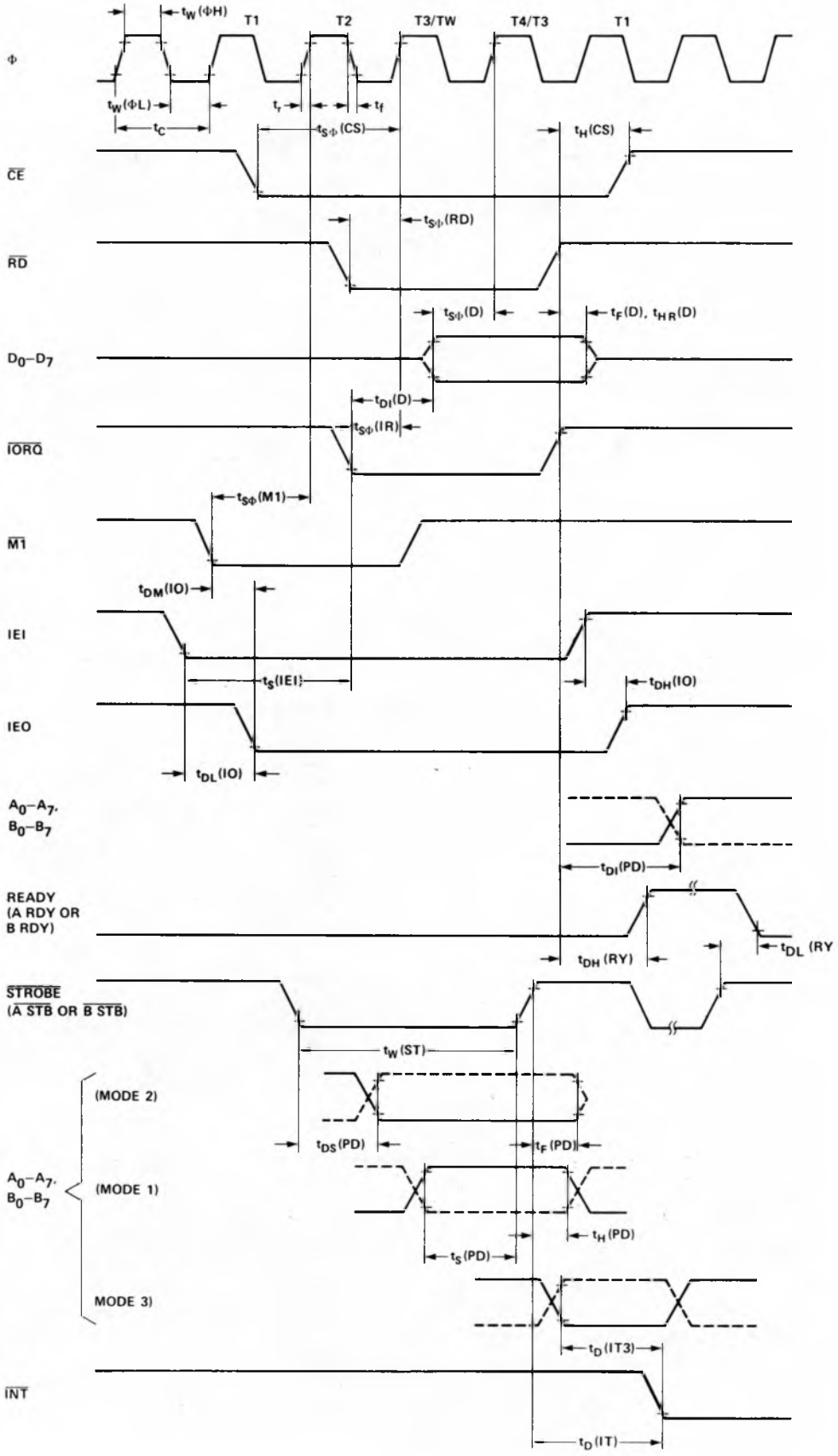
TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	10	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

Zeitverhalten des Z80-PIO

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV =	+0.5V



☐ Grenzdaten

Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.6 W

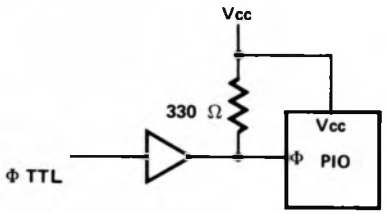
***Comment**
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

☐ Statische Kenndaten

TA = 0° C to 70° C, Vcc = 5 V ± 5% unless otherwise specified

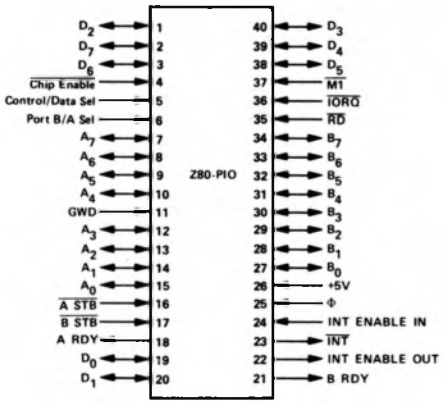
Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6	Vcc+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	Vcc	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} - 250 µA
I_{CC}	Power Supply Current		70	mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to Vcc
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	V_{OUT} = 2.4 to Vcc
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	µA	$0 \leqslant V_{IN} \leqslant Vcc$
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5 V Port B Only

[1] Clock Driver

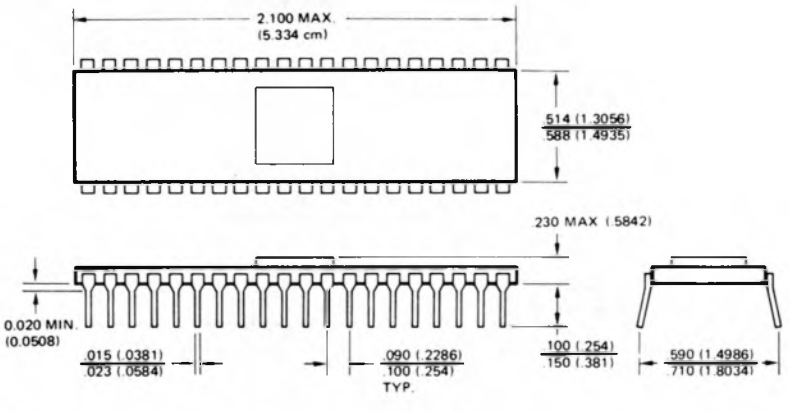


The external pull-up resistor (330 Ω) can satisfy the clock AC and DC requirements.

☐ Pin-Belegung



☐ Gehäuseabmessungen



*Dimensions for metric system are in parenthesis

☐ Bestellbezeichnung

Z80-PIO/PS	Standardversion (0...70°C) in Plastikgehäuse U_B = 5 V ± 5%
Z80-PIO/CS	Standardversion (0...70°C) im Keramikgehäuse U_B = 5 V ± 5%
Z80-PIO/CE	Industrieversion mit erweitertem Temperaturbereich (—40° + 85°C) im Keramikgehäuse U_B = 5 V ± 5%
Z80-PIO/CM	Militärische Version (—55°C...+ 125°C) im Keramikgehäuse U_B = 5 V ± 10%
Z80-PIO/CM-B	Militärische Version (—55°C...+125°C) im Keramikgehäuse nach JAN 883-B

Z80-CTC

Zilog

ZÄHLER/ZEITGEBER-BAUSTEIN

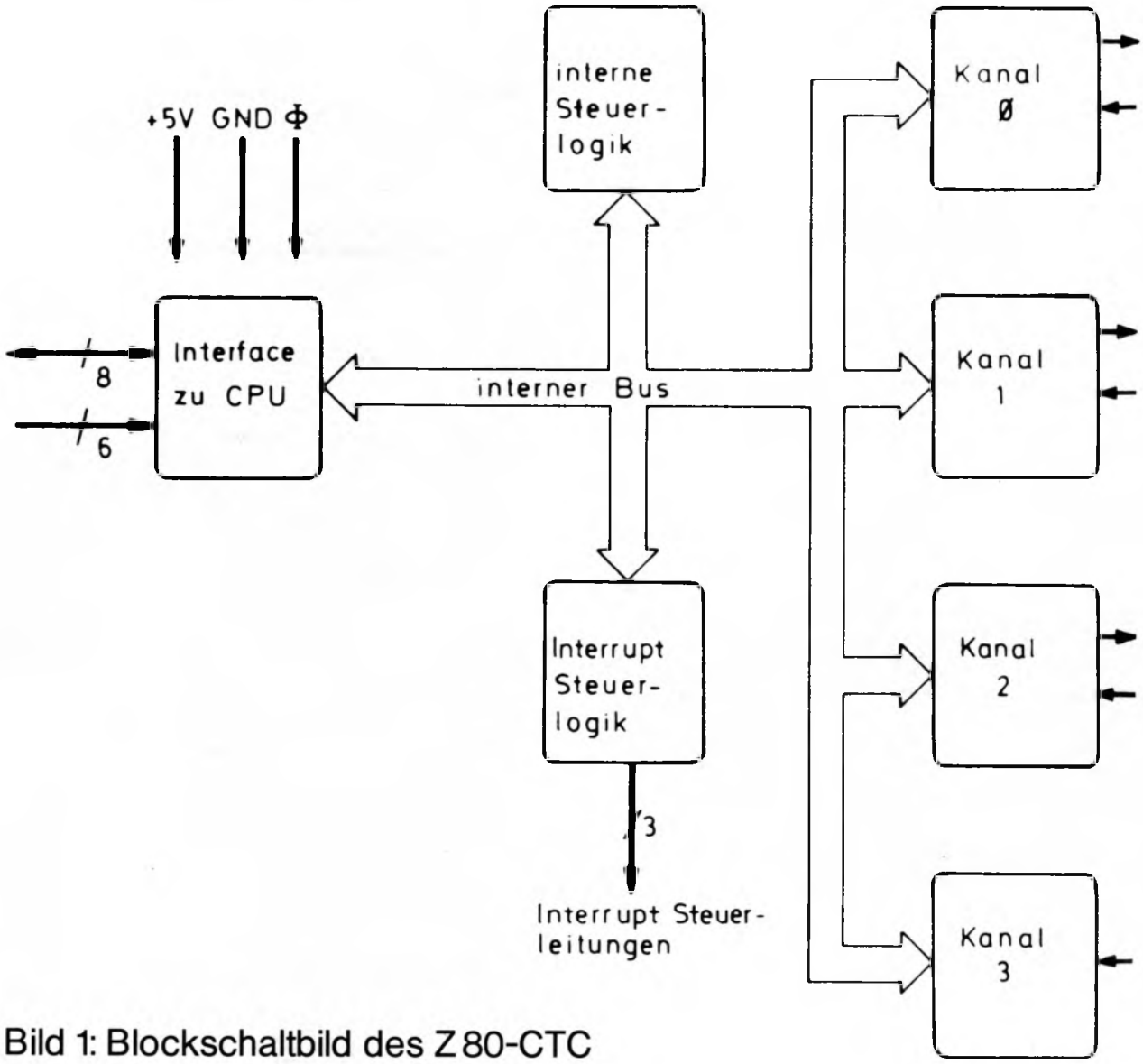


Bild 1: Blockschaltbild des Z80-CTC

2. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

□ Vorbemerkung

Der Baustein Z80 -CTC ist ein Zähler/Zeitgeber-Baustein (="Counter Timer Circuit") für dieses System, der über 4 voneinander unabhängige Software — programmierbare Zähler/Zeitgeber verfügt. Er macht jegliche zusätzliche Interrupt-Steuerungs- und Priorisierungslogik überflüssig und erlaubt verschachtelten, priorisierten Interrupt beliebig vieler Ebenen im gesamten Speicherbereich.

□ Technische Einzelheiten

- N-Kanal-Silicon Gate Depletion Load-Technologie
- 28 Pin DIP-Gehäuse
- Stromversorgung über eine einzige +5 V-Versorgungsspannung
- Einphasen -5 V-Takt
- 4 voneinander unabhängige, software-programmierbare 8 bit Zähler/16 bit Zeitgeber-Kanäle
- jeder dieser Kanäle wahlweise als Zähler oder Zeitgeber verwendbar
- programmierbare Interrupts bei Erreichen von programmmäßig festlegbaren Zähler- oder Zeitgeber-Werten.
- Rückwärtszähler hält Anzahl der bis Null auszuführenden Zählschritte auslesebereit
- Vorteiler durch 16 oder 256 für jeden Zeitgeber-Kanal
- Zeitgeber kann wahlweise durch einen positiven oder negativen Triggerimpuls gestartet werden.
- Die Ausleseausgänge von drei Kanälen sind zum direkten Anschluß von Darlington-Transistoren ausgelegt (Push-Pull).
- Automatische Interrupt-Vektorerzeugung und Prioritätscodierung ohne zusätzlichen Schaltungsaufwand durch Kaskadierung der Bausteine (="Daisy chain priority interrupt logic").
- Alle Ein- und Ausgänge voll TTL-kompatibel
- Maximale Zählfrequenz in Betriebsart „Zähler" =f Φ/2

□ Architektur des Z80-CTC

Ein Blockdiagramm Z80 -CTC ist im Bild 1 wiedergegeben: Die Funktionseinheiten des Bausteins sind

— 4 Zähler/Zeitgeber-Kanäle
— Interface zu Daten- und Steuerbus der Z80-CPU
— interne Steuerlogik und
— Interrupt-Steuerlogik

Jedem Zähler/Zeitgeber-Kanal ist ein Interruptvektor zugeordnet, wobei der Interrupt-Prioritätsrang durch die Kanalnummer mit Nr. 0 als höchste Priorität bestimmt ist.

Jeder der Kanäle besteht aus 2 Registern, (ein 8 bit Zeitkonstantenregister und ein 8 bit Kanal-Steuerregister), 2 Zählern (ein 8 bit Rückwärtszähler und ein auf den Wert „16" oder „256" einstellbaren 8 bit Vorteiler) und wiederum einer eigenen Steuerlogik (Bild 2).

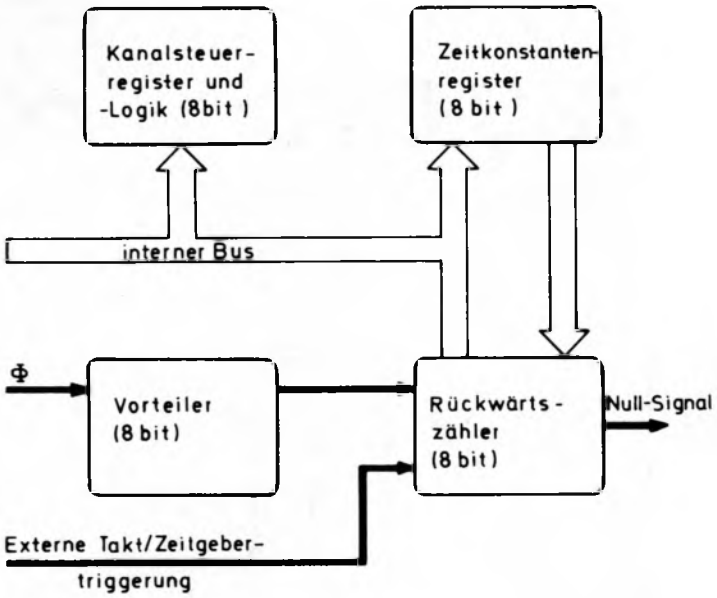
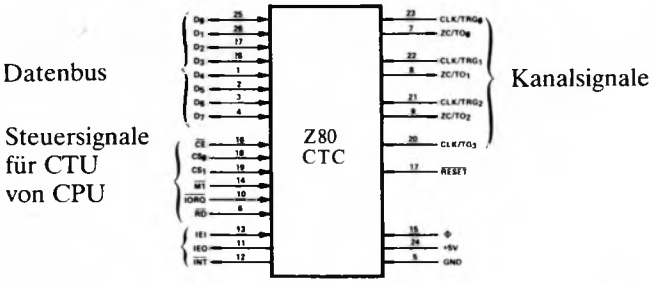


Bild 2: Architektur eines Kanals

Die Funktionen der einzelnen Einheiten sind folgende:

— **Das Zeitkonstantenregister** (8 bit) wird vom Mikroprozessor zum Initialisieren und Wiedersetzen des Rückwärtszählers beim Erreichen des Zählerstandes Null geladen.
— **Das Kanalsteuerregister** (8 bit) wird vom Mikroprozessor zur Bestimmung der Kanalbetriebsart geladen.
— **Der Rückwärtszähler** (8 bit) wird entweder auf Veranlassung des Anwenderprogramms oder automatisch beim Erreichen des Zählerstandes Null auf den Wert des Zeitkonstantenregisters gesetzt. Der Zähler wird in der Kanal-Betriebsart „Zeitgeber" über den Vorteiler, in der Kanal-Betriebsart „Zähler" durch ein Signal am Eingang CLK/TRIG dekrementiert. Der momentane Wert des Rückwärtszählers kann zu jedem beliebigen Zeitpunkt vom Mikroprozessor aus gelesen werden.
— **Der Vorteiler** (8 bit) wird nur in der Betriebsart „Zeitgeber" benützt; er teilt hier den Systemtakt-Puls um den softwaremäßig festlegbaren Wert 16 oder 256.

□ Pinbeschreibung



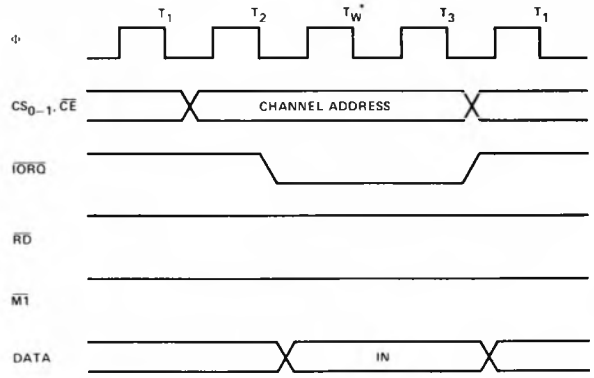
Pin	Funktion	Kommentar
CLK/ TRG 0	Takt/Trigger 0 (="Clock/ Trigger 0")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 0
CLK/ TRG 1	Takt/Trigger 1 (="Clock/ Trigger 1")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 1

Pin	Funktion	Kommentar
CLK/ TRG 2	Takt/Trigger 2 (="Clock/ Trigger 2")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 2
CLK/ TRG 3	Takt/Trigger 3 (="Clock/ Trigger 3")	Eingang High oder Low-aktiv: Externer Takteingang für Zähler bzw. Zeitgeber-Triggereingang für Kanal 3
ZC/TO 0	Nulldurchgang/ Zeitgebermeldung 0 (="Zero Count/ Timeout 0")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 0
ZC/TO 1	Nulldurchgang/ Zeitgebermeldung 1 (="Zero Count/ Timeout 1")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 1
ZC/TO 2	Nulldurchgang/ Zeitgebermeldung 2 (="Zero Count/ Timeout 2")	Ausgang, High-aktiv: Nullsignal des Rückwärtszählers bzw. Meldung des Zeitgebers für Kanal 2
CS 1 und CS 0	Kanalauswahl (="Channel Select")	(Eingang, high-aktiv): Eingabe der 2 Bit-Adresse des des vom Mikroprozessor angesprochenen Kanals
D 7 bis D 0	Datenbus (="Data Bus")	(Bidirektional, Tristate) 8 bit-Datenbus
$\overline{CE}$	Bausteinauswahl (="Chip Enable")	Eingang, Low-aktiv
Φ	Systemtakt (="System Clock")	Eingang
$\overline{M1}$	Maschinenzyklus 1 (="Machine Cycle One")	Eingang, Low aktiv Hier wird das Signal $\overline{M1}$ des Mikroprozessors Z80-CPU angelegt
$\overline{IORQ}$	Ein/Ausgabe-Anforderung (="Input/Output Request")	Eingang, Low-aktiv Hier wird das Signal $\overline{IORQ}$ des Mikroprozessors Z80-CPU angelegt
$\overline{RD}$	Lesen (="Read")	Eingang, Low-aktiv Hier wird das Signal $\overline{RD}$ des Mikroprozessors Z80-CPU angelegt
IEI	Interrupt-Freigabe-Eingang (="Interrupt Enable Input")	Eingang, High-aktiv bildet zusammen mit IEO des vorigen Bausteins die Interrupt-Prioritätskette
IEO	Interrupt-Freigabe-Ausgang (="Interrupt Enable Output")	Ausgang, High-aktiv für Interrupt-Prioritätskette
$\overline{INT}$	Interrupt-Anforderung	Ausgang, Open-Drain, Low-aktiv. Meldet Interruptanforderung an den Mikroprozessor.
$\overline{RESET}$	Rückstelleingang (="Reset")	Eingang, Low-aktiv $\overline{RESET}$ unterbricht den Zählvorgang aller Kanäle und setzt die Interrupt-Freigabebits der Steuerregister aller 4 Kanäle zurück, bringt ZC/TO 0 bis 2 und $\overline{INT}$ in den inaktiven Zustand, setzt Ausgang IEO gleich Wert am Eingang IEI und bringt alle Ausgänge in den hochohmigen Zustand.

☐ CTC-Schreibzyklus

Hier werden Kanalsteuerwort, Zeitkonstante und Interrupt-Vektor eingeschrieben.

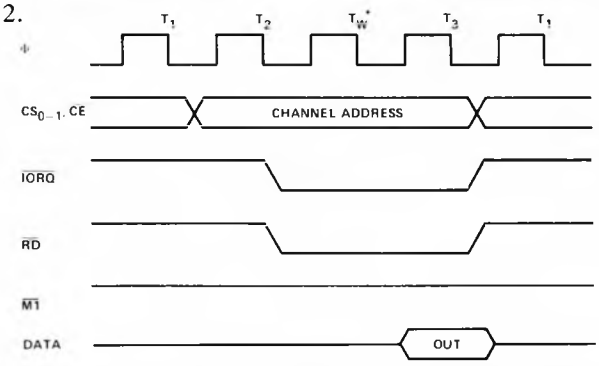
Zur Beachtung: Über den automatisch erzeugten Wartezyklus Tw* dürfen keine Wartezyklen beim Schreiben in die CTC-



Register eingefügt werden. Da der CTC kein spezifisches Schreibsignal geliefert bekommt, erzeugt er es sich intern aus dem RD-Signal.

☐ CTC-Lesezyklus

In der Betriebsart „Zähler" kann der Momentanwert jedes Kanalrückwärtszählers ausgelesen werden. Der auf dem Datenbus ausgelesene Wert repräsentiert die Anzahl der steigenden Taktflanken **vor** der steigenden Taktflanke des Zyklus T2.

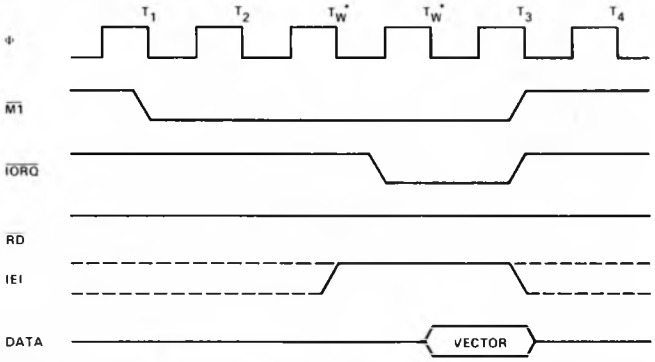


Außer dem automatisch erzeugten Wartezyklus Tw* darf auch beim CTC-Lesen kein Wartezyklus eingefügt werden.

☐ Interrupt-Quittungs-Zyklus

Der Mikroprozessor quittiert nach einer gewissen Zeit den Empfang einer Interrupt-Anforderung vom CTC durch die Signale $\overline{M1}$ und $\overline{IORQ}$.

In der Zwischenzeit ermittelt der CTC intern den anfordernden Kanal mit der höchsten Priorität.



Zum sicheren Einschwingen der Interrupt-Kaskadensignale werden die Unterbrechungsanforderungszustände aller Kanäle „eingefroren", solange $\overline{M1}$ aktiv ist.

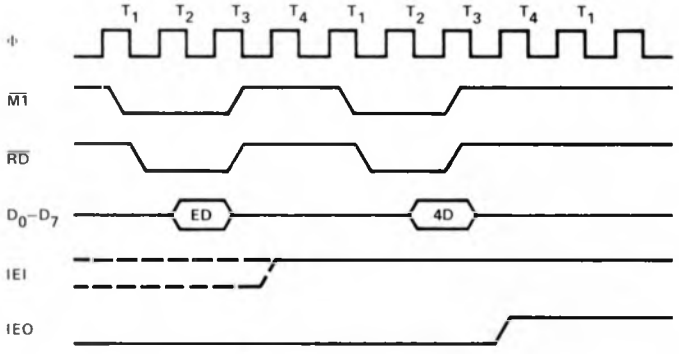
Ist dann der IEI-Eingang des CTC aktiv, bringt der anfordernde Kanal höchster Priorität den Inhalt seines Interrupt-Vektorregisters auf den Datenbus, sobald $\overline{IORQ}$ aktiv wird.

Das Einfügen zusätzlicher WAIT-Zyklen ist hier zulässig.

☐ Rückkehr von Interrupt

Wenn an einem Z80A bzw. Z80-Mikrocomputerbaustein kein Interruptanforderungs-Signal ansteht und gerade keine Interrupt-Service-Routine ausgeführt wird, ist sein Signal IEO = IEI.

Befindet sich eine Interrupt-Service-Routine in Bearbeitung (d. h. der Baustein hat eine Interrupt-Anforderung ausgestellt und eine Interrupt-Bestätigung (= „Interrupt Acknowledge") von der CPU empfangen), ist sein IEO-Ausgang in jedem Fall auf LOW, wodurch niederpriorisierte Bausteine an der Ausstellung einer Interruptanforderung gehindert werden.



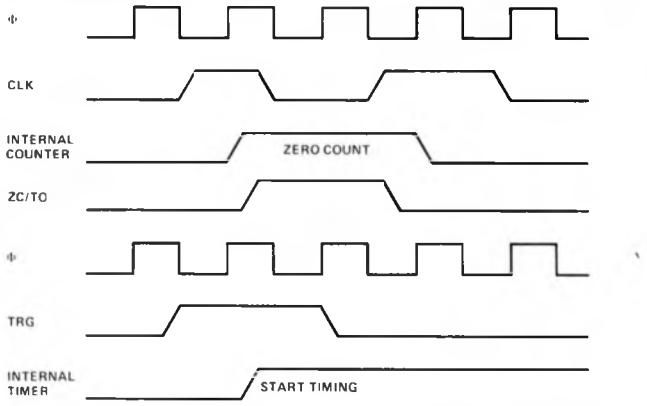
Steht an dem Baustein ein Interruptanforderungs-Signal der ihm zugeordneten Peripherie an, das noch nicht von der CPU quittiert wurde, liegt an IEO LOW-Signal, bis der Hexadezimalwert ED als das erste Byte eines 2 Byte-Opcodes auftritt. Daraufhin wird IEO HIGH bis zum Auftreten des nächsten Opcode-Bytes, wo IEO wieder auf LOW geht. War das zweite Byte dieses Opcodes eine Hexadezimalzahl 4 D, so hat es sich bei dieser Anweisung um einen RETI-Befehl gehandelt.

Nach dem Empfang des Opcodes ED weist ausschließlich **das** Port an IEI High-Signal und an IEO LOW-Signal auf, das die letzte Interruptanforderung an die CPU geschickt hat und nun mit der Interrupt-Behandlung beschäftigt ist; es muß zwangsläufigerweise von den Ports, die zu diesem Zeitpunkt von der CPU eine Interruptquittung bekommen haben, das momentan höchstpriorisierte sein, und bei allen übrigen Ports ist nun IEI = IEO. Beim nächsten OP-Code 4 D verläßt das zuletzt aktive Port seinen Interrupt-Bedienzustand.

In den M1-Zyklen ist an dieser Stelle das Einfügen von WAIT-Zyklen zulässig.

☐ Zähl- und Zeitgeber-Vorgang

In der Betriebsart „Zähler" veranlaßt die steigende Flanke des CLK-Eingang jeweils ein Dekrementieren des Rückwärtszählers. Dieses Signal wird völlig asynchron empfangen, muß jedoch eine gewisse Mindestdauer haben.

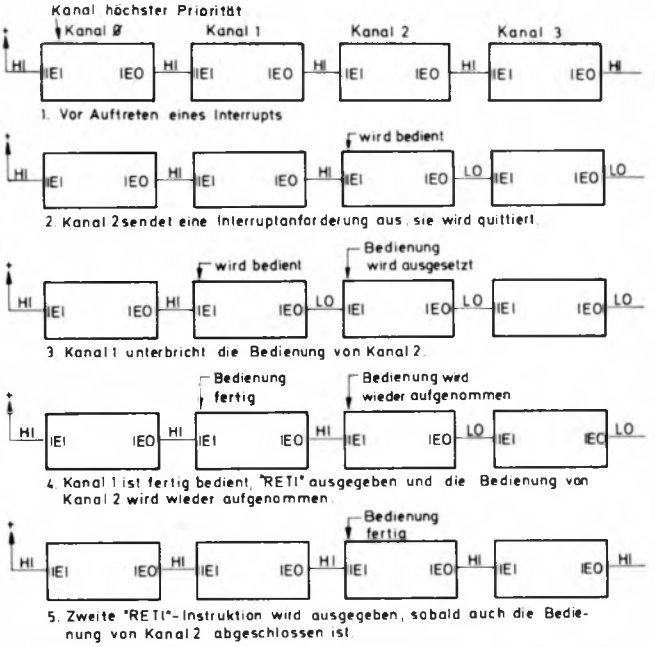


Der Zähler selbst arbeitet jedoch synchron mit Φ, wobei eine gewisse Schaltzeit zur Verfügung stehen muß, wenn der Zähler bereits bei der folgenden steigenden Flanke von Φ nach Aktivierung des CLK-Eingangs dekrementiert werden soll. In der Betriebsart „Zeitgeber" kann der Vorteiler von einer steigenden oder einer fallenden Flanke aus dem TRG-Eingang angesprochen werden.

Wie in der Betriebsart „Zähler" wird diese Flanke völlig asynchron empfangen und muß eine Mindestdauer aufweisen.

Entsprechend muß auch wieder für eine bestimmte Schaltzeit bis zur nächsten steigenden -Flanke gesorgt sein, wenn der Zeitgeberstart zu diesem Zeitpunkt erfolgen soll.

Der Vorteiler zählt steigende Flanken von Φ.



☐ Vorgang der Prioritäts-Kaskadierung

In der Skizze ist eine typische Interruptabfolge als Beispiel dargestellt.

Darin fordert Kanal 2 einen Interrupt an und wird bedient. Währenddessen fordert der Kanal 1 ebenfalls einen Interrupt an und wird wegen seiner höheren Priorität bedient, wobei die Bedienung von Kanal 2 unterbrochen wird. Sobald nun die Bedienung von Kanal 1 abgeschlossen und die RETI-Anweisung ausgeführt ist, wird die niedriger priorisierte Behandlung von Kanal 2 fortgesetzt und zuende geführt.

☐ Laden des Interrupt-Vektors

Dem Mikroprozessor muß vom unterbrechenden Kanal ein Interrupt-Vektor geliefert werden, aufgrund dessen der Mikroprozessor die Startadresse der zugehörigen Interrupt-Bedienroutine ermittelt.

Während des Interrupt-Quittungs-Zyklus (= "Interrupt Acknowledge Cycle") wird dieser Vektor vom anfordernden Kanal höchster Priorität auf den Datenbus gelegt; vorher muß der Vektor dem CTC über Kanal Nr. 0 und Datenbit D 0 gleich Null vorgegeben werden.

D 7 ... D 3 enthalten dabei den Vektor; D 2 und D 1 sind beim Vektorladen unbenützt. Sobald der CTC auf eine Interrupt-Quittung reagiert, enthalten D 2 und D 1 den Binärcode der Nummer des höchstpriorisierten anfordernden Kanals, und D 0 steht auf Null, da die Interrupt-Bedienroutine immer auf einer geraden Adresse beginnt.

Kanal 0 ist hardwaremäßig der höchstpriorisierte Kanal.

Format des zugehörigen Steuerworts:

D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
V 7	V 6	V 5	V 4	V 3	x	x	0

x = beliebiger Binärwert

☐ Laden des Zeitkonstanten-Registers

Nach einem Steuerwort mit Bit 2 = 1 wird das nächstfolgende Steuerwort für den betreffenden Kanal als Zeitkonstante interpretiert und ins Zeitkonstantenregister des Kanals geladen.
Eine Steuerwortkonstante = 0 wird als Zeitkonstante = 256 interpretiert.

Format:

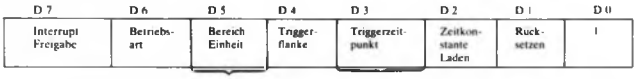
D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0
ZK 7	ZK 6	ZK 5	ZK 4	ZK 3	ZK 2	ZK 1	ZK 0

☐ Programmierung des CTC

Festlegen der Betriebsart

Hat bit 0 den Wert 1, so wird das aktuelle Datenwort in das Kanalsteuerregister übernommen.

Format:



Nur für Betriebsart „Zeitgeber"

Bemerkung:
Werte von D 0 ... D 6 sind beim Einschalten nicht definiert

☐ Erläuterungen zur Programmierung

Bit.-Nr.	Wert	Erklärung
7	0	Interrupt des Kanals gesperrt
	1	Interrupt des Kanals freigegeben; Interruptanforderung erfolgt jedesmal, wenn der Rückwärtszähler den Wert Null erreicht. Durch ein Setzen des Bit 7 wird **kein** Interrupt durch ein **vorangegangenes** Zähler = Null-Ergebnis veranlaßt.
6	0	Betriebsart „Zeitgeber". Der Rückwärtszähler wird vom Verteiler gestartet. Die Periode des Zählers ist dabei $c = t_c \cdot P \cdot TC$ mit t_c = Systemtakt-Periode P = Vorteiler (= 16 oder = 256) TC = 8 bit programmierbare Zeitkonstante (max. = 256)
	1	Betriebsart „Zähler". Der Rückwärtszähler wird von externen Signalen getriggert, der Vorteiler bleibt unbenützt.
5 (nur bei Betriebsart „Zeitgeber")	0	Systemtakt wird vom Vorteiler um den Faktor 16 heruntergeteilt.
	1	Systemtakt wird vom Vorteiler um den Faktor 256 heruntergeteilt.
4	0	Betriebsart Zeitgeber: Zeitmessung wird von negativer Flanke gestartet Betriebsart Zähler: Die negative Flanke dekrementiert den Rückwärtszähler
	1	Betriebsart Zeitgeber: Zeitmessung wird von positiver Flanke gestartet Betriebsart Zähler: Die positive Flanke dekrementiert den Rückwärtszähler.
3 (nur bei Betriebsart „Zeitgeber")	0	Die Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, der auf das Laden der Zeitkonstante folgt, und zwar mit dessen steigender Flanke von T_2
	1	Der externe Triggereingang wird zur Veranlassung des Beginns des Zeitgebervorgangs freigegeben, und zwar nach der steigenden Flanke von T_2 **des** Maschinenzyklus', der auf das Laden der Zeitkonstante folgt. Der Vorteiler wird, soweit die nötige Vorbereitungszeit zur Verfügung stand, nach 2 weiteren Taktzyklen dekrementiert, andernfalls nach 3 weiteren Taktzyklen.
2	0	Auf das Kanal-Steuerwort folgt keine Zeitkonstante. Die Zeitkonstante ist zum Anlaufen lassen des Zeitmeßvorgangs noch einzugeben.
	1	Das nächste Kontrollwort für den betreffenden Kanal stellt die Zeitkonstante für den Rückwärtszähler dar. Wird eine Zeitkonstante während eines Zeitmeßvorgangs eingegeben, wird zuerst die laufende Messung zuende geführt, bevor die neue Zeitkonstante in den Rückwärtszähler geschrieben wird.
1	0	Kanal zählt weiter
	1	Abbruch der momentanen Operation. Falls Bit 2 = 1 ist, wird die Operation nach dem Laden einer neuen Zeitkonstante fortgesetzt. Ist Bit 2 = 0, muß hierfür ein neues Steuerwort an den CTC übermittelt werden.

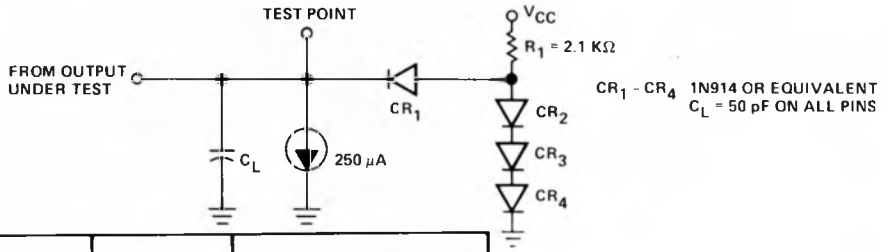
☐ Dynamische Kenndaten des Z80-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	400	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc.	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	160		ns	
D_0–D_7	$t_{DR}(D)$	Data Output Delay from Rising Edge of $\overline{RD}$ During Read Cycle		480	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	60		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During INTA Cycle		340	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		230	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	200		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		220	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		300	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	250		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	210		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	240		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 200$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 200$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		50		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for Enabling of Prescaler on Following Rising Edge of Φ	210			Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200			Counter and Timer Modes
ZC/TO_{0-2}	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190		Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		190		Counter and Timer Modes

Notes: [1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.
[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.
[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum
[4] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

OUTPUT LOAD CIRCUIT



☐ Kapazitäten

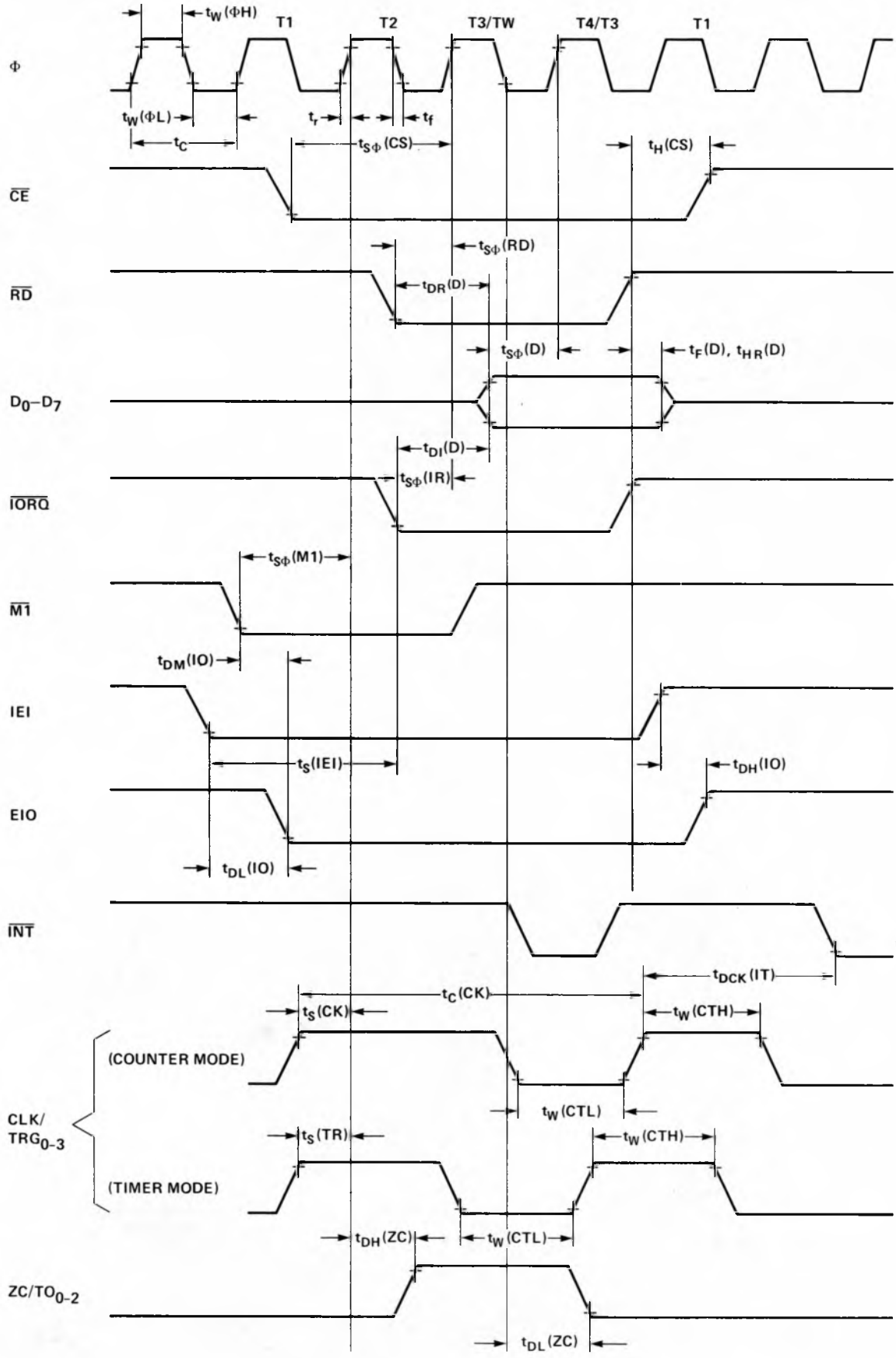
TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	20	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

□ Zeitverhalten des Z80-CTC

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{CC} - .6V	.45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	±0.5V



☐ Absolute Grenzdaten

Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.8W

***Comment**

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

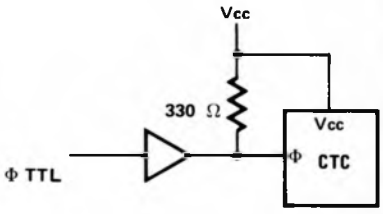
☐ Statische Kenndaten

T_A = 0° C to 70° C, V_{CC} = 5 V ± 5% unless otherwise specified

Z80-CTC

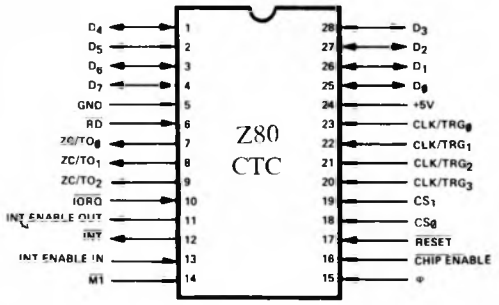
Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage [1]	V_{CC} - .6	V_{CC} + .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
I_{CC}	Power Supply Current		120	mA	T_C = 400 nsec
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5V R_{EXT} = 390Ω

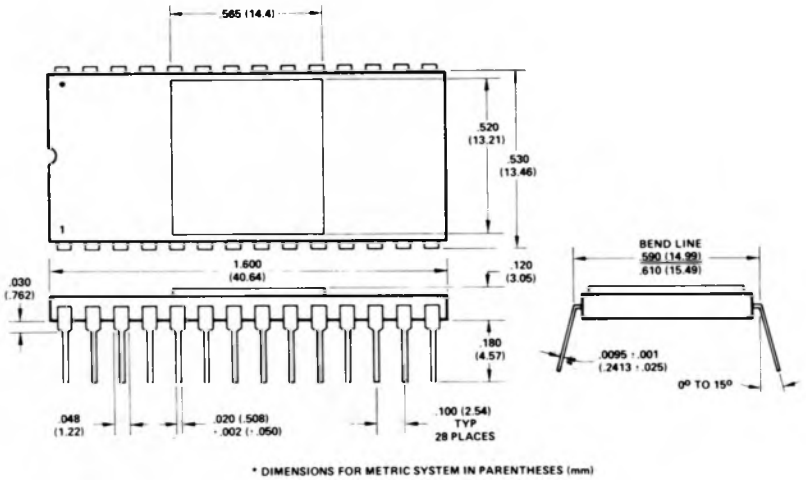
[1] Clock Driver



The external pull-up resistor (330 Ω) can satisfy the clock AC and DC requirements.

☐ Pin-Belegung





☐ Bestellbezeichnung

Z80-CTC/PS	Standardversion (0...70°C) in Plastikgehäuse U_B = 5 V ± 5 %
Z80-CTC/CS	Standardversion (0...70°C) in Keramikgehäuse U_B = 5 V ± 5%
Z80-CTC/CE	Industrieversion mit erweitertem Temperaturbereich (—40° + 85°C) im Keramikgehäuse U_B = 5 V ± 5%
Z80-CTC/CM	Militärische Version (—55°C...+ 125°) im Keramikgehäuse U_B = 5 V ± 10%
Z80-CTC/CM-B	Militärische Version (— 55°C... + 125°C) im Keramikgehäuse, nach JAN 883-B

Z80-DMA

Zilog

BAUSTEIN FÜR DIREKTEN SPEICHERZUGRIFF

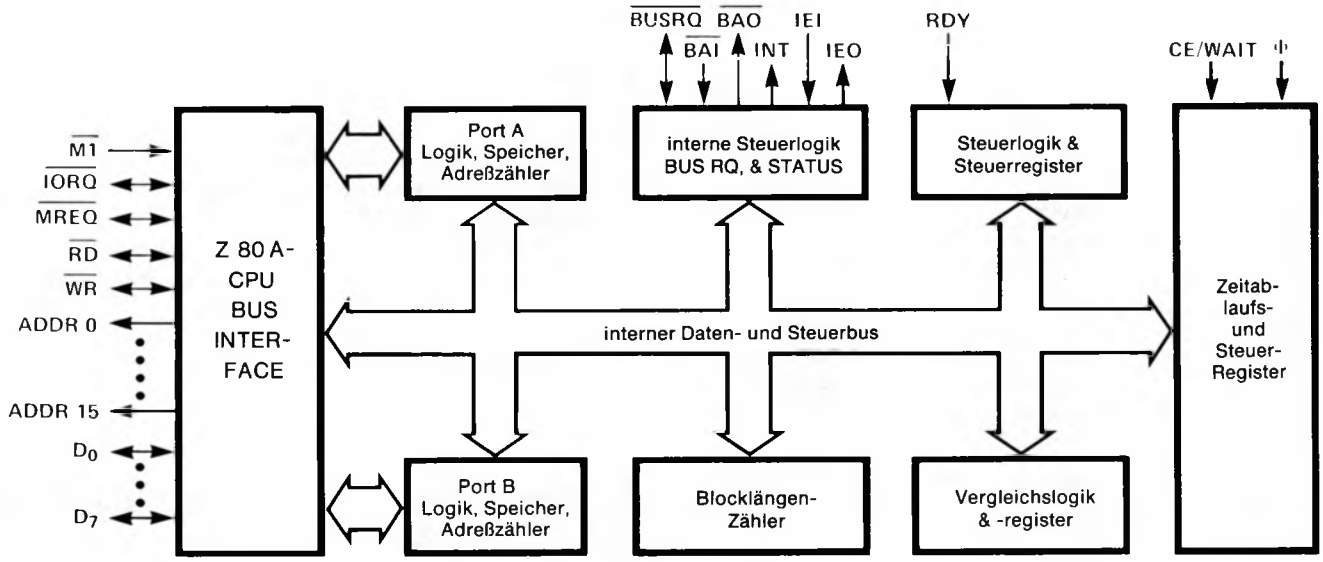


Bild 1: Blockschaltbild des DMA

2. ZILOG-Z80 MIKROCOMPUTER-BAUSTEINE

☐ Vorbemerkung

Das Computersystem Z80A umfaßt ein komplettes Angebot an MC-Halbleiterbausteinen, fertig aufgebauten MC-Platinen, betriebsbereiten OEM-Computersystemen, verschiedenen Typen höchstleistungsfähiger Mikrocomputer-Entwicklungssystemen mit der zugehörigen Betriebssoftware (auch höhere Computer-Sprachen!) und Standard-Anwendersoftware.

Dabei ist das Halbleiterbausteinangebot so geartet, daß auch komplexere MC-Systeme mit minimaler MC-Bausteinanzahl zu realisieren sind. Als Speicherbausteine sind sämtliche Standardchips verwendbar, zusätzliche Logik ist im ganzen System praktisch nicht nötig.

Der Befehlssatz der Z80-CPU verringert durch seine hohe Effizienz Programmentwicklungszeit und Speicherbedard; außerdem wird eine besonders hohe effektive Verarbeitungsgeschwindigkeit erzielt, was für Anwendungen in der Prozeßkontrolle bzw. Steuer- und Regeltechnik ausschlaggebend ist. Befehle zur Einzel-Bit-, 4 bit-, 8 bit- und 16 bit-Verarbeitung und zum Transfer ganzer Datenblöcke machen das System zum idealen Universalstandard für alle Aufgabengebiete, wie Maschinensteuerung, Textverarbeitung, Automatisierung sowie allgemeine Geräterealisierung.

Beim Baustein Z80-DMA (Direct Memory Access) handelt es sich um einen hochintegrierten, programmierbaren Baustein zur Implementierung „direkter Speicherzugriffe"

- ☐ für die besonders schnelle Übertragung von Daten zwischen zwei Funktionsblöcken (= „Ports") innerhalb eines Z80-Systems. Ports können hierbei sowohl Arbeitsspeicher als auch Ein-/Ausgabe-Schnittstellen peripherer Geräte sein.
- ☐ zur schnellen Suche nach einem bestimmten byte oder einer 8-bit Kombination (bit maskierbar) in einem Datenblock. Der Anwender kann zusätzlich festlegen, ob das gesuchte Zeichen gleichzeitig übertragen werden soll oder nicht.

Der Baustein enthält einen solchen DMA-Kanal. Die hierfür notwendigen Zeit- und Steuersignale einschließlich der Adressierung erzeugt die Logik des DMA-Bausteins.

☐ Aufbau

- N-Kanal Silicon-Gate-Depletion Load Technologie
- 40 pin DIP-Gehäuse
- eine einzige 5 V-Spannungsversorgung
- 5 V-Einphasen-Takt
- 1 DMA-Kanal = 2 DMA-Ports

☐ Kenndaten des DMA

- Drei Funktionstypen sind möglich:
  - ☐ nur die Übertragung
  - ☐ nur die Suche
  - ☐ Suche und Übertragung
- Bei einem Übertragungsvorgang werden 2 Adressen gebildet (eine für den Lese-Port, die andere für den Schreib-Port).
- Bei der Übertragung und Suche wird die Portadresse, beginnend von einer vorprogrammierbaren Anfangsadresse, automatisch inkrementiert bzw. dekrementiert (oder bleibt unverändert).
- Seine vier Betriebsarten sind per Programm festzulegen:
  - ☐ Ein-Byte-Übertragung: Pro Bus-request-Anforderung wird 1 byte übertragen.
  - ☐ Peripheriegesteuerte Operation (burst): Operation läuft bis zu einem „Ende" Signal der Peripherie an die DMA-Ports.
  - ☐ Programmgesteuerte Operation (continuous): Operation läuft, bis ein Block mit per Programm festgelegter Länge abgearbeitet ist.
  - ☐ transparente: Operation: („≙ refresh cycles stealing") Die Operation findet während der Speicherrefresh-Zyklen statt.
- Das Zeitverhalten jedes Ports ist programmierbar.
- Beim Auffinden eines gesuchten bytes beim Übertragungsende eines Blocks oder Ready können Interrupts vorprogrammiert werden.
- Eine Operation kann automatisch oder auf Kommando vollständig wiederholt werden (Auto restart oder load).
- Sobald eine bestimmte Anzahl von bytes übertragen worden ist, kann vom DMA-Baustein ein Strobe-Signal ohne Unterbrechung der Übertragung abgegeben werden.
- Mehrfach-DMA-Strukturen mit unterschiedlicher Priorität können in einfacher Weise realisiert werden.
- Der DMA-Kanal kann per Programm freigegeben, gesperrt oder zurückgesetzt werden.
- Per Software kann auch der Status des DMA-Kanals vollständig abgefragt werden.
- Übertragungsraten bis 1,25 Megabyte sind möglich.
- Automatische Interrupt-Vektorerzeugung ohne zusätzlichen Schaltungsaufwand mit Prioritätskaskadierung der Bausteine (daisy chain priority interrupt logic). Die Bestätigung der Busanforderung („BUSRQ") erfolgt hierbei ebenfalls automatisch.
- Alle Ein- und Ausgänge sind TTL-kompatibel.
- Von der CPU können die momentanen Werte von Portadreß-, Bytezählern oder Status-Registern gelesen werden. Die Auswahl des zu lesenden Registers erfolgt durch ein Maskierungs-Steuerwort.

☐ Die Architektur des DMA-Bausteins

Bild 1 zeigt das Blockschaltbild des DMA. Er besteht aus folgenden Funktionseinheiten:

- Bus interfaces
- Steuerlogik und zugehörige Register: Sämtliche Steuerparameter (wie z. B. Betriebsart oder Funktionstyp) werden in dieser Schaltung erkannt und zwischengespeichert.
- Adreß-, Byte-Zähler und Pulsschaltung: Hier werden die Portadressen für Lese- und Schreiboperationen erzeugt und in- bzw. dekrementiert. Ferner setzt die Byte-Zähl-Schaltung im Statusregister ein Flag, wenn das „nullte" Byte transferiert wird. Außerdem erzeugt die Pulsschaltung jedes Mal, wenn die niederwertigen 8 bit des byte-Zählers mit dem Inhalt des „Puls"-Registers übereinstimmen, einen Impuls.
- Schaltung für die Steuerung des DMA-Zeitverhaltens: Das Zeitverhalten bei Lese- und Schreibzugriffen wird für beide Ports per Programm festgelegt.
- Vergleicher: In diesem Block werden durch Vorgabe einer Maske (mask byte) bestimmte bits des aktuellen Worts mit einem gesuchten byte (match byte) verglichen. Sobald das gesuchte byte während eines Übertragungsvorganges gefunden wurde, wird im Statusregister ein Flag gesetzt.

- Interrupt- und BUSRQ-Schaltung: Im Interrupt Steuerregister wird festgelegt, unter welchen Bedingungen der DMA-Baustein einen Interrupt erzeugen darf. Ob dann zu diesem Zeitpunkt ein INT- oder BUSRQ-Signal ausgegeben wird, hängt von der jeweiligen Prioritätscodierungs-Logik ab. Ferner ist in diesem Schaltungsteil ein Interrupt-Vektor-Register enthalten, dessen Inhalt einen Teil des Zeigers einer Interrupt-Bedienroutine darstellt.
- Zustandsregister: In diesem Register sind die laufenden und für den Fortgang bestimmenden DMA-Status-Bedingungen gespeichert.

□ Beschreibung der DMA-Register

Auf folgende interne DMA-Register hat der Anwender Zugriff:

Register	Registerlänge (bit)	L*1	S*2	Erläuterung
● Steuer-Register			X	die für den DMA notwendigen Steuerinformationen sind hier zwischengespeichert, wie z. B. ob ein Interrupt oder Puls erzeugt werden soll, in welcher Betriebsart gearbeitet werden soll, usw.
● Zeitablauf Register	8		X	bestimmt das zeitliche Schreib-/Leseverhalten für die beiden Ports
● Interrupt-Vektor-Register	8	X	X	Bei Interrupt-Anforderung wird der in diesem Register zwischengespeicherte 8 bit-Vektor auf den Datenbus ausgegeben. Der Registerinhalt kann nur während eines Interrupt-Quittungszyklus gelesen werden.
● Blocklängenzähler	16		X	Enthält die gesamte gesuchte und/oder zu übertragende Blocklänge.
● Byte-Zähler	16	X		Die übertragenen oder gesuchten Bytes werden solange aufgezählt bis der Inhalt mit dem Blocklängenregister übereinstimmt. Im Statusregister wird dann das bit „Blocklängenende" gesetzt. Der jeweilige Vorgang kann je nach Programmierung abgebrochen oder einen Interrupt erzeugen. Mit Load oder Continue wird der Byte-Zähler auf Null zurückgesetzt.
● Vergleichs-Register	8		X	gespeichert wird hier ein byte, zu dem ein gleichartiges byte (match byte) während eines Suchvorganges gefunden werden soll.
● Maskierungs Register	8		X	durch eine 8-bit Maske wird festgelegt, welche bits im Vergleichsregister für das Auffinden einer gleichartigen bit-Kombination herangezogen werden.
● Startadressenregister (Port A und Port B)	16		X	gespeichert werden hier die Startadressen (höher- und niederwertige 8-bit) für beide Ports (z. B. Arbeitsspeicher). Bei Suchvorgängen muß nur ein Port spezifiziert und adressiert werden; bei peripheren Ports werden nur die niederwertigen 8-bit als feste Adresse verwendet.
● Adreß-Zähler (Port A und Port B)	16	X		die jeweiligen Startadressen werden in die Adreßzähler mit Load oder Continue geladen. Durch die Programmierung wird festgelegt ob die Adressen fest bleiben oder in- oder dekrementiert werden.
● Pulssteuer-Register	8		X	sobald die hier zwischengespeicherten Sektor- oder Blocklängen (angegeben in bytes) abgearbeitet sind, wird als Rückmeldesignal für ein peripheres Gerät am pin $\overline{INT}$ ein Impuls ausgegeben. Der CPU-Baustein interpretiert dieses Puls-Signal nicht als eine Interrupt-Anforderung, da die Leitung $\overline{BUSRQ}$ vom DMA zuvor aktiviert wurde.
● Status Register	8	X		Enthält die Statusinformationen (siehe Programmierung des DMA-Bausteins).

*1: diese Informationen können nur gelesen werden.
*2: diese Informationen können nur eingeschrieben werden.
*1 und *2: diese Informationen können sowohl gelesen als auch eingeschrieben werden.

□ Funktionstypen des DMA-Bausteins

Drei Funktionstypen sind möglich:
□ nur die Übertragung
□ nur die Suche und
□ kombinierte Übertragung und Suche

Bei einem Übertragungsvorgang werden die Daten von einem Port gelesen und byteweise in das andere Port geschrieben. (Die beiden Ports werden als Port A und Port B bezeichnet). Die Ports können so programmiert werden, daß sie entweder dem Arbeitsspeicher oder peripheren Ein-/Ausgabe-Schnittstellen zugeordnet sind. Folgende Datenübertragungen sind möglich:

- von einer peripheren Einheit zu einer anderen
- von einem Bereich im Arbeitsspeicher zu einem anderen
- von einer peripheren Einheit zum Arbeitsspeicher

Bei einem Suchvorgang werden die Daten nur gelesen und byteweise mit zwei Register im DMA verglichen. In einem

Register steht das für den Vergleich vorgegebene byte (match byte), im anderen (falls erforderlich) ein Masken-byte. Dadurch wird das Aufsuchen einzelner bits oder beliebiger Kombinationen möglich. Sobald das gesuchte 8bit-Wort gefunden ist, wird im DMA-Baustein ein Statusbit gesetzt. Je nachdem, wie der DMA-Baustein programmiert wurde, wird dann der Suchvorgang abgebrochen und/oder ein Interrupt erzeugt.
Bei kombinierten Übertragungs- und Suchvorgängen wird solange ein Datenblock übertragen, bis das gesuchte byte bzw. die gesuchte bit-Kombination gefunden wurde. Bei Übertragung kann dann ähnlich wie beim Suchvorgang abgebrochen und/oder ein Interrupt erzeugt werden. Der Funktionstyp wird mit dem Kommandobyte 1A programmiert.

☐ Betriebsart

Der DMA-Baustein wird vom Anwenderprogramm auf eine seiner 4 möglichen Betriebsarten eingestellt:

- byteweise: die CPU übernimmt wieder die Kontrolle nach jeder Einzelbyte-Operation
- andauernd: (burst) solange der RDY-Eingang aktiv ist, gilt der entsprechende Port als sende- oder empfangsbereit und vom DMA werden Operationen durchgeführt bis das Blockende oder eine vorprogrammierte Vergleichsbedingung erreicht ist
- fortlaufend: (continuous) Die gesamte Suche und/oder Übertragung von Datenblöcken ist abgeschlossen bevor die CPU wieder die Kontrolle übernimmt
- transparent: die DMA-Operation wird während Refreshzyklen durchgeführt. Diese Form von DMA läuft ohne jeden Zeitverlust ab

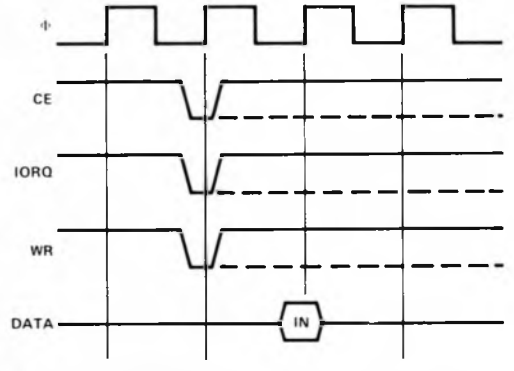
☐ Adressierung

Die Adressierung der DMA-Ports ist entweder fest oder aber ab einer vorgegebenen Startadresse sequentiell inkrementierend bzw. dekrementierend. Die Länge des zu übertragenden Blocks ist durch das vom Anwenderprogramm programmierte Blocklängenregister festgelegt. Bei einem Übertragungsvorgang werden 2 Adressen gebildet (eine für den Lese-Port, die andere für den Schreib-Port).
Der Z80-DMA kann Blocklängen bis zu 64 Kbytes adressieren.

☐ Pin-Belegung

Pin	Funktion	Erläuterung
A_0 — A_{15}	Adreßbus	Tristate Ausgänge, high-aktiv; Adressierung des Arbeitsspeichers oder eines I/0 Ports
D_0 — D_7	Datenbus	Tristate Ein-/Ausgänge (bedirektional) high-aktiv; über den Datenbus erfolgt der Datenaustausch zwischen CPU, Speicher und E/A-Bausteinen
+5 V	Spannungsversorgung	
GND	Bezugspotential	
Φ	Systemtakt	Eingang
$\overline{M1}$	Maschinenzyklus 1	Eingang für M1 Signal von der Z80-CPU, low-aktiv
$\overline{IORQ}$	Ein/Ausgabe-Anforderung („In/Output-Request")	Ein/Ausgabe-Anforderung vom und an den Systembus.
$\overline{MREQ}$	Speicheranforderung (memory-request)	Tristate Ein-/Ausgang, low-aktiv; Speicheranforderung vom und an den Systembus
$\overline{RD}$	Lesen (Read)	Tristate Ein-/Ausgang; Lesesignal vom und für den Systembus
$\overline{WR}$	Schreiben (Write)	Tristate Ein-/Ausgang; Schreibsignal vom und für den Systembus
$\overline{CE/WAIT}$	Bausteinfreigabe/Warten (chip enable/wait)	Eingang, low-aktiv, das Bausteinfreigabesignal kann auch als $\overline{WAIT}$-Eingang programmiert werden, falls $\overline{BAI}$ low ist.
$\overline{BUSRQ}$	Busanforderung (Bus request)	Ein-/Ausgang, low-aktiv, angefordert wird die Kontrolle über Adreß-, Daten und Status/Steuer-Bus
$\overline{BAI}$	Busanforderungsbestätigung-Eingang (Bus acknowledge input)	Zeigt mit seinem Aktiv-(= low) werden an, daß die Buskontrolle an den DMA-Baustein abgegeben wird
$\overline{BAO}$	Busanforderungsbestätigung-Ausgang (Bus acknowledge output)	Ausgang, low-aktiv. $\overline{BAI}$ und $\overline{BAO}$ sind kaskadierbare Verbindungen zur BUS-Prioritätssteuerung z. B. bei Verwendung mehrerer DMA-Kanäle
$\overline{INT}$	Interruptanforderung (interrupt request)	Ausgang, low-aktiv
IEI	Interrupt-Freigabe-Eingang (interrupt enable in)	Eingang, high-aktiv
IEO	Interrupt-Freigabe-Ausgang (Interrupt enable output)	Ausgang, high-aktiv, IEI u. IEO sind kaskadierbare Verbindungen zur Interrupts-Prioritätssteuerung
RDY	Quittierung (Ready)	Eingang, aktiv high oder low je nach Programmierung; hierdurch wird dem DMA mitgeteilt, ob die entsprechende periphere Einheit zum Lesen oder Schreiben bereit ist

☐ Zeitdiagramme des DMA

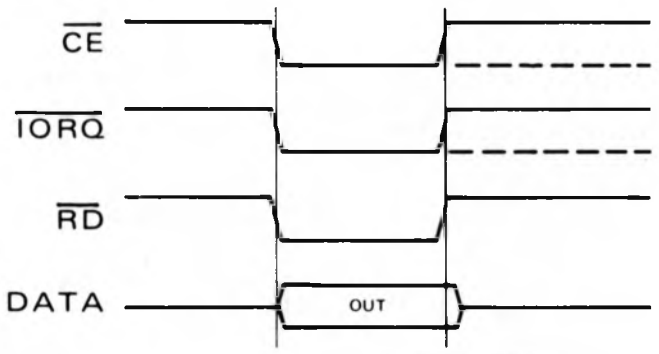


1. Zyklus zum Schreiben eines Kommandobytes

Ein Kommando- oder Steuerbyte wird in ein internes Z80-DMA-Register geschrieben. Das gezeigte Zeitverhalten ist durch die CPU Ausgabebefehle bestimmt.

2. Zyklus zum Lesen eines Registers

Das Lesen des Inhalts des Statusregisters, des Adreßzählers oder anderer lesbarer Register/Zähler geschieht nach folgendem Zeitverhalten, daß durch die CPU-Eingabebefehle bestimmt ist.



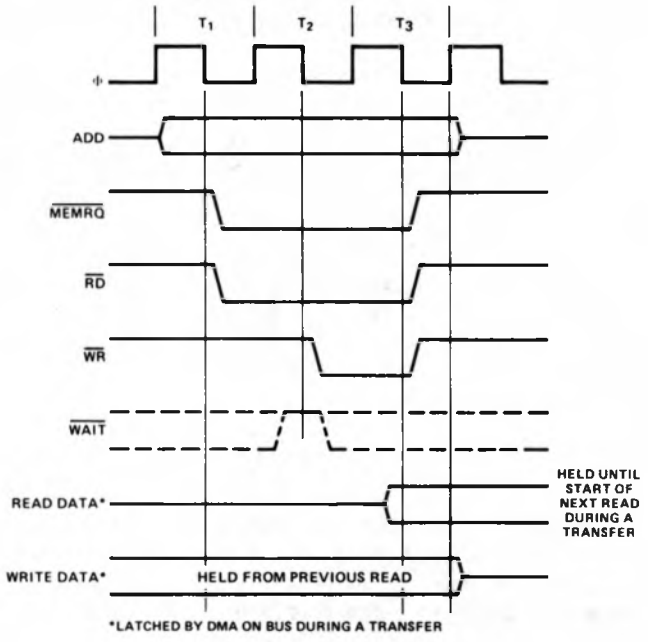
3. Speicherzugriff

Das DMA-Zeitverhalten beim Arbeitsspeicherzugriff entspricht sowohl beim Lese- als auch Schreibvorgang dem der CPU.

Dieses Zeitverhalten wird auch automatisch nach jedem RESET-Signal, beim Einschalten der Speisespannung und nach einem RESET-Kommando programmiert, und wird falls kein anderes Kommandowort gesendet wird, bei allen Such- oder Übertragungsaktivitäten gegenüber dem Arbeitsspeicher beibehalten. Während der Speicherlesephase eines Übertragungszyklus werden die Daten im DMA mit der fallenden Taktflanke während T3 bis zu einem darauffolgenden Schreibzyklus zwischengespeichert. Während der Speicherschreibphase eines Übertragungszyklus werden die Daten vom vorausgegangenen Lesezyklus bis zum Ende des momentanen Zyklus zwischengespeichert.

Zur Beachtung:

Der DMA-Baustein ist für Speicherzugriffe mit 3 Maschinen (T-) Zyklen ausgelegt. Es ist jedoch das Einfügen von „WAIT"-Zyklen möglich, da mit der abfallenden Taktflanke von T2 der log. Zustand des WAIT-Signals abgefragt wird. Ist dieser low, so wird ein weiterer T-Zyklus angehängt und das WAIT-Signal nochmals abgefragt. Auf diese Weise kann die Dauer von Speicherzugriffen beliebig ausgedehnt und ein Anpassen an langsamere Speicher problemfrei vorgenommen werden.

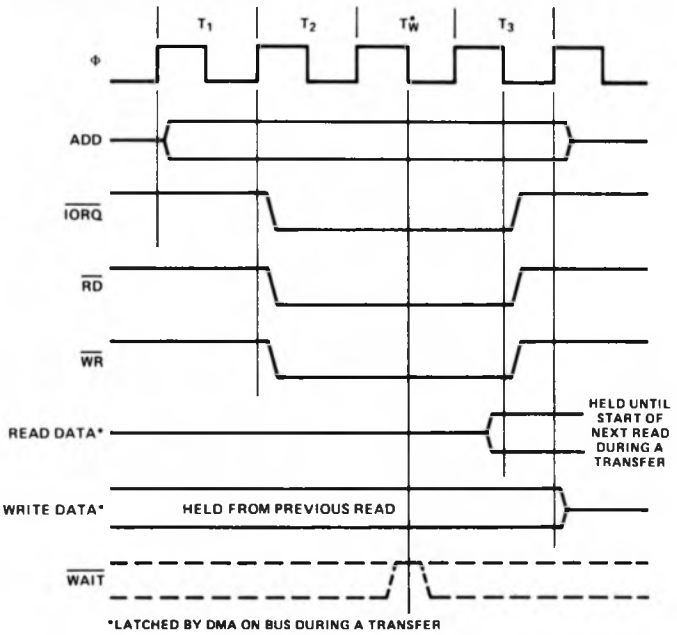


4. Verkehr mit Ein-/Ausgabeeinheiten

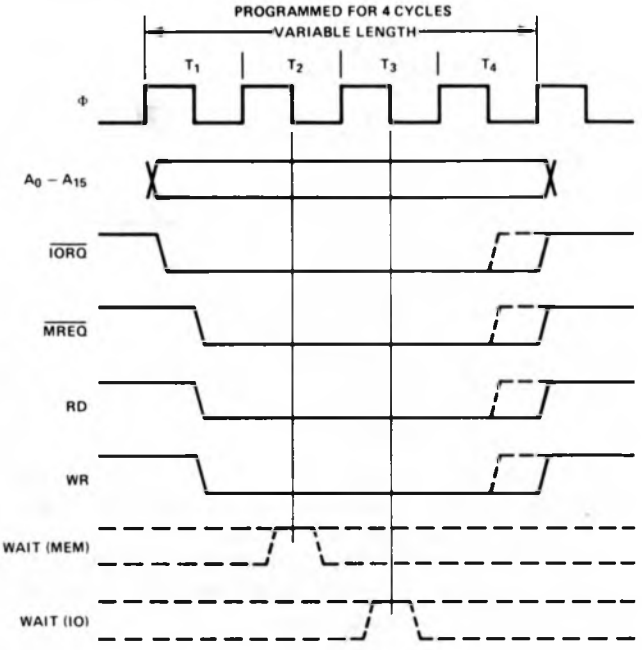
Das Zeitverhalten ist in diesem Fall identisch mit dem der CPU für Lese-Schreiboperationen mit Ein-/Ausgabe-Bausteinen, und wird dem Baustein DMA beim Anlegen der Speisespannung, einem Signal auf der RESET-Leitung oder nach einem Reset-Kommandowort automatisch vorgegeben. Es wird, falls dem DMA keine anderen Kommandowörter zugesendet werden, bei allen Such- und Übertragungsvorgängen mit peripheren Ein-/Ausgabe Schnittstellen beibehalten. Während der Ein-/Ausgabe-Lesephase eines Übertragungszyklus werden die Daten mit der abfallenden Taktflanke des T3-Taktimpulses bis zum nächstfolgenden Schreibzyklus zwischengespeichert. Während der Ein-/Ausgabe-Schreibphase eines Übertragungszyklus werden die Daten vom vorangegangenen Lesezyklus bis zum Ende des momentanen Zyklus zwischengespeichert.

Zur Beachtung:

Ist das WAIT-Signal während der fallenden Taktflanke von T_W^*, gleich low, dann wird ein neuer T-Zyklus angehängt und das WAIT-Signal nochmals abgefragt. Auf diese Weise kann die Dauer von peripheren Ein-/Ausgabezugriffen beliebig ausgedehnt werden.



5. Variabler Zyklus



Eine besondere Eigenschaft des Z80-DMA ist seine Möglichkeit, das Zeitverhalten von Speicher oder Ein/Ausgabe-Vorgängen frei zu programmieren. Damit kann

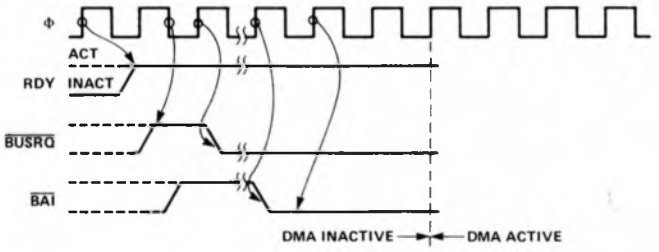
a) das Zeitverhalten an die jeweiligen Anforderungen der gewählten Systemkomponenten angepaßt und

b) die Datentransferrate optimiert werden,

ohne daß hierfür zusätzliche Schaltungen oder Programmteile nötig werden. Die Zykluslänge kann auf ein bis 4 Maschinen-(T-)Zyklen (und mehr bei Verwendung von $\overline{WAIT}$-zyklen) vorprogrammiert werden. Die Signallängen können entsprechend nebenstehendem Impulsdiagramm variiert werden. Die Datenzwischenspeicherung während des Übertragungsvorganges erfolgt nach der Taktflanke, die eine steigende Flanke von $\overline{RD}$- zur Folge hat. Bis zum Ende des folgenden Schreibzyklus werden diese Daten gespeichert.

6. Busverwaltung

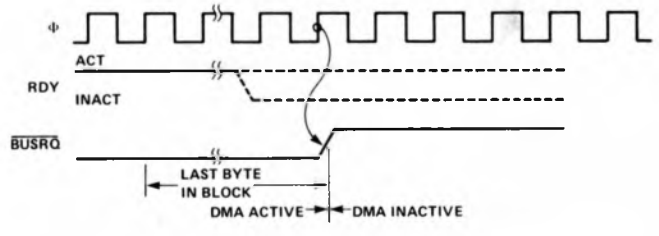
6.1. Busanforderungen bei den Betriebsarten:
byteweise, andauernd und fortlaufend
Das Ready-Signal wird mit jeder ansteigenden Flanke des Taktes Φ abgefragt. Sobald die Ready-Leitung als aktiv (=„1") erkannt wird, erzeugt die nächstfolgende ansteigende Takt-Flanke das $\overline{BUSRQ}$-Signal.



Die CPU aktiviert dann die $\overline{BUSAK}$-Leitung, die entweder direkt mit der BAI-Leitung verbunden wird oder über die kaskadierbare Busprioritätssteuerung.
Sobald $\overline{BAI}$ = „0" (wiederum abgefragt mit jeder steigenden Taktflanke) wird mit der nächstfolgenden ansteigenden Taktflanke ein DMA-Zyklus eingeleitet.

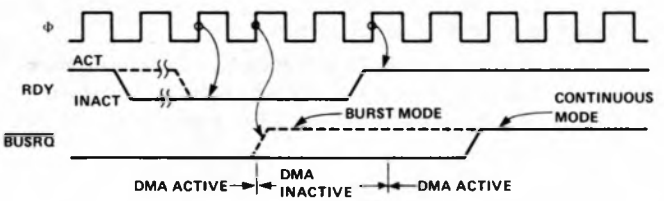
6.2. Busfreigaben beim DMA
Die Busfreigabe (also Rückgabe der Kontrolle über die Busse an die CPU oder anderen DMA's) erfolgt je nach Betriebsart oder Sonderbedingung auf unterschiedliche Weise bei

a) Blockende:
Betriebsart andauernd und fortlaufend, falls eine „automatisches Wiederholen" nicht programmiert wurde.

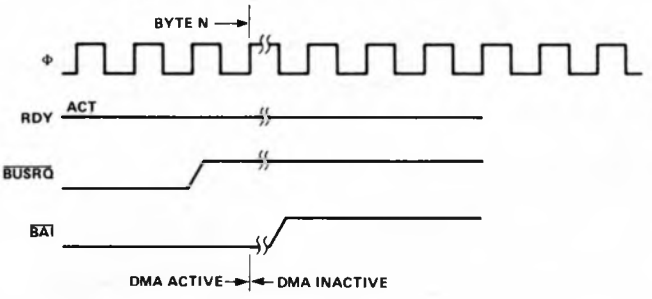


b) der Betriebsart andauernd (burst):
Busfreigabe erfolgt, nachdem die RDY-Leitung inaktiv (= „0") wird (Zeitdiagramm siehe c).

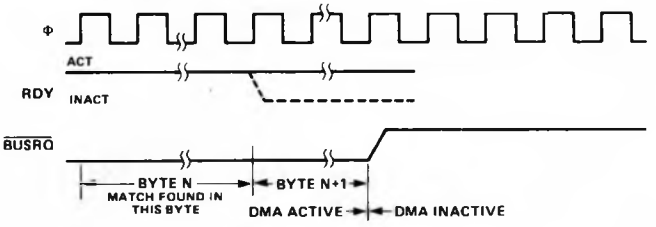
c) der Betriebsart fortlaufend:
Busfreigabe erfolgt, nachdem Blockende oder eine (byte) Vergleichbedingung gefunden wurde.
Die Kontrolle über die Busse wird auch mit RDY = „0" solange ausgeübt ($\overline{BUSRQ}$ = „0") bis der Zyklus bei RDY abgeschlossen werden kann.



d) der Betriebsart byteweise:
Die Busfreigabe erfolgt gleichgültig vom RDY-Zustand nach der steigenden Taktflanke Φ vor dem Ende jeder Lesezyklen bei Suchvorgängen und jeder Schreibzyklen bei Übertragungsvorgängen.
Erst wenn $\overline{BUSRQ}$ = „1" und $\overline{BAI}$ = „1" kann der DMA erneut die Kontrolle über die Busse übernehmen.

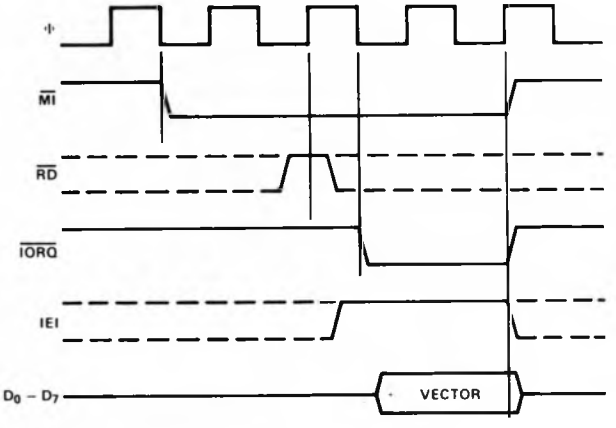


e) Vergleichen bei den Betriebsarten andauernd und fortlaufend:
Sobald eine (-Byte-) Vergleichsbedingung gefunden worden ist, wird vom DMA eine weitere byte-Operation eingeleite und danach die Busse freigegeben. Das „stop on compare"-bit (siehe Commandowort 2A, D2) muß zuvor natürlich aktiviert sein.



7. ($\overline{M1}$ ∧ $\overline{IQ}$)-Interrupt-Quittungszyklus

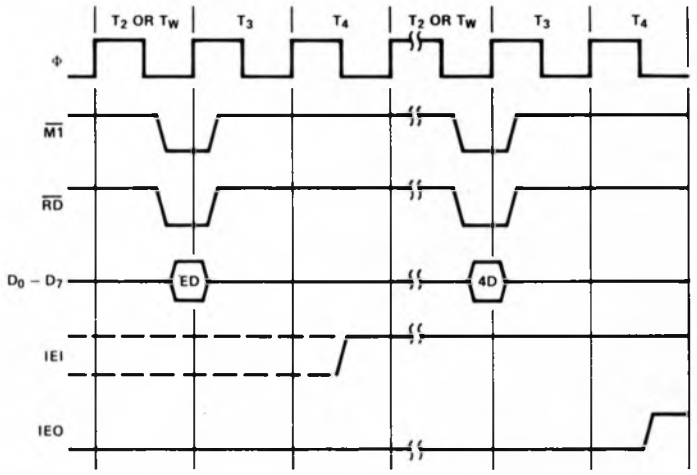
Nachdem vom DMA ein Interrupt angefordert wurde, wird von der CPU eine Interrupt-Quittung ausgegeben ($\overline{M1}$ ∧ $\overline{IORQ}$). In der Zwischenzeit wird durch die Interruptlogik festgestellt, ob der DMA-Kanal die höchste Priorität hat.



(Die Interruptkaskade muß dazu erst einschwingen. Es dürfen sich deshalb die Zustände der Interruptanforderungen nicht ändern, solange $\overline{M1}$ aktiv ist). Ein DMA-Vektor wird dann auf den Datenbus ausgegeben, sobald $\overline{IORQ}$ aktiv wird.

8. Rücksprung vom Interruptzyklus

Durch eine RETI-Anweisung am Ende einer Interrupt-Service-Routine wird der eindeutige Ablauf bei priorisierten, verschachtelten Interrupts gewährleistet. Die RETI-Anweisung wird im DMA-Baustein zu Identifizierungszwecken dekodiert.

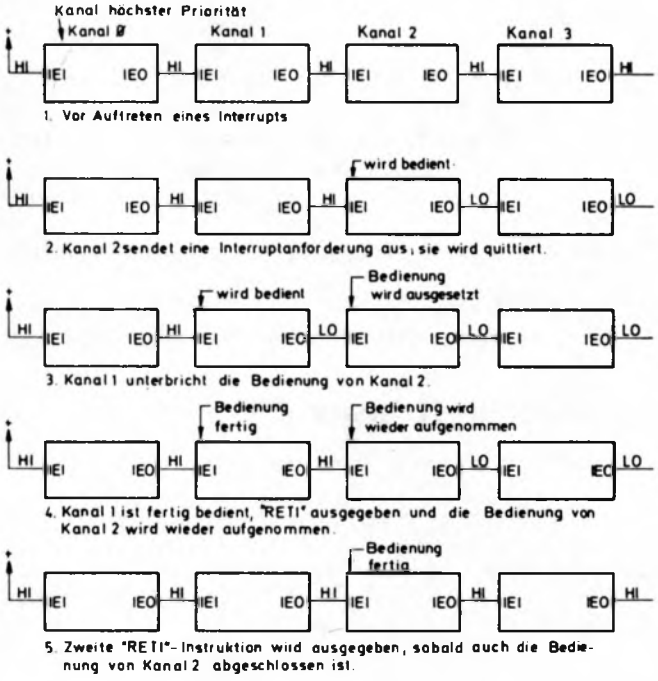


Bei Erkennen des Codeworts EDH wird das IEI-Signal des jeweils angesprochenen Bausteines aktiv. Wird 4 DH empfangen, so wird der momentan bediente Kanal neu initialisiert und sein IEO-Ausgang aktiviert.

☐ Vorgang der Prioritäts-Kaskadierung

In der Skizze ist eine typische Interruptabfolge als Beispiel dargestellt.

Darin fordert Kanal 2 einen Interrupt an und wird bedient. Währenddessen fordert der Kanal 1 ebenfalls einen Interrupt an und wird wegen seiner höheren Priorität bedient, wobei die Bedienung von Kanal 2 unterbrochen wird. Sobald nun die Bedienung von Kanal 1 abgeschlossen und die RETI-Anweisung ausgeführt ist, wird die niedriger priorisierte Behandlung von Kanal 2 fortgesetzt und zuende geführt.



☐ Lesen der DMA-Register

Der DMA-Baustein enthält sieben Register, die sequentiell in der unten aufgezählten Reihenfolge gelesen werden können:

- Status-Register
- Blocklängenregister (niederwertiger Teil)
- Blocklängenregister (höherwertiger Teil)
- Portadresse A (niederwertiger Teil)
- Portadresse A (höherwertiger Teil)
- Portadresse B (niederwertiger Teil)
- Portadresse B (höherwertiger Teil)

Das Weiterschalten von Register zu Register erfolgt durch ein internes Zeigerregister, das bei jedem READ-Signal fortgeschrieben wird. Soll ein Register nicht gelesen werden, kann es durch Setzen einer „0" an der entsprechenden Stelle der Lese-Maske übersprungen werden.

Nach jedem Reset Chip. Reset timing, RESTART, Continue, Enable Chip, Disable Chip oder Reset RD wird der interne Zeiger auf das erste durch die Lesemarke festgelegte Register gesetzt. Nach jedem RD-Status-Signal weist der Zeiger immer auf das Status Register (unabhängig von der Lese-(„Response")-Maske).

☐ Programmierung des DMA-Bausteines

Im aktiven Zustand übernimmt der DMA-Baustein die Kontrolle über die Systembusse, den Datentransfer und die Interruptverarbeitung.

Ein Kommandowort kann dem DMA-Baustein sowohl im aktiven als auch im inaktiven Zustand übermittelt werden; es bewirkt automatisch einen Übergang in den nicht aktiven Zustand. In diesem Zustand befindet sich der Baustein auch, wenn die Stromversorgung eingeschaltet oder der DMA-Baustein auf eine andere Weise zurückgesetzt wird.

Bevor der DMA-Baustein seine eigentliche Aufgabe erfüllen kann, muß er durch das Anwenderprogramm über die CPU wie ein Ein-/Ausgabe Port initialisiert werden. D.h. durch Ausgabebefehle wird eine Folge von Kommandoworte über den Datenbus in die verschiedenen DMA-Register geladen. Die Kommandoworte enthalten 8bit-Steuerinformationen, die den Zustand und die Arbeitsweise des DMA-Bausteins bestimmen und beeinflussen wie z.B. das Setzen eines Interruptfreigabebits oder zusätzlich benötigte Informationen, die nach dem jeweiligen Kommandobyte geladen werden wie z.B. die Startadresse eines Ports.

Die Funktion jedes einzelnen bits der 6 verschiedenen Kommandobytes wird anhand folgender Zusammenstellung klar:

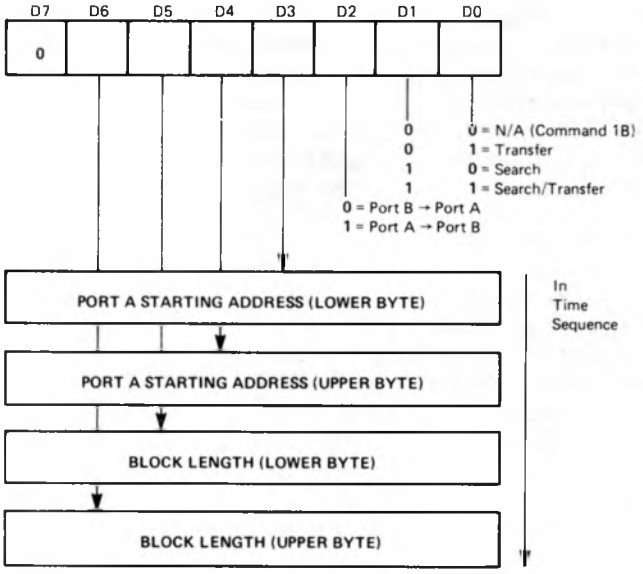
Die Kommandobytes sind in 2 Gruppen eingeteilt

Gruppe 1 mit den bytes 1 A, 1 B

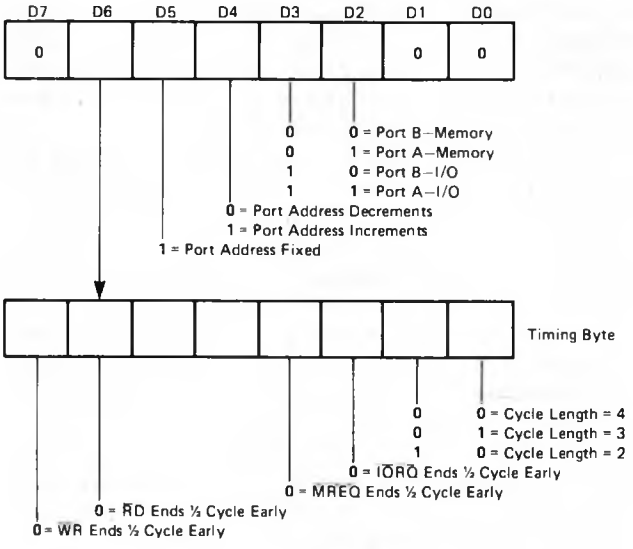
Gruppe 2 mit den bytes 2 A, 2 B, 2 C, 2 D

und sind durch die bits D_0, D_1 und D_7 gekennzeichnet.

Kommando Register 1A



Kommando Register 1B

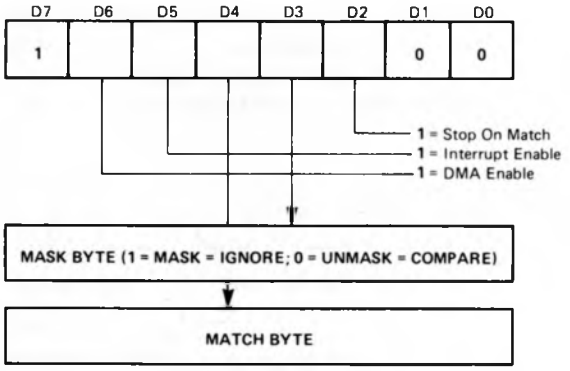


Für Übertragungen wird dieses Byte normalerweise zweimal geschrieben, nämlich einmal für Port A und dann für Port B.

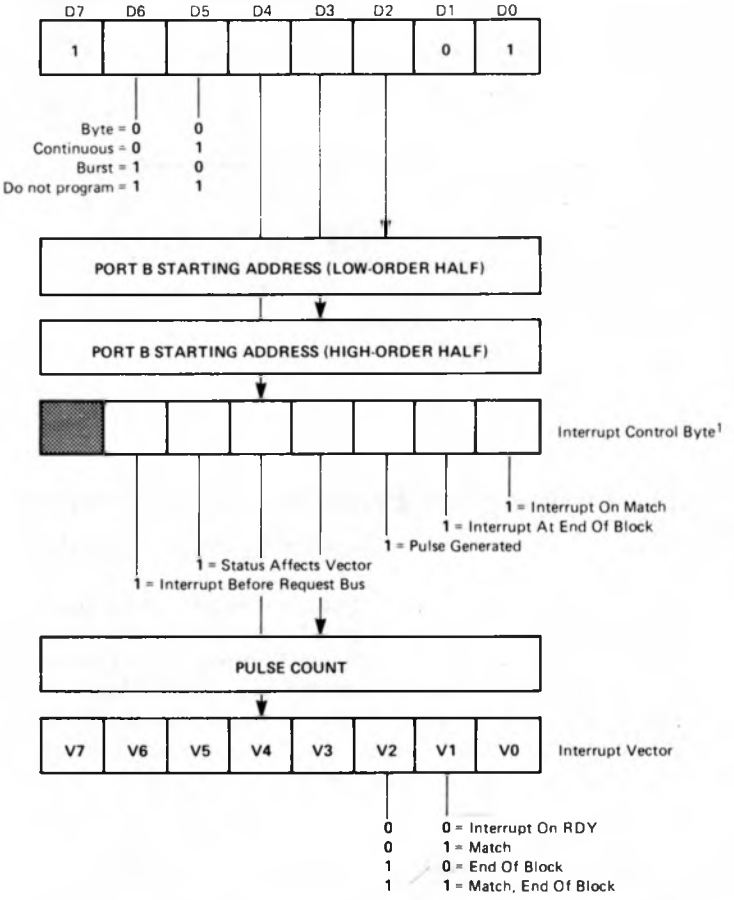
Eine „1" in Bit D_3 bis D_6 bedeutet, daß das betreffende Byte im Folgenden an den DMA gesendet wird. Beachten Sie, daß die Reihenfolge der Bytes zwingend ist.

Der DMA-Baustein überträgt oder sucht grundsätzlich ein Byte mehr als die Zahl im Blocklängenregister angibt. Eine „0" im Blocklängenregister veranlaßt das Übertragen oder Suchen von $(2^{16}+1)$ Bytes. Die kürzeste programmierbare Blocklänge ist daher 2 Byte (durch eine „1" im Blocklängenregister festgelegt).

Kommando Register 2A



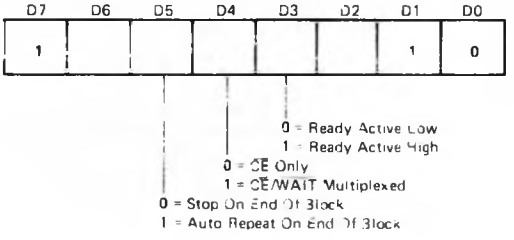
Kommando Register 2B



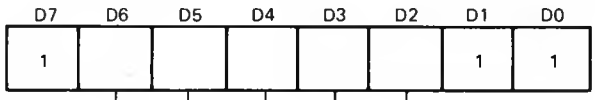
Bei „Interrupt before Requesting Bus" (eine 1 in Bit 6 des Interrupt-Control-Bytes) stellt der DMA-Baustein seine Interrupt-Anforderung erst dann aus, wenn der DMA-Baustein vorher die folgenden Kommandos empfangen hat:

- ☐ „Freigabe nach RETI"-Kommando (B7 in Kommando Byte 2D).
- ☐ „DMA-Freigabe"-Kommando (87 in Kommando-Byte 2D).
- ☐ Eine RETI-Anweisung, die das IUS-Flipflop des DMA-Bausteins rücksetzt (IUS = Interrupt under Service-Latch)

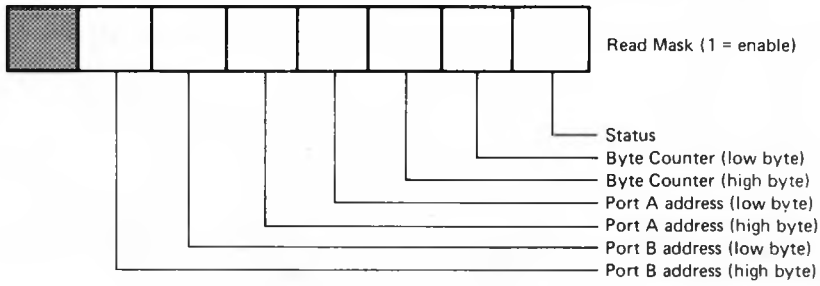
Kommando Register 2C



Kommando Register 2D



HEX					
C3	1	0	0	0	**0** = **Reset** Interrupt circuitry, disable interrupt and bus request logic, unforce internal ready condition, disable "MUXCE" and stop auto repeat.
C7	1	0	0	0	**1** = **Reset Port A Timing** to standard Z-80 CPU timing.
CB	1	0	0	1	**0** = **Reset Port B Timing** to standard Z-80 CPU timing.
CF	1	0	0	1	**1** = **Load** starting address for both ports, clear byte counter.*
D3	1	0	1	0	**0** = Addresses **continue** from present locations, clear byte counter.
AB	0	1	0	1	**0** = **Enable interrupts**
AF	0	1	0	1	**1** = **Disable interrupts**
A3	0	1	0	0	**0** = **Reset and disable interrupt** circuits (like RETI) and unforce the internal ready condition
87	0	0	0	0	**1** = **Enable DMA**] Both affect all operations except interrupts, but do not
83	0	0	0	0	**0** = **Disable DMA**] reset any functions.
A7	0	1	0	0	**1** = **Initiate read** sequence to the first register designated as readable by the Read Mask register.
BF	0	1	1	1	**1** = Set **read status** so next read is from status register.
B3	0	1	1	0	**0** = **Force** an internal **ready** condition independent of the RDY input. Used for memory-to-memory operations where no RDY signal is needed. This command does not function in the "byte-at-a-time" mode.
8B	0	0	0	1	**0** = **Clear** Match and End of Block **status** bits.
B7	0	1	1	0	**1** = **Enable after RETI** so DMA will request bus only after receiving a RETI. Must be followed by an Enable DMA command.
BB	0	1	1	1	**0** = **Read mask** is the following byte.



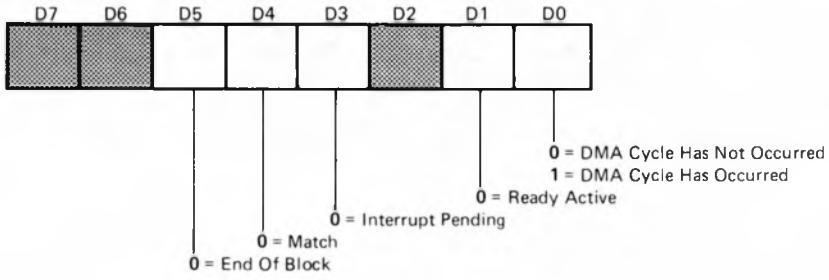
* Loading Port Addresses. The "Load" command (CF in Command Register 2D) loads a fixed address only into a port selected as the source, not into a port selected as the destination. Therefore, the destination address must be loaded by temporarily mislabeling the destination as the source.

The following example is a set-up procedure for a transfer from Port A to Port B:

1. Command byte 1A with B as source port
2. Command byte 2D with CF = load
3. Command byte 1A with A as source port
4. Command byte 2D with CF = load
5. Command byte 2D with 87 = Enable DMA

This manipulation is required only when the destination has a fixed address.

Status Register



□ Erläuterung zur Programmierung

Funktionstypenschlüssel

C_1	C_0	Funktion
0	0	nicht erlaubt
0	1	nur die Übertragung
1	0	nur die Suche
1	1	Suche und Übertragung

Betriebsartenschlüssel

M_1	M_0	Ablaufweise
0	0	byteweise
0	1	fortlaufend
1	0	burst
1	1	transparent

Zykluslängenschlüssel

T_1	T_2	Zykluslänge
0	0	4
0	1	3
1	0	2
1	1	1

(1) Eine „0" in D_2, D_3, D_6, oder D_7 hat zur Folge, daß das entsprechende Steuersignal eine halbe Taktzeit zuvor beendet ist. Für den Übertragungs- (Lesen u. Schreiben) und Suchvorgang (Lesen) sind mindestens 2 Taktzyklen erforderlich.

Interruptstatusschlüssel

(falls bit 5 des Interruptsteuerbytes gesetzt („1") wurde)

D_2	D_1	Funktion
0	0	Interruptanforderungen nach
0	1	Vergleichsbyte gefunden, Blocklängenende nicht gefunden
1	0	Vergleichsbyte nicht gefunden, Blocklängenende gefunden
1	1	Vergleichsbyte und Blocklängenende gefunden

Zustandsschlüssel

Hex	f_4	f_3	f_2	f_1	f_0	Bedeutung	Kurzbeschreibung
C3	1	0	0	0	0	Chip zurücksetzen	Setzt alle Interruptbedingungen zurück, sperrt Interrupts und Busse
C7	1	0	0	0	1	Setzt Zeitverhalten der Ports A/B zurück	gleicht das Zeitverhalten der Ports A/B an das Zeitverhalten der Z80-CPU an
CB	1	0	0	1	0		
CF	1	0	0	1	1	Load	setzt den Bytezähler auf Null zurück und lädt die Startadressen für beide Ports
D3	1	0	1	0	0	Restart bei momentanem Adreßstand	lädt nochmals die Blocklänge. Die Ausführung wird beim momentanen Adreßstand begonnen.
AB	0	1	0	1	0	Interruptfreigabe	Interrupts werden erlaubt
AF	0	1	0	1	1	Interruptsperre	Interrupts werden nicht erlaubt
A3	0	1	0	0	0	Interrupt-rücksetzung	Alle Interrupts werden zurückgesetzt (vergleiche mit RETI)
87	0	0	0	0	1	Chip Freigabe	Alle Operationen mit Ausnahme des Interrupts werden freigegeben oder
83	0	0	0	0	0	Chip Sperre	blockiert; es wird nichts zurückgesetzt
BB	0	1	1	1	0	Lesemarke folgt	welche Register ausgegeben werden, steht im nächst folgenden byte
A7	0	1	0	0	1	Zurücksetzen von Read	Das erste auslesbare, durch die Lesemarke bestimmte Register wird als nächstes gelesen.
BF	0	1	1	1	1	Lesen des Status (RD Status)	Als nächstes wird das Status Register ausgelesen
B3	0	1	1	0	0	Erzwungenes Ready-Signal (Force Ready)	Bei gewissen Operationen wird kein Ready-Signal benötigt (z.B. bei der Speicher zu Speicherübertragung). Das Ready-Signal wird hier als aktiv betrachtet, unabhängig vom tatsächlichen log. Zustand.
B7	0	1	1	0	1	Freigabe der Busanforderung nach einem RETI-Signal	Vom DMA-Baustein kommt keine Busanforderung, solange des RETI-Signal nicht empfangen wurde.
8B	0	0	0	1	0	Zurücksetzen des Status (RST Status)	Das Match und Blockende-Statusbit wird zurückgesetzt.

Masken-Byte

Ein Vergleich zwischen dem gelesenen und dem vorgegebenen Bit in einem Datenwort findet statt, falls an der entsprechenden Stelle eine log „0" im Masken byte vorgegeben wurde.

Match-Byte

Wörter bis zu 8-bit (D_0-D_7) können während des Lesevorganges verglichen werden. Näheres siehe Masken-Byte.

☐ Interrupt-Vektor

Während der Interrupt-Quittungsphase steht der CPU dieser 8-bit Vektor zur Erzeugung der Tabellenadresse für die Bedienroutinenstartadresse zur Verfügung (falls der DMA-Baustein die höchste Priorität hat).

☐ Anzahl der Operationen

Mit diesem 8-bit Wort wird der Puls-Zähler vorgesetzt. Nach jeder Operation wird der Zähler dekrementiert. Sobald der Zählerstand Null erreicht, erscheint ein Impuls auf der INT-Leitung. Anschließend wird der Zähler wieder vorgesetzt. Eine Interruptanforderung wird hierbei nicht erzeugt.

☐ Programmbeispiel

In dem unten gezeigten Beispiel sollen Daten (Blocklänge 1001 H Bytes) aus dem Arbeitsspeicher (Port A) auf ein peripheres Gerät (Port B). In diesem Beispiel ist die Port A-Anfangsadresse 1050 H und die feste Peripherieadresse 05 H.

	D7	D6	D5	D4	D3	D2	D1	D0	HEX
1) Command Register 1A sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0 Group One	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Addr Follows	1 Port A Lower Addr Follows	0 B →A Temporary For Loading B Address	0	1 Command Byte 1A Transfer, No Search	79
2) Port A address (lower)	0	1	0	1	0	0	0	0	50
3) Port A address (upper)	0	0	0	1	0	0	0	0	10
4) Block length (lower)	0	0	0	0	0	0	0	0	00
5) Block length (upper)	0	0	0	1	0	0	0	0	10
6) Command Register 1B defines Port A as memory with incrementing address.	0 Group One	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port Is Memory	1 This Is Port A	0	0 Byte 1B	14
7) Command Register 1B defines Port B as peripheral with fixed address.	0 Group One	0 No Timing Follows	1 Fixed Address	0 Not Used	1 Port Is I/O	0 This Is Port B	0	0 Byte 1B	28
8) Command Register 2B sets mode to Burst, sets DMA to expect Port B address.	1 Group Two	1	0 Burst Mode	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Addr Follows	0	1 Byte 2B	C5
9) Port B address (lower)	0	0	0	0	0	1	0	1	05
10) Command Register 2C sets Ready active High.	1 Group Two	0 Not Used	0 No Auto Restart	0 No Wait States	1 RDY Active HIGH	0 Not Used	1	0 Byte 2C	8A
11) Command Register 2D loads Port B address and resets block counter.	1 Group Two	1	0	0 Load	1	1	1	1 Byte 2D	CF
12) Command Register 1A sets Port A as source. *	0 Group One	0	0	0 No Addr Or Block Length Bytes	0	1 A →B	0	1 Byte 1A, Transfer No Search	03
13) Command Register 2D loads Port A address and resets block counter. *	1 Group Two	1	0	0 Load	1	1	1	1 Byte 2D	CF
14) Command byte 2D enables DMA to start operation.	1 Group Two	0	0	0 Enable DMA		1	1	1 Byte 2D	87

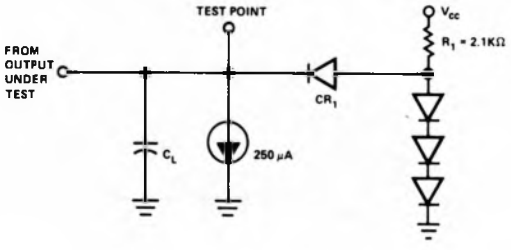
NOTE: The actual number of bytes transferred is one more than specified by the block length.
* These commands are necessary only in the case of a fixed destination address.

☐ Kapazitäten

$T_A = 25°C, f = 1\ MHz$

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	35	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

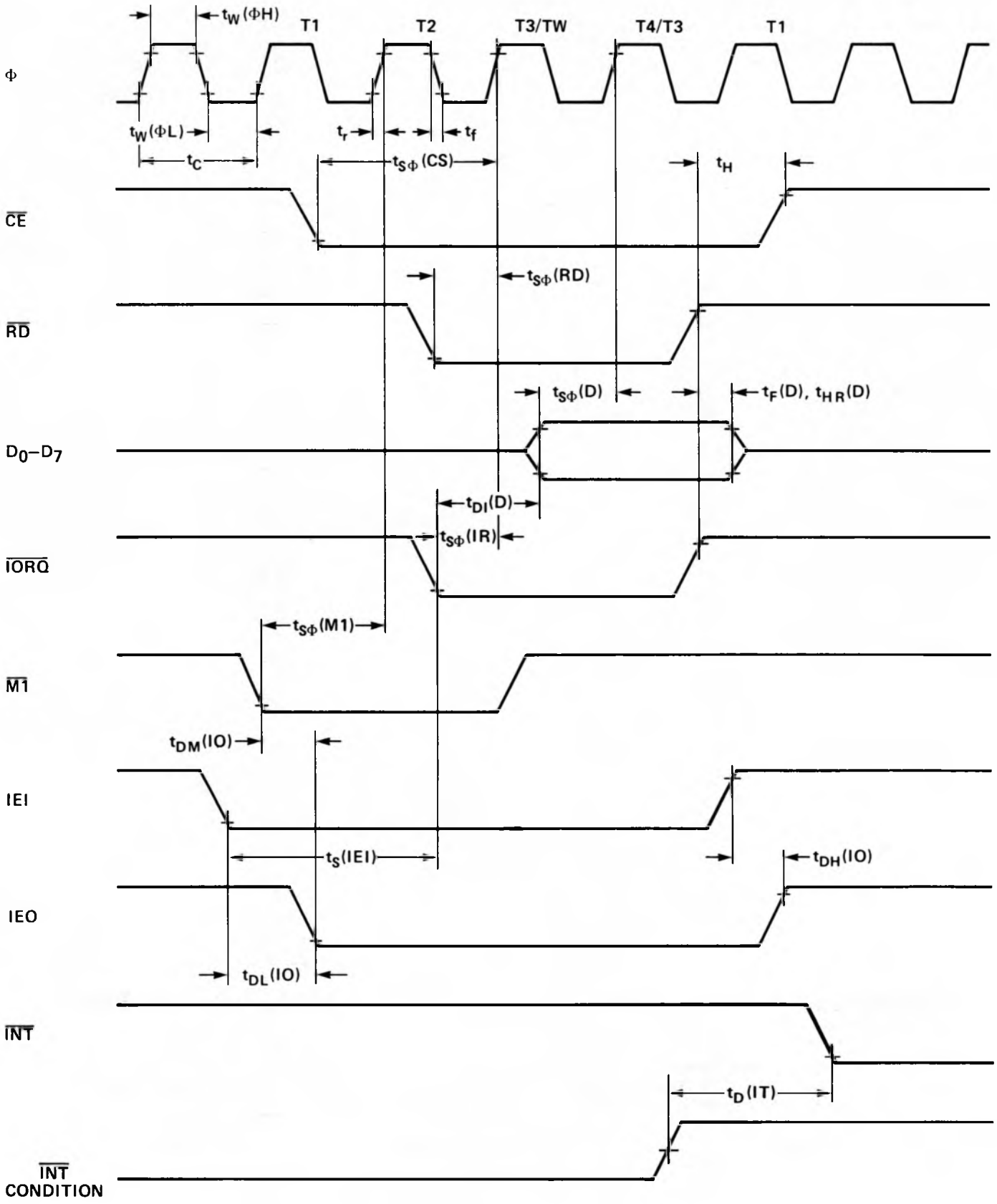
Load Circuit for Output



□ Zeitverhalten des Z80-DMA als periphere (inaktive) Einheit

Timing measurements are made at the following voltages, unless otherwise specified:

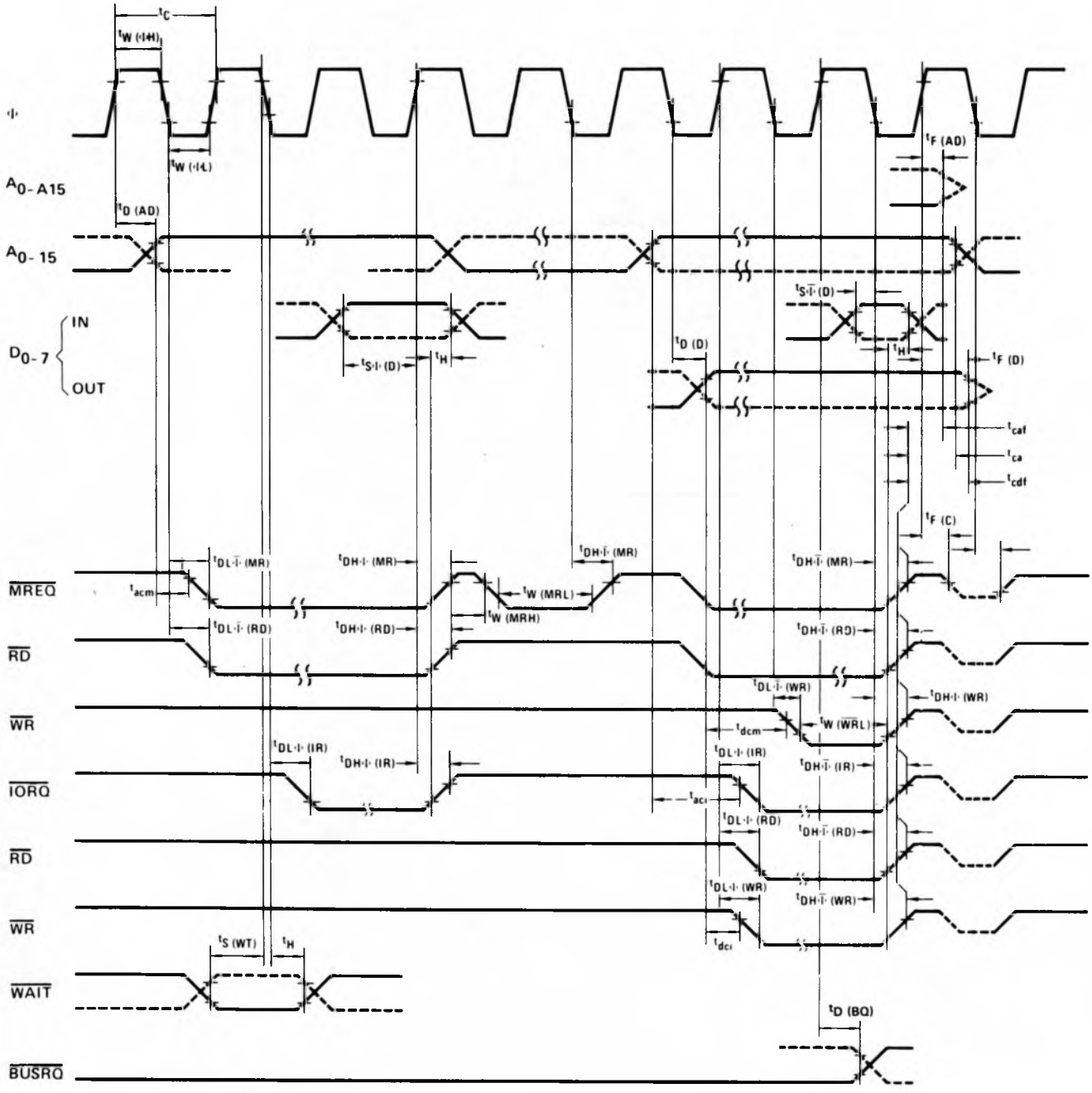
	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV =	+0.5V



□ Zeitverhalten des Z80-DMA als aktive Einheit (= Bus Controller)

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



□ Dynamische Kenndaten des Z80-DMA als periphere (= inaktive) Einheit

SIGNAL	SYMBOL	PARAMETER	Z-80 DMA	
			MIN [nsec]	MAX [nsec]
Φ	t_c	Clock Period	400	[1]
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	170	2000
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	170	2000
	t_r, t_f	Clock Rise and Fall Times		30
	t_H	Any Hold Time for Specified Setup Time	0	
$\overline{CE}$, $\overline{WR}$ $\overline{IORQ}$	$t_{S\Phi(CS)}$	Control Signal Setup Time to Rising Edge of Φ during Write Cycle ($\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$)	280	
D_{0-7}	$t_{DR(D)}$	Data Output Delay from Falling Edge of $\overline{RD}$		500
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Φ during Write or $\overline{M1}$ Cycle	50	
	$t_{DI(D)}$	Data Output Delay from Falling Edge of $\overline{IORQ}$ during INTA Cycle		340
	$t_{F(D)}$	Delay to Floating Bus (Output Buffer Disable Time)		160
IEI	$t_{S(IEI)}$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ during INTA Cycle	140	
IEO	$t_{DH(IEI)}$	IEO Delay Time from Rising Edge of IEI		210
	$t_{DL(IO)}$	IEO Delay Time from Falling Edge of IEI		190
	$t_{DM(IO)}$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		300
$\overline{M1}$	$t_{S\Phi(M1)}$	$\overline{M1}$ Setup Time to Rising Edge of Φ during INTA or $\overline{M1}$ Cycle. See Note B.	210	
$\overline{RD}$	$t_{S\Phi(RD)}$	$\overline{RD}$ Setup Time to Rising Edge of Φ during $\overline{M1}$ Cycle.	240	
$\overline{INT}$	$t_{D(IT)}$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		500
$\overline{BAO}$	$t_{DH(BO)}$	BAO Delay from Rising Edge of BAI	150	200
	$t_{DL(BO)}$	BAO Delay from Falling Edge of BAI	150	200

[1] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

☐ Dynamische Kenndaten des Z80-DMA als aktive Einheit (= Bus Controller)

T_A = 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted.

SIGNAL	SYMBOL	PARAMETER	Z-80 DMA	
			MIN [nsec]	MAX [nsec]
Φ	t_C	Clock Period	400	
	$t_{W(\Phi H)}$	Clock Pulse Width, Clock High	180	2000
	$t_{W(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000
	$t_{r,f}$	Clock Rise and Fall Time		30
A_0–A_{15}	$t_{D(AD)}$	Address Output Delay		145
	$t_{F(AD)}$	Delay to Float		110
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	$t_{W(\Phi H)} + t_f - 75$	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	$t_C - 80$	
	t_{ca}	Address Stable from $\overline{RD}$ or $\overline{WR}$	$t_{W(\Phi L)} + t_r - 40$	
	t_{caf}	Address Stable from $\overline{RD}$ or $\overline{WR}$ during Float	$t_{W(\Phi L)} + t_r - 60$	
D_0–D_7	$t_{D(D)}$	Data Output Delay		230
	$t_{F(D)}$	Delay to Float during Write Cycle		90
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock during Read when Falling Edge Ends $\overline{RD}$	50	
	$t_{S\bar{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock during Read when Falling Edge Ends $\overline{RD}$	60	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	$t_C - 210$	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	$t_{W(\Phi L)} + t_r - 210$	
	t_{cdf}	Data Stable from $\overline{WR}$	$t_{W(\Phi L)} + t_r - 80$	
	t_H	Any Hold Time where Setup Time is Specified	0	
$\overline{MREQ}$	$t_{DL\Phi(MR)}$	$\overline{MREQ}$ Delay from Rising Edge of Clock, $\overline{MREQ}$ Low		100
	$t_{DL\bar{\Phi}(MR)}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low		100
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay from Rising Edge of Clock, $\overline{MREQ}$ High		100
	$t_{DH\bar{\Phi}(MR)}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ High		100
	$t_{W(MRL)}$	Pulse Width, $\overline{MREQ}$ Low	$t_C - 40$	
	$t_{W(MRH)}$	Pulse Width, $\overline{MREQ}$ High	$t_{W(\Phi H)} + t_f - 30$	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ Low		90
	$t_{DL\bar{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ Low		110
	$t_{DH\bar{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ High		100
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ Low		110
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ Low		100
	$t_{DL\bar{\Phi}(RD)}$	$\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ Low		130
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ High		100
	$t_{DH\bar{\Phi}(RD)}$	$\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ High		110
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ Low		80
	$t_{DL\bar{\Phi}(WR)}$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ Low		90
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ High		100
	$t_{DH\bar{\Phi}(WR)}$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ High		100
	$t_{W(WRL)}$	Pulse Width, $\overline{WR}$ Low	$t_C - 40$	
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70	
$\overline{BUSRQ}$	$t_{D(BQ)}$	$\overline{BUSRQ}$ Delay Time from Rising Edge of Clock		100
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100

NOTES:

- Data must be enabled onto the DMA data bus when $\overline{RD}$ is active.
- All equations imply standard Z-80 CPU and Z-80A CPU timing.

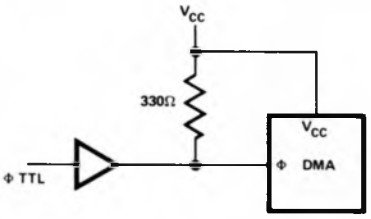
☐ Absolute Grenzdaten

Temperature Under Bias	Specified operating range.
Storage Temperature	−65°C to +150°C
Voltage On Any Pin with Respect to Ground	−0.3V to +7V
Power Dissipation	1.5W

Comment

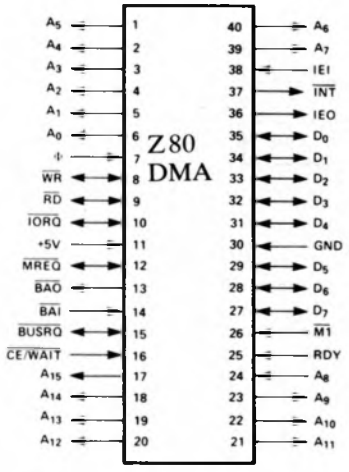
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

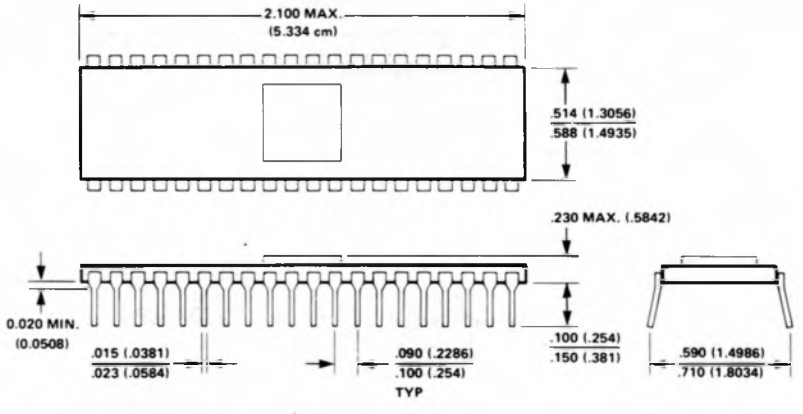
☐ Statische Kenndaten



An external clock pull-up resistor of (330Ω) will meet both the AC and DC clock requirements.

☐ Pin-Belegung





*Dimensions for metric system are in parentheses

☐ Bestellbezeichnung

Z80-DMA/PS Standardversion (0...70°C) in Plastikgehäuse U_B = 5 V ± 5%

Z80-DMA/CS Standardversion (0...70°C) im Keramikgehäuse U_B = 5 V ± 5%

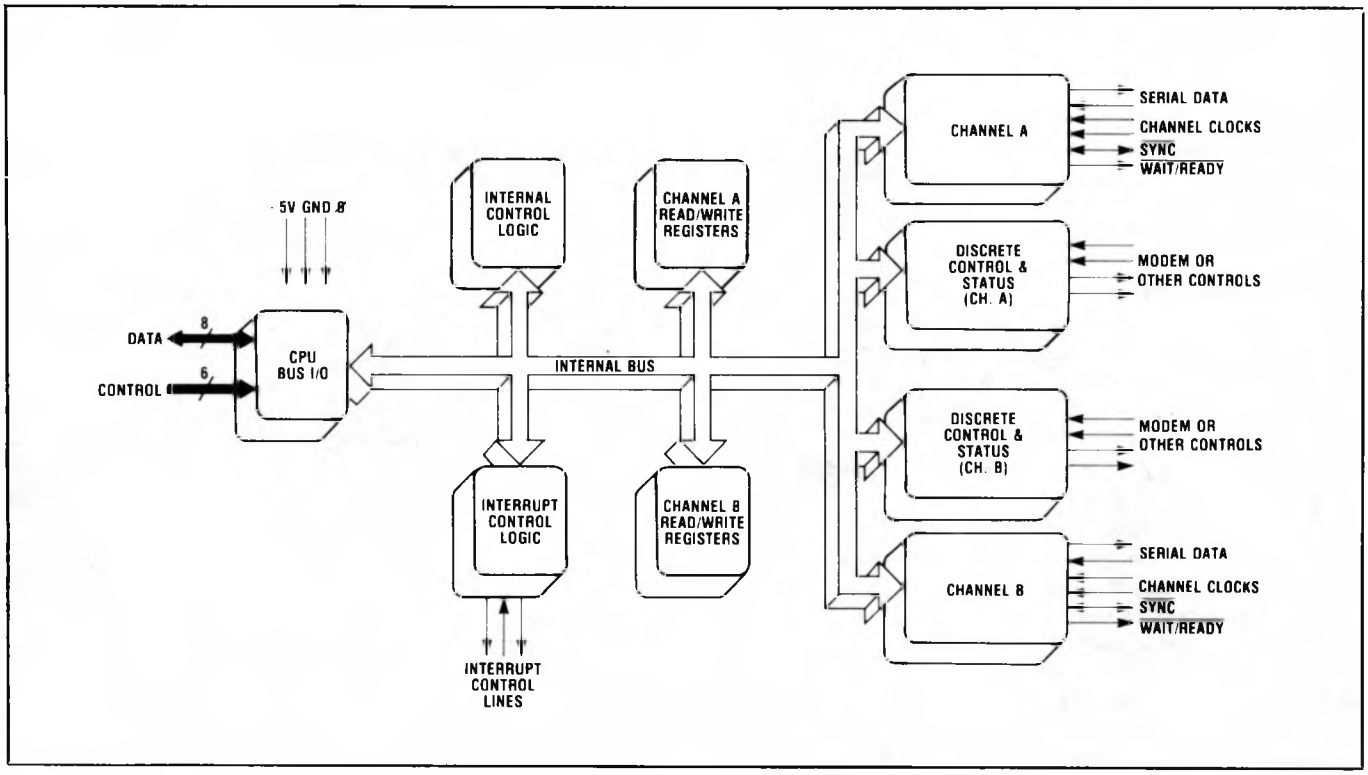
Z80-DMA/CE Industrieversion mit erweitertem Temperaturbereich (—40° +85°C) im Keramikgehäuse U_B = 5 V ± 5%

Z80-DMA/CM Militärische Version (—55°C...+125°C) im Keramikgehäuse U_B = 5 V ± 10%

Z80-SIO

Zilog

UNIVERSELLER SERIEN-EIN/AUSGABE-BAUSTEIN



2. ZILOG-Z80-MIKROCOMPUTER-BAUSTEINE

Der Baustein Z80-SIO (**S**erial **I**nput/**O**utput) ist ein komplexer, programmierbarer, unsiverseller Baustein zur Abwicklung von seriellen Ein/Ausgabe-Aktivitäten in 8bit-Mikrocomputer-Systemen. Er beinhaltet 2 vollständige Duplex-Serien-Ein/Ausgabekanäle, also insgesamt 2 Eingabe- und 2 Ausgabekanäle.
Für Anwendungen, bei denen ein einziger Duplex-Serienkanal ausreicht, ist die Version SIO/9 verfügbar, bei der lediglich die Anschlüsse des Kanals A herausgeführt sind.
Er führt die Umwandlung von paralleler in serielle Information und umgekehrt durch, wobei er für asynchrone, synchrone und bitweise synchrone Datenübertragungen geeignet ist. Dadurch ist er u.a. in der Lage, Datenübertragungsprotokolle wie das IBM, BiSync, HDLC, SDLC und serielle Übertragungen nach anderen Verfahren abzuwickeln.
Bei jeder synchronen Datenübertragung kann der Baustein zyklische Redundanzprüfungssignale (CRC) erzeugen.
In der asynchronen Betriebsart ist er über Kommandowörter der CPU jedem asynchronen Übertragungsformat anzupassen.
Der Baustein eignet sich auch besonders als universeller Magnetplatten- (z.B. Floppy-Disk-) Controller.

☐ Aufbau

- N-Kanal Silocon Gate Depletion Load Technologie
- 40 pin-DIP-Gehäuse
- Eine einzige +5 V Speisespannungsversorgung
- Einphasen 5 V-Taktanschluß
- 2 volle Duplexkanäle.

☐ Eigenschaften

- Beide Duplexkanäle voneinander völlig unabhängig
- Übertragungsrate 0...880 kBit/sec bei 4 MHz Systemtakt
- Datenempfangsregister: 4fache Pufferung
- Datensenderregister: 2fache Pufferung
- Asynchrone Betriebsweise:
  - ☐ 5, 6, 7 oder 8 Datenbits / übertragenem Zeichen
  - ☐ 1, 1½ oder 2 Stop-Bits
  - ☐ Gerade, ungerade oder keine Parität
  - ☐ x1, x16, x32 und x64 Taktzyklen-Betriebsart
  - ☐ Erzeugung und Prüfung von Break (= „Unterbrechungs"-) Bedingungen
  - ☐ Parity-Overrun und Wortformatfehler (= „Framing Error")-Prüfung
- Binär synchrone Betriebsweise:
  - ☐ Interne oder externe Zeichen-Synchronisation
  - ☐ 1 oder 2 Sync-Zeichen in getrennten Registern
  - ☐ automatisches Einfügen/Ausblenden von Sync-Zeichen
  - ☐ CRC-Erzeugung und Prüfung
- HDLC- oder IBM-SDLC-Betriebsart:
  - ☐ Null-Einfügung und -Ausblendung
  - ☐ Automatisches Flag-Einfügen
  - ☐ Adreßfeld-Erkennung
  - ☐ I-Feld-Residuum-Behandlung
  - ☐ Gültige Empfangs-Messages Overrun-geschützt
  - ☐ CRC-Erzeugung und -Prüfung
- getrennte Modem-Control-Eingänge und Ausgänge für beide Duplexkanäle
- Sowohl CRC 16 als auch CRC-CCITT (−0 und −1) sind implementiert.
- Modem-Status ist abfragbar.

☐ Architektur des SIO

Ein Blockschaltbild des Bausteins finden Sie auf der vorhergehenden Seite. Daraus sind die Hauptfunktionselemente des Bausteins ersichtlich:

- Zwei Vollduplex-Datenübertragungskanäle mit Parallel/Serien- und Serien/Parallel-Wandlern, bzw. ein solcher Vollduplex-Kanal bei SIO/9.
- Steuerlogik, Interruptlogik und das Interface zum Anschluß des Bausteins an die Mikrocomputer-Systembusse.

Die beiden Datenübertragungskanäle verfügen jeweils über fünf 8bit-Steuerregister, zwei 8bit-Statusregister und zwei 8bit Sync-Zeichen-Register.
Jeder Empfänger verfügt über drei 8bit-Pufferregister, die nach dem FIFO-Prinzip (="first in; first out") organisiert sind. Jeder Sender hat über das 8bit-Ausgabeschieberegister hinaus 1 zusätzliches Puffer-Register. Die 16bit CRC-Generatoren/Prüfer sind in geeigneter Weise intern rückgekoppelte 16bit-Schieberegister, die über Kommandowörter per Programm für zwei verschiedene CRC-Codes programmiert werden können.
Die Interrupt-Steuerlogik bestimmt, welchem Baustein und welchem Kanal innerhalb des Bausteins die momentan höchste Priorität zugeordnet ist; dadurch ist automatische priorisierte Vektorinterrupt-Behandlung ohne jegliche zusätzliche Priorisierungs- oder Interruptcontroller-Bausteine möglich.
SIO-Intern hat Kanal A die höhere Priorität.
Innerhalb beider Kanäle gilt folgende Prioritätshierarchie: Empfänger, Sender, Extern/Status.
Der Interrupt-Vektor wird in ein zusätzliches Register des Kanal B geschrieben und kann über diesen Kanal auch von der CPU gelesen werden.

☐ Anschlußbeschreibung

Bezeichnung	Funktion	Kommentar
D_0—D_7	Datenbus	Bidirektionale Tri-State-Datenleitungen zum Anschluß an den System-Datenbus
B/$\overline{A}$	Kanal „B/A-Select"	Eingang zur Kanalauswahl (High bedeutet „Kanal B selektiert")
C/$\overline{D}$	„Control/ Data-Select"	Eingang Steuerwort/Datenübertragung (High bedeutet „Steuerwort übertragen)
$\overline{CE}$	„Chip enable"	Eingang, low-aktiv: Chip-Freigabe
$\overline{M1}$	„Machine-cycle 1"	Eingang, low-aktiv: Maschinenzyklus M1 oder CPU (Steuerbus-Signal)
$\overline{IORQ}$	„I/O-Request"	Eingang, low-aktiv: Ein/Ausgabe-Anforderung von der CPU
$\overline{RD}$	„Read"	Eingang, low-aktiv: Lesezyklus der CPU (Steuerbus-Signal)
Φ	„Clock"	Eingang: Systemtakt
$\overline{INT}$	„Interrupt-request"	Ausgang: low-aktiv: Interrupt-Anforderung von der Peripherie
IEI	„Interrupt enable In"	Eingang, high-aktiv
IEO	„Interrupt enable Out"	Ausgang, high-aktiv Mit diesen beiden Signalen wird die Interrupt-Priorisierungskette (Daisy Chain) gebildet.
$\overline{RESET}$	„Reset"	Eingang, low-aktiv: Sperrt sowohl Sender als auch Empfänger T×DA und T×DB werden in den aktiven Zustand gebracht, Modem Control in den High-Zustand. Nach dem Rücksetzen müssen die Steuerregisterinformationen neu eingeschrieben werden, bevor irgendeine Datenübertragung stattfindet. Sämtliche Interrupts werden gesperrt.

Bezeichnung	Funktion	Kommentar
W/RDYA W/RDYB*	„Wait/Ready"	1 Ausgang pro Kanal, der als „Ready-Leitung für den Anschluß von DMA-Controllern oder als „Wait"-Leitung zur Synchronisierung der CPU mit der Datenübertragungsrate programmiert werden kann; low-aktiv.
CTSA CTSB*	„Clear to Send"	1 Anschluß pro Kanal, Schmitt-Trigger-Eingänge, low-aktiv: In der Betriebsart „Auto-Enable" sperrt dieses Signal den Sender seines Kanals. Wird der Anschluß nicht zur Senderfreigabe verwendet, steht er als allgemeiner Eingang zur freien Verfügung. Die beiden Leitungen haben Schmitt-Trigger-Eingänge, sodaß auch Eingangssignale geringer Flankensteilheit einwandfrei verarbeitet werden.
DCDA DCDB*	„Data Carrier Detect"	1 Anschluß pro Kanal, Eingänge, low-aktiv: Die Funktion ist ähnlich der von CTS, jedoch wird von DCD der Empfänger des zugehörigen Kanals gesperrt.
RxDA RxDB*	„Receive Data"	1 Anschluß pro Kanal, Eingänge, high-aktiv: Serielle Dateneingänge
TxDA TxDB*	„Transmit Data"	1 Anschluß pro Kanal, Ausgänge, high-aktiv: Serielle Datenausgänge
RxCA RxCB*	„Receiver Clock"	1 Kontaktfleck pro Kanal, Schmitt-Trigger-Eingänge, low-aktiv: (bitte beachten Sie den untenstehenden Kontaktierungshinweis!) Empfängertakteingang; als Taktgeschwindigkeit in der asynchronen Betriebsart können ×1, ×16, ×32 oder ×64 programmiert werden.
TxCA TxCB*	„Transmitter Clock"	1 Kontaktfleck pro Kanal, Schmitt-Trigger-Eingänge, high-aktiv: (bitte beachten Sie den untenstehenden Kontaktierungshinweis!) Sendertakteingang; als Taktgeschwindigkeit sind hier ebenfalls ×1, ×16, ×32 oder ×64 möglich, jedoch müssen Sender- und Empfänger-Taktmultiplikator gleich sein.
RTSA RTSB*	„Request to Send"	1 Anschluß pro Kanal, Ausgänge low-aktiv: Sobald das RTS-Bit eines Kanals gesetzt ist, geht die zugehörige RTS-Leitung in den low-Zustand über. Wird das RTS-Bit in der asynchronen Betriebsart rückgesetzt, geht die zugehörige RTS-Leitung in den high-Zustand, sobald das Sender-Register leer ist. In der synchronen Betriebsart fungiert der Anschluß einfach als Ausgang, an dem dauernd der Wert des RTS-Bits liegt.
DTRA DTRB*	„Data Terminal Ready"	1 Kontaktfleck pro Kanal, Ausgänge, low-aktiv: (bitte beachten Sie untenstehenden Kontaktierungshinweis!) Ausgang, an dem der Wert des DTR-Bits liegt.
SYNCA SYNCB*	„External Character Synchronisation"	2 Ein/Ausgabe-Anschlüsse, low-aktiv: In der Betriebsart Externe Synchronisation beginnt das Aufbauen des Zeichens mit der steigenden Flanke von R×C, die direkt auf die fallende Flanke des Sync-Signals folgt. In der Betriebsart Interne Synchronisation sind diese Anschlüsse Ausgänge, die in dem Teil eines Taktzyklus, aktiv sind, in dem ein Sync-Zeichen erkannt wird. Die Sync-Bedingung ist nicht zwischengespeichert, sodaß der Anschluß zu jeder Zeit unabhängig von der Wortlänge bis Erkennen eines Sync-Bitmusters aktiv ist. In der asynchronen Betriebsart sind diese Anschlüsse einfach direkte Eingänge für die Hunt/Sync-Bits im Statusregister 0 und können zu beliebigen Eingabezwecken herangezogen werden. Bitte beachten Sie: Wird der Anschluß zur externen Synchronisation benützt, darf er frühestens 2 Taktzyklen nach der steigenden Flanke des R×C-Signals aktiv werden, bei dem das letzte Bit des Sync-Zeichens empfangen wurde. Diese Bedingung läßt sich im allgemeinen einfach erfüllen, indem man eine Änderung des SYNC-Signals ausschließlich bei der fallenden Flanke von R×C zuläßt.

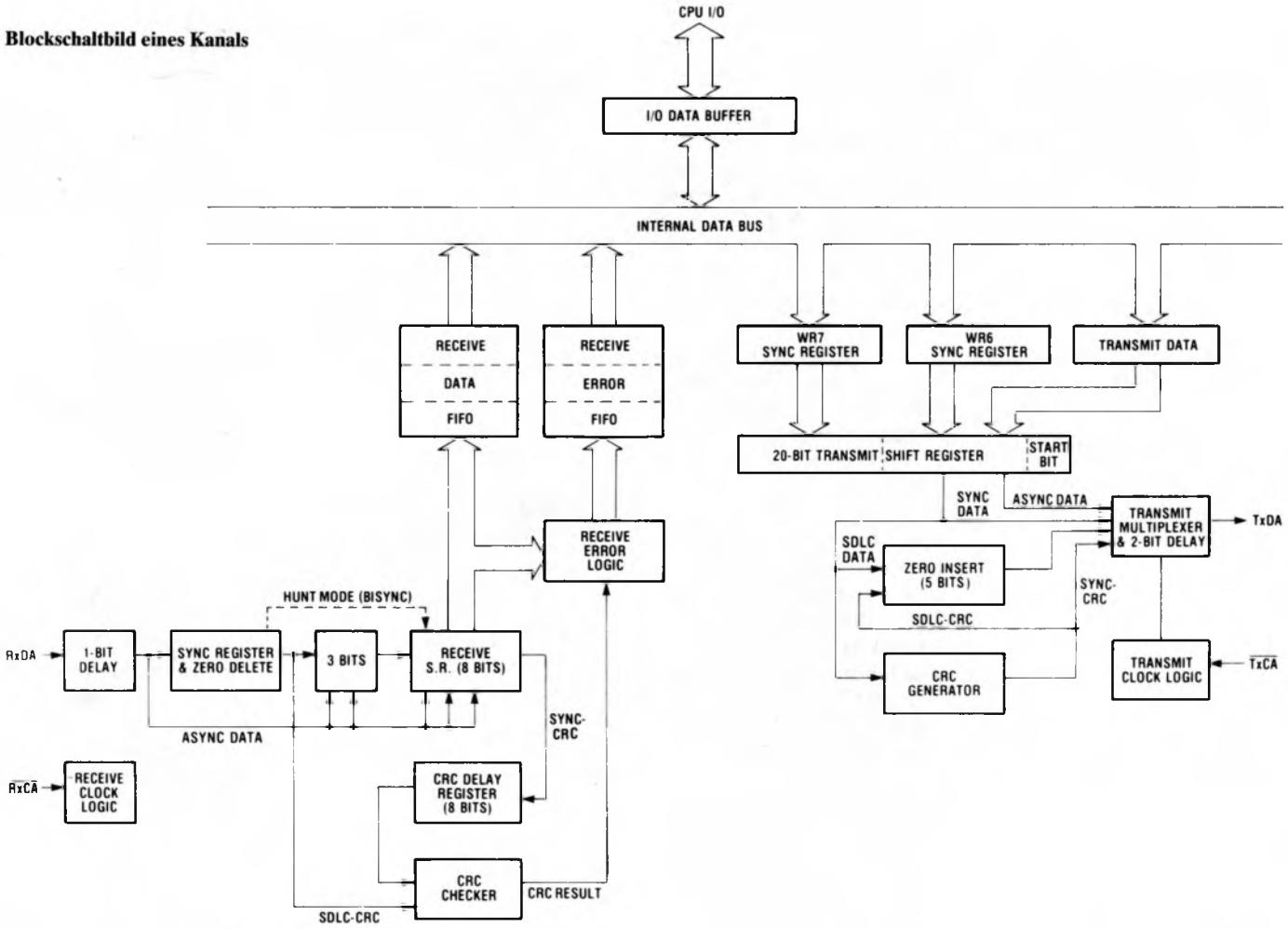
*: Nicht bei SIO/9

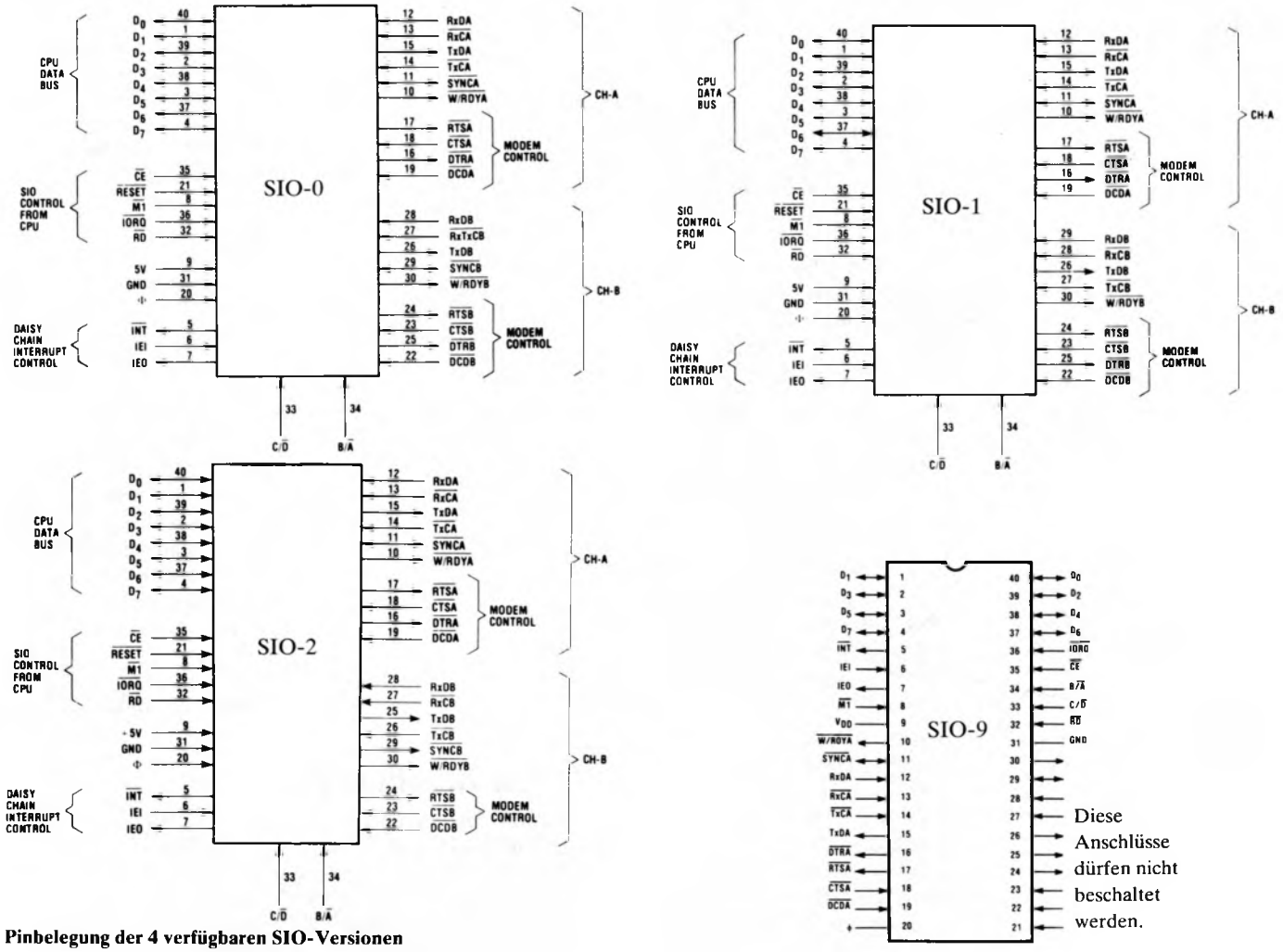
☐ Zur Beachtung*:

Aufgrund der begrenzten Anschlußzahl (40) stehen für die 4 Signale TxCB, RxCB Sync. und DTRB nur 3 pins zur Verfügung. Aus diesem Grund werden folgende 3 Kontaktierversionen des SIO angeboten:

- ☐ SIO/0: TxCB und RxCB sind miteinander verbunden
- ☐ SIO/1: DTRB ist nicht herausgeführt
- ☐ SIO/2: SYNCB ist nicht herausgeführt

Blockschaltbild eines Kanals



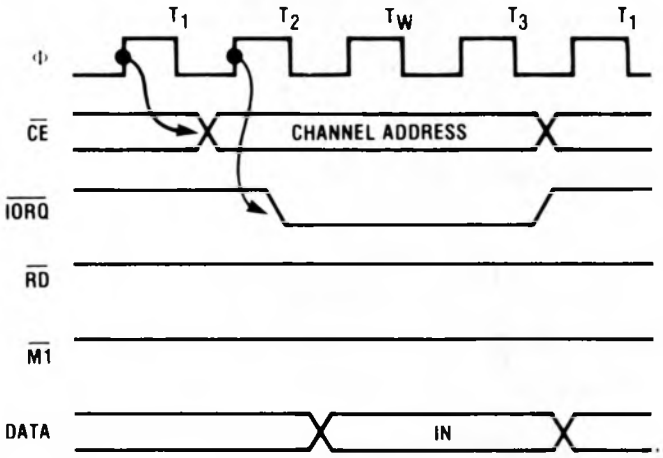


Diese Anschlüsse dürfen nicht beschaltet werden.

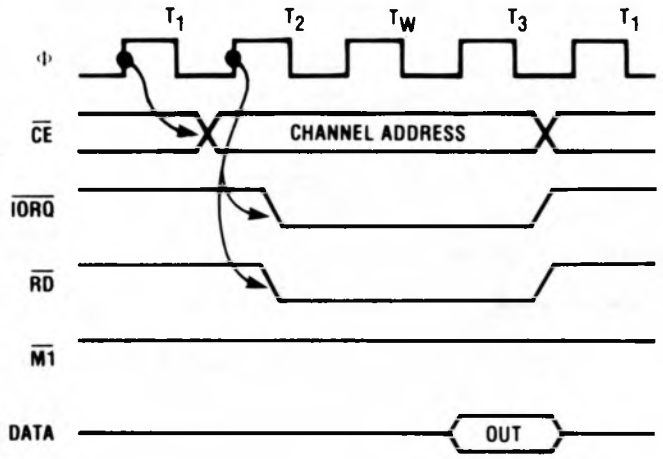
Pinbelegung der 4 verfügbaren SIO-Versionen

SIO-Zeitverhalten

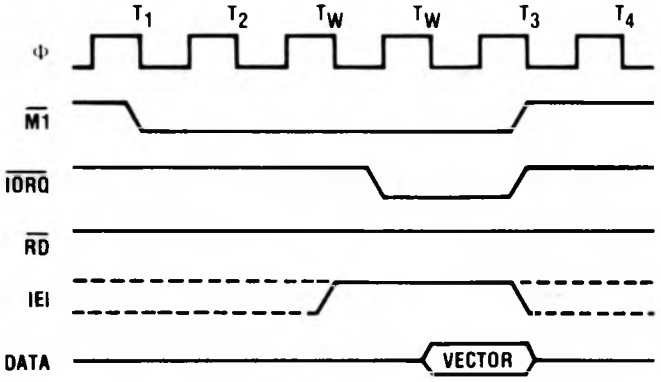
Schreib-Zyklus
Der nebenstehende Zyklus zeigt den Zeitablauf beim Schreiben eines Daten- oder Kontrollbytes in den SIO. Sämtliche Z80 Ausgabebefehle entsprechen diesem Zeitablauf.



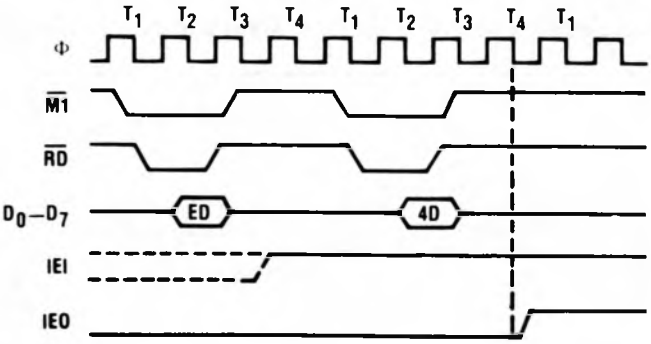
Lese-Zyklus
Der nebenstehende Zeitablauf zeigt das Lesen von Daten oder Statusinformationen aus dem SIO.
Sämtliche Z80 Eingabebefehle entsprechen diesem Zeitablauf.



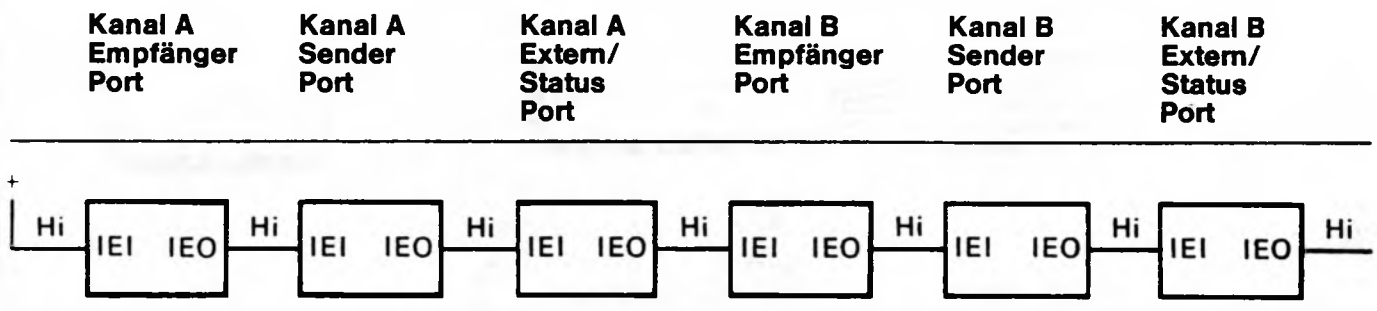
Interrupt-Quittungs-Zyklus
Bei der Annahme eines Interrupts vom SIO durch die CPU sendet diese eine Interrupt-Annahme-Bestätigung (M1 und IORQ). Die Interrupt-Logik des SIO bestimmt darauf die höchstpriorisierte Funktion, die einen Interrupt anmeldet. Um die Stabilisierung der Daisy-Chain-Leitungen zu garantieren, sind die Kanäle während des M1-Signals blockiert und können ihren momentanen Zustand nicht ändern. Der SIO plaziert den entsprechenden Interrupt-Vektor während des Signals IORQ auf den Datenbus.



Rücksprung vom Interrupt-Zyklus
Falls ein Z80 Peripherie-Baustein weder einen Interrupt angemeldet hat noch seine Service-Routine laufen hat, gilt IEO = IEI. Bei der momentanen Abarbeitung einer Interrupt-Service-Routine liegt der Ausgang IEO auf null und blockiert die nachfolgenden Peripheriebausteine. Falls ein Interrupt angemeldet, aber noch nicht bestätigt ist, liegt IEO auf null bis der Code „ED" als erstes Byte eines 2-Byte-Befehls decodiert wird. Zu diesem Zeitpunkt liegt IEO auf 1 bis zur Decodierung des nächsten Befehls-Bytes. Anschließend geht IEO auf null. War das 2. Byte des Befehls „4D", so entspricht es einem RETI-Befehl. Nach der Decodierung des Befehlsbytes „ED" hat nur der Peripheriebaustein IEI auf eins und IEO auf null, dessen Interrupt-Anmeldung akzeptiert wurde. War das nächste Byte ein „4D" geht IEO wieder auf eins.

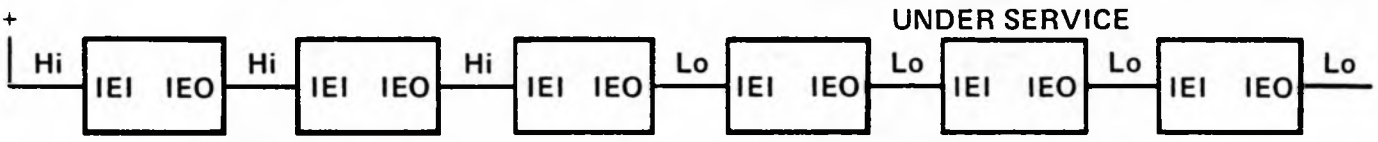


Das folgende Bild erläutert die Priorisierung der einzelnen interruptanfordernden Funktionaleinheiten innerhalb des SIO. Selbstverständlich kann jederzeit eine Kaskadierung mit weiteren SIO's oder anderen Peripherie-Bausteinen der Z80-Familie erfolgen.

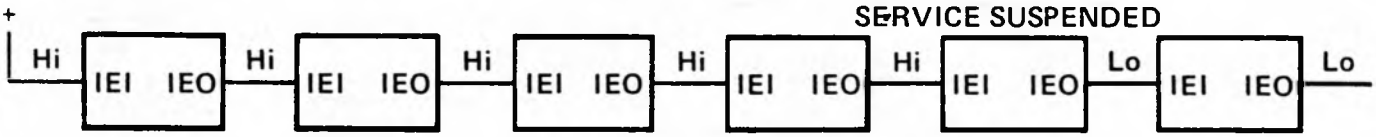


1. Daisy Chain vor irgendeinem Interrupt.

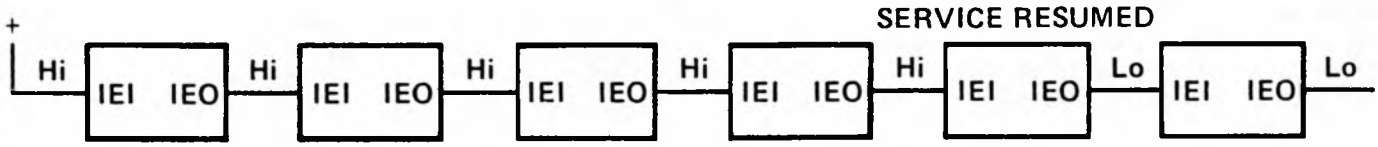
Kanal A Empfänger Port	Kanal A Sender Port	Kanal A Extern/ Status Port	Kanal B Empfänger Port	Kanal B Sender Port	Kanal B Extern/ Status Port



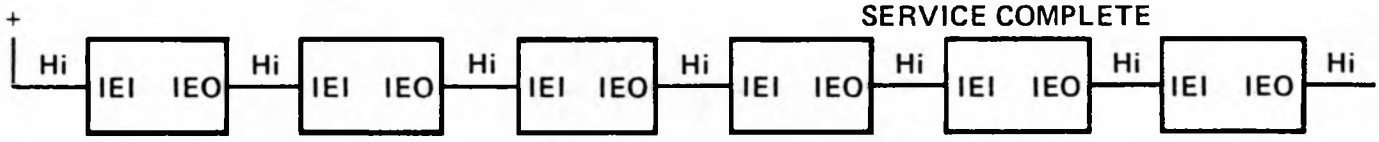
2. Sender port von Kanal B fordert Interrupt an und erhält Bestätigung.



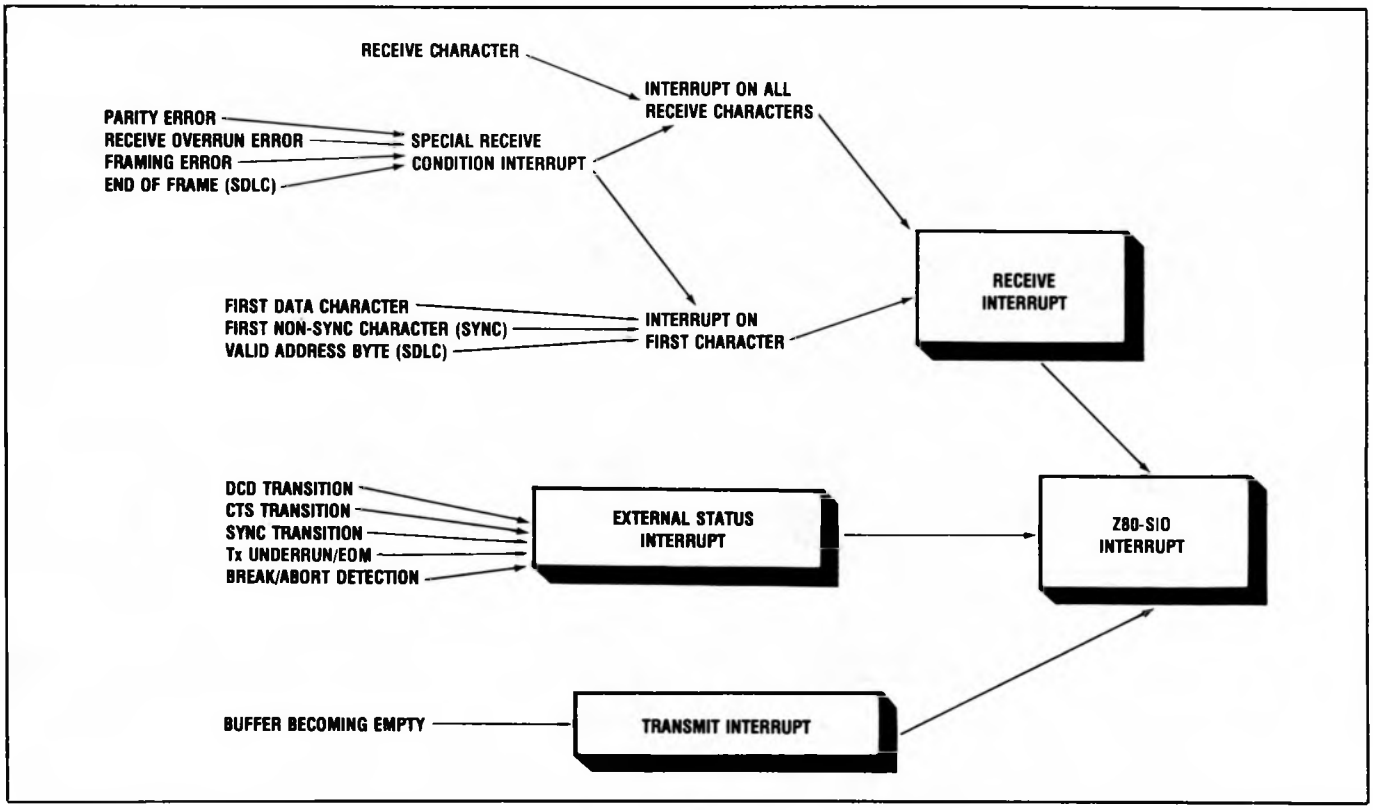
3. Extern/Status-Port von Kanal A fordert Interrupt an und unterbricht die Bedienung von Kanal B.



4. Bedienroutine für Extern/Status-Port von Kanal A ist fertig. Die RETI-Anweisung wird auf den Bus gelegt und die Bearbeitung der Service-Routine für das Senderport von Kanal B wieder aufgenommen.



5. Service-Routine für das Sender-Port von Kanal B ist fertig, zweite RETI-Instruktion liegt auf dem Bus.



Arbeitsweise des SIO

Das Verhalten des SIO wird durch den Inhalt seiner Steuerregister bestimmt. Bevor der SIO einen Datentransfer durchführen kann, müssen seine sämtlichen Steuerregister programmiert werden, wobei einige Kommandos und Verhaltensweisen des SIO auch während des Programmablaufes modifiziert werden können. Die Statusregister sind zu jedem Zeitpunkt auslesbar.

Asynchronbetrieb
Die Eingangskanäle sind 4-fach gepuffert, d.h. es existieren 3 Speicherregister zusätzlich zum Eingabeschieberegister. Die CPU erhält damit Zeit für den Start einer Interrupt-Service-Routine, auch wenn der zu übernehmende Datenblock mit hoher Geschwindigkeit eintrifft. Fehlerbits sind ebenfalls 4-fach gepuffert und werden zur selben Zeit wie die Zeichen geladen. Die Flags-, Empfängerüberlauf- und Paritätsfehler (Receiver Overrun und Parity-Error) werden nur durch einen Error Reset-Befehl (Kommando 6) rückgesetzt. Rahmenfehler und CRC-Fehler zeigen den Status des momentan im Puffer befindlichen Zeichens an. Beim Auslesen des Fehlerstatus zeigt dieser Fehler des momentan im Empfangsregister befindlichen Zeichens sowie irgendwelche Paritäts- oder Überlauffehler, die seit den letzten Fehlerrücksetzkommandos aufgetreten sind, an. Um die Übereinstimmung zwischen dem Zustand des Fehlerregisters und dem Inhalt des Empfangsregisters aufrechtzuerhalten, sollte das Statusregister vor den Daten gelesen werden. Bei vektorisiertem Interrupt wird wegen der Erzeugung eines speziellen Interruptvektors bei Auftreten von Fehlern diese Bedingung leicht eingehalten.
Eine Ausnahme tritt auf bei der Auswahl des Arbeitsmodus „Empfänger-Interrupt nur beim 1. Zeichen". Bei diesem speziellen Interrupt werden sowohl der Fehler als auch das Zeichen so lange gespeichert, bis das Fehlerrücksetzkommando ausgeführt wird. Dadurch wird die Übernahme von neuen Daten bis zur Übernahme der Fehlermeldung verhindert.
Wird der Modus „Interrupt bei jedem Zeichen" gewählt, ändert sich der Interrupt-Vektor abhängig von auftretenden Fehlern. Bei Überlauf wird das letzte eintreffende Zeichen übernommen und das vorhergehende Zeichen geht verloren.
Beim Lesen dieses Zeichens, welches über vorhergehende Zeichen geschrieben wurde, ist das Überlauf-Bit gesetzt und der Interrupt-Vektor „Special Receive Condition" wird geliefert („Status Affects Vector" muß aktiviert sein). Es ist möglich, den SIO in einem Polling-System zu verwenden. Hier muß das Bit „Receive Character Available" als Empfangsmeldung geprüft werden. Sind alle Empfangspufferregister leer, wird dieses Bit automatisch rückgesetzt. Das Bit „Transmit Buffer Empty" muß bei Polling-Systemen geprüft werden, bevor neue Daten in den Sender geschrieben werden können.

Senden
Im Asynchron-Betrieb sendet der SIO ein Zeichen in folgender Form; Im Ruhezustand liegt die Ausgangsleitung auf 1, vorausgesetzt, daß kein „BREAK" programmiert wurde. In diesem Fall liegt die Leitung auf 0 bis das „Send-Break"-Kommando gelöscht und der Baustein rückgesetzt wurde.
Senden kann erst nach dem Setzen des Bits „Transmit Enable" beginnen. Bei der Wahl der Option „Auto Enables" muß der Eingang CTS auf null liegen. Werden die Zeichen mit 5 Bit codiert, müssen die nicht verwendeten Bits D_5, D_6 und D_7 in den eingeschriebenen Datenbytes null sein.

Empfang
Asynchroner Empfang wird nach dem Setzen des Bits „Receiver Enable" möglich. Bei der Wahl der Option „Auto Enables" muß der Eingang DCD auf null liegen. Ein Übergang von 1 auf 0 am RxD-Eingang markiert ein Startbit. Beim Anliegen von mindestens einer halben Bit-Länge wird dieses Startbit als gültig betrachtet und die eintreffenden Datenbits werden so lange zu ihrem mittleren Zeitpunkt abgetastet, bis das vollständige Zeichen übernommen ist. Diese Form der Erkennung eines Startbits verbessert die Fehlererkennung beim Auftreten von Störspitzen auf der Empfangsleitung.

Synchronbetrieb
Alle Formen der synchronen Datenübertragung benötigen einen ungeteilten Takt (x1) für Senden und Empfang. Daten werden zum Zeitpunkt der steigenden Flanke von RxC abgetastet. Sendedatenänderungen erfolgen während der fallenden Flanke von TxC. In allen Fällen befindet sich der Empfangsteil in einem Suchlauf (nach einem internen oder externen Reset). Der Suchlauf startet nur nach der Aktivierung des Empfängers und der eigentliche Datentransfer benötigt eine vorhergehende Zeichensynchronisierung. Nach einem Synchronisierungsverlust kann der Suchlauf durch Schreiben eines Steuerwortes neu gestartet werden, wenn das „Enter Hunt Mode"-Bit gesetzt war. Die Unterschiede der Übertragungsformen, Monosync, Bisync und externe Synchronisierung, ergeben sich nur durch die Form der Erstsynchronisation.

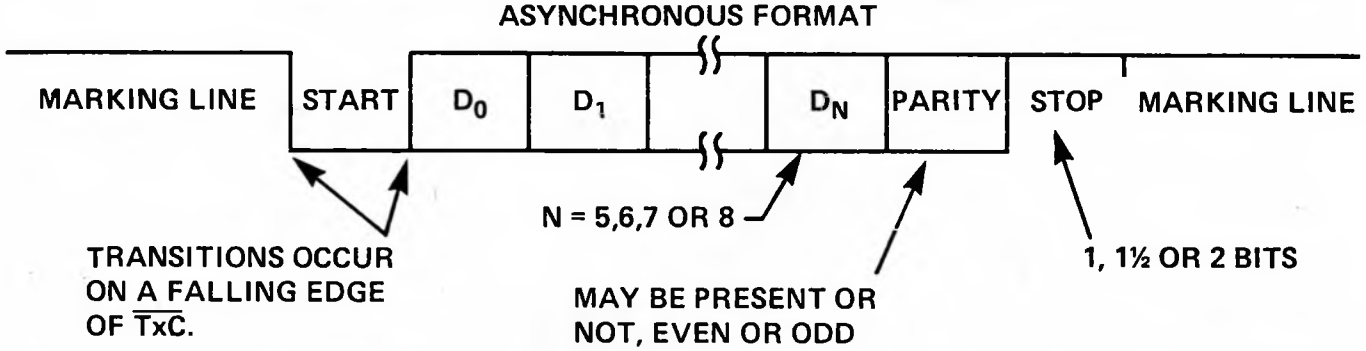
Monosync (8 Bit-Synchronisierung)
Beim Erkennen eines einfachen Synchronzeichens (programmiert in Steuerregister 7) beginnt der Datentransfer.

Bisync (16 Bit-Synchronisierung)
Beim Erkennen von zwei aufeinanderfolgenden Synchronzeichen (programmiert in Steuerregister 6 und 7) beginnt der Datentransfer.
In beiden Fällen (monosync und bisync) geht die SYNC-Leitung beim Erkennen einer Synchronzeichenfolge auf null. Sie bleibt null während des gesamten folgenden Taktimpulses.

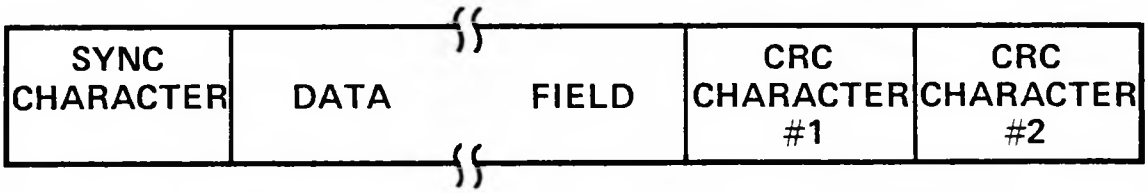
Externe Synchronisierung
Datentransfer beginnt mit der ersten ansteigenden Flanke von RxC nach Aktivierung des Sync-Einganges (aktiv = 0). Der Sync-Eingang sollte während eines kompletten Taktes aktiv bleiben.
Bei allen drei Varianten wird die Übernahme von Daten bis zum Auftreten folgender Ereignisse durchgeführt:

- Externe oder interne Rücksetzung des SIO
- Blockierung des Empfangskanals (per Kommando oder DCD)
- Setzen des Bits „Enter Hunt Mode"

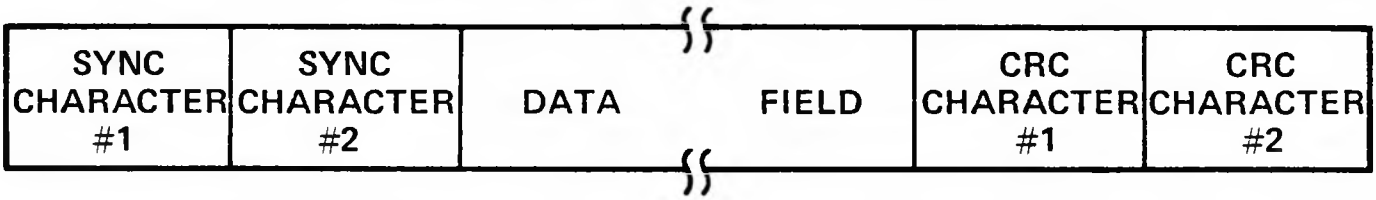
Nach der Erstsynchronisierung unterscheiden sich die drei Varianten nur mehr durch die nachfolgenden Formate:



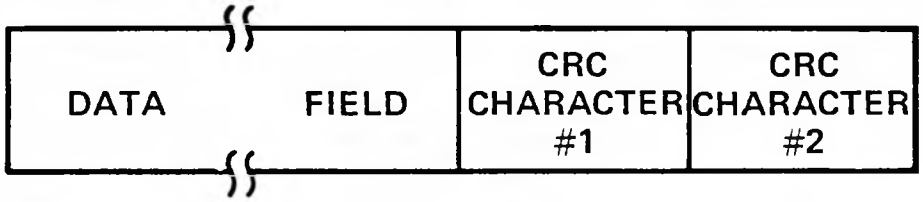
MONOSYNC MESSAGE FORMAT (Internal Sync Detect)



BISYNC MESSAGE FORMAT (Internal Sync Detect)



EXTERNAL SYNC DETECT FORMAT



Senden in Synchronbetrieb (ohne SDLC)

a) Ruhezustand
Im Grundzustand ist die Leitung aktiv (nach Reset oder blockiertem Sendekanal). Break kann zur Erzeugung des Zustandes null auf der Sendeleitung programmiert werden. Der Zustand beginnt im Moment der Programmierung unabhängig vom Inhalt des Senderegisters. Nach der Aktivierung des Sendekanals und der Auswahl der Synchronübertragungsform werden automatisch 8 oder 16 Bit-Synchronzeichen ausgesandt.

b) Verschiedene Interrupt-Aufrufe sind möglich:
- Sendeinterrupt aktiv
  Falls das „Sendeinterrupt Aktiv"-Bit gesetzt ist, wird nach Ausgabe jedes Zeichens (leeres Senderegister) ein Interrupt erzeugt. Dieser Interrupt kann durch Schreiben eines neuen Zeichens in den Sender oder durch Kommando 5 (Reset Transmitter Interrupt Pending) abgearbeitet werden. Bleibt das Senderegister leer, werden keine weiteren Interrupts erzeugt. Nach Schreiben eines neuen Zeichens wird das Senderegister erneut leer und erzeugt den nächsten Interrupt.
- Extern/Statusinterrupt aktiv
  Nach Setzen des Bits „Extern/Statusinterrupt aktiv" können verschiedene Sendebedingungen (Senden von CRC-Zeichen, Senden von SYNC-Zeichen, Änderung von CTS) einen Interrupt erzeugen.
- Alle Interrupts können bei Anwendung in einem Polling-System verhindert werden.

c) Falls CRC inaktiv ist, werden Sync-Zeichen automatisch vom Sender dann ausgeschickt, wenn keine Daten für den Transfer zur Verfügung stehen. Bei aktiviertem CRC wird in der ersten Datenpause das 16 Bit-CRC ausgesandt, gefolgt von Sync-Zeichen.
Während des Aussendens des CRC ist das Bit „Sending CRC/SYNCS" gesetzt und das Bit „Transmit Buffer Empty" zeigt ein volles Senderegister an. CRC wird nicht während des automatischen Aussendens von Sync-Zeichen berechnet.
Kontrolle des CRC-Generators:
Rücksetzen erfolgt mit dem Kommando „Reset Transmit CRC Generator". Nach dem Aktivieren des CRC und des Sendeteils können Daten geladen werden. Vor dem Aussenden von CRC (aber nach dem Laden der ersten Daten) muß das „CRC/SYNC SENT/SENDING" Flag mit dem Kommando „RESET CRC/SYNC SENT/SENDING" rückgesetzt werden.

d) Wird der Sender während der Ausgabe eines Zeichens deaktiviert, wird zwar das Zeichen vollständig ausgegeben, anschließend folgen aber weder CRC noch Sync-Zeichen. Ein im Puffer vorhandenes Zeichen bleibt dort erhalten. Wird während des Aussendens eines Zeichens ein BREAK aktiviert, geht das Zeichen verloren.

e) Grundsätzlich erfolgt die Reihenfolge der Aussendung der Bits vom untersten zum obersten. Daher müssen alle Daten (bei einer kleineren Wortlänge als 8 Bit) nach rechts justiert werden. Bei einer Wortlänge von 5 Bit und darunter muß eine Spezialtechnik (Sektion „Transmit Bits/Char") angewandt werden.

Empfang (außer SDLC)

a) Nach der Programmierung der Übertragungsart und der Sync-Zeichen (in dieser Reihenfolge) kann der Empfänger aktiviert werden. Er geht dann in den Suchlauf und bleibt dort so lange bis einer der folgenden Zustände eintritt:

1) Empfang eines einfachen Sync-Zeichens (Monosync)
2) Empfang eines doppelten Sync-Zeichens (Bisync)
3) Der externe Sync-Eingang geht auf null

In den Fällen 1 und 2 ist die Sync-Leitung ein Ausgang, der die Zeichensynchronisierung meldet. Im Fall 3 ist Sync ein Eingang.

b) Zeichenübernahme beginnt nach Erkennung eines entsprechenden Sync-Zeichens. 4 Formen des Interrupts sind möglich:

1) Interrupt deaktiviert für ein Polling-System oder im „Off-line"-Betrieb

2) Interrupt nur beim ersten Zeichen
Diese Form startet normalerweise eine Abfrageschleife oder einen Block-Transferbefehl, wobei mit Hilfe des „Wait/Ready"-Ausgangs die CPU auf die ankommende Datentransfergeschwindigkeit synchronisiert werden kann. Ebenso kann der SIO mit einem DMA-Baustein verbunden werden, wobei der SIO einen Interrupt beim ersten Zeichen aber anschließend nur mehr bei Auftreten von Fehlern liefert. Diese Interrupt-Form wird mit dem Kommando 4 (Reset Receive Interrupt On First Character) rückgesetzt. Das erste Zeichen nach diesem Kommando erzeugt noch einen Interrupt. Sind externe Statusinterrupts aktiviert, können sie jederzeit auftreten. Paritätsfehler erzeugen keinen Interrupt, jedoch Rahmenfehler und Empfängerüberlauf.

3) Interrupt bei jedem Zeichen
Bei jedem Empfang eines Zeichens wird ein Interrupt erzeugt. Fehler und spezielle Bedingungen erzeugen einen Spezialvektor falls die Form „Status Affects Vektor" aktiv wird. Die Erzeugung eines speziellen Vektors beim Auftreten eines Paritätsfehlers kann optional unterdrückt werden.

c) CRC-Erzeugung und -Kontrolle

1) Die Berechnung eines CRC eines bestimmten Zeichens beginnt 8 Bit-Zeiten bevor das Wort in den Empfänger transferiert wird. Wird der CRC vor dem Transfer des nächsten Zeichens aktiviert, berechnet sich der CRC aufgrund dieses Zeichens. Wird der CRC vor dem nächsten Transfer desaktiviert, erfolgt die Berechnung noch für das momentan verarbeitete Zeichen, aber nicht mehr für das nachfolgende.

2) Der CRC kann so oft wie nötig für eine vorgegebene Berechnung aktiviert und desaktiviert werden.

3) CRC-Codes werden während des Auswahlvorganges der Übertragungsart programmiert. Entweder das CRC 16 Polynom $x^{16}+x^{15}+x^2+1$ oder das SDLC-Polynom $x^{16}+x^{12}+x^5+1$ kann gewählt werden. In allen Fällen außer SDLC wird der CRC-Generator und Kontroller auf null rückgesetzt. Sender und Empfänger müssen dasselbe Polynom verwenden.

4) In Monosync. Bisync und externer Synchronisation enthält das CRC/FRAMING ERROR Bit das Ergebnis der CRC-Kontrolle. Sie erfolgt 16 Bit-Zeiten nach dem Transfer des Zeichens vom Empfangsschieberegister zum Puffer. Eine fehlerfreie Übertragung ist durch null gekennzeichnet. Der Vergleich wird bei jedem Transfer durchgeführt und ist nur gültig, so lange das Zeichen im Empfangs-FIFO enthalten ist.

SDLC-Senden

a) Üblicherweise wird der CRC-Generator vor der Übertragung von Datenblocks rückgesetzt (mit dem Kommando „Reset Transmit CRC Generator"). Dieses Kommando kann jederzeit nach dem Aussenden des CRC der vorhergehenden Meldung erfolgen. Während des Aussendens des CRC ist das Bit „CRC/SYNC SENT/SENDING" gesetzt, jedoch das Bit „Trans Buffer Empty" ist nicht gesetzt. Nach dem Aussenden des CRC wird das „Trans Buffer Empty"-Bit gesetzt. Falls Sendeinterrupts aktiviert sind, zeigt ein stehender Interrupt die vollständige Aussendung des CRC an.

b) Im Ruhezustand werden kontinuierliche Flags gesendet (Sender aktiv), ansonsten wird eine Marke ausgesandt (Log. 1).

c) Eine Abbruchsequenz kann durch das Kommando 1 (Send Abort) erzeugt werden. Dadurch werden 8 bis 14 1-Bits ausgesandt, dann folgen Flags. Alle Sendedaten im Senderegister und Puffer gehen verloren.

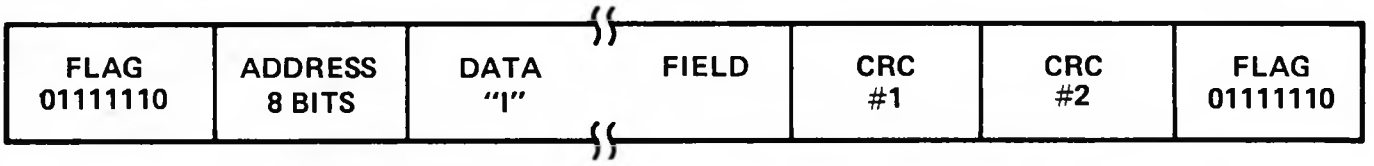
d) Ein bis 8 Bit pro Zeichen können gesendet werden (siehe Beschreibung des Steuerregister 5). Da die Anzahl der Bits pro Zeichen jederzeit geändert werden kann, ist das Füllen eines Datenfeldes mit einer beliebigen Anzahl von Bits möglich. In Verbindung mit den Receiver-Residue-Codes ist der SIO imstande, eine Meldung beliebiger Bit-Anzahl zu empfangen und sie in exakt gleicher Form rückzusenden ohne zusätzliche Information über die Zeichenstruktur des Datenfeldes (I-Feld). Ein Wechsel in der Zahl der Bits pro Zeichen beeinflußt nicht das momentan gesendete Zeichen. Zeichen werden mit der Anzahl von Bits ausgesandt, die im Moment der Übertragung des Zeichens vom Puffer zum Senderegister programmiert sind.

e) Sind keine Daten zum Senden verfügbar, wird automatisch die 2 Byte CRC-Sequenz ausgesandt. Beim Aussenden des CRC wird das Bit CRC/SYNCS SENT/SENDING gesetzt und ein Interrupt (Statuswechsel) erzeugt, sofern Extern/Status Interrupt aktiviert ist. Diese Information kann zur Anzeige des leeren Senders verwendet werden. Nach dem Aussenden des CRC werden kontinuierlich Flags gesendet. Die Kontrolle des CRC-Generators kann in folgender Weise stattfinden:
1) Auswahl der Übertragungsart
2) Rücksetzen des CRC-Generators
3) Schreiben der ersten zwei Datenbytes
4) Rücksetzen des Bits „CRC/SYNCS SENT/SENDING"
5) Schreiben der restlichen Daten
6) Nach der vollständigen Aussendung der Daten werden CRC und Flags automatisch gesendet und der Ablauf kann ab Punkt 2 wiederholt werden.

f) Null-Bits können automatisch in den Datenstrom eingefügt werden, um die Bedingung von max. 5 Eins-Bits in einer Reihe zu erfüllen (gilt nicht für Flags oder Datentransfer-Abbruch).

g) Bei der Auswahl der SDLC-Datenübertragung ergibt das Rücksetzen des CRC-Generators tatsächlich Eins-Bits. Der SDLC/CRC-Code muß eigens gewählt werden.

SDLC/HDLC Message Format



SDLC-Empfang

a) Der Datentransfer beginnt mit dem ersten Zeichen nach mindestens einem Flagzeichen (01111110), sofern Adress-Search-Mode nicht aktiv ist. Bei Aktivierung muß dem Flag entweder die Standard-Adresse (11111111) oder eine programmierte folgen, bevor Datentransfer möglich ist.

1) Falls Interrupts inaktiv sind, kann das Vorhandensein eines Zeichens durch Prüfen des Bits „Receive Character Available" erkannt werden.

2) Falls „Interrupt On First Character Only" gewählt ist, wird damit meist ein Block-Transfer aktiviert. Bei unbekannter Datenblock-Länge kann der „Special Condition"-Interrupt (End of Frame) zum Ausstieg aus der Software-Schleife verwendet werden.
Vor Einlangen des nächsten Datenblocks muß das Kommando 4 „Reset Interrupt On First Character" erfolgen bevor ein neuerlicher Interrupt mit dem ersten Zeichen des nachfolgenden Blocks durchgeführt werden kann.

3) Flags werden nicht übernommen. Die Nullen, die während der Übertragung eingeschaltet wurden, werden automatisch gelöscht.
4) Abbrüche werden als 7 oder mehr Einsen entdeckt und erzeugen einen Statusinterrupt (falls aktiviert) und setzen das „Break Abort"-Bit in Statusregister 0. Nach Ausführung des Kommandos 2 „Reset External/Status Interrupts" wird ein zweiter Interrupt nach Beendigung der Sendung von kontinuierlichen Einsen durchgeführt.

b) Im SDLC-Format ist die Kontrolle des Empfangs-CRC-Generators automatisch. Er wird mit der führenden Flag rückgesetzt und CRC wird berechnet bis zur letzten Flag. Das Byte, welches das „End of Frame"-Bit setzt, ist gleichzeitig das Byte, welches das Ergebnis der CRC-Kontrolle enthält. Ist das Bit „CRC/Framing Error" nicht gesetzt, zeigt der CRC eine gültige Meldung an. Bei der SDLC-Kontrolle muß das Endergebnis 0001110100001111 sein.

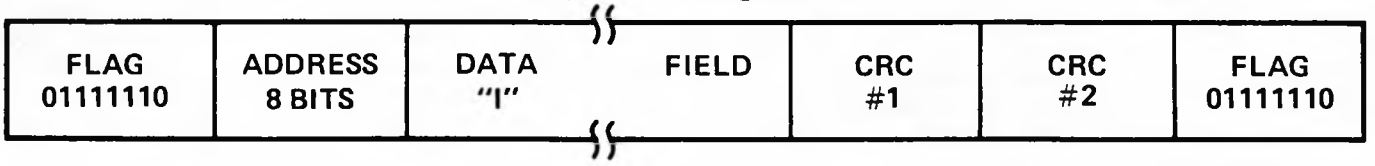
c) Zeichenbitlänge kann während des Ablaufs geändert werden. Falls Adreß- und Steuerbit als 8 Bit-Zeiten transferiert werden, kann der Empfänger während der Übernahme des ersten Informationszeichens auf eine kürzere Zeichenbit-Länge geändert werden.

d) Falls Adreß-Suchlauf nicht gewünscht wird oder Sendungen mit Mehr-Byte-Adressen eintreffen, muß die ungewünschte Meldung nicht komplett übernommen werden. Das Setzen des Bis „Enter Hunt Mode" verhindert die Übernahme von Meldungen bis eine neue Meldung (mit einem Flag am Anfang) empfangen wird.

e) Beim Empfang des führenden Flags wird ein Interrupt mit einem Spezialvektor erzeugt (falls aktiviert). Damit wird der Empfang des Bytes mit einem gesetzten Bit „End of Frame" signalisiert (zuzüglich zum Ergebnis der CRC-Kontrolle). Statusregister 1 enthält 3 Bits des Residue-Codes, der im Moment gültig ist. Im Falle, daß die Anzahl der Bits im I-Feld kein ganzzahliges Vielfaches der Zeichenlänge ist, geben diese Bits den Zwischenraum zwischen den CRC-Steuerbits und den I-Feld-Bits an (siehe Statusregister 1).

f) Paritätskontrolle kann bei Daten im I-Feld nur bei 5 bis 7 Bit-Zeichen und nur bei Halb-Duplex-Protokollen durchgeführt werden.

SDLC/HDLC Message Format



Programmieren des SIO

Allgemeines

Der Z80-SIO ist eine Multifunktions-Peripheriekomponente, die ein weites Feld von seriellen Datenübertragungsprotokollen überstreicht. Seine grundlegende Funktion ist die eines seriell/parallel, parallel/seriell Umsetzers, aber innerhalb dieser Funktion kann man mit Hilfe von Programmbefehlen ihn auf die jeweilige Anwendung festlegen. Zur Programmierung benötigt der SIO eine Serie von Kommandos, die die gewünschte Grundfunktion einstellen und andere Kommandos, die innerhalb der gewählten Betriebsart bestimmte Bedingungen auswählen. Jeder der 2 Kanäle des SIO enthält 8 Steuerregister, die vom System vor der Durchführung eines gewünschten Datentransfers programmiert werden müssen. Der Kanalauswahleingang (B/$\overline{A}$) und der Steuerdateneingang (C/$\overline{D}$) werden über den Adreßbus der Z80-CPU gesteuert.

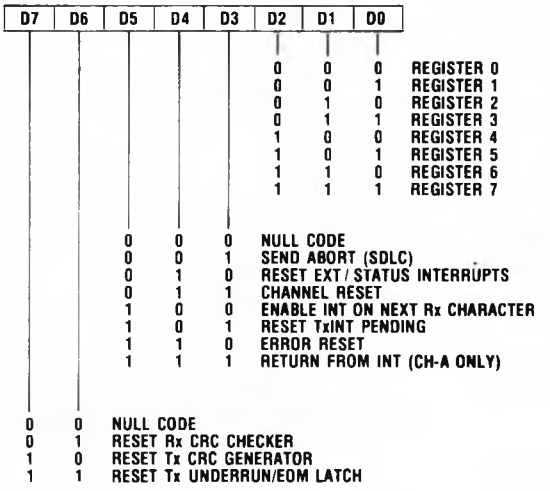
C/$\overline{D}$	B/$\overline{A}$	Funktion
0	0	Kanal A Daten
0	1	Kanal B Daten
1	0	Kanal A Kommando/Status
1	1	Kanal B Kommando/Status

Steuerregister

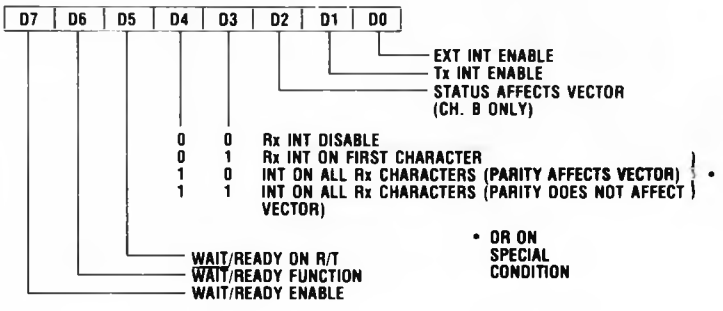
Der Z80-SIO enthält 8 Steuerregister, in die die Codewörter für die gesamten Funktionen eines Kanals eingeschrieben werden. Alle Steuerregister außer Steuerregister 0 benötigen 2 Bytes zur funktionsfähigen Programmierung. Das erste Byte enthält 3 Bits, die das ausgewählte Register markieren (D0 - D2). Das zweite Byte ist das tatsächliche Steuerwort.

Das Steuerregister 0 ist ein Spezialfall. RESET (entweder internen Befehl oder externer Eingang) initialisiert den SIO auf das Steuerregister 0. Alle grundlegenden Kommandos (CMD 2-CMD 0) und CRC-Befehle (CRC 0, CRC 1) können durch ein einziges Steuerbyte im Steuerregister 0 aktiviert werden.

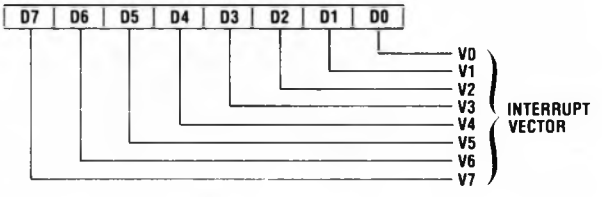
WRITE REGISTER 0



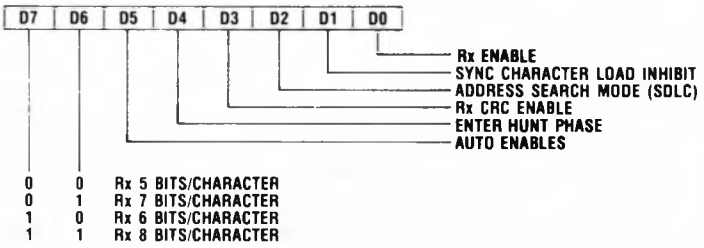
WRITE REGISTER 1



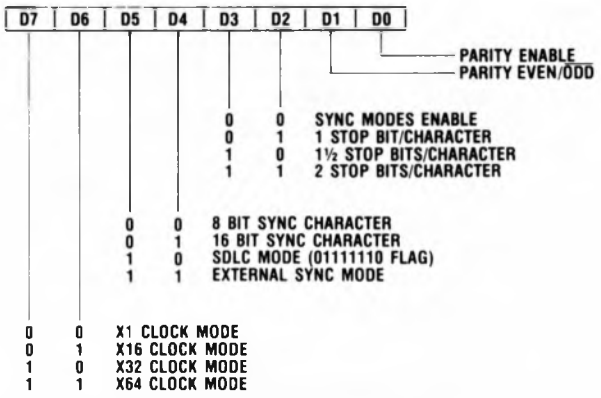
WRITE REGISTER 2 (CHANNEL B ONLY)



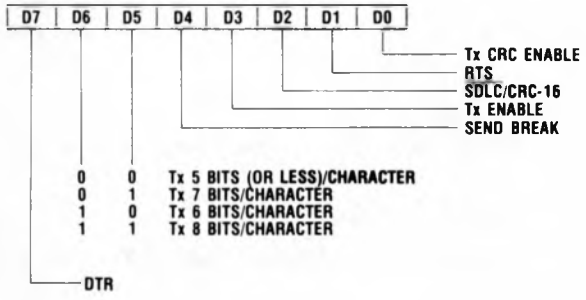
WRITE REGISTER 3



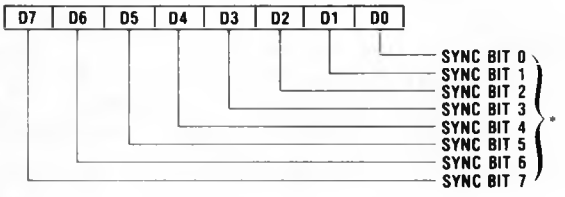
WRITE REGISTER 4



WRITE REGISTER 5

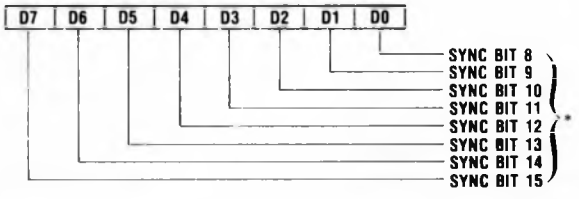


WRITE REGISTER 6



*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7

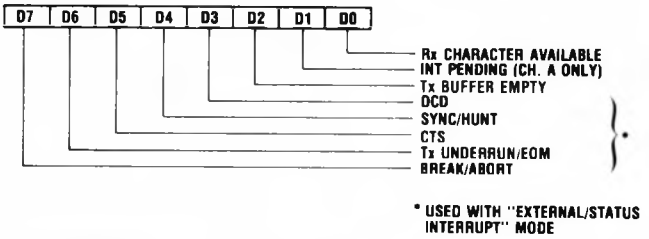


*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

Statusregister

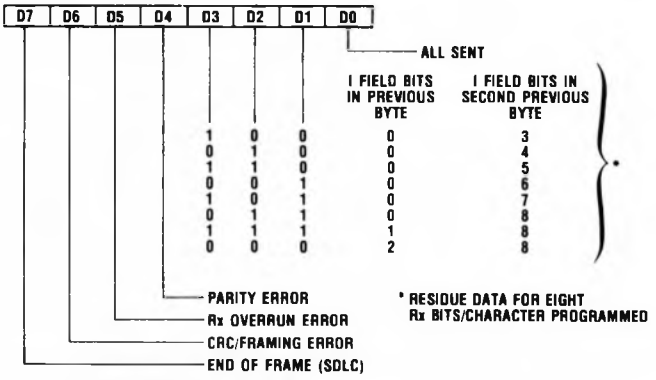
Der Z80-SIO enthält 3 Statusregister, die von der CPU gelesen werden können und Informationen über den Status jedes Kanals liefern. Die Statusinformation enthält Fehlermeldungen, den Interrupt-Vektor und Datentransfersignale. Um den Inhalt eines ausgewählten Statusregisters lesen zu können, muß dem SIO ein Steuerbyte eingeschrieben werden, welches die selbe Zeigerinformation (D0-D2) enthält wie bei Zugriffen zu Steuerregistern. Mit einem Lesezugriff zum SIO kann anschließend der Inhalt des adressierten Statusregisters in die CPU transferiert werden. Beim Entwurf der zugehörigen Programmiersoftware zur Initialisierung des SIO bietet sich die Verwendung der BLOCK I/O-Befehle an.

READ REGISTER 0



* USED WITH "EXTERNAL/STATUS INTERRUPT" MODE

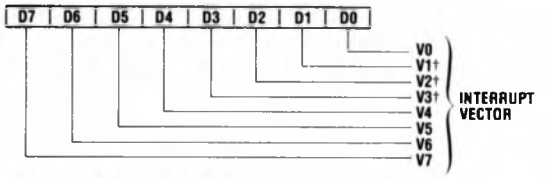
READ REGISTER 1 †



* RESIDUE DATA FOR EIGHT Rx BITS/CHARACTER PROGRAMMED

† USED WITH SPECIAL RECEIVE CONDITION MODE

READ REGISTER 2



†VARIABLE IF "STATUS AFFECTS VECTOR" IS PROGRAMMED

Registerbeschreibung

Jeder Kanal enthält die nachfolgenden Steuerregister, die als Kommando adressiert werden.

Steuerregister 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code 1	CRC Reset Code 0	CMD 2	CMD 1	CMD 0	PNT 2	PNT 1	PNT 0

$PNT_0 - PNT_2$ (D_0-D_2)

Diese Bits sind Zeigerbits, die dem SIO die Adresse jenes Registers liefern, in welches das folgende Byte geschrieben wird. Das erste in einen Kanal eingeschriebene Byte nach einem Reset wird grundsätzlich ins Steuerregister 0 transferiert. Jedes Byte, welches einem Lese- oder Schreibbefehl zu irgendeinem Register (nicht zur Register 0) folgt, wird in Register 0 geschrieben.

CMD_0 to CMD_2 (D_3-D_5)

Command	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no affect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Reset Receive Interrupt on First Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A only)

Kommando 0 — Es wird üblicherweise zum Setzen des Zeigers für ein nachfolgendes Byte verwendet (Adressierung eines Registers).

Kommando 1 (Send Abort) — wird nur in SDLC-Format verwendet und erzeugt eine Sequenz von 8 bis 13 Einsen.

Kommando 2 (Reset External/Status Interrupts) — nach einem externen oder Status-Interrupt (z.B. Anzeige der Änderung auf einer Modem-Leitung oder BREAK) werden die Statusbits vom Statusregister 0 gespeichert. Dieses Kommando aktiviert sie erneut und erlaubt einen neuerlichen Interrupt. Die Zwischenspeicherung ermöglicht die Erkennung von kurzen Impulsen an den Eingängen und gibt der CPU Zeit für einen Lesevorgang.

Kommando 3 (Channel Reset) — Dieses Kommando wirkt wie ein externer Reset, aber nur auf einem Kanal. Der Reset am Kanal A setzt außerdem die Interrupt-Prioritäts-Logik zurück. Nach diesem Kommando müssen alle Steuerregister neu beschrieben werden. 4 zusätzliche Systemtakte sollten anschließend abgewartet werden, bevor neuerliche Kommandos in den Kanal eingeschrieben werden.

Kommando 4 (Reset Receive Interrupt on First Receive Character) — Bei der Programmierung des Formates „Interrupt nur beim 1. Empfangszeichen" ist es nicht nötig, das Kommando nach jeder vollständigen Meldung neu zu aktivieren.

Kommando 5 (Reset Transmitter Interrupt Pending) — Beim Format „Interrupt bei jedem Zeichen" liefert der Sender einen Interrupt, wenn seine Senderegister leer sind. In Fällen, in denen keine weiteren Zeichen gesendet werden, verhindert dieses Kommando weitere Sende-Interrupts (Interrupts erfolgen erst wieder, nachdem ein neuerliches Zeichen in den Sendeteil geladen wurde).

Kommando 6 (Error Reset, Latches) — Paritäts- und Überlauffehler bleiben im Statusregister 1 so lange gespeichert, bis sie mit diesem Befehl rückgesetzt werden. Dadurch braucht die Fehlerkontrolle des Blockdatentransfers erst am Ende des Blocks zu erfolgen.

Kommando 7 (Return from Interrupt) — Dieses Kommando (nur für Kanal A) wird vom SIO wie ein RETI-Befehl auf dem Datenbus interpretiert, d.h. das Interrupt Under Service-Latch der internen Baugruppe (Sender, Empfänger) wird rückgesetzt. Damit wird Baugruppen niederer Priorität Interrupt erlaubt. Die interne Daisy-Chain kann auch in Systemen, die keine externe Daisy-Chain enthalten, verwendet werden.

CRC RESET CODE 0 (D_6) und CRC RESET CODE 1 (D_7)

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (keine Wirkung)
0	1	Empfangsseitigen CRC-Prüfer rücksetzen
1	0	Senderseitigen CRC-Generator rücksetzen
1	1	CRC/SYNCS Sende-Flip Flop rücksetzen

Steuerregister 1

Steuerregister 1 enthält die Steuerbits für die verschiedenen Arten des Interrupts und WAIT/READY.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Wait/ Ready Enable	Ready/Wait	FN/FN	W/Ready On R/T	Empfang Interrupt Mode 1	Empfang Interrupt Mode 0	Status Affects Vector	Trans Interrupt Enable	Ext Interrupts Enable

EXT INT ENABLE (D_0)

Das Ext Int Enable-Bit erlaubt Interrupts als Folge von Zustandsänderungen auf den DCD, CTS oder SYNC-Leitungen, weiters als Folge eines BREAKS oder beim Senden eines CRC oder Sync-Zeichens.

TRANS INT ENABLE (D_1)

Dieses Bit erlaubt Interrupts jedes Mal beim Leerzustand des Sendepuffers.

STATUS AFFECTS VECTOR (D_2)

Bei Auswahl dieses Formats wird der entstehende Interrupt-Vektor variabel:

	V_3	V_2	V_1	
	0	0	0	Ch B Senderpuffer leer
	0	0	1	Ch B External/Status Änderung
Ch B	0	1	0	Ch B Empfangenes Zeichen steht an
	1	0	0	Ch A Senderpuffer leer
	1	0	1	Ch A External/Status Änderung
Ch A	1	1	0	Ch A Empfangenes Zeichen steht an
	1	1	1	Ch A Besonderer Zustand des Empfängers

Ist Bit D_2 gleich null, wird der Grundvektor ausgegeben.

REC INT MODE 0 (D_3), REC INT MODE 1 (D_4)

Die beiden Bits gemeinsam spezifizieren die 4 variablen Meldebedingungen:

MODE	D_4 REC INT MODE 1	D_3 REC INT MODE 0	
0	0	0	Receiver interrupts disabled
1	0	0	Empfänger-Interrupt bei Fehler zum ersten Zeichen
2	1	0	Interrupt on all Receive Characters-Parity/affects Vector
3	1	1	Interrupt on all Receive Characters-Parity error does not affect Vector.

W/READY on R/T (D_5)

Wenn die W/R-Leitung aktiviert ist, wählt dieses Bit Meldung bei: Empfänger ist leer (Bit = 1) oder Senderpuffer ist voll (Bit = 0).

READY FN/WAIT FN (D_6)
Soll die W/R-Leitung mit der CPU in Wait arbeiten, sollte dieses Bit null sein. In Zusammenarbeit mit einem DMA als READY muß es 1 sein. Die Ready-Funktion kann jederzeit auftreten unabhängig davon, ob der SIO adressiert ist. Die Wait-Funktion wird nur dann aktiv, wenn die CPU versucht, Daten aus dem SIO zu lesen, die noch nicht vollständig empfangen sind bzw. versucht, Daten zu schreiben, während der Senderpuffer noch immer voll ist. In der Wait-Funktion ist der Ausgang als Open Drain geschaltet und folgt der fallenden Flanke des Systemtaktes. Bei der Ready-Funktion ist der Ausgang aktiv und folgt der steigenden Flanke des Systemtaktes.

WAIT/READY ENABLE (D_7)
Die W/R-Leitung bleibt 1 (ready) oder tristate (wait) bis dieses Bit auf 1 programmiert wird.

Steuerregister 2
Register 2 ist das Interrupt-Vektor-Register und existiert nur in Kanal B. Beim Lesen werden $V_4 - V_7$ und V_0 wie beim Schreiben geliefert. V_1 bis V_3 folgen „Status Affects Vektor" (falls gewählt).

Steuerregister 3
Register 3 enthält Steuerbits für einen Teil der Empfangslogik.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
RCVR Bits/ Char 0	RCVR Bits/ Char 1	Auto Enables	Enter Hunt Mode	RECVR CRC Enabl	Address Search Mode	Sync Char Load Inhibit	Receiver Enabl

RECEIVER ENABLE (D_0)
„Eins" erlaubt grundsätzlich Empfangsfunktionen.

SYNC CHAR LOAD INHIBIT (D_1)
Bei Auswahl dieser Option werden die einer Meldung vorangehenden Sync-Zeichen nicht in die Empfangspuffer-Register geladen. Die CRC-Berechnung wird während der Entfernung der SYNC-Zeichen nicht gestoppt.

ADDRESS SEARCH MODE (D_2)
Im SDLC-Format verhindert das Setzen dieses Bits Interrupts, sofern die Adressen der Meldungen nicht mit der programmierten Adresse oder der Global-Adresse (IIIIIIII) übereinstimmen.

RECVR CRC ENABLE (D_3)
Ist dieses Bit gesetzt, beginnt die Berechnung des CRC beim Start des letzten Zeichens vom Empfangsregister zum Puffer.

ENTER HUNT MODE (D_4)
Geht die Zeichensynchronisierung verloren oder wird bestimmt, daß der Inhalt einer ankommenden Meldung nicht benötigt wird (SDLC), kann mit dem Schreiben einer 1 in dieses Bit der Sync-Zeichen-Suchlauf neu beginnen.

AUTO ENABLES (D_5)
Bei gesetztem Bit fungieren die DCD und CTS-Eingänge als Empfangs- und Senderaktivierleitungen. Bei rückgesetztem Bit wirken die beiden Eingänge nur auf ihre entsprechenden Bits im Statusregister 0.

RCVR BITS/CHAR 1 (D_6), RCVR BITS/CHAR 0 (D_7)
Diese Bits bestimmen gemeinsam, wie viele Empfangene Bits gemeinsam einem Zeichen entsprechen. Diese Bits können während der Zusammenstellung eines Zeichens verändert werden, falls dies geschieht, bevor die Anzahl der programmierten Bits erreicht wurde.

D_6 Receiver Bits/Character 1	D_7 Receiver Bits/Character 0	Bits/Character
0	0	5
0	1	6
1	0	7
1	1	8

Steuerregister 4
Dieses Register enthält Steuerbits zur Steuerung von Sender und Empfänger.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate 1	Clock Rate 0	Sync Modes 1	Sync Modes 0	Stop Bits 1	Stop Bits 0	Parity Even/ Odd	Parity

PARITY (D_0)
Falls D_0 gesetzt, bewirkt es ein zugefügtes Paritätsbit beim Senden und Erwartung eines Paritätsbits beim Empfangen.

PARITY EVEN/ODD (D_1)
Auswahl gerader (Bit = 0) oder ungerader (Bit = 1) Parität.

STOP BITS 0 (D_2), STOP BITS 1 (D_3)
Bestimmung der Anzahl der Stopbits beim Senden, der Empfänger erwartet immer ein Stopbit (Asynchron).
Bei 00 wird Synchronbetrieb markiert

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync Modes
0	1	1 Stop Bit pro Zeichen
1	0	$1^1/_2$ Stop Bits pro Zeichen
1	1	2 Stop Bits pro Zeichen

SYNC MODES 0 (D_4), SYNC MODES (D_5)
Auswahl der verschiedenen SYNC-Arten

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDCL Mode (01111110 sync pattern)
1	1	External Sync Mode

CLOCK RATE 0 (D_6), CLOCK RATE 1 (D_7)
Bestimmung des Multiplikationsfaktors zwischen Takt und Datentransferrate (im Synchronbetrieb ist „x 1" Bedingung), wobei der Faktor für Senden und Empfang gilt.
Grundsätzlich muß der Systemtakt (0) mindestens 4,5-fach der Datenrate sein. Beim Faktor x 1 muß die Bit-Synchronisierung extern erfolgen.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate X 1 = Clock Rate
0	1	Data Rate X16 = Clock Rate
1	0	Data Rate X32 = Clock Rate
1	1	Data Rate X64 = Clock Rate

Steuerregister 5
Die hier enthaltenen Bits steuern meist den Sender.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Transmit Bits/ Char 0	Transmit Bits/ Char 1	Send Break	Transmit Enable	SDLC/ CRC16	RTS	Transmit CRC Enable

TRANSMIT CRC ENABLE (D_0)
Bei Setzen dieses Bits wird das momentan ausgesandte Zeichen mit CRC berechnet. Der CRC wird nicht automatisch bei leerem Senderkanal ausgesandt.

RTS (D_1)
Steuerbit für RTS-Leitung. Bit = 1: RTS geht auf 0 (aktiv), wird das Bit = 0 geht RTS erst dann auf 1 (inaktiv), wenn der Sender leer ist.

SDLC/CRC/16 (D_2)
Auswahl des CRC-Codes. Bit = 1 bewirkt das SDLC-Polynom

$$X^{16} + X^{12} + X^5 + 1,$$

Bit = 0 das CRC-16-Polynom

$$X^{16} + X^{15} + X^2 + 1.$$

TRANSMIT ENABLE (D_3)
Datenausgabe bleibt blockiert und TxD auf „1" (Markting) bis das Bit = 1 wird. Bei Rücksetzen wird das momentane Zeichen voll ausgesandt (jedoch: CRC wird unterbrochen!)

SEND BREAK (D_4)
Gesetzt, bewirkt dieses Bit das Aussenden von „0" (Spacing) auf TxD, unabhängig von der momentanen Ausgabe.

TRANSMIT BITS/CHAR 0 (D_5), TRANSMIT BITS/CHAR 1 (D_6)
Diese Bits kontrollieren die Bitzahl/Zeichen, die vom eingeschriebenem Datenbyte tatsächlich ausgegeben werden.

D_5 Transmit Bits/Character 1	D_6 Transmit Bits/Character 0	Bits/Character
0	0	5 or less
0	1	6
1	0	7
1	1	8

Alle Bits sind rechtsorientiert, D_0 wird zuerst gesendet. Bei „5 or less" können 1 – 5 Bits transferiert werden, wobei die Daten laut Tafel formatiert sein müssen.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	sendet 1 Bit
1	1	1	0	0	0	D	D	sendet 2 Bits
1	1	0	0	0	D	D	D	sendet 3 Bits
1	0	0	0	D	D	D	D	sendet 4 Bits
0	0	0	D	D	D	D	D	sendet 5 Bits

DTR (D_7)
Dieses Bit kontrolliert die DTR-Leitung. Bit = 1 bewirkt DTR = 0 (aktiv)

Steuerregister 6
Dieses Register enthält die ersten 8 Bit einer BiSync-Sequenz, muß in SDLC-Format mit der Prüfadresse programmiert sein und das SYNC-Zeichen bei 8 Bit-Synchronbetrieb enthalten. Bei EXT/SYNC-MODE nicht verwendet.

Steuerregister 7
Enthält das 2. Byte in BiSync-Transfer oder das 8 Bit-SYNC-Zeichen. In SDLC-Format muß es mit 01111110 programmiert sein.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SYN15	SYN14	SYN13	SYN12	SYN11	SYN10	SYN9	SYN8

Statusregister 0
Wird mit Registerzeiger 000 ausgewählt.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Sending CRC/ Syncs	CTS	Sync/ Hunt	DCD	Transmit Buffer Empty	Interrupt Pending	Receive Character Available

RECEIVE CHARACTER AVAILABLE (D_0)
Gesetzt, falls mindestens ein Zeichen im Puffer vorhanden.

INTERRUPT PENDING (D_1)
Bei Anmeldung irgendeines INTERRUPTS im SIO wird dieses Bit gesetzt, allerdings nur im Kanal A vorhanden, in B immer 0.

TRANSMITTER BUFFER EMPTY (D_2)
Gesetzt, wenn Sende-Puffer leer, außer bei Senden eines CTC.

DCD (D_3)
Zeigt den Zustand der DCD-Leitung zum Zeitpunkt der letzten Änderung eines der fünf Status-Bits (DCD, CTS, SYNC/ HUNT, BREAK/ABORT, SENDING CRC/SYNCS). Zum Lesen des momentanen Status muß dieses Bit unmittelbar nach dem Kommando 2 gelesen werden.

SYNC/HUNT (D_4)
Im Asynchronbetrieb zeigt dieses Bit den Zustand der SYNC-Leitung. Bei Synchronbetrieb wird dieses Bit nach erfolgter Zeichensynchronisierung rückgesetzt und kann nur durch Schreiben des „Enter Hunt Mode"-Bits wieder gesetzt werden.

CTS (D_5)
Zeigt den Zustand der CTS-Leitung (invertiert)

BREAK/ABORT (D_6)
Im Asynchronbetrieb wird dieses Bit durch Erkennen eines „BREAK" gesetzt. Nach Neuaktivierung der Eingänge mit Kommando 2 wird dieses Bit beim Ende des „BREAK" rückgesetzt. Falls „EXTERNAL/STATUS"-Interrupts erlaubt sind, erzeugt dieses Bit einen Interrupt.
Im SDLC-Format wird dieses Bit beim Erkennen einer Abbruch-Sequenz gesetzt (sieben oder mehr Einsen).

SENDING CRC/SYNCS (D_7)
Im Synchronbetrieb wird der CRC beim ersten Auftreten eines leeren Senderegisters innerhalb einer Meldung gesendet. Interrupts werden nur mit einem gesetzten Bit erlaubt. Ist dieses Bit gesetzt und das „TRANSMITTER BUFFER EMPTY"-Bit rückgesetzt, wird der CRC ausgegeben. Sind beide Bits gesetzt, werden automatisch SYNC-Zeichen ausgegeben.

Statusregister 1
Dieses Register wird mit dem auf 001 gesetzten Zeiger ausgesprochen. Nach dem Lesen steht der Zeiger automatisch wieder auf 000.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End Of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

ALL SENT (D_0)
Im Asynchronbetrieb ist dieses Bit nach Aussenden aller im Sendekanal enthaltenen Zeichen gesetzt.

RESIDUE CODE 0 (D_1) – RESIDUE CODE 2 (D_3)
Diese Bits markieren die Länge des I-Feldes im SDLC-Format, in Fällen, in denen das I-Feld kein ganzzahliges Vielfaches der Zeichenlänge ist. Nur bei Transfers mit gesetztem „END OF FRAME" (SDLC)-Bit sinnvoll. Bei einem auf 8 Bit/Zeichen gesetzten Empfänger zeigen die Bits folgendes:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field Bits In Previous Byte	I-Field Bits In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-Felder sind rechts justiert.
Bei anderer Zeichenlänge können ähnliche Code-Tafeln konstruiert werden. Falls kein Rest bleibt, gilt immer:

Residue Code 2	Residue Code 1	Residue Code 0
1	0	1

PARITY ERROR (D_4)
Bei aktivierter Paritätskontrolle wird dieses Bit bei falscher Parität gesetzt. Es bleibt bis zu einem Kommando 6 gespeichert.

RECEIVER OVERRUN ERROR (D_5)
Zeigt Empfänger-Überlauf an (mehr als 4 Zeichen), Rücksetzung nur durch Kommando 6. Ist „Status Affects Vector" aktic, liefert das Überlauf-Zeichen einen Interrupt mit dem „Special Receive Condition"-Vektor.

CRC/FRAMING ERROR (D_6)
Bei Auftreten eines Rahmenfehlers (im Asynchronbetrieb) ist diese Bit nur während des jeweiligen falschen Zeichens gesetzt. Bei der Erkennung eines Rahmenfehlers wird automatisch eine halbe Bitzeit addiert, um eine Interpretation als Startbit zu vermeiden.
Im Synchronbetrieb zeigt das Bit den CRC-Vergleich an.

END OF FRAME (D_7)
Zeigt in SDLC-Format eine gültige Ende-FLAG nebst gültigen CRC und RESIDUE-Codes an.

Statusregister 2
Enthält den Interrupt-Vektor wie das Steuerregister 2, falls das „Status Affects Vector"-Bit nicht gesetzt ist.
Bei gesetztem Bit liefert es den Interrupt-Vektor, der momentan verarbeitet wird. Falls kein Interrupt anliegt, sind $V_3 = 0$ und $V_1 = 1$, der Rest wie programmiert.
Lesen nur aus Kanal B möglich.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0
				Variable if „Status Affects Vector" is enabled			

Programmierbeispiele:
Eine typische Kaltstartroutine (nach einem internen oder externen RESET) würde folgendermaßen aussehen:

B/A	C/D	RD	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Erläuterung
1	1	1	0	0	0	0	0	0	1	0	Pointer set to Register 2B
1	1	1	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0	Interrupt Vector loaded
1	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4B
1	1	1	0	1	X	X	0	1	1	1	Even parity, 1 stop bit, X16 clock, asynchronous mode selected
1	1	1	0	0	0	0	0	1	0	1	Pointer set to Write Register 5B
1	1	1	0	0	1	0	1	0	1	0	7 bits/transmit character, transmitter enabled
1	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3B
1	1	1	0	1	1	0	0	0	0	1	7 bits/receive character, DCD and CTS enable Receiver and Transmitter, Receiver enabled
1	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1B
1	1	1	0	0	0	1	0	1	1	1	Interrupt on every character, status affects Vector external/status interrupts enabled

Nun wird Kanal B zum asynchronen Senden und Empfang vorbereitet. Als nächstes wird eine entsprechende Vorbereitung für Kanal A durchgeführt.

0	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4A
0	1	1	0	0	1	0	0	0	0	0	SDLC mode and X1 clock selected, no parity

Programmierbeispiel:

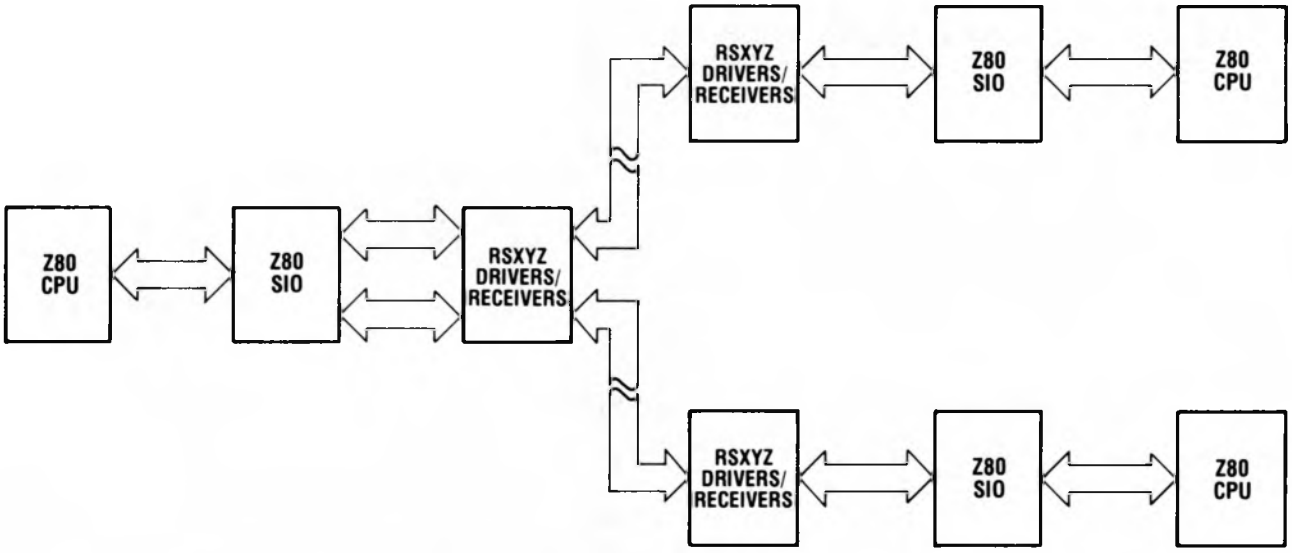
B/A	C/D	RD	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Erläuterung
0	1	1	0	1	0	0	0	1	1	0	Pointer set to Write Register 6A, Reset Receive CRC Checker
0	1	1	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	AD_1	AD_0	SDLC message address entered
0	1	1	1	0	0	0	0	1	1	1	Pointer set to Write Register 7A, Reset Transmit CRC generator
0	1	1	0	1	1	1	1	1	1	0	SDLC Flag entered
0	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1A
0	1	1	0	0	0	1	0	1	1	1	Interrupt every character, status affects vector, external/status interrupts enabled
0	1	1	0	0	0	1	0	1	0	1	Pointer set to Write Register 5A, Reset External/Status Interrupts
0	1	1	1	1	1	0	1	0	0	0	SDLC CRC Code selected, 8 bits/transmit character, CRC and transmitter enabled
0	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3A
0	1	1	1	1	1	0	1	1	0	1	8 bits/receive character, DCD and CTS enable receiver and transmitter, receiver is enabled, SIO searches for programmed address

Nun wird Kanal A für SDCL-Übertragung vorbereitet.

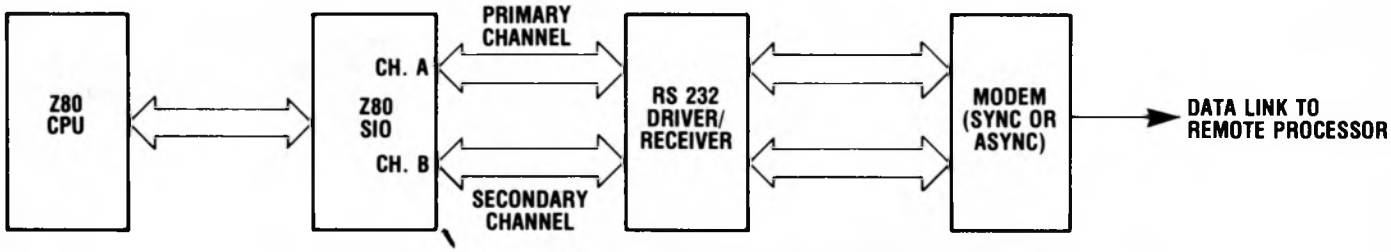
0	0	1	D	D	D	D	D	D	D	D	Address byte to be sent Ch. A
0	1	1	1	1	0	0	0	0	0	0	Reset CRC/SYNCS SENT/SENDING, pointer to register 0, so CRC can be automatically sent at end of message

Anwendungshinweise:

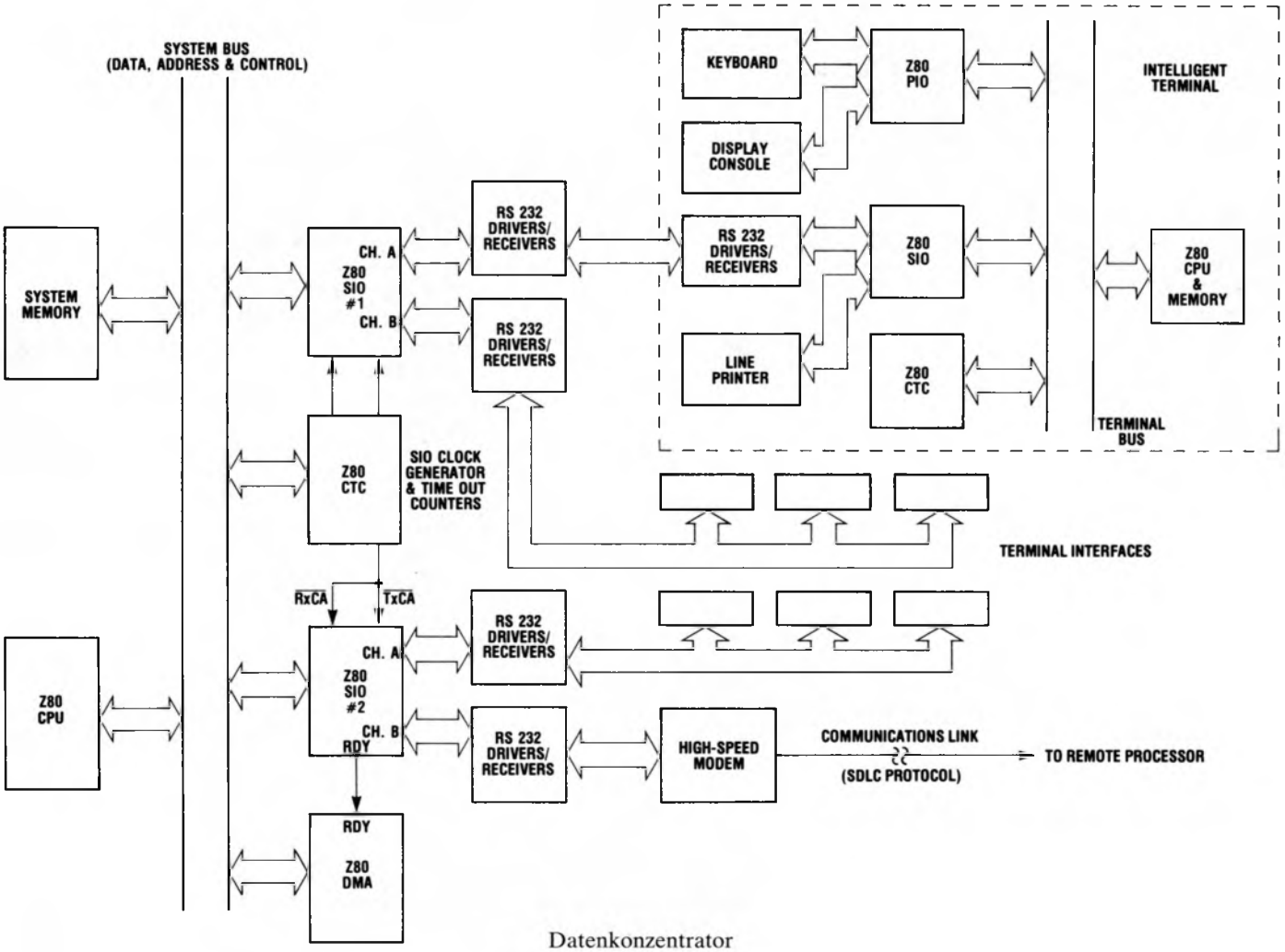
Die folgenden Abbildungen illustrieren typische Anwendungen der SIO-Bausteine bzw. ihrer verschiedenen Ausführungen. Die Zeichnungen bedürfen kaum einer Erläuterung und stellen nur einen kleinen Ausschnitt aus dem Anwendungsbereich dieses extrem universellen Bausteins dar. Fertige Standardhardware dieser Art ist im Rahmen der Z80-Baugruppenserie ECB unter der Bezeichnung ECB/F und ECB/C8 verfügbar.



Synchrone/Asynchrone Rechner/Rechner-Serienkommunikation für einen Haupt- und zwei Sub-Computer

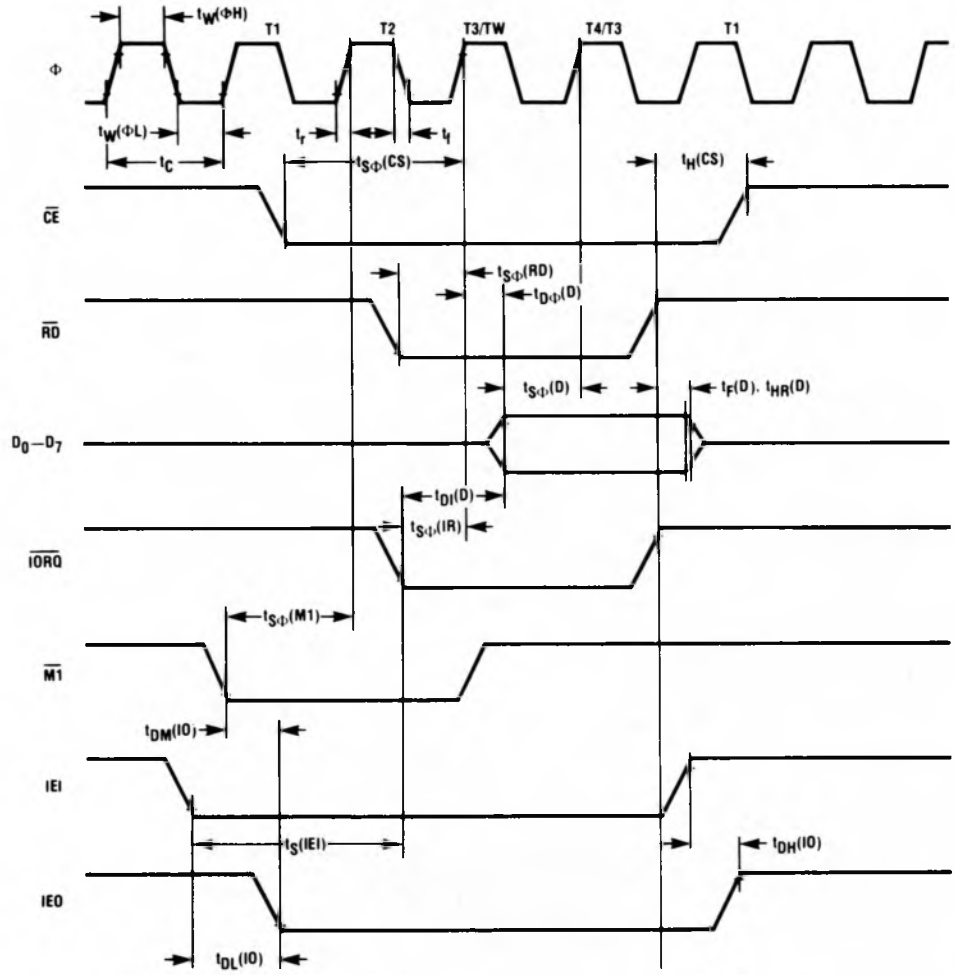


Sychrone/Asynchrone Rechner/Rechner-Kopplung über Telefonnetz



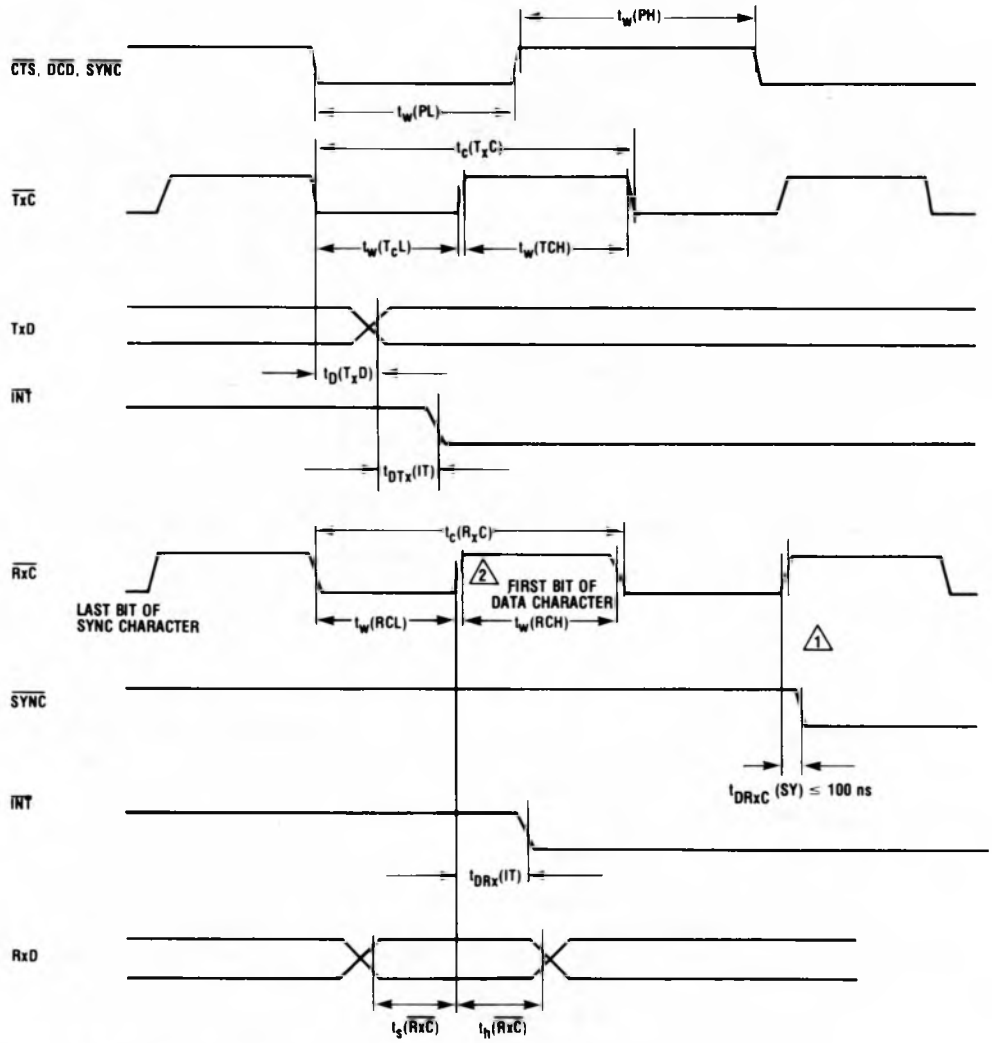
Datenkonzentrator

$T_A = 0°C$, $V_{CC} = +5V$, $\pm 5\%$



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Unit
	$t_C(\phi)$	Clock Period	400	4000	ns
	$t_w(\phi H)$	Clock Pulse Width, clock HIGH	170	2000	ns
ϕ	$t_w(\phi L)$	Clock Pulse Width, clock LOW	170	2000	ns
	t_r, t_f	Clock Rise and Fall Times	0	30	ns
	t_h	Any Unspecified Hold Time for setup times specified below	0		ns
$\overline{CE}$, B/$\overline{A}$ C/$\overline{D}$, $\overline{IORQ}$	$t_{s\phi}(CS)$	Control Signal Setup Time to rising edge of ϕ during Read or Write Cycle	160		ns
D_0-D_7	$t_{D\phi}(D)$	Data Output Delay from rising edge of ϕ during Read Cycle		240	ns
	$t_{s\phi}(D)$	Data Setup Time to rising edge of ϕ during Write or M1 Cycle	50		ns
	$t_{DI}(D)$	Data Output Delay from falling edge of $\overline{IORQ}$ during INTA Cycle		340	ns
	$t_F(D)$	Delay to Floating Bus (output buffer disable time)		230	ns
IEI	$t_s(IEI)$	IEI Setup Time to falling edge of $\overline{IORQ}$ during INTA Cycle	200		ns
	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI (after "ED" decode)		150	ns
IEO	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		150	ns
	$t_{DM}(IO)$	IEO Delay Time from falling edge of $\overline{M1}$ (interrupt occurring just prior to M1)		300	ns
M1	$t_{s\phi}(M1)$	$\overline{M1}$ Setup Time to rising edge of ϕ during INTA or M1 Cycle	210		ns
$\overline{RD}$	$t_{s\phi}(RD)$	$\overline{RD}$ Setup Time to rising edge of ϕ during Read or M1 Cycle	240		ns

***If WAIT from the SIO is to be used, $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ and $\overline{M1}$ must be valid for as long as the Wait condition is to persist.**



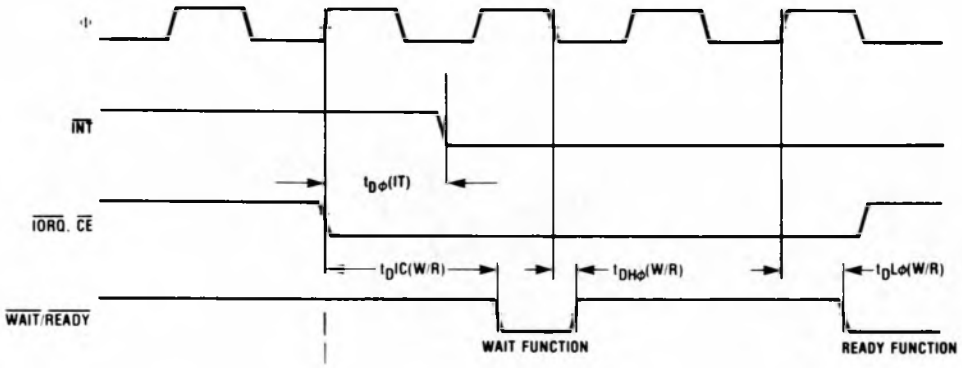
NOTES:

1. The $\overline{SYNC}$ input must be driven Low on the rising edge of $\overline{RxC}$ delayed two complete clock cycles from the last bit of the sync character.
2. Data character assembly begins on the next Receive Clock cycle after the last bit of the sync character is received.

Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Unit
$\overline{INT}$	$t_{DRx}(IT)$	INT Delay Time from rising edge of RxC	10	13	ϕ penods
	$t_{DTx}(IT)$	$\overline{INT}$ Delay Time from transition of Xmit Data Bit	5	9	ϕ penods
$\overline{CTSA}$, $\overline{CTSB}$, $\overline{DCDA}$, $\overline{DCDB}$, $\overline{SYNCA}$, $\overline{SYNCB}$	$t_{w}(PH)$	Minimum HIGH Pulse Width for latching state into register and generating interrupt	200		ns
	$t_{w}(PL)$	Minimum LOW Pulse Width for latching state into register and generating interrupt	200		ns
$\overline{SYNCA}$, $\overline{SYNCB}$	$t_{DL}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{RxC}$. Output Modes	4	7	ϕ penods
	$t_{DRxC}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{RxC}$ External Sync Mode		100	ns
$\overline{TxCA}$, $\overline{TxCB}$	$t_{c}(TxC)$	Transmit Clock Period	400	∞	ns
	$t_{w}(TCH)$	Transmit Clock Pulse Width, clock HIGH	180	∞	ns
	$t_{w}(TCL)$	Transmit Clock Pulse Width, clock LOW	180	∞	ns
TxDA, TxDB†	$t_{D}(TxD)$	TxD Output Delay from falling Edge of $\overline{TxC}$ (x1 Clock Mode)		400	ns
$\overline{RxCA}$, $\overline{RxCB}$	$t_{c}(RxC)$	Receive Clock Period	400	∞	ns
	$t_{w}(RCH)$	Receive Clock Pulse Width, clock HIGH	180	∞	ns
	$t_{w}(RCL)$	Receive Clock Pulse Width, clock LOW	180	∞	ns
RxDA, RxDB†	$t_{s}(RxC)$	Setup Time to rising edge of $\overline{RxC}$, x1 mode	0		ns
	$t_{h}(RxC)$	Hold Time from rising edge of $\overline{RxC}$, x1 mode	140		ns

†In all modes, the system clock (ϕ) rate must be at least 4.5 times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete ϕ cycle.

Dynamische Kenndaten



Signal	Symbol	Parameter	Z80-SIO Min	Max	Unit
$\overline{INT}$	$t_{D\phi(IT)}$	$\overline{INT}$ Delay Time from rising edge of ϕ		200	ns
	$t_{DIC(W/R)}$	$\overline{WAIT/READY}$ Delay Time from $\overline{IORQ}$ or $\overline{CE}$ in Wait Mode		180	ns
	$t_{DH\phi(W/R)}$	$\overline{WAIT/READY}$ Delay Time from falling edge of ϕ, $\overline{WAIT/READY}$ HIGH, Wait Mode		150	ns
$\overline{WAIT/READY}$	$t_{DRx(W/R)}$	$\overline{WAIT/READY}$ Delay Time from rising edge of $\overline{RxC}$ Data Bit, Ready Mode	10	13	ϕ periods
	$t_{DTx(W/R)}$	$\overline{WAIT/READY}$ Delay Time from center of Transmit Data Bit, Ready Mode	5	9	ϕ periods
	$t_{DL\phi(W/R)}$	$\overline{WAIT/READY}$ Delay Time from rising edge of ϕ, $\overline{WAIT/READY}$ LOW, Ready Mode		120	ns

Statische Kenndaten

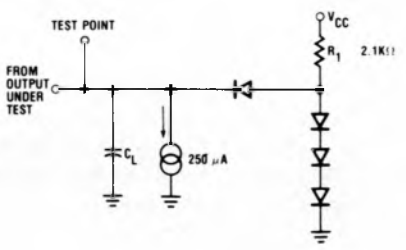
$T_A = 0\,°C$ to $70\,°C$, $V_{CC} = +5V$, $\pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	− 0.3	+ 0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} − 0.6	+ 5.5	V	
V_{IL}	Input Low Voltage	− 0.3	+ 0.8	V	
V_{IH}	Input High Voltage	+ 2.0	+ 5.5	V	
V_{OL}	Output Low Voltage		+ 0.4	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	+ 2.4		V	$I_{OH} = -250\,\mu A$
I_{LI}	Input Leakage Current	− 10	+ 10	µA	$0 \leq V_{IN} \leq V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	− 10	+ 10	µA	$0 \leq V_{IN} \leq V_{CC}$
$I_{LI(SY)}$	$\overline{SYNC}$ Pin Leakage Current	− 40	+ 10	µA	
I_{CC}	Power Supply Current		100	mA	

Kapazitäten

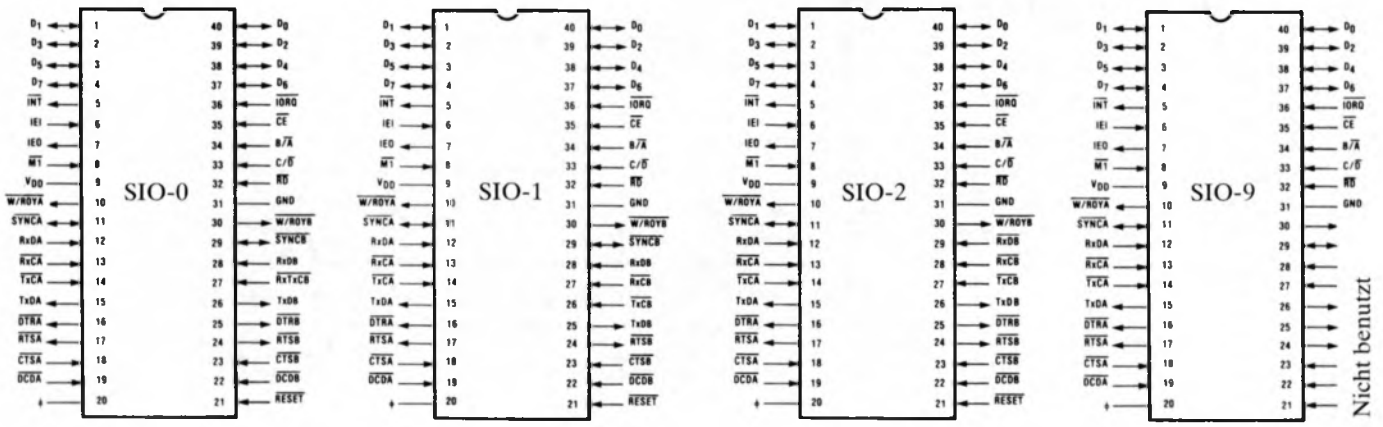
$T_A = 25\,°C$, $f = 1$ MHz

Symbol	Parameter	Min.	Max.	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

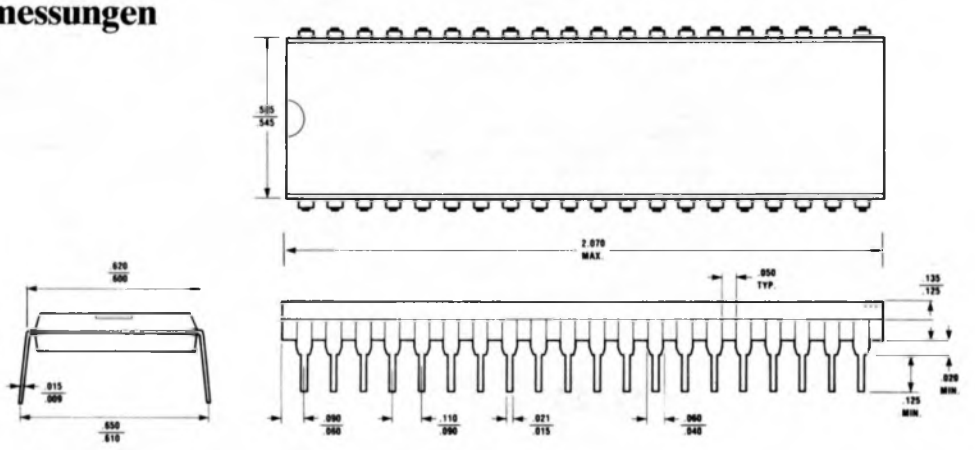


$C_L = 50$ pF. Increase delay by 10 ns for each 50 pF increase in C_L, up to 200 pF maximum.

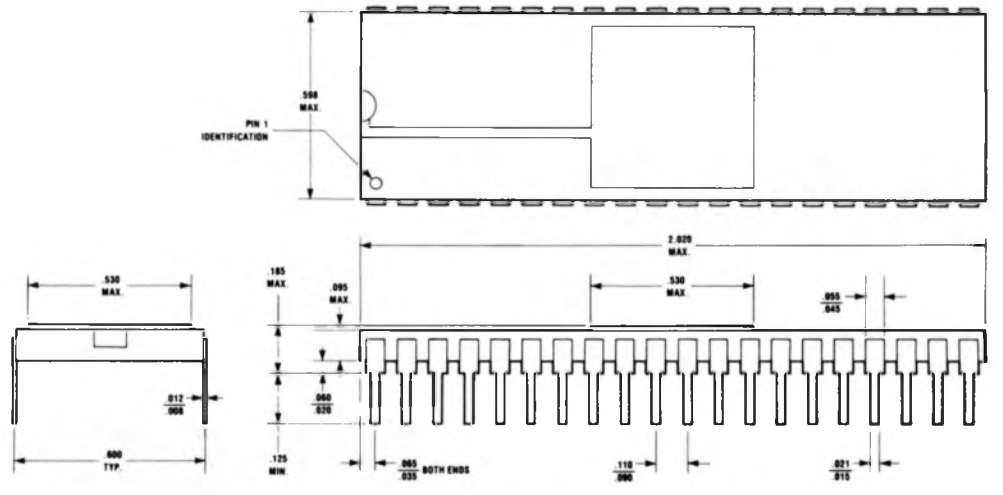
Pin-Belegung



Gehäuseabmessungen



40-Pin Plastik



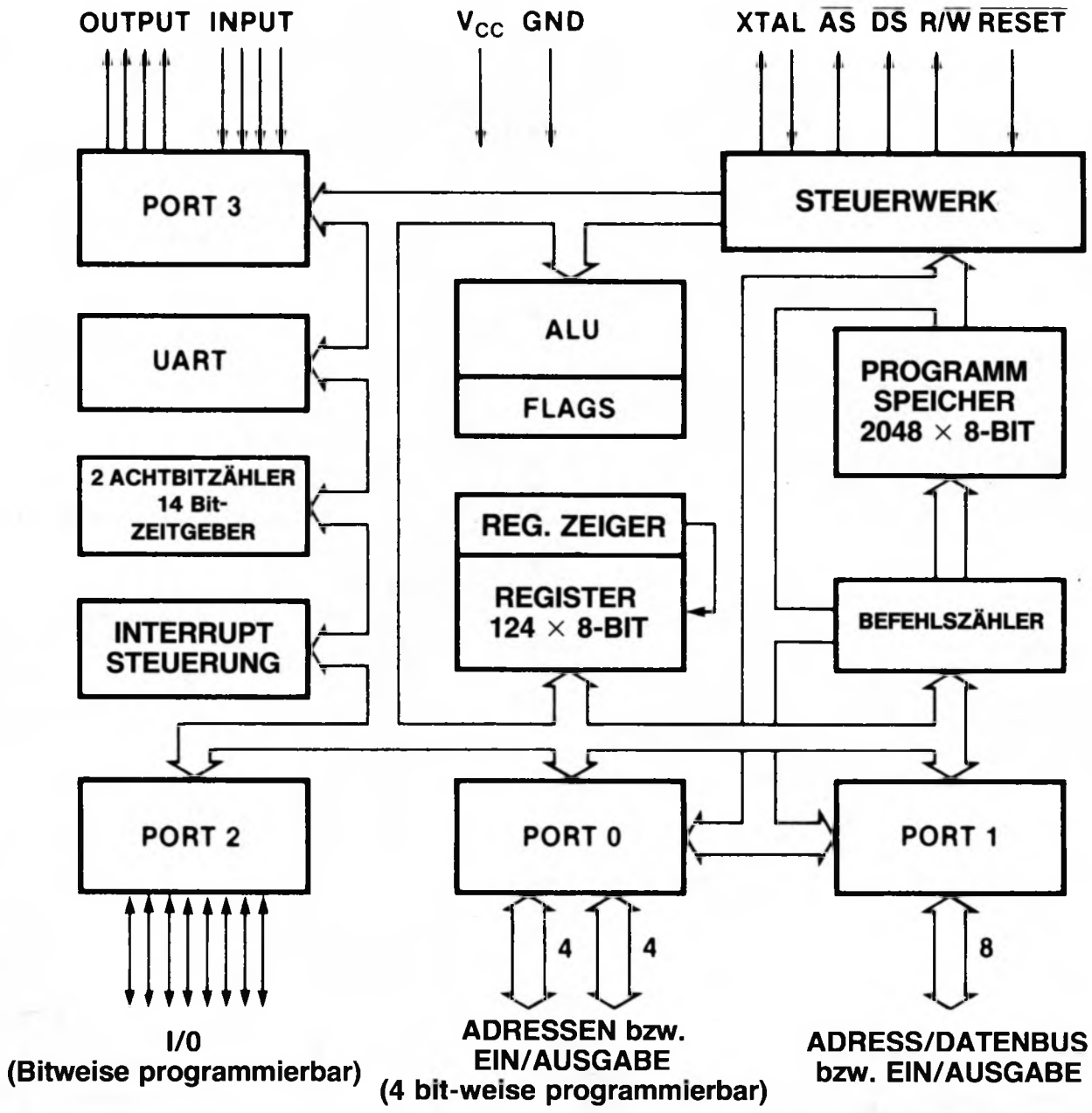
40-Pin Keramik

□ Bestellbezeichnung

Z80 -SIO/PS	Standardversion (0 . . . 70°C) in Plastikgehäuse $U_B = 5\ V \pm 5\%$
Z80 -SIO/CS	Standardversion (0 . . . 70°C im Keramikgehäuse $U_B = 5\ V \pm 5\%$

Die Zusätze nach den oben angegebenen Bestellbezeichnungen -0, -1, -2 und -9 bezeichnen die verschiedenen Kontaktierungsformen, in denen der SIO geliefert werden kann. Bei der 1-Kanal-SIO-Version Z80-SIO/...-9 ist Kanal B nicht benutzt; die bezeichneten Eingänge dürfen nicht verwendet werden.

Z8 Zilog **Einchip-Mikrocomputer**



Vorläufige Beschreibung **Bild 1: Blockschaltbild**

3. ZILOG-Z8-EINCHIP-MIKROCOMPUTER

Beschreibung und Anwendung:

Der 1 Chip-Mikrocomputer Z8 ist ein Halbleiterbaustein, bei dem sämtliche Computerfunktionen in einem 40 pin-DIL-Gehäuse untergebracht sind, und zwar Zentraleinheit, Taktgenerator, Programmspeicher, Schreib/Lesespeicher, Parallel-Ein/Ausgabe, Serien-Ein/Ausgabe und sogar Echtzeitzähler und Zeitgeber.

Es ist also möglich, ein funktionsfähiges Mikrocomputersystem mit diesem einen, hochintegrierten Baustein zu realisieren.

Dadurch eignet sich der Baustein in optimaler Weise für Anwendungen, in denen die Verwendung eines Universalprozessors (wie Z80-CPU, Z80A-CPU oder Z8000-CPU) mit den zugehörigen, universellen Peripheriebausteinen zu aufwendig wäre, wie bei einfachen Waagen, intelligenten Terminals, Spielen, und Anwendungen der Konsumelektronik.

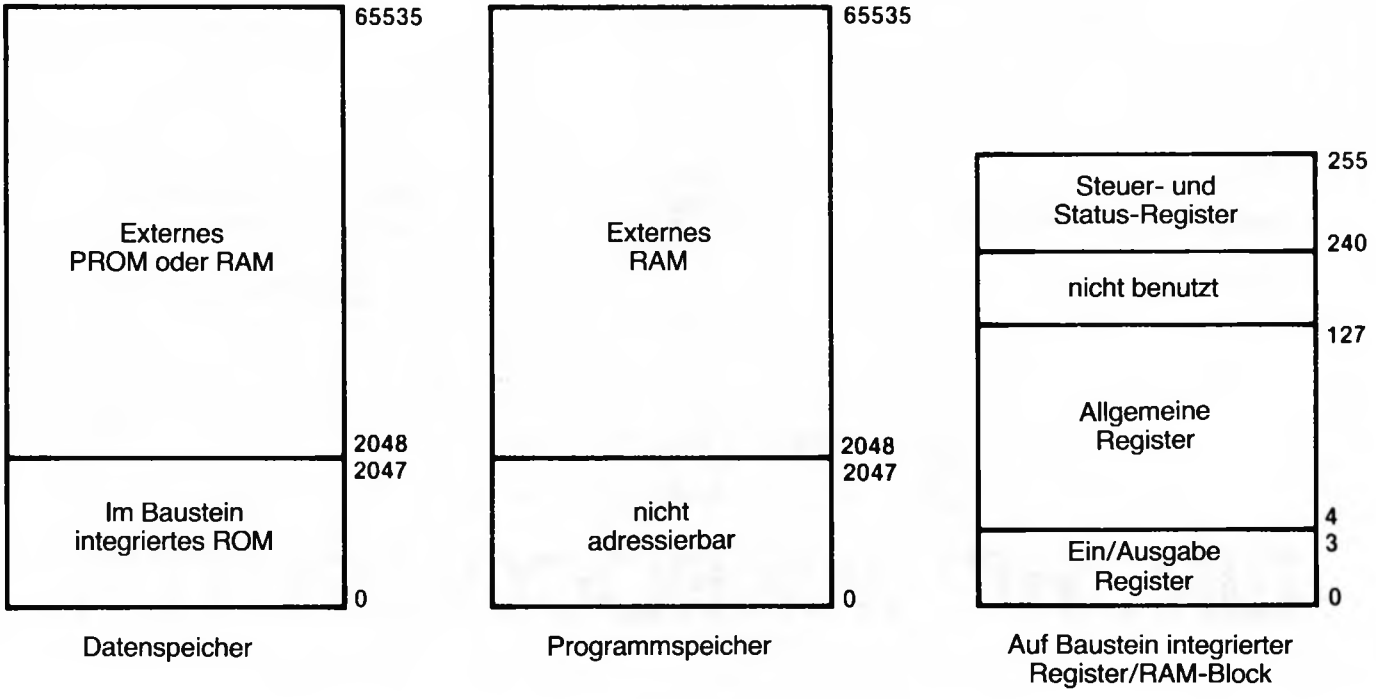
Darüberhinaus ist der Baustein ausgezeichnet als intelligenter und programmierter Controller in größeren Mikrocomputersystemen und als Subprozessor in Multi-Prozessorsystemen einzusetzen.

Durch freie Programmierbarkeit der Bestimmung einzelner Anschlüsse wurden hohe Universalität und hervorragende Anpaßbarkeit an das jeweilige Anwenderproblem erreicht. Gleichzeitig ist dadurch volle Erweiterbarkeit — der Z8 läßt sich auf 64 kByte und um Ein/Ausgabe-Bausteine erweitern. Die Programmentwicklung erfolgt mit Hilfe des Zilog-Entwicklungssystems ZDS mit dementsprechenden Zusätzen.

Die Ausrüstung des Bausteins mit einem maskenprogrammierbarem Programmspeicher prädestiniert ihn als preisgünstige Lösung für Anwendersysteme, die in Großstückzahlen gefertigt werden.

Im Einzelnen hat der Baustein folgende Eigenschaften:

- Vollständiger Mikrocomputer (CPU, Taktgenerator, RAM, ROM, parallel Ein/Ausgabe, UART und zwei Zähler/Zeitgeber) in einem Baustein integriert.
- Höchstmögliche Ausnutzung der 40 Gehäuseanschlüsse für Ein/Ausgabe; hierfür stehen 32 Leitungen (vier 8 bit-Ports) zur Verfügung.
- Zei dieser Ein/Ausgabeleitungen sind als asynchrone, serielle Duplexschnittstelle programmierbar (hardwaremäßig integrierter UART).
- Integrierter Festwertspeicherbereich: 2 kByte (Masken-ROM).
- Integrierter Schreib/Lesespeicherbereich: 144 Byte, davon sind 124 Byte allgemeine Register, die alle als Akkus benützt werden können (Bild 3).
- 16 Bit-Befehlszähler.
- Der zur Verfügung stehende Speicherbereich ist bis auf 62 kByte Datenspeicher (extern) und 64 kByte Programmspeicher (62 kByte extern) erweiterbar. Die Erweiterung erfolgt durch programmgesteuerte Benutzung von Ein/Ausgabe-Leitungen als Adreßleitungen; bei max. Speicherausbau sind also 16 der 32 verfügbaren Ein/Ausgabe-Leitungen für Speicheradressierzwecke belegt.
- Die Ein/Ausgabe-Puffer können als allgemeine Register behandelt werden (d.h. **es sind Registeroperationen in den Ein/Ausgabe-Puffern** möglich).
- Zwei 8 bit Zähler/Zeitgeber und je ein 6bit Vorteiler (programmierbar von $2^4 \ldots 2^8$) zur prozessorzeitsparenden Abwicklung von Echtzeitaufgaben.
- Integrierte Mehrebenen Vektor-Interrupt-Steuerung.
- Eingebauter Taktgenerator.
- Verwendung des internen oder auch eines externen Taktgenerators.
- Max. Taktfrequenz: 4 MHz.
- Befehlsausführungszeit bei $\Phi = 4$ MHz: min. 1,5 μsec, durchschnittlich 2,2 μsec, max. 4,5 μsec.
- Stromversorgung über eine einzige +5 V Spannungsquelle.
- Alle Anschlüsse sind TTL-kompatibel.



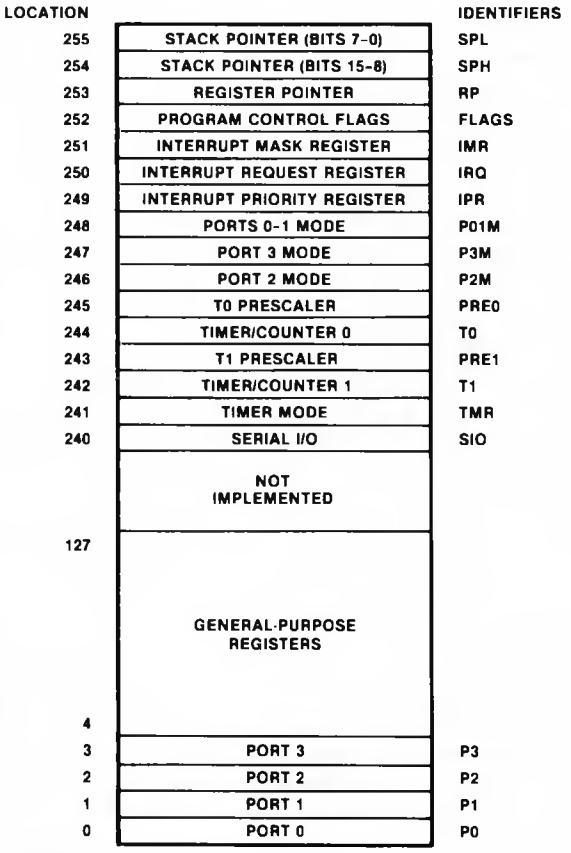


Bild 3. Register-Organisation

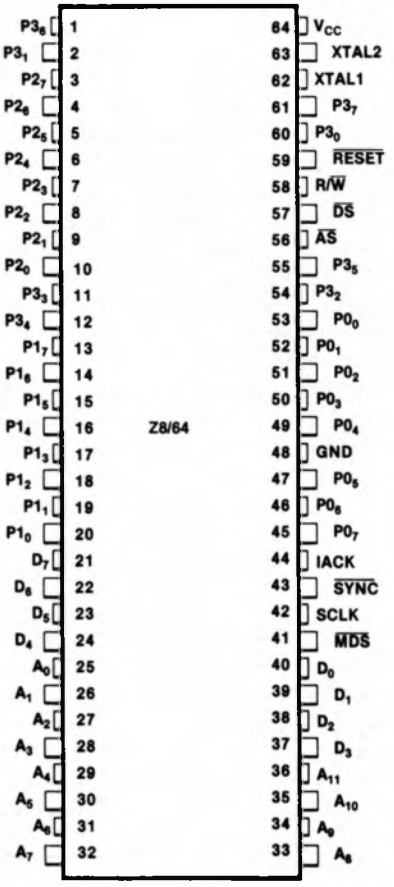


Bild 5

Pin-Belegung

Der Baustein wird standardmäßig in einer 40 Pin-Masken-ROM-Version geliefert (Bild 4). Für Testzwecke und zur Prototypenerstellung steht eine 64 Pin-Version des Bausteins zur Verfügung, bei dem die internen Busse zusätzlich herausgeführt sind (Bild 5 und 6).

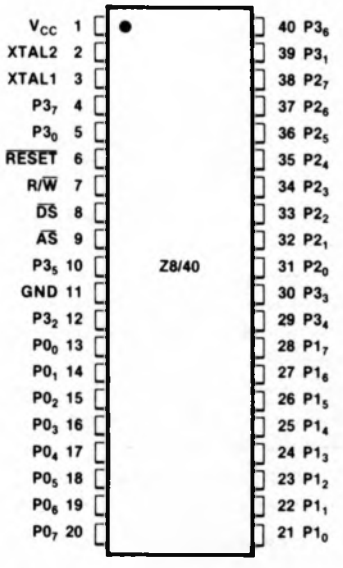


Bild 4

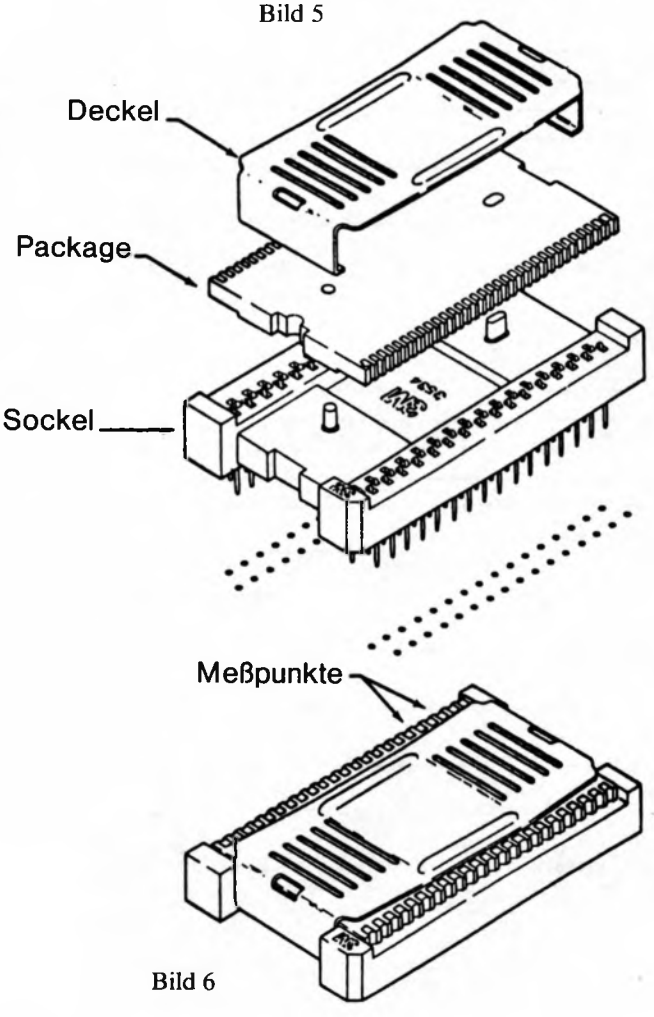


Bild 6

Befehlsverarbeitung

Zur Erhöhung der effektiven Verarbeitungsgeschwindigkeit erfolgt die Befehlsabarbeitung im Z8 nach dem „Pipeline"-Verfahren.

Dabei wird während der Ausführung eines Befehls bereits mit dem Einziehen des Befehls begonnen, der durch den aktuellen Inhalt des Befehlszählers +1 adressiert ist. Bei diesem Verfahren wird bei der Abarbeitung all **der** Befehle Verarbeitungszeit gespart, die keine Sprünge auf einen anderen als den nächstfolgenden Befehl bewirken (Bild 7). Bitte beachten Sie, daß durch diese Architektur sinnvollerweise zwei unterschiedliche Verarbeitungszeiten in der Befehlsliste angegeben werden:

a) die Gesamtabarbeitungszeit, die immer dann benötigt wird, wenn vor diesem (n+1)-ten Befehl **nicht** der n-te Befehl zur Ausführung gekommen war („Execution Cycles").

b) die effektive Abarbeitungszeit, die andernfalls durch das frühzeitige Einziehen des (n+1)-ten Befehls lediglich die Verarbeitungszeit minus die Befehlseinziehzeit beträgt („Pipeline Cycles"). Das Zeitverhalten des Z8-Computers ist durch die folgenden Bilder 8—13 beschrieben.

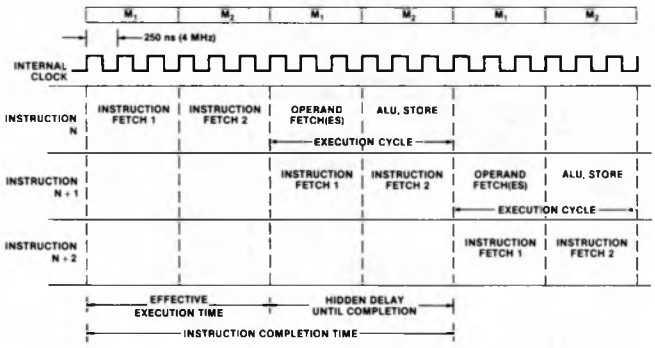


Bild 7: Pipelining

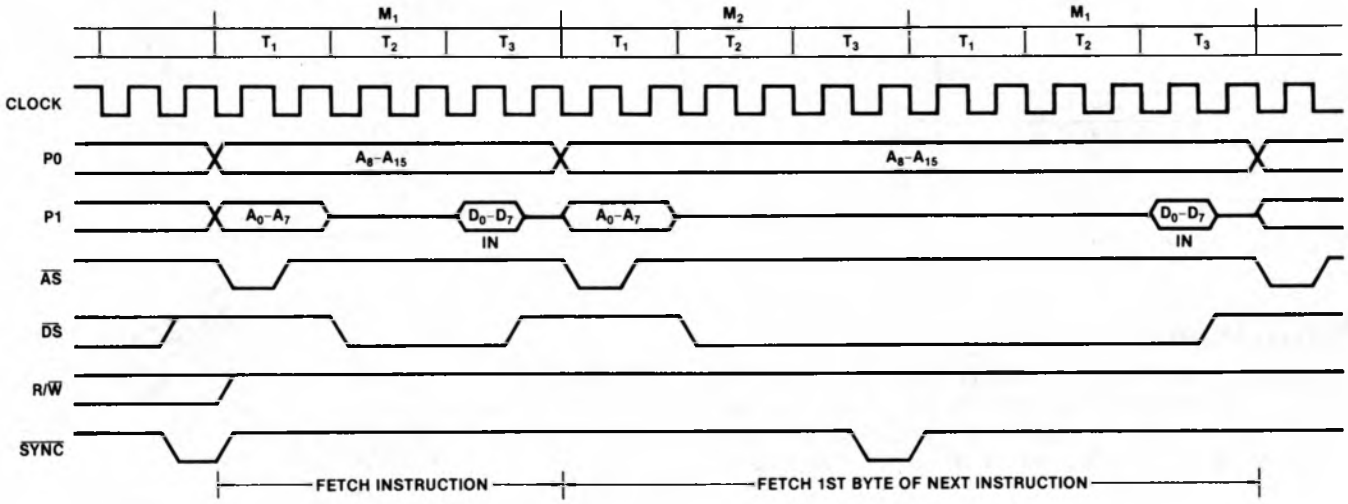


Bild 8: Zeitverhalten bei 1 Byte-Befehlen

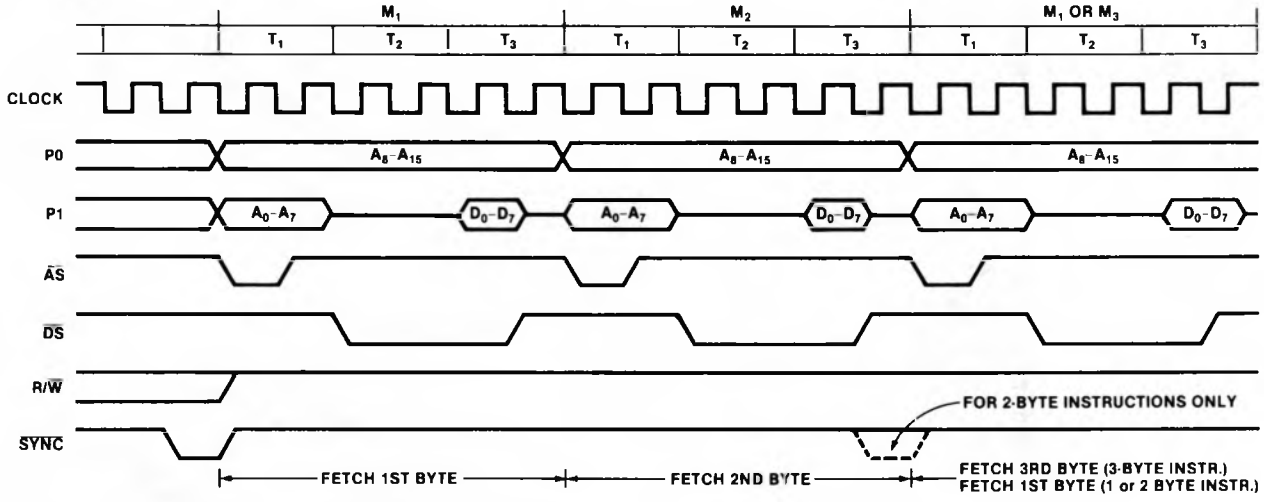


Bild 9: Zeitverhalten bei 2- und 3-Byte-Befehlen

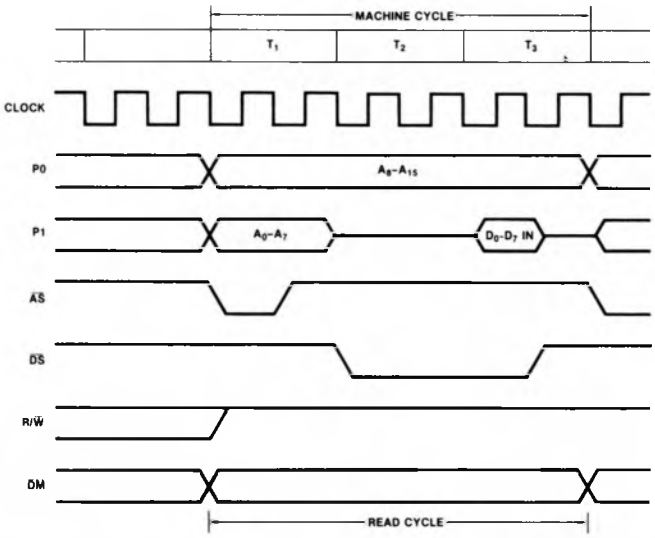


Bild 10: Zeitverhalten bei Einziehen extern abgespeicherter Befehle und bei Eingabe oder Speicherlesezugriffen

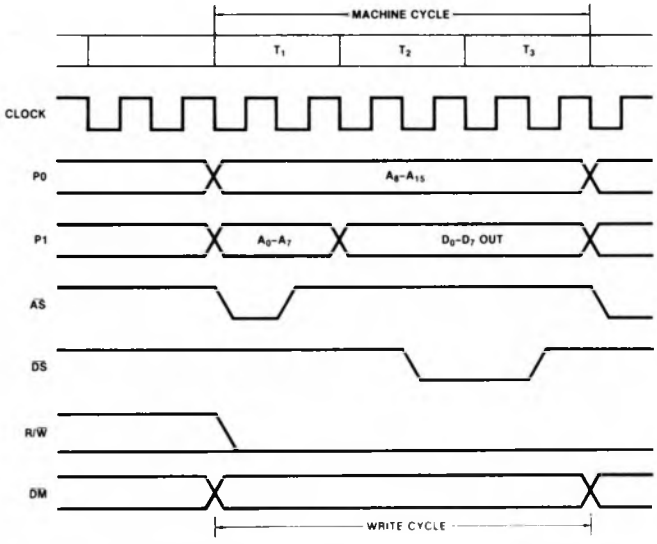


Bild 11: Zeitverhalten bei Ausgabe- oder Speicherschreibzugriffen

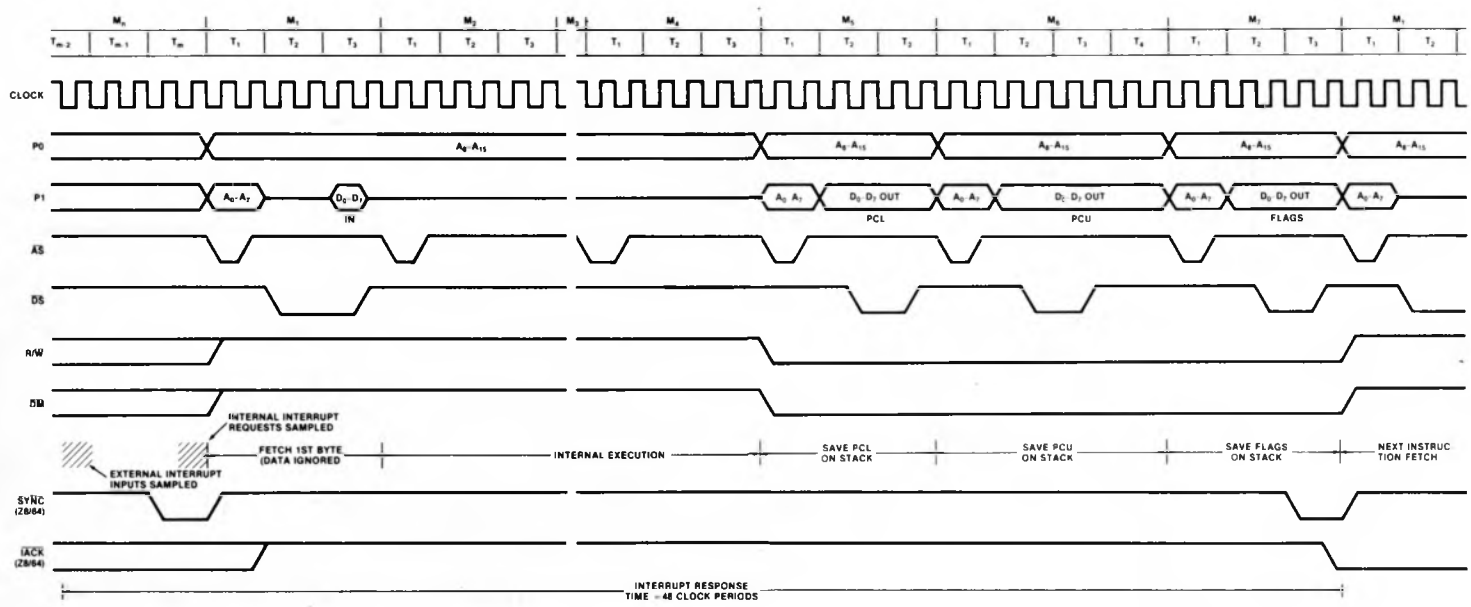


Bild 12: Interrupt-Zeitverhalten

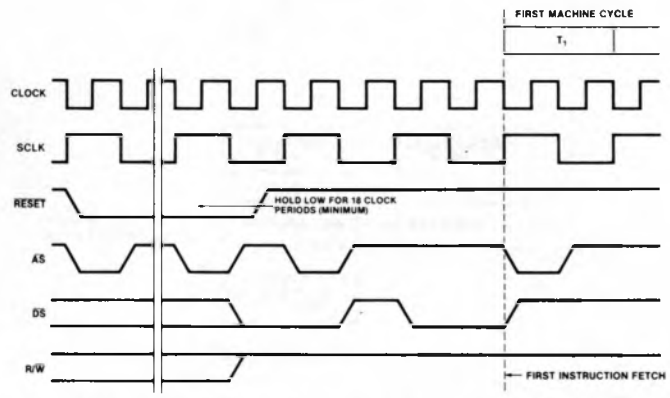


Bild 13: Zeitverhalten bei RESET

Ein/Ausgabe

Die vier auf dem Z8-Baustein integrierten 8 bit-Ein/Ausgabe-Ports sind anwenderprogramm-programmierbare Schnittstellen, von denen zwei (Port 0 und Port 1) teilweise oder ganz per Anwenderprogramm als Adreßleitungen definierbar sind. Port 2 ist ein allgemeiner Ein/Ausgabe-Kanal, dessen Anschlüsse sich sogar per Anwenderprogramm mit Pull-Up-Widerständen beschalten lassen.
Blockschaltbilder von Port 0—2 finden Sie in Bild 14; Port 3 in Bild 15 schematisch dargestellt.
Diese Schnittstellen werden durch die hardwaremäßig mit auf dem Baustein integrierter asynchromer Serienschnittstelle (UART, Bild 16 und 17) und die zwei 8bit-Zähler/Zeitgeber mit ihren zwei Vorteilern (je 6 bit) ergänzt (Bild 18). Einzelheiten sind dem „Z8 Technical Manual" zu entnehmen.

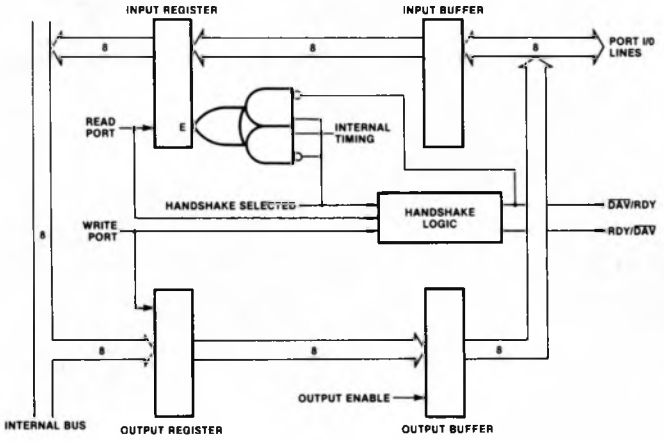


Bild 14: Blockschaltung von Ports 0, 1 und 2

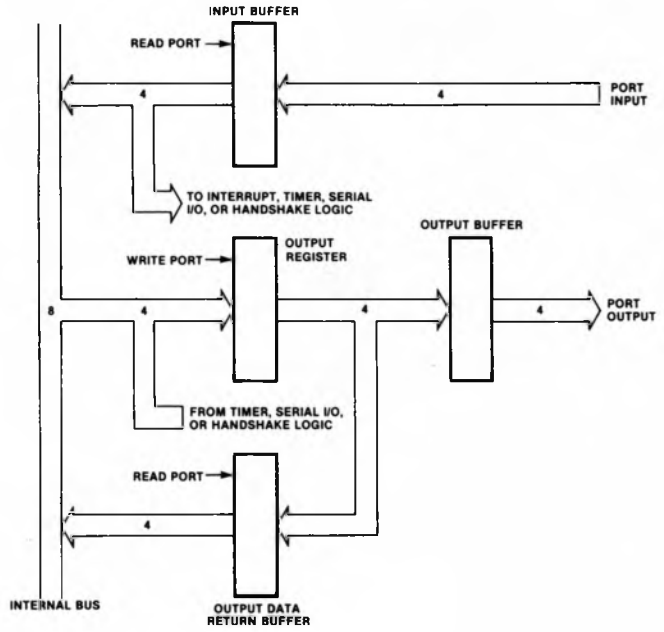


Bild 15: Blockschaltung von Port 3

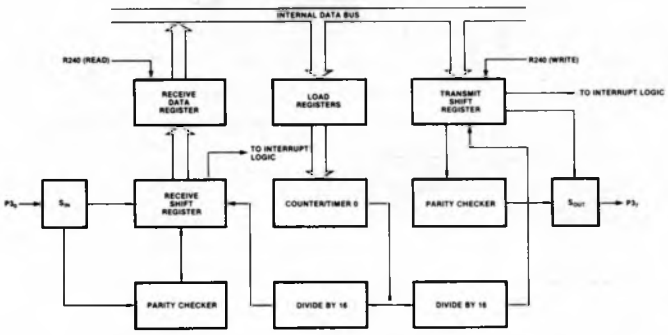


Bild 16: Integrierte Serienschnittstelle (UART)

Übertragungsraten (Bit/sec)*	in To vorprogrammierter Wert
19200	1
9600	2
4800	4
2400	8
1200	16
600	32
300	64
150	128
110	175 (error 0.3%)

* Bei Quarzfrequenz = 7.3728 MHz

Bild 17: Datenübertragungsraten der integrierten Serienschnittstelle.

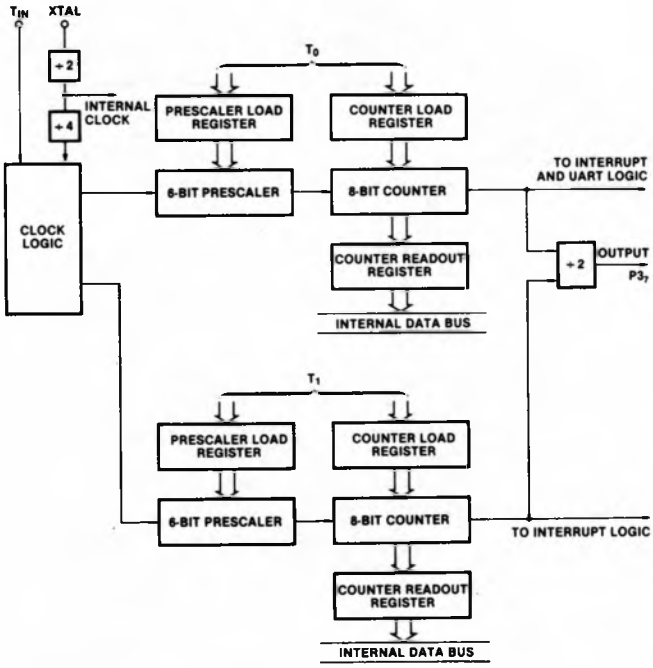


Bild 18: Blockschaltung der zwei integrierten Zähler/Zeitgeberkanäle

Interrupt-Architektur

Z8-Systeme erlauben 6 verschiedene Interrupts, die von 8 Quellen ausgelöst werden können:

Diese 8 möglichen Quellen sind die 4 Anschlüsse P30-P33 von Port 3, die serielle Eingabe, die serielle Ausgabe und die beiden Zähler/Zeitgeber. Diese Interrupts sind über das Interrupt-Masken-Register (IMR) R251 maskierbar und über das Interrupt-Prioritäts-Register (IPR) R249 priorisierbar.

Alle sechs Interrupts lassen sich durch Rücksetzen des Haupt-Interrupt-Freigabebits im Interruptmaskenregister R251 global sperren. Sämtliche Z8-Interrupts sind vektorisiert (Bild 19).

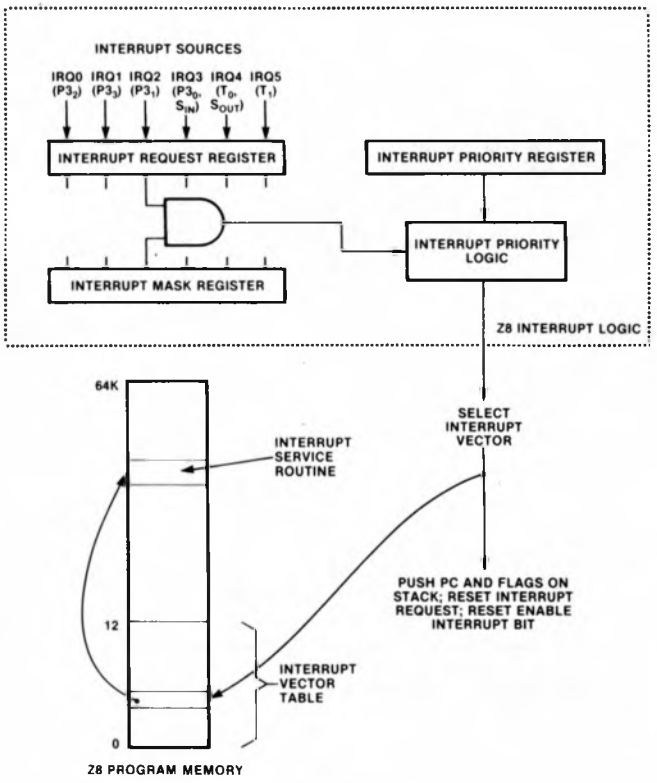


Bild 19: Z8-Interrupt-Architektur

Status-Flags

Die Z8 verfügt über ein Flag-Register (R252) mit den folgenden Status/Bedingungs-Bits:

C Carry	S Sign	D Decimal Adjust	F1 User Flag 1
Z Zero	V Overflow	H Half Carry	F2 User Flag 2

Die beiden Flags F1 und F2 stehen für allgemeine Anwendungen im Anwenderprogramm zur Verfügung. Alle Flags außer dem Half-Carry und dem Decimal-Adjust können über 19 verschiedene Z8-Befehle zur Behandlung von Bedingungen benutzt werden.

Z8-Befehlssatz

Der Befehlssatz des Z8 zeichnet sich durch seine strikte Regularität aus, die besonders einfache und effiziente Programmstellung ermöglicht.

Folgende Adressierweisen sind in diesem Rahmen implementiert:

- Registeradressierung
- Register-indirekte Adressierung
- Indizierte Adressierung
- Direkte Adressierung
- Relative Adressierung
- „Immediate" Adressierung
- „Immediate" Adressierung (Konstantenzuweisung)

Die Registeradressierung kann entweder über eine 8bit-Adresse mit den zulässigen Werten 0—127 und 240—255 (Bild 20) oder über eine 4bit-Zeiger-(„Working-Register"-) Adressierung (Bild 21).

Das Verfahren der Register-indirekten Adressierung zeigen die Bilder 22 und 23, das der indizierten Adressierung Bild 24.

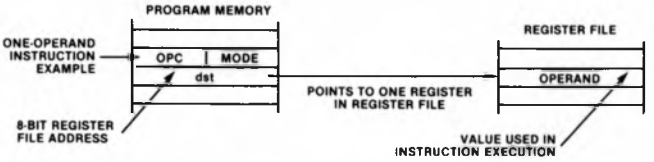


Bild 20: Register-Adressierung

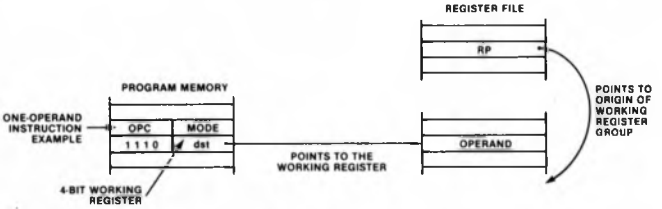


Bild 21: 4bit-Zeiger-(„Arbeitsregister"-)Adressierung

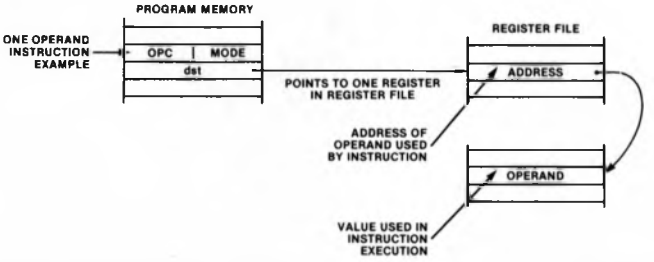


Bild 22: Register — Indirekte Adressierung in den internen Registern

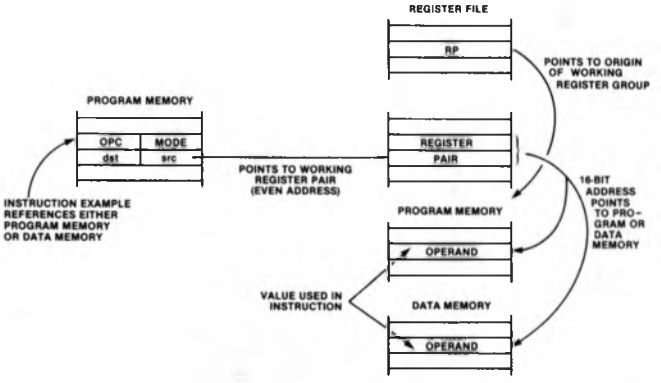


Bild 23: Register — Indirekte Adressierung für Programm- bzw. Datenspeicher

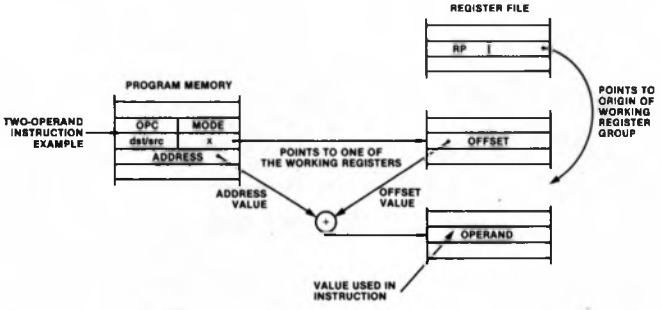


Bild 24: Indizierte Adressierung

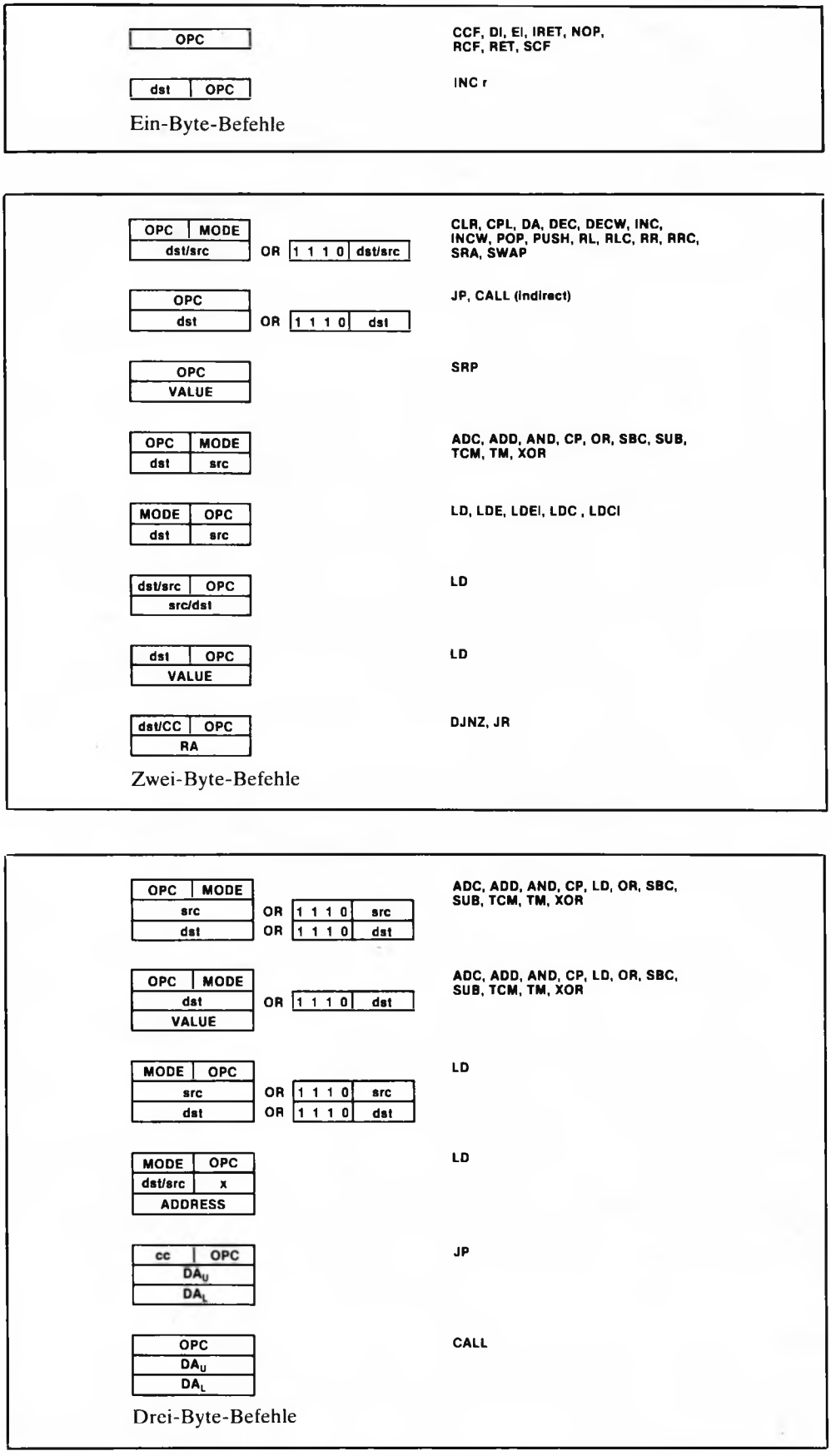


Bild 25: Übersicht über Befehlsformate

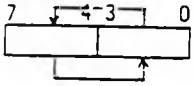
Eine Übersicht über die Befehlsformate ist in Bild 25 gegeben; den vollständigen Befehlsvorrat zeigt die folgende Befehlsliste.

Instruction and Operation	Addr Modes dst	src	Hex Opcode	Execution Bytes	Cycles	Pipline Cycles	Flags Affected C	Z	S	V	D	H
ADC dst,src	r	r	12	2	6	5	*	*	*	*	0	*
dst<-dst+src+C	r	Ir	13	2	6							
	R	R	14	3	10							
	R	IR	15	3	10							
	R	IM	16	3	10							
	IR	IM	17	3	10							
ADD dst,src	r	r	02	2	6	5	*	*	*	*	0	*
dst<-dst+src	r	Ir	03	2	6							
	R	R	04	3	10							
	R	IR	05	3	10							
	R	IM	06	3	10							
	IR	IM	07	3	10							
AND dst,src												
dst<-dst AND src	r	r	52	2	6	5	-	*	*	0	-	-
	r	Ir	53	2	6							
	R	R	54	3	10							
	R	IR	55	3	10							
	R	IM	56	3	10							
	IR	IM	57	3	10							
CALL dst	DA		D6	3	20	0	-	-	-	-	-	-
SP<-SP-2	IRR		D4	2	20	0						
@SP<-PC												
PC<-dst												
CCF			EF	1	6	5	*	-	-	-	-	-
C<-NOT C												
CLR dst	R		B0	2	6	5	-	-	-	-	-	-
dst<-0	IR		B1	2	6							
COM dst	R		60	2	6	5	-	*	*	0	-	-
dst<-NOT dst	IR		61	2	6							
CP dst,src	r	r	A2	2	6	5	*	*	*	*	-	-
dst-src	r	IR	A3	2	6							
	R	R	A4	3	10							
	R	IR	A5	3	10							
	R	IM	A6	3	10							
	IR	IM	A7	3	10							
DA dst	R		40	2	8	5	*	*	*	X	-	-
dst<-DA dst	IR		41	2	8							

Instruction and Operation	Addr Modes dst	src	Hex Opcode	Bytes	Execution Cycles	Pipline Cycles	Flags Affected C Z S V D H
DEC dst	R		00	2	6	5	- * * * - -
dst<-dst-1	IR		01	2	6		
DECW dst	RR		80	2	10	5	- * * * - -
dst<-dst-1	IR		81	2	10		
DI			8F	1	6	1	- - - - - -
IMR(7)<-0							
DJNZ r,dst	RA		rA	2	12/10	3/5	- - - - - -
r<-r-1			r=0-F		(taken/		
if r ≠ 0					not taken)		
PC<-PC+dst							
Range: +127,-128							
EI			9F	1	6	1	- - - - - -
IMR(7)<-1							
INC dst	r		rE	1	6	5	- * * * - -
dst<-dst+1			r=0-F				
	R		20	2	6		
	IR		21	2	6		
INCW dst	RR		A0	2	10	5	- * * * - -
dst<-dst+1	IR		A1	2	10		
IRET			BF	1	16	0	* * * * * *
FLAGS<-@SP							
SP<-SP+1							
PC<-@SP							
SP<-SP+2							
IMR(7)<-1							
JP cc,dst	DA		cD	3	12/10	0	- - - - - -
if cc is true,			c=0-F		(taken/		
PC<-dst					not taken)		
	IRR		30	2	8		
JR cc,dst	RA		cB	2	12/10	3	- - - - - -
if cc is true,			c=0-F		(taken/		
PC<-PC+dst					not taken)		
Range: +127,-128							
LD dst,src	r	IM	rC	2	6	5	- - - - - -
dst<-src	r	R	r8	2	6		

Instruction and Operation	Addr Modes dst	src	Hex Opcode	Bytes	Execution Cycles	Pipeline Cycles	Flags Affected C Z S V D H
	R	r	r9 r=0-F	2	6		
	r	X	C7	3	10		
	X	r	D7	3	10		
	r	Ir	E3	2	6		
	Ir	r	F3	2	6		
	R	R	E4	3	10		
	R	IR	E5	3	10		
	R	IM	E6	3	10		
	IR	IM	E7	3	10		
	IR	R	F5	3	10		
LDC dst,src dst<-src	r Irr	Irr r	C2 D2	2 2	12 12	0	- - - - - -
LDCI dst,src dst<-src r<-r+1 rr<-rr+1	Ir Irr	Irr Ir	C3 D3	2 2	18 18	0	- - - - - -
LDE dst,src dst<-src	r Irr	Irr r	82 92	2 2	12 12	0	- - - - - -
LDEI dst,src dst<-src r<-r+1 rr<-rr+1	Ir Irr	Irr Ir	83 93	2 2	18 18	0	- - - - - -
NOP			FF	1	6	0	- - - - - -
OR dst,src dst<-dst OR src	r r R R R IR	r Ir R IR IM IM	42 43 44 45 46 47	2 2 3 3 3 3	6 6 10 10 10 10	5	- * * 0 - -
POP dst dst<-@SP SP<-SP+1	R IR		50 51	2 2	10 10	5	- - - - - -
PUSH src SP<-SP-1 @SP<-src		R IR	70 71	2 2	10/12 (int/ext stack) 12/14 (int/ext stack)	1	- - - - - -

Instruction and Operation	Addr Modes dst	src	Hex Opcode	Execution Bytes	Cycles	Pipeline Cycles	Flags Affected C	Z	S	V	D	H
RCF C<-0			CF	1	6	5	0	-	-	-	-	-
RET PC<-@SP SP<-SP+2			AF	1	14	0	-	-	-	-	-	-
RL dst	R		90	2	6	5	*	*	*	*	-	-
	IR		91	2	6							
RLC dst	R		10	2	6	5	*	*	*	*	-	-
	IR		11	2	6							
RR dst	R		E0	2	6	5	*	*	*	*	-	-
	IR		E1	2	6							
RRC dst	R		C0	2	6	5	*	*	*	*	-	-
	IR		C1	2	6							
SBC dst,src dst<-dst-src-C	r	r	32	2	6	5	*	*	*	*	1	*
	r	Ir	33	2	6							
	R	R	34	3	10							
	R	IR	35	3	10							
	R	IM	36	3	10							
	IR	IM	37	3	10							
SCF C<-1			DF	1	6	5	1	-	-	-	-	-
SRA dst	R		D0	2	6	5	*	*	*	0	-	-
	IR		D1	2	6							

Instruction and Operation	Addr Modes dst	src	Hex Opcode	Bytes	Execution Cycles	Pipeline Cycles	Flags Affected C Z S V D H
SRP src RP<-src		IM	31	2	6	1	- - - - - -
SUB dst,src dst<-dst-src	r	r	22	2	6	5	* * * * 1 *
	r	Ir	23	2	6		
	R	R	24	3	10		
	R	IR	25	3	10		
	R	IM	26	3	10		
	IR	IM	27	3	10		
SWAP dst  	R		F0	2	8	5	X * * X - -
	IR		F1	2	8		
TCM dst,src (NOT dst)AND src	r	r	62	2	6	5	- * * 0 - -
	r	Ir	63	2	6		
	R	R	64	3	10		
	R	IR	65	3	10		
	R	IM	66	3	10		
	IR	IM	67	3	10		
TM dst,src dst AND src	r	r	72	2	6	5	- * * 0 - -
	r	Ir	73	2	6		
	R	R	74	3	10		
	R	IR	75	3	10		
	R	IM	76	3	10		
	IR	IM	77	3	10		
XOR dst,src dst<-dst XOR src	r	r	B2	2	6	5	- * * 0 - -
	r	Ir	B3	2	6		
	R	R	B4	3	10		
	R	IR	B5	3	10		
	R	IM	B6	3	10		
	IR	IM	B7	3	10		

Elektrische Kenndaten

Absolute Grenzdaten

Voltages on all inputs and outputs with respect to GND .....-0.3V to +7.0V

Operating Ambient Temperature .....0°C to +70°C

Storage Temperature ....-65°C to +150°C

Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test-Bedingungen

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows: $+4.75V \leq V_{CC} \leq +5.25V$, GND = 0V, $0°C \leq T_A \leq +70°C$.

Statische Kenndaten

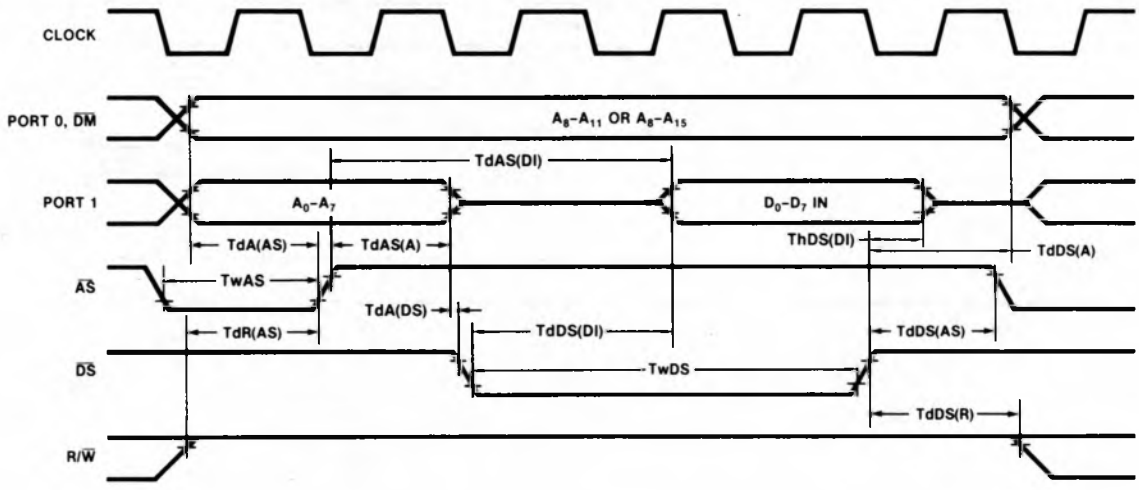
Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage			V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage			V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage			V		
V_{RL}	Reset Input Low Voltage			V		
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$	1
V_{OL},	Output Low Voltage		0.45	V	$I_{OL} = +2.0$ mA	1
I_{IL}	Input Leakage			μA	$0 \leq V_{IN} \leq +5.25V$	
I_{OL}	Output Leakage			μA	$0 \leq V_{IN} \leq +5.25V$	
I_{IR}	Reset Input Current			μA	$V_{RL} = 0V$, $V_{CC} = +5.25V$	
I_{DD}	V_{DD} Supply Current		200	mA		
I_{MM}	V_{MM} Supply Current		20	mA		

1. For A_0-A_{11}, $\overline{MDS}$, $\overline{SYNC}$, SCLK and IACK on the Z8/64 pin version, $I_{OH} = -100\ \mu A$ and $I_{OL} = 1.0$ mA.

Zeitverhalten beim Einlesen von Befehlen, Eingabe oder Speicherlesezugriffe

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Float Delay Time	60		ns	Test Load 1	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 1	3
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdA(DS)	Address Float to Data Strobe Delay Time	0		ns	Test Load 1	
TwDS	Data Strobe Width	230		ns	Test Load 1	2
TdDS(DI)	Data Strobe to Data In Valid Delay Time		160	ns	Test Load 1	3
ThDS(DI)	Data In Hold Time	0		ns		
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		ns	Test Load 1	1
TdR(AS)	Read Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(R)	Data Strobe to Read Change Delay	60		ns	Test Load 1	1

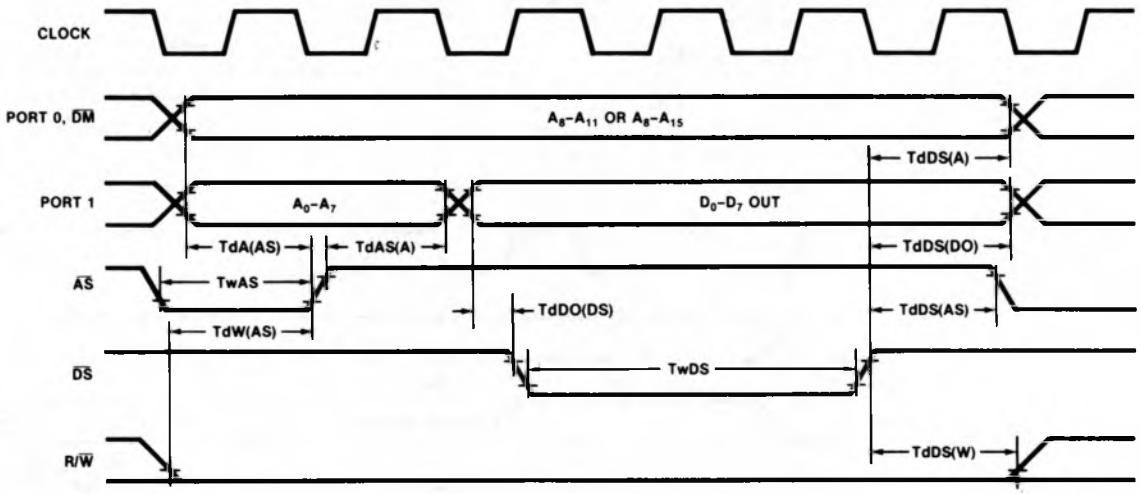
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
2. Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-9 and 1-10.
3. Address Strobe and Data Strobe to Data In Valid delay times represent memory system access times and are given for an 8 MHz crystal input frequency. For lower frequencies; the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdDS(DI).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



Zeitverhalten bei Ausgabe oder Speicherschreibzugriffen

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdAS(A)	Address Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TwAS	Address Strobe Width	60		ns	Test Load 1	1
TdDO(DS)	Data Out Valid to Data Strobe Delay Time	30		ns	Test Load 1	1
TwDS	Data Strobe Width	150		ns	Test Load 1	2
TdDS(A)	Data Strobe to Address Change Delay Time	60		ns	Test Load 1	1
TdDS(DO)	Data Strobe to Data Out Change Delay Time	60		ns	Test Load 1	1
TdDS(AS)	Data Strobe to Address Strobe Delay Time	50		ns	Test Load 1	1
TdW(AS)	Write Valid to Address Strobe Delay Time	30		ns	Test Load 1	1
TdDS(W)	Data Strobe to Write Change Delay Time	60		ns	Test Load 1	1

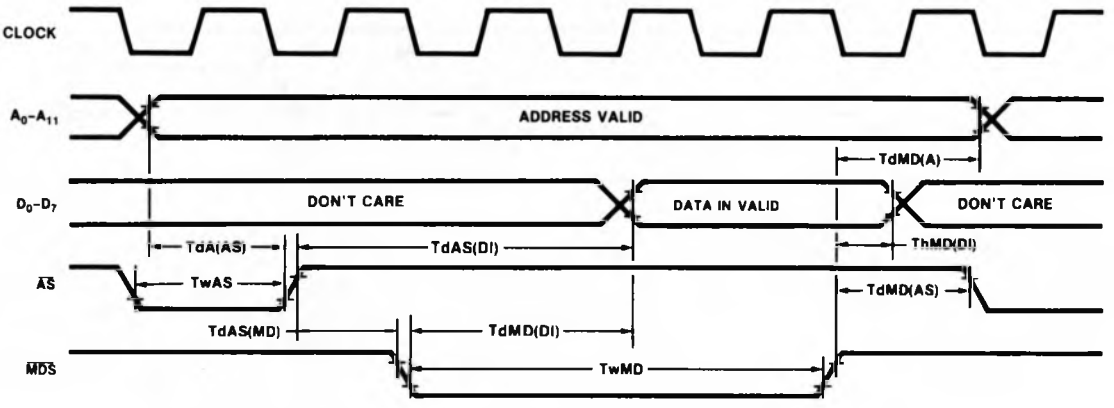
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
2. Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-9 and 1-10.
3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



Zeitverhalten der 64 pin-Version (Z8/64) beim „Memory-Port“

Symbol	Parameter	Min	Max	Unit	Condition	Notes
TdA(AS)	Address Valid to Address Strobe Delay Time	30		ns	Test Load 2	1
TdAS(DI)	Address Strobe to Data In Valid Delay Time		280	ns	Test Load 2	3
TwAS	Address Strobe Width	60		ns	Test Load 2	1
TdAS(MD)	Address Strobe to Memory Data Strobe Delay Time	60		ns	Test Load 2	1
TwMD	Memory Data Strobe Width	230		ns	Test Load 2	2
TdMD(DI)	Memory Data Strobe to Data In Valid Delay Time		160	ns	Test Load 2	1
ThMD(DI)	Data In Hold Time	0		ns		
TdMD(A)	Memory Data Strobe to Address Change Delay Time	60		ns	Test Load 2	1
TdMD(AS)	Memory Data Strobe to Address Strobe Delay Time	50		ns	Test Load 2	1

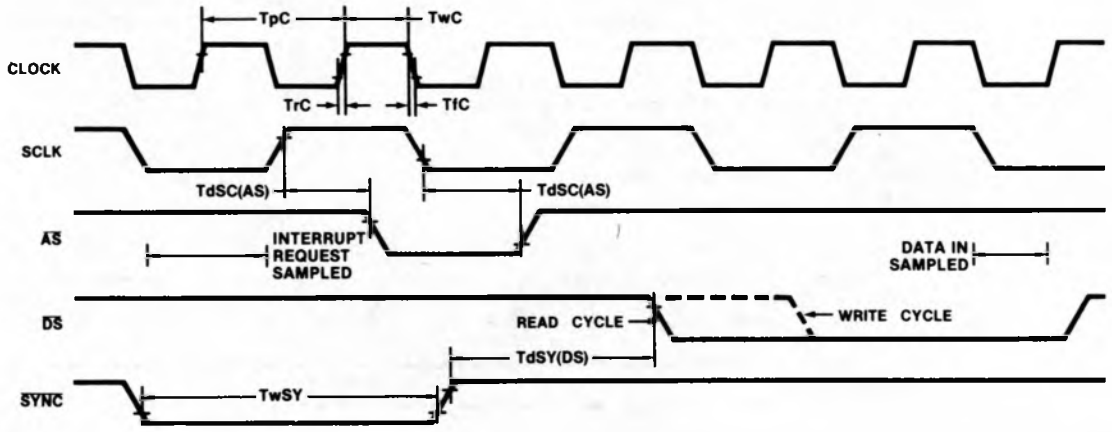
1. Delay times given are for an 8 MHz crystal input frequency. For lower frequencies, the change in clock period must be added to the delay time.
2. Memory Data Strobe Width is given for an 8 MHz crystal input frequency. For lower frequencies the change in three clock periods must be added to obtain the minimum width. The Memory Data Strobe Width varies according to the instruction being executed. Refer to Figures 1-9 and 1-10.
3. Address Strobe and Memory Data Strobe to Data In Valid delay times represent memory system access times and are given at an 8 MHz crystal input frequency. For lower frequencies the change in four clock periods must be added to TdAS(DI) and the change in three clock periods added to TdMS(DI).
4. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."



Übriges Zeitverhalten

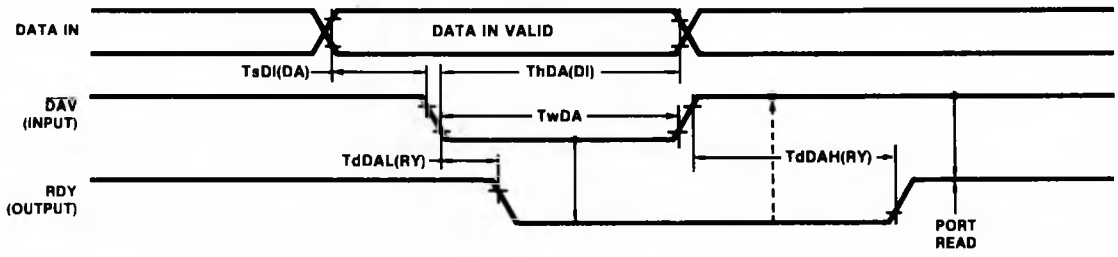
Symbol	Parameter	Min	Max	Unit	Condition	Notes
TpC	Input Clock Period	125		ns		
TrC, TfC	Input Clock Rise and Fall Times			ns	From External Clock Generator	
TwC	Input Clock Width			ns	From External Clock Generator	
TdSC(AS)	System Clock Out to Address Strobe Delay Time			ns		1
TdSY(DS)	Instruction Sync Out to Data Strobe Delay Time			ns		1, 2
TwSY	Instruction Sync Out Width			ns		1, 2

1. Test Conditions use Test Load 1 for SCLK and $\overline{\text{SYNC}}$ when output through their respective Port 3 pins and Test Load 2 on the SCLK and $\overline{\text{SYNC}}$ direct outputs on the 64 pin version.
2. Times given assume an 8 MHz crystal input frequency. For lower frequencies, the change in two clock periods must be added.
3. All timing references assume 2.0V for a logic "1" and 0.8V for a logic "0."

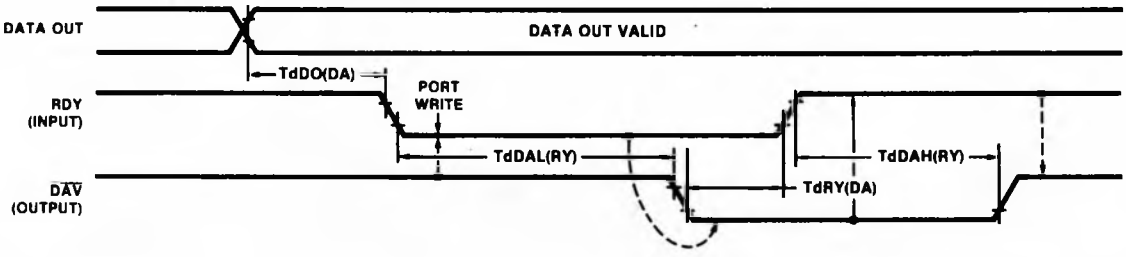


Zeitverhalten im Quittungsbetrieb

Symbol	Parameter	Min	Max	Unit	Condition
TsDI(DA)	Data In Setup Time	0		ns	
ThDA(DI)	Data In Hold Time	190		ns	
TwDA	Data Available Width			ns	Input Handshake Test Load 1
TdDAL(RY)	Data Available Low to Ready Delay Time			ns	Input Handshake Test Load 1
		0		ns	Output Handshake Test Load 1
TdDAH(RY)	Data Available High to Ready Delay Time			ns	Input Handshake Test Load 1
		0		ns	Output Handshake Test Load 1
TdDO(DA)	Data Out to Data Available Delay Time			ns	Test Load 1
TdRY(DA)	Ready to Data Available Delay Time			ns	Test Load 1

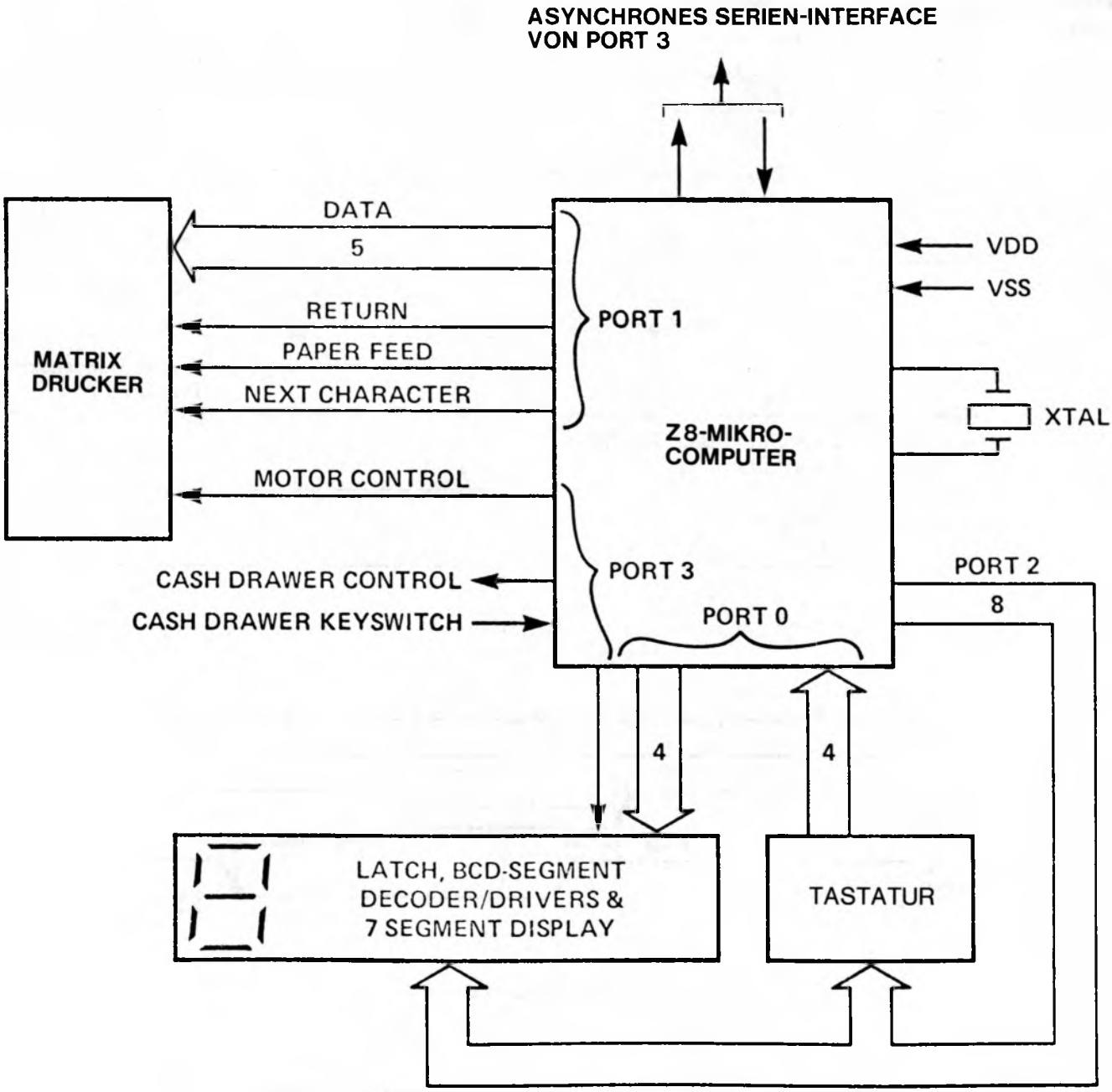


Input Handshake

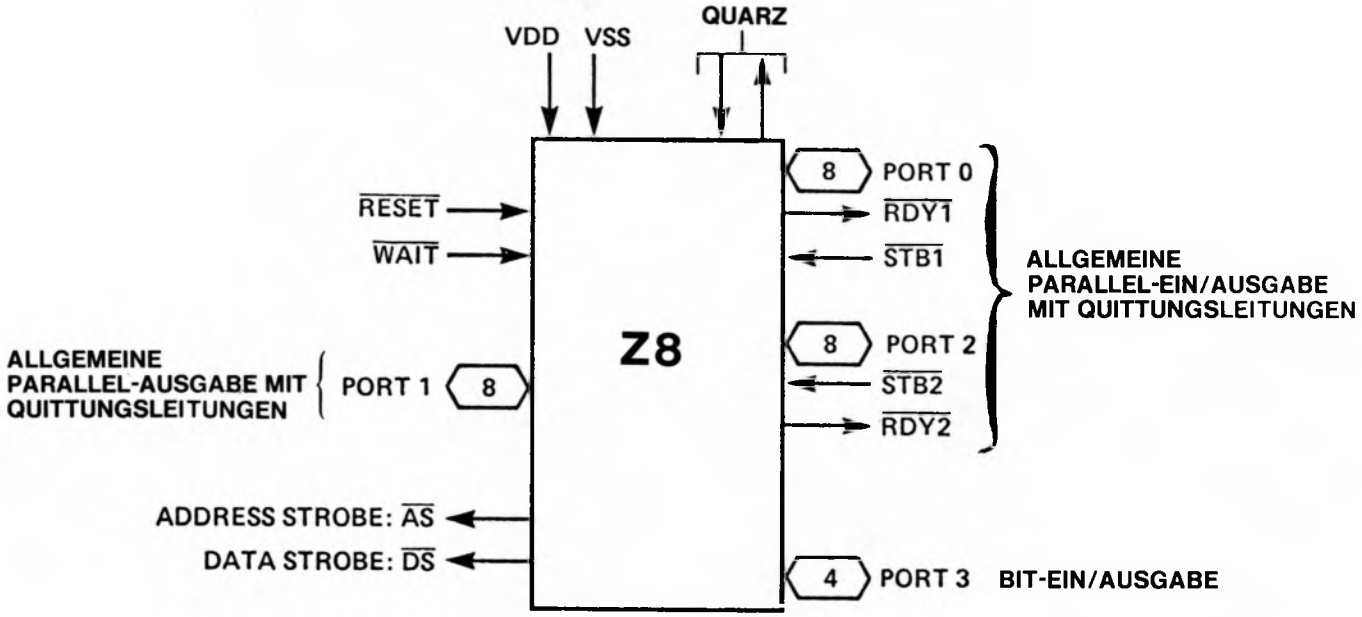


Output Handshake

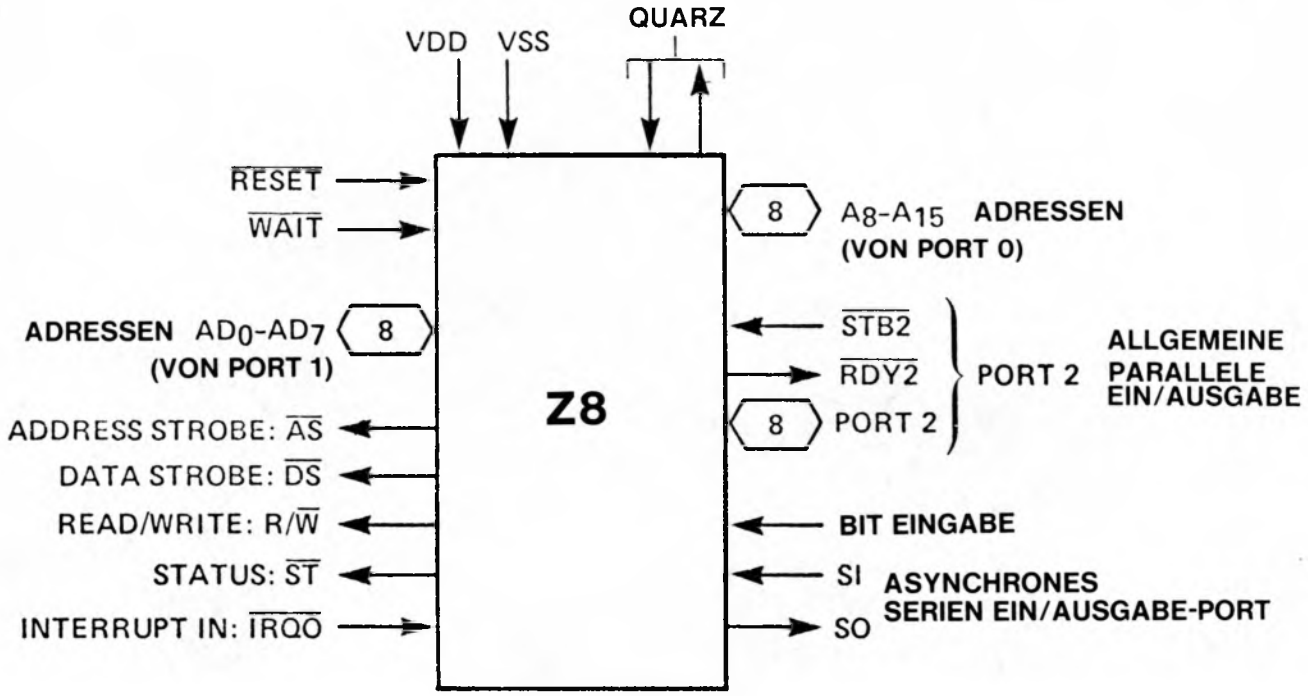
Z-8 MIKROCOMPUTER-KONFIGURATIONSBEISPIEL: REGISTRIERKASSE



KONFIGURATIONS-BEISPIEL 1:
STAND-ALONE-MIKROCOMPUTER AUF 1 CHIP



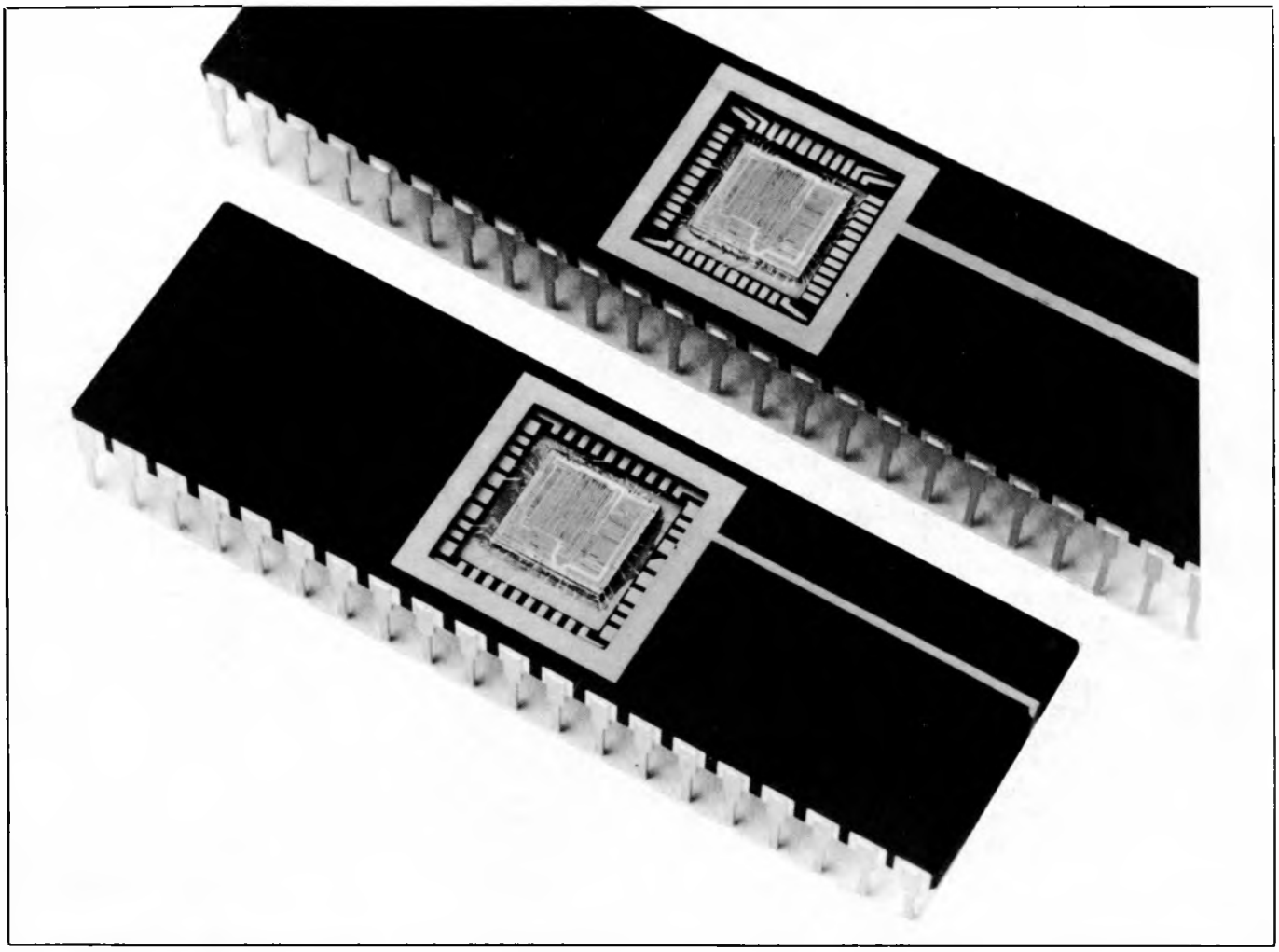
KONFIGURATIONS-BEISPIEL 2:
Mikrocomputer auf 1 Chip mit Serien- und Parallelschnittstelle und 16 Adreßleitungen.



Z 8001
Z 8002

Zilog

**16 bit-
Mikroprozessoren**



4. ZILOG-Z 8000-MIKROCOMPUTER-BAUSTEINE

Die Mikrocomputerfamilie Z8000

Das Ergebnis des Z8000 Entwicklungsprogrammes ist ein 16 Bit Mikroprozessor, der sich durch seine richtungsweisende Architektur ein breites Anwendungsfeld und eine lange Produktlebensdauer sichert. Die umfangreichen Möglichkeiten des Z8000 Speicher zu adressieren und zu verwalten, stellen hervorstechende Eigenschaften dar: Es können bis zu 8 Megabyte direkt adressiert werden. Die weitreichenden Fähigkeiten der CPU, wie zahlreiche Register, viele Datentypen, ein großer Befehlsvorrat sowie eine bisher bei Mikroprozessoren nicht vorhandene Regelmäßigkeit der Registerorganisationen, vereinfachen den Programmieraufwand erheblich und reduzieren die Programmlänge.

Die Z8000 erreicht ihren hohen Durchsatz mit relativ niedriger Taktfrequenz. Daher können Speicher mit einer vergleichsweise langen Zugriffszeit verwendet werden. Spezielle Befehle und die Systemarchitektur erlauben es, optimal mit Compilern zu arbeiten, sowie wirkungsvolle Betriebssysteme zu erstellen. Schließlich wird der Aufbau von Multiprozessor-Systemen sowohl durch Hardware als auch Software unterstützt.

1. Die Zentralbausteine Z8001 und Z8002

Bei einem zukunftsweisenden Mikroprozessorkonzept darf die Architektur sich nicht nur auf die Adressierbarkeit großer Speicherräume beschränken. Die Fähigkeiten der CPU müssen ebenfalls ausreichend sein, komplexe Probleme optimal zu lösen.

1.1 Die Register

Die Z8000 bietet sechzehn allgemeine 16-Bit Register sowie zusätzlich einige Systemregister. Alle 16 Register (R0-R15) können als Akkumulatoren verwendet werden. Alle — bis auf ein Register — können als Indexregister dienen (R1-R15)-R0-R7 sind als sechzehn 8-Bit Register verwendbar (RHO, RLO-RH7, RL7) wobei wiederum alle als Akkumulatoren verwendbar sind.

Zur Verarbeitung von 32-Bit Doppelworten werden 8 Doppelregister gebildet (RR0-RR14), für 64-Bit Daten entsprechend Vierergruppen (RQ0-RR14). Zwei Registerblöcke dienen als System-Register. Dies sind die Programmstatusregister sowie ein Register, das einen Vektor für die Verarbeitung von Interrupts und Traps enthält.

Ein weiteres Register steuert von der CPU aus den Refresh dynamischer Speicher. Neun Bit dieses Registers dienen als Zähler zur Ausgabe der Refresh-Adressen. Weitere sechs Bit dienen als Vorteiler zur Einstellung der Refresh-Intervalle. Die Intervalle können zwischen 1 und 64 sec. liegen. Bit 15 des Refresh-Registers dient zur Abschaltung der Refresh-Automatik. Abbildung 1a, b zeigt die Registerstruktur.

1.1.1 Stack und Stackpointer

In der „nichtsegmentierten" Version (Z8002, auf die Segmentierung wird später eingegangen) enthält R 15 den Stackpointer, in der segmentierten Art RR14 (32 Bit). Der Stackpointer kann durch alle Befehle beeinflußt werden, da er Teil des allgemeinen Registersatzes ist. Dieser Stackpointer wird implizit von den Befehlen „Call" und „RET" verwendet und kann explizit für die Befehle „PUSH" und „POP" angewendet werden.

Die Prozessoren der Z8000-Serie können in zwei Betriebsarten arbeiten: Systembetrieb und Normalbetrieb. Daher wurde ein zusätzlicher Stackpointer vorgesehen. Einer wird für Systemsoftware benutzt, der andere für die Anwendersoftware. Damit wird vermieden, daß der Systemstack, der z.B. die Statusinformationen bei Interruptbetrieb enthält, mit Daten aus dem Anwenderprogramm (PUSH, POP) vermischt wird und ebenso der Anwenderstack frei von Systemteilen ist. Über besondere Befehle ist es jedoch im Systembetrieb möglich, auf den Anwenderstack zuzugreifen.

1.2 Befehle

Die Prozessoren der Z8000-Serie bieten über 110 verschiedene Befehlstypen. Ein Befehlswort enthält mehrere Felder (Abbildung 2,3).

1. „OP-Code"-Feld, hier wird die Befehlsart festgelegt.
2. „Mode"-Feld, Definition der Adressierungsart.
3. „Datenelement"-Feld, Definition der Datenart. (Bit, Byte, Wort . . . . .)
4. Operandenfeld (z.B. Register)

Je nach Adressierungsart folgt dem Befehl eine Adresse oder Daten.

Die Länge des gesamten Befehls kann von eins bis fünf Worten (16 Bit) variieren. Die Befehle können bis zu vier Operanden definieren.

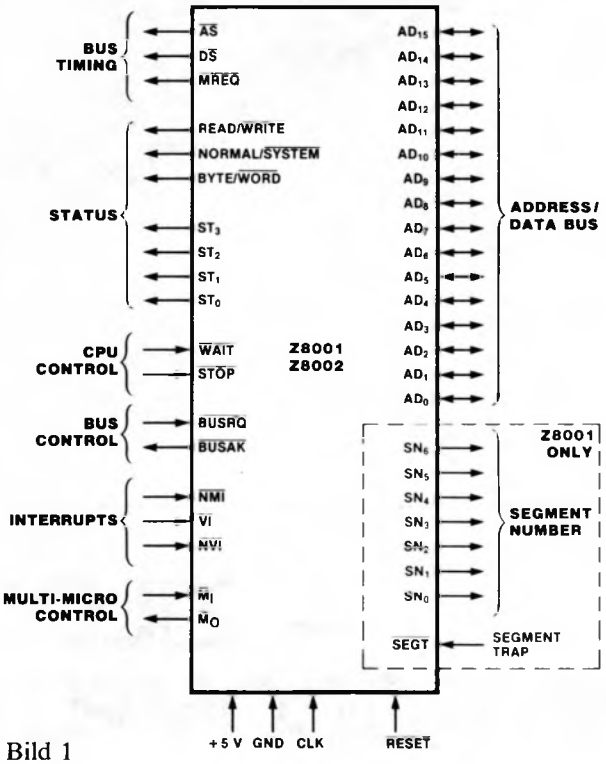


Bild 1

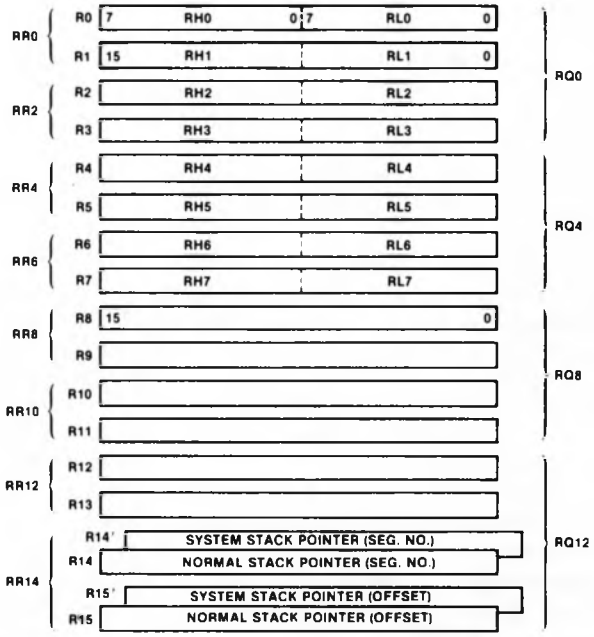


Bild 1a
Die Register der Z8001-CPU (segmentierte Version)

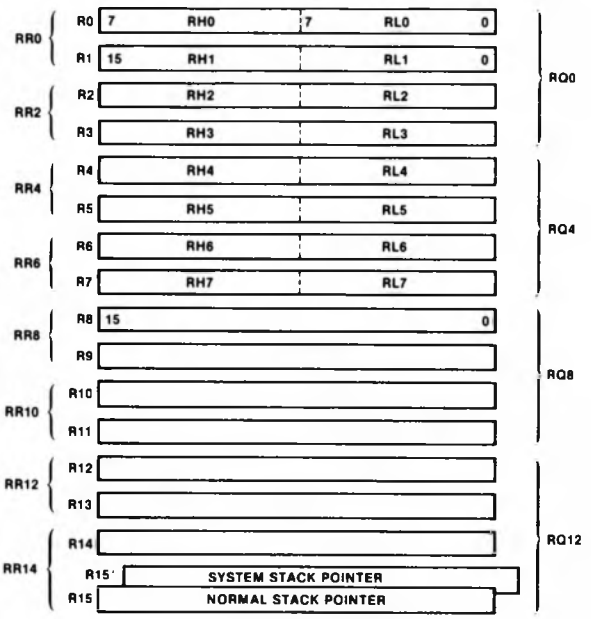


Bild 1b
Die Register der Z8002-CPU (nicht segmentierte Version)

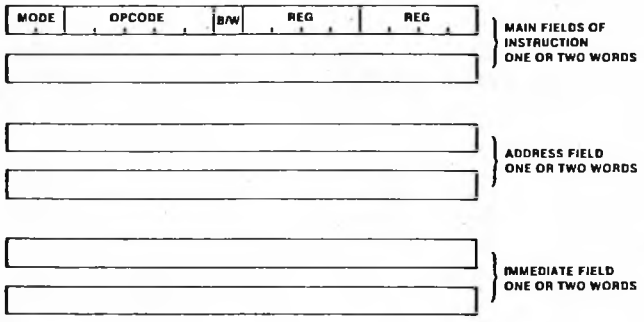


Bild 2
Typischer Aufbau eines Befehls

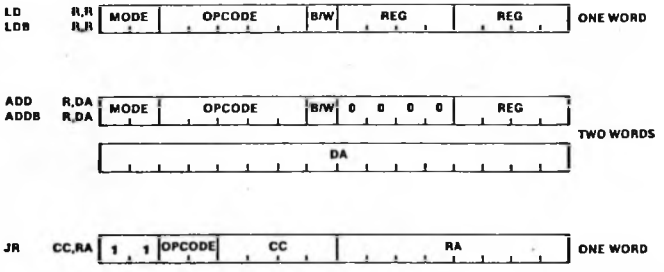


Bild 3
Beispiele einiger Befehlsformate (nicht segmentierte Version)

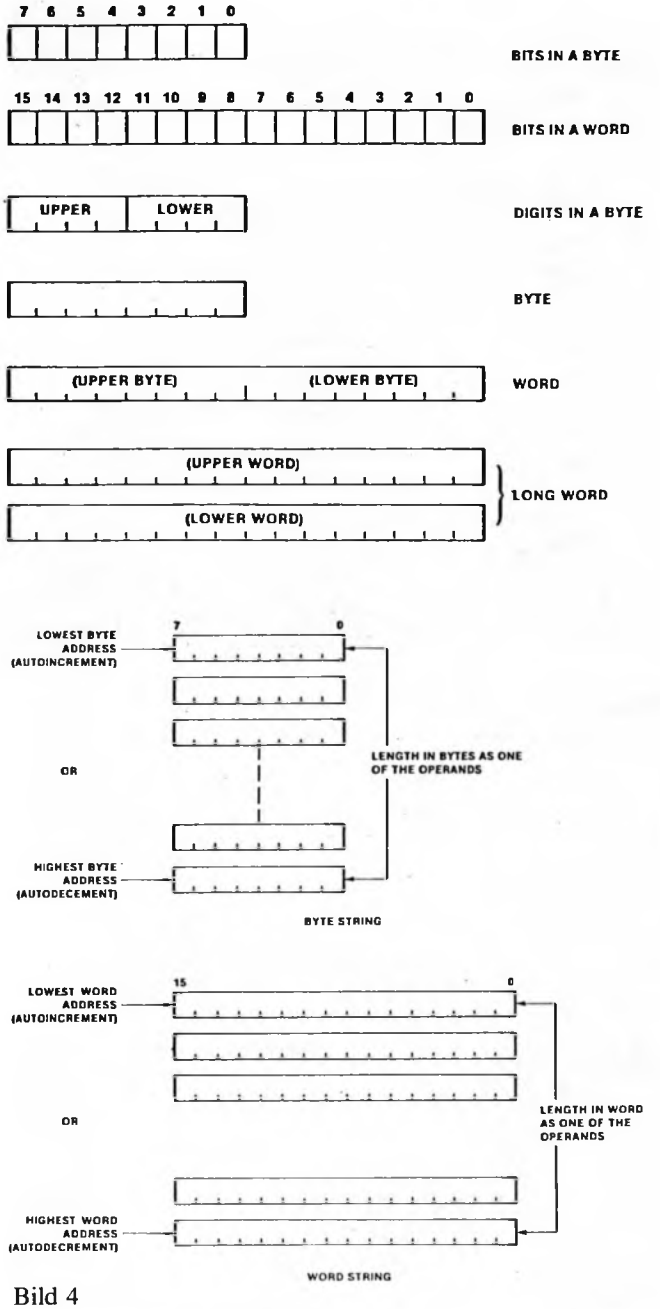


Bild 4
Adressierbare Datenelemente

1.3 Datentypen

In den Prozessoren der Z8000-Serie werden die folgenden Typen von Daten verwendet (Bild 4).

1. Bits
2. BCD Digits (4 Bit)
3. Bytes (8 Bit)
4. Worte (16 Bit)
5. Doppelworte (32 Bit)
6. Vierfachworte (64 Bit bei Multiplikation und Division)
7. Wort-Sequenzen
8. Byte-Sequenzen

Das Basis-Datenelement ist das Byte. Die Anzahl der zu verarbeitenden Bytes ist entweder implizit gegeben oder sie wird vom Programmierer bei Sequenz-Verarbeitungen festgelegt. Ein Datenelement in einem Register wird durch eine Byte-Register-Adresse für Bits, Digits oder Bytes festgelegt, durch eine 16 Bit Register Adresse für ein Wort oder für ein Doppelwort durch eine Registerpaar-Adresse. Sequenzen sind im Speicher abgelegt. Sie werden durch die höchste oder niedrigste Byte-(Wort-) Adresse und die Länge in Bytes (Worten) festgelegt.

1.4 Speicheradressierung

Zum Erreichen einer optimalen Anpassung des Speicherraumes an die Anwendung wird die Z8000 in zwei Versionen angeboten: Eine 40 Pin Version mit 64 KByte Adreßbereich (Z8002), sowie eine 48 Pin Version mit 8 Megabyte Adreßbereich (Z8001). Die Z8002 wird als „nicht segmentiert", die Z8001 als „segmentiert" bezeichnet.

Aufwärtskompatibilität ist gegeben. Jedes Programm für die nichtsegmentierte Version läuft in einem Segment der segmentierten Version.

Wird auf einen Speicherplatz zugegriffen, dann unterscheidet die CPU zwischen Befehls-, Daten- und Stack-Zugriffen, und zwar sowohl im System — wie im Anwenderbetrieb. Jede dieser sechs Fälle wird durch die Statusleitungen angezeigt. Unter Ausnutzung dieser Status-Information ist es möglich die Speicherbereiche voneinander zu trennen und somit 384 K Byte bzw. 48 Megabyte zu adressieren. Jeder Adreßbereich kann als eine Folge von Bytes angesehen werden, die nacheinander in aufsteigender Reihenfolge numeriert sind. Jede dieser Nummern kann als Adresse verwendet werden. Somit ist das Byte das adressierbare Grundelement. Im Falle von Bits und Digits ist das dazugehörige Byte zu adressieren, bei Datentypen von 16 und mehr Bit wird das höchstwertige Byte adressiert (Bild 5). Ein Bit kann entweder über seine Nummer in einer Byte-Adresse oder Wort-Adresse angesprochen werden. Ein Wort, wie ein Doppelwort wird über die Byte-Adresse seines höchstwertigen Bytes spezifiziert.

Befehle werden grundsätzlich als Wort adressiert, wobei das am weitesten links stehende Byte immer eine geradzahlige Adresse hat.

1.5 Eingabe/Ausgabe

Die Z8001 und Z8002 verwenden Ein-/Ausgabe-Befehle und vom Speicherbereich getrennte E/A-Adressen.

Die E/A-Adressen sind 16 Bit lang und werden im Multiplex mit den dazugehörigen Daten ausgegeben. Blockbefehle sorgen für die einfache Möglichkeit, größere Datenmengen zu transferieren.

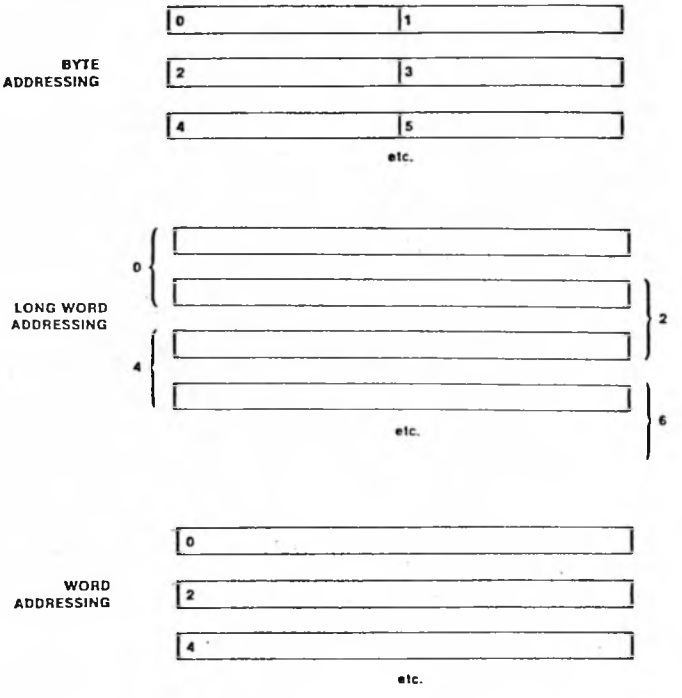


Bild 5
Format der Datenelemente im Speicher

1.6 Interrupts, Traps und Reset

Interrupts sind asynchron auftretende Ereignisse, die von Peripherie-Einheiten als Anforderung an die CPU ausgelöst werden. **Traps** sind synchron auftretende Ereignisse, die aus einer Abarbeitung besonderer Befehle entstehen und bei der Wiederverarbeitung der Befehle immer wieder entstehen können. Interrupts und Traps werden von der Hardware in gleicher Weise verarbeitet.

Programmstatus. Es bedeuten:

1. Flags:		2. Statusbits	
C	Übertrag	VIE	vekt, interr, frei
Z	Null	MVIE	nicht vekt. interr. frei
S	Vorzeichen	STOP	Stop
P/V	Parität/Überlauf	SEG	segment. / nicht segment
DA	Dezimal Korrektur	S/N	System/Anwender
H	Digit Übertrag		

Schattierte Bereiche sind reserviert.

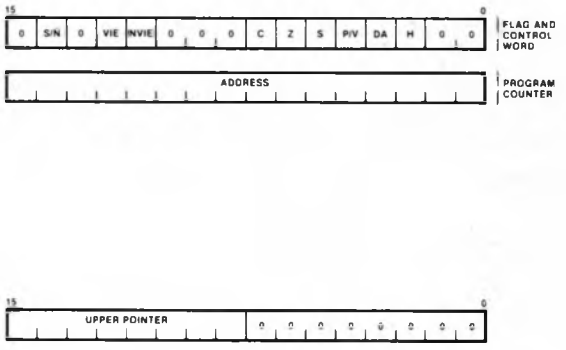
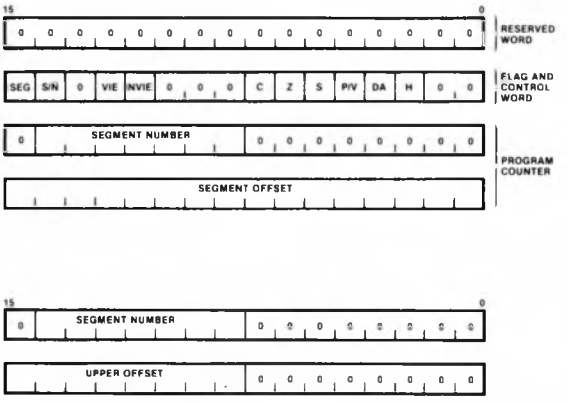


Bild 6

Es werden drei Interrupts verwendet, nicht maskierbarer Interrupt (NMI), nicht vektorisierter Interrupt (NVI) und vektorisierter Interrupt (VI) sowie fünf Traps (System-Aufruf, nicht erlaubter Befehl, bevorzugter E/A-Befehl, andere bevorzugte Befehle, Segmentierung).

Erfolgt ein Interrupt oder Trap, dann wird der alte Programmstatus im Systemstack gerettet. Es werden der Programmzähler, das Statuswort und der „Grund" für die Aktivität abgelegt. Ein neuer Programmstatus wird aus dem Bereich geladen, der durch einen Zeiger (New Programstatus Areapointer **NPSAP** definiert ist. (Bild 6, 7, 8). Reset hat die höchste Wertigkeit. Bei einer Reset-Sequenz wird ein Programmstatus von zwei Worten (vier Worten bei Segmentierung) geladen. Diese Informationen werden aus Adresse 0 (Segment 0) gelesen. Es wird keine Stackoperation durchgeführt.



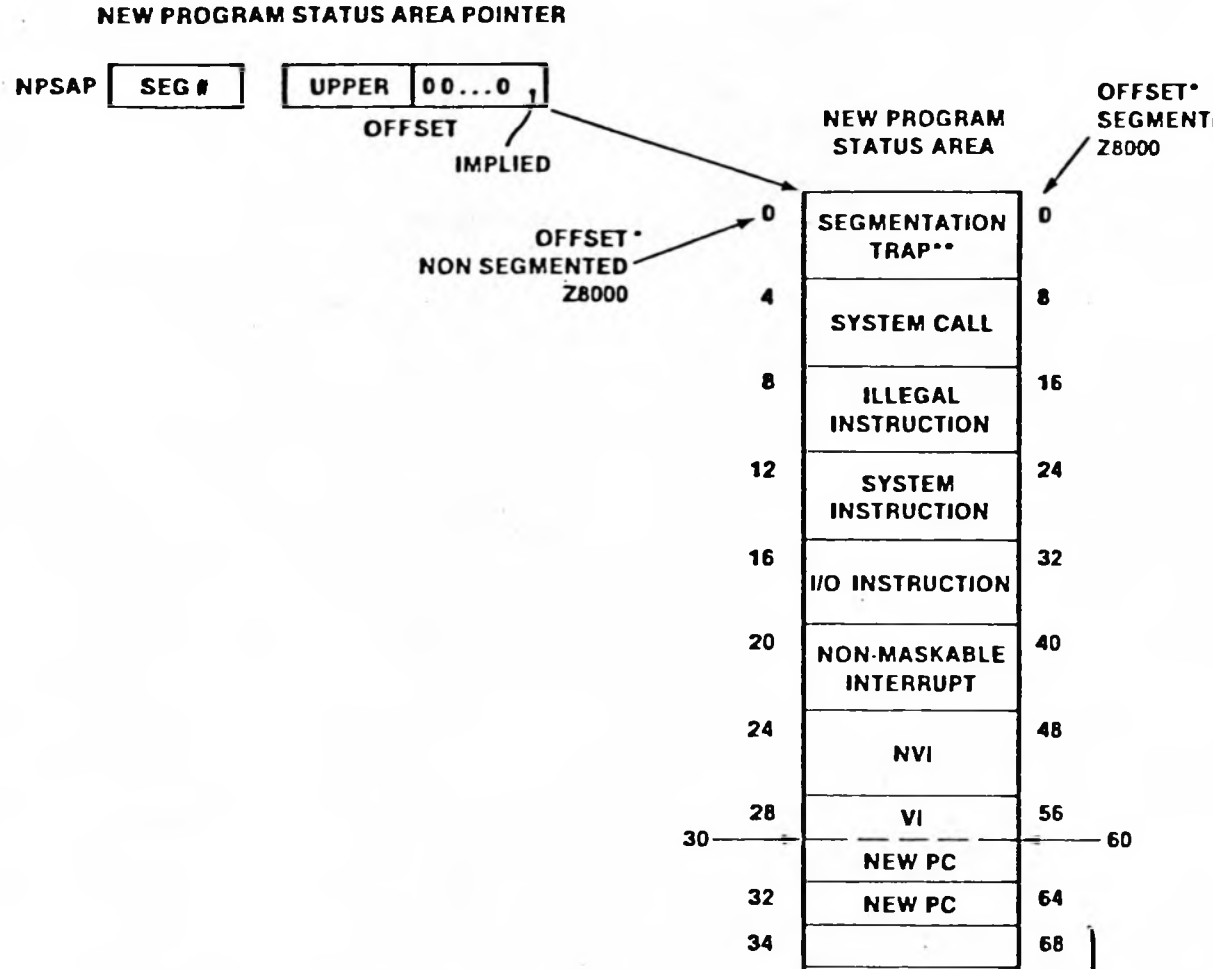


Bild 7
Ablauf der Programmstatus-Rettung bei Interrupts oder Traps.

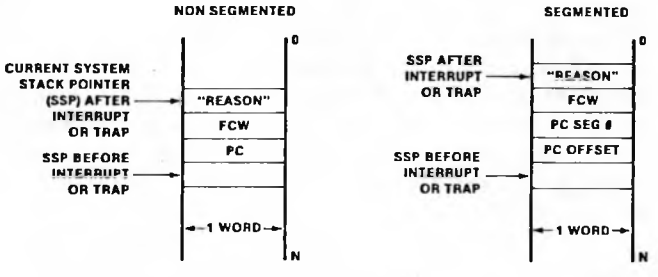
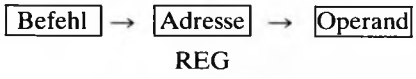


Bild 8
Aufteilung des Speicherbereiches, der die Statusinformationen enthält, die nach Interrupts oder Tra

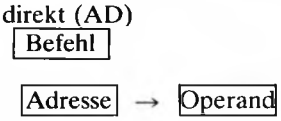
1.7 Adressierweisen

Die zu einer Befehlsausführung notwendigen Operanden werden durch Adressen spezifiziert. Dies können Register-, Speicher-, oder E/A-Adressen sein.
Die Adressierweisen sind entweder explizit oder durch den Befehl implizit gegeben.
Die Z8001 und Z8002 verwenden acht Adressierweisen:
1. Register (R)

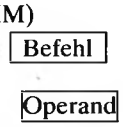
Der Operandenwert ist der Inhalt des Registers.
2. Indirekter Register (IR)



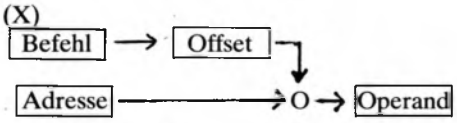
Der Operand ist der Inhalt der Speicherzelle, deren Adresse das Register enthält.
3. Adresse direkt (AD)



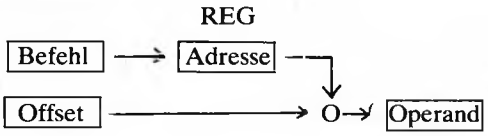
Der Operandenwert ist der Inhalt der Speicherzelle, deren Adresse im Befehl enthalten ist.
4. Direkt (IM)



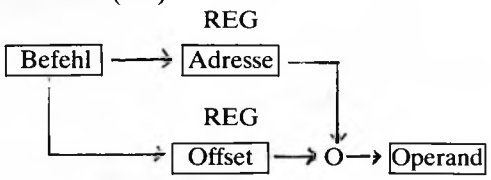
Der Operandenwert ist im Befehl enthalten.
5. Indiziert (X)



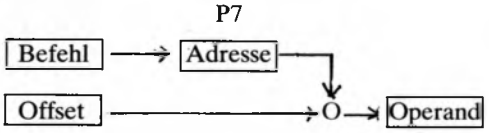
Der Operandenwert ist der Inhalt der Speicherzelle, deren Adresse durch die Summe aus der im Befehl enthaltenen Adresse und einem Offset, der in einem Register steht, gebildet wird.
6. Basis Adresse (BA)



Der Operandenwert ist der Inhalt der Speicherzelle, deren Adresse durch die Summe aus der in einem Register stehenden Adresse und einem Offset, der im Befehl enthalten ist gebildet wird.
7. Basis indiziert (BX)



Der Operandenwert ist der Inhalt der Speicherzelle, deren Adresse durch die Summe einer Adresse und einem Offset, die beide in einem Register stehen, gebildet werden.
8. Relativ (RA)



Der Operandenwert ist der Inhalt der Speicherzelle, deren Adresse durch die Summe aus dem Inhalt des Programmzählers (PZ) und einem Offset, der im Befehl enthalten ist, gebildet wird. Der Offset ist im Zweierkomplement dargestellt.

2. Segmentierung und Speicherverwaltung

Wie bereits angedeutet, wird die Z8000 in zwei Versionen angeboten. Die Z8002 wird in einem 40-Pin-Gehäuse geliefert und kann mit den vorhandenen 16 Adreßleitungen 64 K-Byte Speicher adressieren. Um Anwendungen, die einen größeren Speicherbedarf erfordern, gerecht zu werden, ist außerdem eine segmentierte Version (Z8001) im 48-Pin-Gehäuse erhältlich. Hier können direkt 8 Megabyte Speicher adressiert werden. Die 23 vorhandenen Adreßleitungen sind in zwei Gruppen aufgeteilt.
Eine 7 Bit Segmentnummer zeigt auf den Beginn eines Speicherbereiches, der Segment genannt wird;
ein 16 Bit Offset adressiert jede Speicherzelle relativ zum Segmentbeginn.
Beide Adreßteile können getrennt bearbeitet werden. Die Z8001 kann in beiden Arten (segmentiert und nicht segmentiert) betrieben werden.
Wenn Speicher-Segmente verschiebbar sein sollen, wird eine Adreßumwandlung notwendig. Diese Umwandlung wird von einem getrennten Speicherverwalter durchgeführt. Dieser Baustein überträgt segmentierte Adressen — auch als logische Adressen bezeichnet — in Adreß-Signale, die dann zu den Speicherbausteinen führen.
Die Segmentnummer ist eine vorzeichenlose 7 Bit lange Zahl von 0 bis 127, der Offset ist eine vorzeichenlose 16 Bit lange Zahl von 0 bis 64 K.
Wird eine segmentierte Adresse in einem Register dargestellt, wird ein Registerpaar verwendet. Sowohl die Segmentnummer als auch der Offset können getrennt oder gemeinsam mit

allen zur Verfügung stehenden Befehlen bearbeitet werden. Steht eine solche Adresse im Speicher, so belegt sie ein Doppelwort (Bild 9).

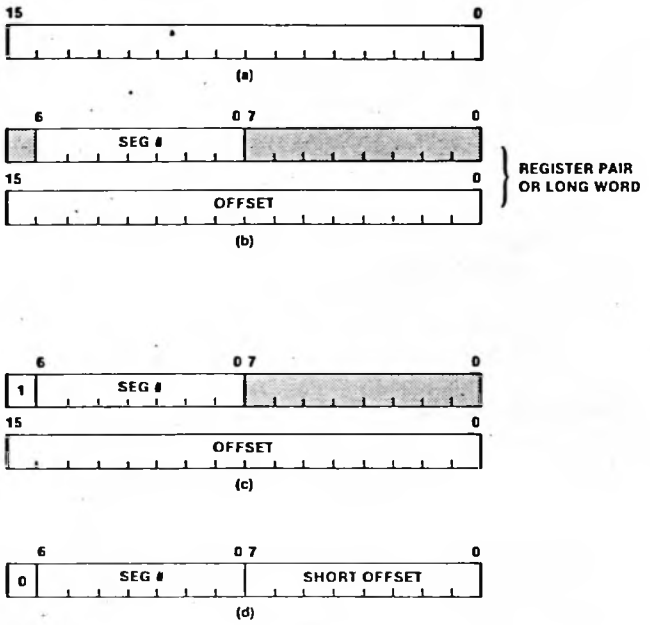


Bild 9
Hardwaremäßige Adreßdarstellung in einem Register oder Speicher.

In einem Befehl bestehen für die segmentierte Adresse zwei Darstellungsmöglichkeiten:
Langer Offset und kurzer Offset.
Adressen mit langem (16 Bit) Offset belegen zwei Worte, mit kurzem (8 Bit) Offset ein Wort. Bei dem kurzen Offset sind die oberen 8 Bit des normalen (langen) Offset null. Sowohl Programmkode-Länge wie auch Verarbeitungsgeschwindigkeit werden hierdurch positiv beeinflußt.
Der Baustein zur Speicherverwaltung („Memory Manager", MMU) hat zwei wesentliche Funktionen:
1. Segment Zuordnung durch die Übersetzung von Logischen in physikalische Adressen.
2. Speicherschutz

Adressen, die der Programmierer verwendet, die in den Befehlen enthalten sind und die von der Z8001 ausgegeben werden, bezeichnet man als logische Adressen. Der Speicherverwalter MMU benutzt diese logischen Adressen (7 Bit und 16 Bit) und erzeugt daraus 24 Bit physikalische Adressen, (Bild 10, 11).

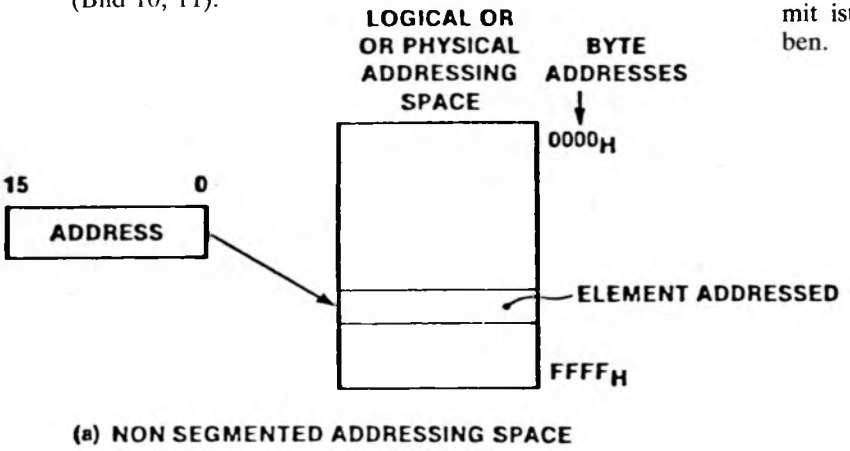


Bild 10
Die von der CPU ausgehenden Adreßleitungen sind direkt zum Speicher geführt. Logische Adressen sind hier identisch mit den Hardwareadressen.

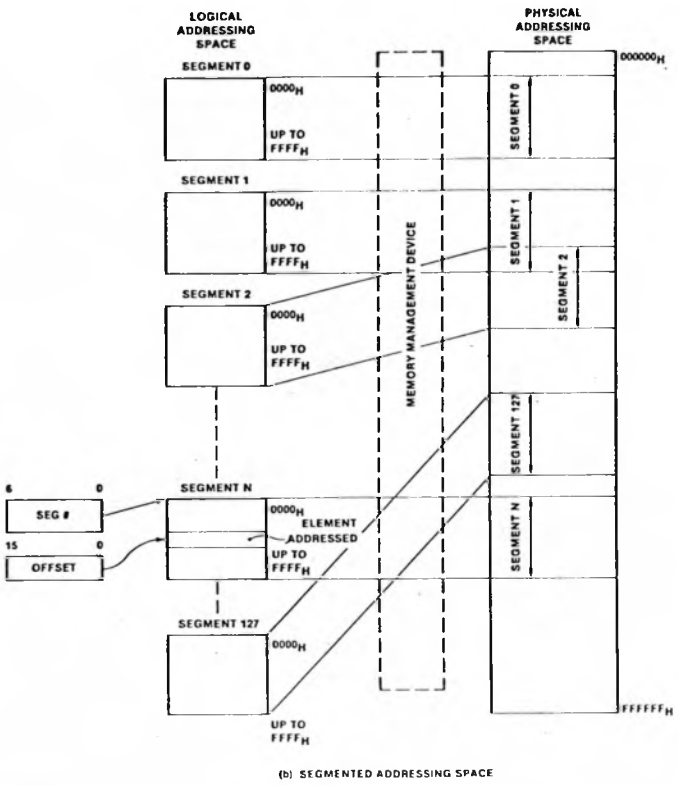


Bild 11
Der Speicherverwalter MMU (Memory Management Unit) übersetzt die logischen Adressen in physikalische (=Hardware)-Adressen.

Diese Umwandlung wird „Relokation" genannt.
Eine 24 Bit Basisadresse ist jedem Segment zugeordnet. Diese Adresse wird über Software in den Speicherverwalter geladen. Die Hardware Adresse wird dann durch eine Addition des 16 Bit Offset und der Basisadresse gebildet (Bild 12). Das bedeutet, daß 8 Megabyte innerhalb eines 16 Megabyte Speicherraumes direkt adressiert werden können. Bei jedem Speicherzugriff vergleicht der Speicherverwalter den Zustand der CPU-Statusleitungen mit vorgegebenen Bedingungen. Prüfbar sind: Systemstatus/Anwenderstatus eines Segments, Datenbereich/Programmbereich, Schreib-Lese-Status/ nur Lesestatus von Daten, unerlaubter Segmentzugriff und Segmentgröße. Wird eine vorgegebene Bedingung nicht eingehalten, wird ein Segmentierungstrap erzeugt. Hiermit ist eine wirkungsvolle Speicherschutzmöglichkeit gegeben.

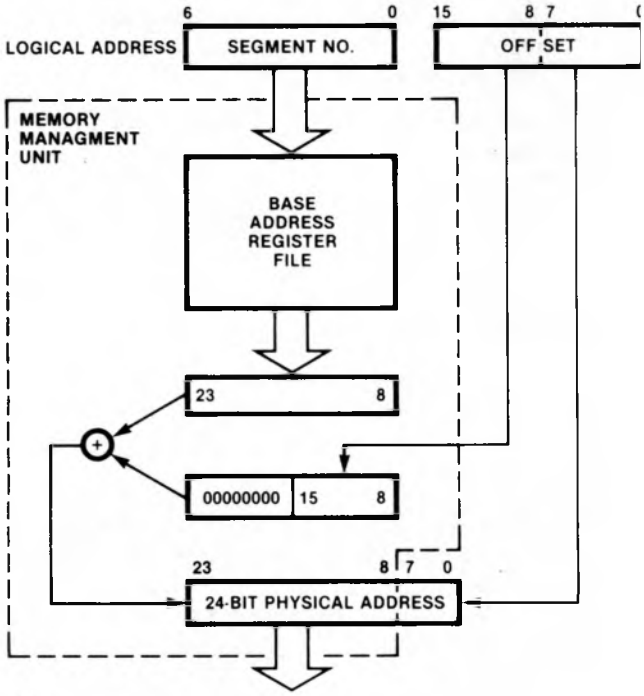


Bild 12
Funktionsablauf der Adreßbildung durch den Speicherverwalter.

Der Speicherverwalter MMU ist ein 48 Bit Baustein. Die Eingänge sind die Segmentnummern, die höherwertigen acht Bit des Offset und die Statusinformationen der CPU. Die Ausgänge sind die oberen 16 der 24 Hardware-Adreßleitungen sowie die Segmentierungstrap-Leitung. Die restlichen 8 Adreßleitungen führen direkt von der CPU zum Speicher (Bild 12). Der Speicherverwalter arbeitet bei jedem Speicherzugriff. Werden Informationen geladen, so wird er von der CPU als Peripherie-Baustein angesehen. Die anzuwendenden Befehle sind spezielle E/A-Befehle, die die oberen acht Bit des Datenbus verwenden. Daher benötigt der Verwalterbaustein als zusätzliche Leitungen „Chip Select", „Schreiben/Lesen" sowie „Adressen aktiv" und „Daten aktiv".

2.1 Einfluß der Segmentierung auf die Adressierung

Die Z8002 kennt nur das Wort als Adreßdarstellung. In der Z8001 hingegen sind mehrere Formen möglich. In einem Befehl ist es bei Kurz-Offset ein Wort und bei vollem Offset ein Doppelwort. In Registern wird grundsätzlich ein Doppelwort verwendet. Für alle Adressierungsarten außer „Adresse direkt" und „Indiziert" ist das Befehlsformat für beide Versionen gleich, da sie keine Adresse enthalten. Die Adressierungsarten „Register" und „Indiziert" erfordern ein 16 Bit Register in beiden Fällen. Anders die Arten „Register indirekt", „Basis Adresse" und „Basis indiziert". Es wird ein Doppelwort bei der segmentierten und ein einfaches bei der nichtsegmentierten Version benötigt. Das hat zur Folge, daß die Arten „Basis Adresse" und „Indiziert" in der nicht segmentierten Version den gleichen Bereich abdecken. Bei Segmentierung unterscheidet er sich natürlich erheblich. Es ist wichtig zu beachten, daß bei der indizierten Adressierung die Adressen bei der Assemblierung festgelegt werden, der Offset kann hingegen während des Programmlaufes variiert werden. Bei der Adressierung „Basis Adresse" gilt das umgekehrt.
Der Programmstatus (PS), die Stackpointer (NSP, SSP) und der Zeiger auf den Bereich neuer Statusinformationen (NPSAP) sind in den beiden Versionen unterschiedlich (Bild 6, 7, 8).

3. Befehlsvorrat

Der Befehlsvorrat der Z8001 und Z8002 umfaßt mehr als 110 Befehlstypen. Unter Anwendung der Adressierungsarten ergeben sich daraus über 410 sinnvolle Kombinationen mit den daraus resultierenden verschiedenen Operationscodes. Wie bereits erwähnt ergeben sich Befehlslängen von einem bis zu fünf Worten. Umfangreiche statistische Untersuchungen von Programmen (Anwenderprogramme, Systemprogramme, Compiler und Compilergenerierte Programme) führten dazu, daß insbesondere häufig wiederkehrende Befehle als Ein-Wort-Befehle implementiert wurden.

Im Folgenden wird ein Überblick über die Befehlstypen gegeben:

1. Transfer
   Löschen, austauschen, Register-Register, Speicher-Register, Register-Speicher, POP, PUSH.
2. Arithmetische Befehle
   Addieren, subtrahieren, Dezimalkorrektur, Vergleich, multiplizieren, dividieren, increment (1-16), Decrement (1-16).
3. Logische Befehle:
   Negation, UND, ODER, komplement, EXKLUSIV-ODER, Test
4. Programmsteuerung
   Unterprogrammaufruf, Vergleich und Sprung relativ, decrement und Sprung bei $\neq 0$, bedingter Sprung, Systemaufruf, Rücksprung von Interrupt, Rücksprung bedingt.
5. Bitbefehle
   testen, setzen, rücksetzen, testen und setzen
6. Rotieren und Schieben
   Rotieren links und rechts: Digit, mit Carry, ohne Carry, eine Stelle, zwei Stellen.
   Schieben links oder rechts: logisch, arithmetisch, Anzahl bis zur Datenelementgröße; dynamisch, rechts oder links je nach Vorzeichen 31 bis -32 Stellen.
7. Block- und Sequenzbefehle
   Vergleichen-Decrement (Increment), Vergleichen — Decrement (Increment) mit Wiederholung, wie oben für Sequenzen, Laden-Decrement (Increment), Laden Decrement (Increment) mit Wiederholung.
8. Ein-/Ausgabebefehle
   Eingabe, Eingabe-Decrement (Increment), Eingabe-Decrement (Increment)-wiederhole;
   Ausgabe, Ausgabe-Decrement (Increment), Ausgabe-Decrement (Increment)-wiederhole;
   Ein- und Ausgabe für Bytes und Worte
9. CPU Steuerung
   Flagbeeinflussung, Interruptmaskierung, Multiprozessorsteuerung, Halt, No Operation.

4. Ausführungszeiten

Die Z8000 verwendet eine Taktfrequenz von 4 MHz. Eine Taktperiode beträgt daher 250 ns. Jeder Befehl benötigt mehrere Maschinenzyklen, wobei ein Maschinenzyklus aus mindestens drei Taktperioden besteht. Die meisten Zyklen enthalten einen Speicherzugriff, der drei Taktperioden benötigt, und weitere Perioden für die Verarbeitung. Bei einigen Befehlen wird die Verarbeitungszeit durch ein Überlappen der Befehlsabarbeitung und dem Lesen und Dekodieren des nächsten Befehles verkürzt.

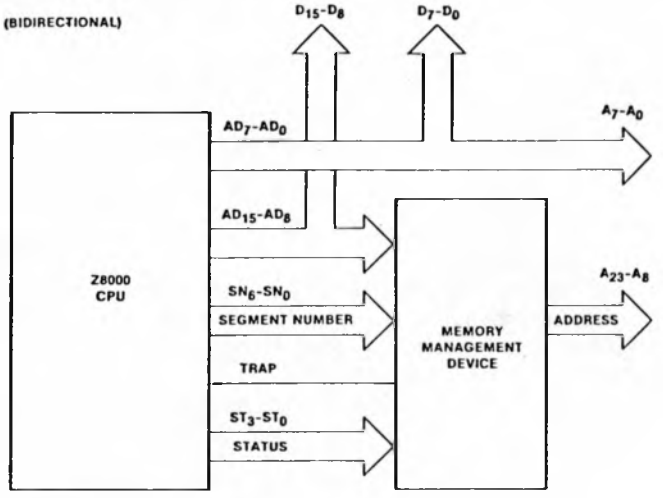


Bild 13
Anschluß des Speicherverwalters an die Z8000 CPU.

(Bild 12). Der Speicherverwalter arbeitet bei jedem Speicherzugriff. Werden Informationen geladen, so wird er von der CPU als Peripherie-Baustein angesehen. Die anzuwendenden Befehle sind spezielle E/A-Befehle, die die oberen acht Bit des Datenbus verwenden. Daher benötigt der Verwalterbaustein als zusätzlich Leitungen „Chip Select", „Schreiben/Lesen" sowie „Adressen aktiv" und „Daten aktiv".

Dabei bedeuten:

Im Operanden:	R	Register
	Q	Quelle
	Z	Ziel
	CC	Bedingungswort
Im Befehl:	W	Wort
	L	Doppelwort
	B	Byte
Segmentierungsart:	NS	nicht segmentiert
	SO	segmentiert kurzer Offset
	FO	segmentiert langer Offset

5. Anschlußbelegung (Bild 14)

AD_0 — AD_{15}	Adreß-/Datenleitungen
AS	Adreß-Strobe, steigende Flanke zeigt Gültigkeit der Adressen an (aktiv-Low-Ausgang)
BUSAK	CPU-Bus ist hochohmig (aktiv-LOW-Ausgang)
BUSRQ	Busanforderung (aktiv-LOW-Eingang)
DS	Daten-Strobe, zeigt Gültigkeit der Daten an (3-State-aktiv-LOW-Ausgang)
MREQ	Gültig wenn auf Adreß/Daten-Bus Speicheradresse anliegt (3-State-aktiv-LOW-Ausgang)
M_I	Daisy Chain-Eingang für Multiprocessing (Eingang, aktiv-LOW)
M_O	Daisy Chain für Multiprocessing (Ausgang, aktiv-LOW)
NMI	Nicht maskierbarer Interrupt, durch fallende Flanke ausgelöst (flankengetriggerter, LOW-aktiver Eingang)
NVI	Nicht vektorisierter Interrupt (Eingang, aktiv-LOW)
CLK	System Takt (+5 V Einphasentakt)
RESET	Rücksetzen der CPU (Eingang, aktiv-LOW)
R/W	Schreib-/Lese-Leitung (3-State-Ausgang; LOW bedeutet Schreibvorgang)
SA_0 — SA_6	Segmentnummern (Ausgänge, aktiv-HIGH)
ST_0 — ST_3	Statusinformationen (Ausgänge, altiv-HIGH)
STOP	Signal zur Einzelbefehl-Verarbeitung (Eingang, aktiv-LOW)
SEGT	Segmentierungstrapsignal der MMU an die CPU (Eingang, aktiv-LOW)
VI	Vektorisierte Interruptanforderung (Eingang, aktiv-LOW)
WAIT	Speicher, E/A nicht bereit für Datenübertragung (Eingang, aktiv-LOW)
B/W	Byte/Wort-Zugriffssignal (gibt Auskunft über Art des Speicherzugriffs; 3-State-Ausgang; LOW = wortweiser Zugriff)
N/S	Normal-/Anwender-Betriebsart (3-State-Ausgang; LOW = „System-Mode“)

In der nichtsegmentierten Version (Z8002) entfallen die Anschlüsse SA_0 — SA_6 und SEGT.

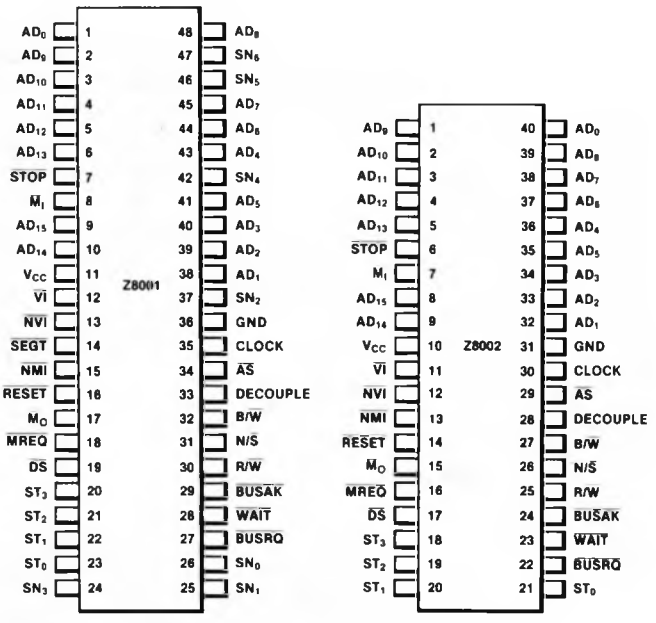


Bild 14
Anschlußbelegung der Z8001 und Z8002

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CLR **CLRB**	dst	R	7	–	–				**Clear** dst ← 0
		IR	8	–	–				
		DA	11	12	14				
		X	11	12	15				
EX **EXB**	R, src	R	6	–	–				**Exchange** R ↔ src
		IR	12	–	–				
		DA	15	16	18				
		X	16	16	19				
LD **LDB** **LDL**	R, src	R	3	–	–	5	–	–	**Load into Register** R ← src
		IM	7	–	–	11	–	–	
		IM	5 (byte only)						
		IR	7	–	–	11	–	–	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	–	–	17	–	–	
		BX	14	–	–	17	–	–	
LD **LDB** **LDL**	dst, R	IR	8	–	–	11	–	–	**Load into Memory** (Store) dst ← R
		DA	11	12	14	14	15	17	
		X	12	12	15	15	15	18	
		BA	14	–	–	17	–	–	
		BX	14	–	–	17	–	–	
LD **LDB**	dst, IM	IR	11	–	–				**Load Immediate into Memory** dst ← IM
		DA	14	15	17				
		X	15	15	18				
LDA	R, src	DA	12	13	15				**Load Address** R ← source address
		X	13	13	16				
		BA	15	–	–				
		BX	15	–	–				
LDAR	R, src	RA	15	–	–				**Load Address Relative** R ← source address
LDK	R, src	IM	5	–	–				**Load Constant** R ← n (n = 0 ... 15)
LDM	R, src, n	IR	11	–	–				**Load Multiple** R ← src (n consecutive words) (n = 1 ... 16)
		DA	14	15	17	+3n			
		X	15	15	18				
LDM	dst, R, n	IR	11	–	–				**Load Multiple** (Store Multiple) dst ← R (n consecutive words) (n = 1 ... 16)
		DA	14	15	17	+3n			
		X	15	15	18				
LDR **LDRB** **LDRL**	R, src	RA	14	–	–	17	–	–	**Load Relative** R ← src (range -32768 ... +32767)
LDR **LDRB** **LDRL**	dst, R	RA	14	–	–	17	–	–	**Load Relative** (Store Relative) dst ← R (range -32768 ... +32767)
POP **POPL**	dst, IR	R	8	–	–	12	–	–	**Pop** dst ← IR Autoincrement contents of R
		IR	12	–	–	19	–	–	
		DA	15	16	18	22	23	25	
		X	16	16	19	23	23	26	
PUSH **PUSHL**	IR, src	R	9	–	–	12	–	–	**Push** Autodecrement contents of R IR ← src
		IM	12	–	–	–	–	–	
		IR	13	–	–	20	–	–	
		DA	13	14	16	20	21	23	
		X	14	14	17	21	21	24	

*** NS = Non-Segmented SS = Segmented Short Offset SL = Segmented Long Offset**

Arithmetic

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
ADC **ADCB**	R, src	R	5	-	-				**Add with Carry** R ← R + src + carry
ADD **ADDB** **ADDL**	R, src	R	4	-	-	8	-	-	**Add** R ← R + src
		IM	7	-	-	14	-	-	
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP **CPB** **CPL**	R, src	R	4	-	-	8	-	-	**Compare with Register** R - src
		IM	7	-	-	14	-	-	
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP **CPB**	dst, IM	IR	11	-	-				**Compare with Immediate** dst - IM
		DA	14	15	17				
		X	15	15	18				
DAB	dst	R	5	-	-				**Decimal Adjust**
DEC **DECB**	dst, n	R	4	-	-				**Decrement by n** dst ← dst - n (n = 1 ... 16)
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
DIV **DIVL**	R, src	R	95	-	-	723	-	-	**Divide** (signed) Word: $R_{n+1} \leftarrow R_{n,n+1} \div src$ $R_n \leftarrow$ remainder Long Word: $R_{n+2,n+3} \leftarrow R_{n\ldots n+3} \div src$ $R_{n,n+1} \leftarrow$ remainder
		IM	95	-	-	723	-	-	
		IR	95	-	-	723	-	-	
		DA	96	97	99	724	725	727	
		X	97	97	100	725	725	728	
EXTS **EXTSB** **EXTSL**	dst	R	11	-	-	11	-	-	**Extend Sign** Extend sign of low order half of dst through high order half of dst
INC **INCB**	dst, n	R	4	-	-				**Increment by n** dst ← dst + n (n = 1 ... 16)
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
MULT **MULTL**	R, src	R	70	-	-	282*	-	-	**Multiply** (signed) Word: $R_{n,n+1} \leftarrow R_{n+1} \cdot src$ Long Word: $R_{n\ldots n+3} \leftarrow R_{n+2,\,n+3} \cdot src$ *Plus seven cycles for each 1 in the multiplicand
		IM	70	-	-	282*	-	-	
		IR	70	-	-	282*	-	-	
		DA	71	72	74	283*	284*	286*	
		X	72	72	75	284*	284*	287*	
NEG **NEGB**	dst	R	7	-	-				**Negate** dst ← 0 - dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
SBC **SBCB**	R, src	R	5	-	-				**Subtract with Carry** R ← R - src - carry
SUB **SUBB** **SUBL**	R, src	R	4	-	-	8	-	-	**Subtract** R ← R - src
		IM	7	-	-	14	-	-	
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

Logical

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
AND **ANDB**	R, src	R	4	-	-				**AND** R ← R AND src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
COM **COMB**	dst	R	7	-	-				**Complement** dst ← NOT dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
OR **ORB**	R, src	R	4	-	-				**OR** R ← R OR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
TEST **TESTB** **TESTL**	dst	R	7	-	-	13	-	-	**Test** dst OR 0
		IR	8	-	-	13	-	-	
		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
TCC **TCCB**	cc, dst	R	5	-	-				**Test Condition Code** Set LSB if cc is true
XOR **XORB**	R, src	R	4	-	-				**Exclusive OR** R ← R XOR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				

Program Control

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CALL	dst	IR	10	-	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
		DA	12	18	20				
		X	13	18	21				
CALR	dst	RA	10	-	15				**Call Relative** Autodecrement SP @ SP ← PC PC ← PC + dst (range -4094 to +4096)
DJNZ **DBJNZ**	R, dst	RA	11	-	-				**Decrement and Jump if Non-Zero** R ← R - 1 If R ≠ 0: PC ← PC + dst (range -254 to 0)
IRET*	-	-	13	-	16				**Interrupt Return** PS ← @ SP Autoincrement SP
JP	cc, dst	IR	10	-	15	(taken)			**Jump Conditional** If cc is true: PC ← dst
		IR	7	-	7	(not taken)			
		DA	7	8	10				
		X	8	8	11				
JR	cc, dst	RA	6	-	-				**Jump Conditional Relative** If cc is true: PC ← PC + dst (range -256 to +254)
RET	cc	-	10	-	13	(taken)			**Return Conditional** If cc is true: PC ← @ SP Autoincrement SP
			7	-	7	(not taken)			
SC	src	IM	33	-	39				**System Call** Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS

Bit Manipulation

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
BIT **BITB**	dst, b	R IR DA X	4 8 10 11	- - 11 11	- - 13 14				**Test Bit Static** Z flag ← NOT dst bit specified by b
BIT **BITB**	dst, R	R	10	-	-				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R
RES **RESB**	dst, b	R IR DA X	4 11 13 14	- - 14 14	- - 16 17				**Reset Bit Static** Reset dst bit specified by b
RES **RESB**	dst, R	R	10	-	-				**Reset Bit Dynamic** Reset dst bit specified by contents R
SET **SETB**	dst, b	R IR DA X	4 11 13 14	- - 14 14	- - 16 17				**Set Bit Static** Set dst bit specified by b
SET **SETB**	dst, R	R	10	-	-				**Set Bit Dynamic** Set dst bit specified by contents of R
TSET **TSETB**	dst	R IR DA X	7 11 14 15	- - 15 15	- - 17 18				**Test and Set** S flag ← MSB of dst dst ← all 1s

*Privileged instruction. Executed in system mode only.

Rotate and Shift

RLDB	R, src	R	9	-	-	**Rotate Digit Left**
RRDB	R, src	R	9	-	-	**Rotate Digit Right**
RL **RLB**	dst, n	R R	6 for n = 1 7 for n = 2			**Rotate Left** by n bits (n = 1, 2)
RLC **RLCB**	dst, n	R R	6 for n = 1 7 for n = 2			**Rotate Left through Carry** by n bits (n = 1, 2)
RR **RRB**	dst, n	R R	6 for n = 1 7 for n = 2			**Rotate Right** by n bits (n = 1, 2)
RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2			**Rotate Right through Carry** by n bits (n = 1, 2)
SDA **SDAB** **SDAL**	dst, R	R	(15 + 3 n)		(15 + 3 n)	**Shift Dynamic Arithmetic** Shift dst left or right by contents of R
SDL **SDLB** **SDLL**	dst, R	R	(15 + 3 n)		(15 + 3 n)	**Shift Dynamic Logical** Shift dst left or right by contents of R
SLA **SLAB** **SLAL**	dst, n	R	(13 + 3 n)		(13 +3 n)	**Shift Left Arithmetic** by n bits
SLL **SLLB** **SLLL**	dst, n	R	(13 + 3 n)		(13 + 3 n)	**Shift Left Logical** by n bits
SRA **SRAB** **SRAL**	dst, n	R	(13 + 3 n)		(13 + 3 n)	**Shift Right Arithmetic** by n bits
SRL **SRLB** **SRLL**	dst, n	R	(13 + 3 n)		(13 + 3 n)	**Shift Right Logical** by n bits

Block Transfer and String Manipulation

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPD **CPDB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Decrement** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$
CPDR **CPDRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Decrement and Repeat** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPI **CPIB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Increment** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$
CPIR **CPIRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Increment and Repeat** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPSD **CPSDB**	dst, src, R, cc	IR	25	-	-				**Compare String and Decrement** dst - src Autodecrement dst and src addresses R ← R - 1
CPSDR **CPSDRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Decr. and Repeat** dst - src Autodecrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
CPSI **CPSIB**	dst, src, R, cc	IR	25	-	-				**Compare String and Increment** dst - src Autoincrement dst and src addresses R ← R - 1
CPSIR **CPSIRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Incr. and Repeat** dst - src Autoincrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst, src, R	IR	20	-	-				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R - 1
LDDR **LDDRB**	dst, src, R	IR	(11 + 9 n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R - 1 Repeat until R = 0
LDI **LDIB**	dst, src, R	IR	20	-	-				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R - 1
LDIR **LDIRB**	dst, src, R	IR	(11 + 9 n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R - 1 Repeat until R = 0
TRDB	dst, src, R	IR	25	-	-				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R - 1

	Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Block Transfer and String Manipulation (Continued)	**TRDRB**	dst, src, R	IR	(11 + 14 n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R - 1 Repeat until R = 0
	TRIB	dst, src, R	IR	25	-	-				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R - 1
	TRIRB	dst, src, R	IR	(11 + 14 n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R - 1 Repeat until R = 0
	TRTDB	src 1, src 2, R	IR	25	-	-				**Translate and Test, Decrement** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1
	TRTDRB	src 1, src 2, R	IR	(11 + 14 n)						**Translate and Test, Decr. and Repeat** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
	TRTIB	src 1, src 2, R	IR	25						**Translate and Test, Increment** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1
	TRTIRB	src 1, src 2, R	IR	(11 + 14 n)						**Translate and Test, Incr. and Repeat** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
Input/ Output	**IN***	R, src	IR	10	-	-				**Input** R ← src
	INB*		DA	12	-	-				
	IND* **INDB***	dst, src, R	IR	21	-	-				**Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
	INDR* **INDRB***	dst, src, R	IR	(11 + 10 n)						**Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	INI* **INIB***	dst, src, R	IR	21	-	-				**Input and Increment** dst ← src Autoincrement dst address R ← R - 1
	INIR* **INIRB***	dst, src, R	IR	(11 + 10 n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	OUT*	dst, R	IR	10	-	-				**Output** dst ← R
	OUTB*		DA	12	-	-				
	OUTD* **OUTDB***	dst, src, R	IR	21	-	-				**Output and Decrement** dst ← src Autodecrement src address R ← R - 1

Input/ Output (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
OTDR* **OTDRB***	dst, src, R	IR	(11 + 10 n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
OUTI* **OUTIB***	dst, src, R	IR	21	-	-				**Output and Increment** dst ← src Autoincrement src address R ← R - 1
OTIR* **OTIRB***	dst, src, R	IR	(11 + 10 n)						**Output, Increment and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
SIN* **SINB***	R, src	DA	12	-	-				**Special Input** R ← src
SIND* **SINDB***	dst, src, R	IR	21	-	-				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
SINDR* **SINDRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
SINI* **SINIB***	dst, src, R	IR	21	-	-				**Special Input and Increment** dst ← src Autoincrement dst address R ← R - 1
SINIR* **SINIRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
SOUT* **SOUTB***	dst, src	DA	12	-	-				**Special Output** dst ← src
SOUTD* **SOUTDB***	dst, src, R	IR	21	-	-				**Special Output and Decrement** dst ← src Autodecrement src address R ← R - 1
SOTDR* **SOTDRB***	dst, src, R	IR	(11 + 10 n)						**Special Output, Decr. and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
SOUTI* **SOUTIB***	dst, src, R	IR	21	-	-				**Special Output and Increment** dst ← src Autoincrement src address R ← R - 1
SOTIR* **SOTIRB***	dst, src, R	R	(11 + 10 n)						**Special Output, Incr. and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0

*Privileged instructions. Executed in system mode only.

CPU Control

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
COMFLG	flags	–	7	–	–				**Complement Flag** (Any combination of C, Z, S, P/V)
DI*	int	–	6	–	–				**Disable Interrupt** (Any combination of NVI, VI)
EI*	int	–	6	–	–				**Enable Interrupt** (Any combination of NVI, VI)
HALT*	–	–	(8 + 3 n)						**HALT**
LDCTL*	CTLR, src	R	7	–	–				**Load into Control Register** CTLR ← src
LDCTL*	dst, CTLR	R	7	–	–				**Load from Control Register** dst ← CTLR
LDCTLB	FLGR, src	R	7	–	–				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dst, FLGR	R	7	–	–				**Load from Flag Byte Register** dst ← FLGR
LDPS*	src	IR	12	–	16				**Load Program Status** PS ← src
		DA	16	20	22				
		X	17	20	23				
MBIT*	–	–	7	–	–				**Test Multi-Micro Bit** Set S if $\overline{M_I}$ is Low; reset S if $\overline{M_I}$ is High.
MREQ*	dst	R	(12 + 7 n)						**Multi-Micro Request**
MRES*	–	–	5	–	–				**Multi-Micro Reset**
MSET*	–	–	5	–	–				**Multi-Micro Set**
NOP	–	–	7	–	–				**No Operation**
RESFLG	flag	–	7	–	–				**Reset Flag** (Any combination of C, Z, S, P/V)
SETFLG	flag	–	7	–	–				**Set Flag** (Any combination of C, Z, S, P/V)

*Privileged instructions. Executed in system mode only.

Bedingungs-bits

Code	Meaning	Flag Settings	CC Field
	Always false	–	0000
	Always true	–	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No Carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	P/V = 1	0100
NOV	No overflow	P/V = 0	1100
PE	Parity is even	P/V = 1	0100
PO	Parity is odd	P/V = 0	1100
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
LT	Less than (signed)	(S XOR P/V) = 1	0001
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

Status-Informationen

ST_3-ST_0	Definition	ST_3-ST_0	Definition
0 0 0 0	Internal operation	1 0 0 0	Data memory request
0 0 0 1	Memory refresh	1 0 0 1	Stack memory request
0 0 1 0	I/O reference	1 0 1 0	Reserved
0 0 1 1	Special I/O reference (e.g., to an MMU)	1 0 1 1	Reserved
0 1 0 0	Segment trap acknowledge	1 1 0 0	Program reference, nth word
0 1 0 1	Non-maskable interrupt acknowledge	1 1 0 1	Instruction fetch, first word
0 1 1 0	Non-vectored interrupt acknowledge	1 1 1 0	Reserved
0 1 1 1	Vectored interrupt acknowledge	1 1 1 1	Reserved

Absolute Grenzdaten

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

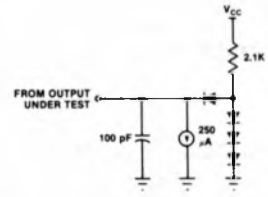
Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test-Bedingungen

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- □ +4.75 V ≤ V_{CC} ≤ +5.25 V
- □ GND = 0 V
- □ 0°C ≤ T_A ≤ +70°C



All ac parameters assume a load capacitance of 100 pF max, except for parameter 6 (50 pF max). Timing references between two output signals assume a load difference of 50 pF max.

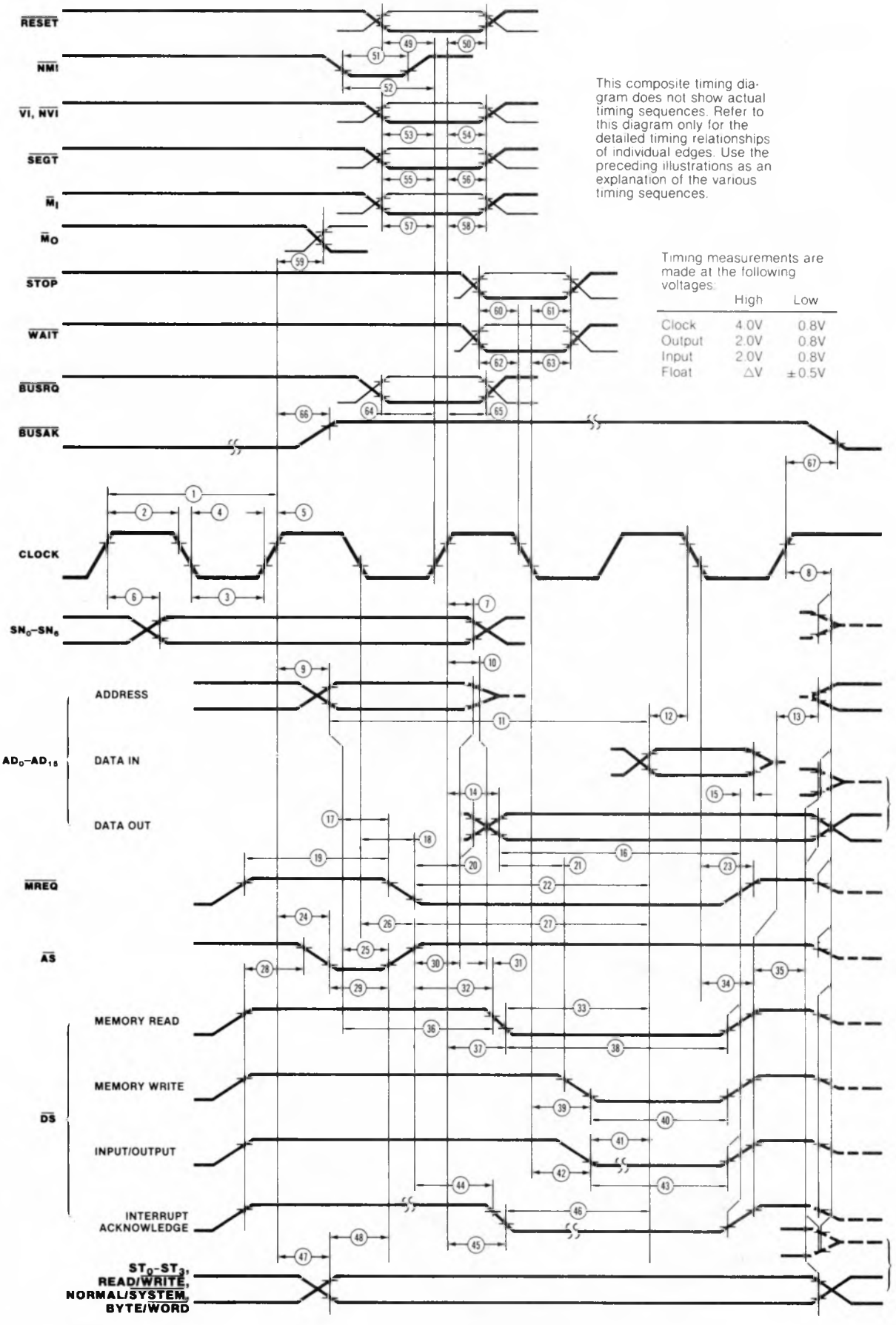
Statische Kenndaten

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{OL}	Output Leakage		±10	µA	0.4 ≤ V_{OUT} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		300	mA	

Bestellbezeichnungen

Part Number	Temperature Range	Number of Pins	Package	Description
Z8001 CPU	0°C to +70°C	48	Ceramic	Segmented 16-Bit Microprocessor
Z8002 CPU	0°C to +70°C	40	Ceramic	Non-Segmented 16-Bit Microprocessor

Übersicht über das Zeitverhalten der Z8001 und Z8002



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages:

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	ΔV	±0.5V

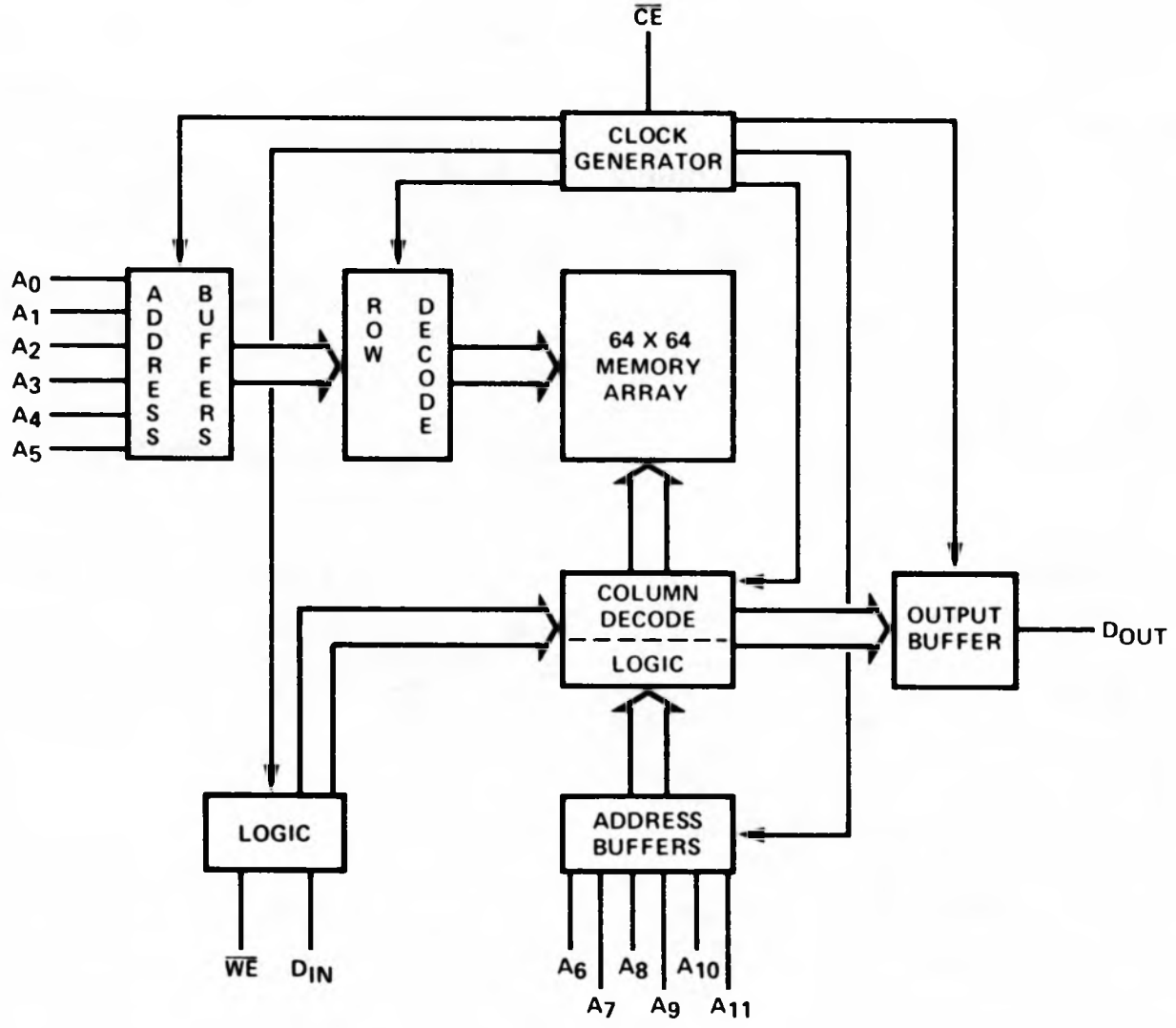
Dynamische Kenndaten (vorläufige Daten)

Number	Symbol	Parameter	Min	Max	Unit
1	TcC	Clock Cycle Time	250	2000	ns
2	TwCh	Clock Width (High)	105	2000	ns
3	TwCl	Clock Width (Low)	105	2000	ns
4	TfC	Clock Fall Time		20	ns
5	TrC	Clock Rise Time		20	ns
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130	ns
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		ns
8	TdC(Bz)	Clock ↑ to Bus Float		65	ns
9	TdC(A)	Clock ↑ to Address Valid		100	ns
10	TdC(Az)	Clock ↑ to Address Float		65	ns
11	TdA(DI)	Address Valid to Data In Required Valid	400		ns
12	TsDI(C)	Data In to Clock ↓ Setup Time	70		ns
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80		ns
14	TdC(DO)	Clock ↑ to Data Out Valid		100	ns
15	ThDI(DS)	Data In to $\overline{DS}$ ↑ Hold Time	0		ns
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	230		ns
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		ns
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80	ns
19	TwMRh	$\overline{MREQ}$ Width (High)	190		ns
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70		ns
21	TdDO(DSW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55		ns
22	TdMR(DI)	$\overline{MREQ}$ ↓ to Data In Required Valid	330		ns
23	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		80	ns
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80	ns
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55		ns
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90	ns
27	TdAS(DI)	$\overline{AS}$ ↑ to Data In Required Valid	290		ns
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		ns
29	TwAS	$\overline{AS}$ Width (Low)	80		ns
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	60		ns
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		ns
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	70		ns
33	TdDSR(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid	155		ns
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70	ns
35	TdDS(DO)	$\overline{DS}$ ↑ to Data Out and STATUS Not Valid	80		ns
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	120		ns
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120	ns
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275		ns
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95	ns
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	160		ns
41	TdDSI(DI)	$\overline{DS}$ (Input) ↓ to Data In Required Valid	315		ns
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120	ns
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	400		ns
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	960		ns
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120	ns
46	TdDSA(DI)	$\overline{DS}$ (Ack.) ↓ to Data In Required Delay	420		ns
47	TdC(S)	Clock ↑ to Status Valid Delay		110	ns
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	40		ns
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		ns
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		ns
51	TwNMI	$\overline{NMI}$ Width (Low)	100		ns
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		ns
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		ns
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	0		ns
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		ns
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		ns
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		ns
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		ns
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		120	ns
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		ns
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		ns
62	TsWT(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	70		ns
63	ThWT(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	0		ns
64	TsBRQ(C)	$\overline{BUSRQ}$ to Clock ↑ Setup Time	90		ns
65	ThBRQ(C)	$\overline{BUSRQ}$ to Clock ↑ Hold Time	0		ns
66	TdC(BAKr)	Clock ↑ to $\overline{BUSAK}$ ↑ Delay		100	ns
67	TdC(BAKf)	Clock ↑ to $\overline{BUSAK}$ ↓ Delay		100	ns

Z6104

Zilog

Statischer Schreib/Lesespeicher Baustein 4096 × 1 Bit



INTERNE BLOCKSCHALTUNG

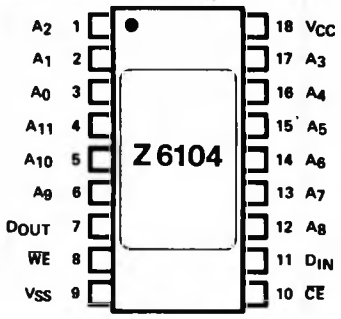
5. ZILOG-SPEICHERBAUSTEINE

- 4096 × 1 Organisation
- Statische Datenspeicherung (kein Refresh nötig)
- Einphasen-Bausteinfreigabetakt-Erzeugung
- Getrennter Daten-Ein- und -Ausgang
- Zugriffs/Zyklus-Zeit / Leistungsverbrauch / Bausteintyp

Zugriffs/Zyklus-Zeit	Leistungsverbrauch	Bausteintyp
150/250 ns	440 mW	Z6104-2
220/340 ns	385 mW	Z6104-3
250/380 ns	220 mW	Z6104-4
300/440 ns	220 mW	Z6104-5
350/510 ns	220 mW	Z6104-6

- Alle Eingänge sind TTL-Pegel-kompatibel
- Stromversorgung über eine einzige + 5 V ± 10%-Quelle
- 18 pin DIP-Gehäuse
- Kein pullup-Widerstand am Ausgang erforderlich
- Depletion load N channel silicon gate-Technologie

Anschlußbelegung



Pin-Bezeichnung

ADRESS EINGÄNGE	$A_0 \ldots A_{11}$
BAUSTEINFREIGABE	$\overline{CE}$
DATENEINGANG	D_{IN}
DATENAUSGANG	D_{OUT}
MASSE	V_{ss}
+ 5 VH	V_{cc}
SCHREIB/LESE-STEUERUNG	$\overline{WE}$

Beschreibung:

Aufgrund seiner fortgeschrittenen Technologie und seinem hohen Integrationsgrad erlaubt der statische 4 k × 1 Schreib/Lese-Speicherbaustein sehr schnellen Zugriff und kurze Zykluszeit bei geringem Leistungsverbrauch.
Seine 12 Adreßeingänge wählen eine Speicherzelle aus der 64 × 64 bit großen Speichermatrix in diesem Baustein aus.
Die Adreßinformation ($A_0 \ldots A_{11}$) wird dem Baustein mit der fallenden Flanke des CE-Signals übergeben.
Dieser Eingang steuert außerdem den internen Datentransfer.
Da es sich beim Z6104 um einen statischen Baustein handelt, bewirkt ein Verlängern des CE-Signals keinen Verlust von Speicherzellen-Inhalten. Bei CE = HIGH ist der Baustein gesperrt, wobei sein Leistungsverbrauch reduziert ist, und der Baustein wird auf den folgenden Zyklus vorbereitet (= „precharge-mode").
Der WE-Eingang bewirkt ein Schreiben der am D_{IN} anliegenden Information in den Baustein, wenn WE LOW ist.
Bei WE = HIGH wird das in der adressierten Speicherstelle des Bausteins stehende Bit über den Ausgang D_{OUT} ausgelesen. Die ausgelesene Information hat die gleiche Phase wie die eingelesene, d.h. die Information wird zwischen Schreib- und Lesevorgang **nicht** invertiert.
Der Ausgang D_{OUT} ist mit max. 2 Standard-TTL- bzw. 8 Low-Power-Schottky-TTL-Eingängen belastbar (V_{OH} = 2,4 V (min) und V_{OL} = 0,4 V (max.)).
Alle übrigen Anschlüsse sind voll TTL-kompatible Eingänge.

☐ Zeitverhalten

Lese-Zyklus:

Die 12 Adreßeingänge wählen eine Zelle der 64 × 64 bit-Speichermatrix an.
Bei $\overline{WE}$ = HIGH wird mit der fallenden Flanke von $\overline{CE}$ der Inhalt der adressierten Zelle auf D_{OUT} angelesen. Dabei geht D_{OUT} nach der Zugriffszeit vom inaktiven, hochohmigen Zustand, sobald $\overline{CE}$ wieder HIGH wird, geht D_{OUT} zurück in den inaktiven, hochohmigen Zustand.

Schreibzyklus:

Die auf dem Dateneingang D_{IN} anstehende Information wird bei der nächsten Flanke von $\overline{WE}$ oder $\overline{CE}$ auf die angesprochene Speicherstelle gebracht. Die Eingangsinformation muß über die erforderlichen Vorbereitungs- und Halte-Zeiten ab der $\overline{WE}$- oder-$\overline{CE}$ Flanke stabil gehalten werden, wobei diese Zeit ab der von beiden zuerst auftretenden Flanke zu rechnen ist.
Beim Schreibvorgang wird D_{OUT} inaktiv, wird jedoch aktiv, sobald $\overline{WE}$ = HIGH wird, wobei der Speicherzelleninhalt an D_{OUT} erscheint, bis auch $\overline{CE}$ wieder HIGH-Signal bekommt.

Lese/Änderungs/Schreib-Zyklus

Diese Betriebsart ist eine Sonderform von Lese- und Schreib-Zyklus.
Dabei wird die Speicherzelle nach der Zugriffszeit ausgelesen und im gleichen Zugriffszyklus ihr Inhalt durch eine Schreib-Operation geändert.
In dieser Betriebsart ist D_{OUT} zunächst inaktiv und wird nach dem Schreibzyklus aktiv. Bei der steigenden Flanke von $\overline{CE}$ wird D_{OUT} wieder inaktiv.

Dynamische Kenndaten

Symbol	Parameter	Z6104-2		Z6104-3		Z6104-4 Z6104L-4		Z6104L-5		Z6104L-6		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
t_{AC}	Read Access Time[1, 2]		150		200		250		300		350	ns
t_C	Read or Write Cycle Time[3]	240		320		380		440		510		ns
t_{CE}	Chip Enable Pulse Width (Low)	150	10^5	200	10^5	250	10^5	300	10^5	350	10^5	ns
t_P	Chip Enable Precharge Time (High)	50		80		90		100		120		ns
t_{AS}	Address Set-up Time	0		0		0		0		0		ns
t_{AH}	Address Hold Time	50		80		100		120		140		ns
t_{CW}	Chip Enable to Write Disable	90		110		140		180		220		ns
t t_{WC}	Write Enable to Chip Disable	40		50		60		80		100		ns
t_{WW}	Write Enable Pulse Width	40		50		60		80		100		ns
t_{DS}	Data Input Set-up Time	60		70		80		100		120		ns
t_{DH}	Data Input Hold Time	0		0		0		0		0		ns
t_{WD}	Write Enable and Disable to Data Out	0	50	0	60	0	80	0	100	0	120	ns
t_{OFF}	Output Buffer Turn-off Delay[4]		40		50		60		80		100	ns
t_T	Transition Time		50		50		50		50		50	ns

1. Output loaded with 100 pF and TTL loading. Output levels are measured at V_{OH} = 2.0 V and V_{OL} = 0.8 V.
2. Typical change in access time vs load capacitance is 0.1 ns/pF.
3. $t_C = t_{CE} + t_P + 2 \cdot t_T$. Test conditions use t_T = 20 ns.
4. Output waveform is dependent on output load. D_{OUT} is guaranteed to be off within t_{OFF}.

Leistungsaufnahme

V_{CC} = 5.5V

Symbol	Parameter		Z6104-2, 3, 4		Z6104L-4, 5, 6		Unit	Condition
		T_A =	0°C	70°C	0°C	70°C		
$I_{CC(SB)}$	Standby Power Supply Current		60	45	40	30	mA	$\overline{CE}$ = High
$I_{CC(SEL)}$	Selected Power Supply Current		85	65	60	45	mA	$\overline{CE}$ = Low
Q	Dynamic Power Supply Charge		2000	2000	2000	2000	pC[1]	

[1] pC = mA • ns

☐ Absolute Grenzdaten

Voltage on any pin relative to V_{SS} . . . . . –0.3V to +7.0V
Storage Temperature (Ambient) . . . . . –65°C to +150°C
Power Dissipation . . . . . 1.5 Watts
Temperature under bias. . . . . Specified operating range

*Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Statische Kenndaten

V_{CC} = +5.0 V ±10%; T_A = 0°C to +70°C

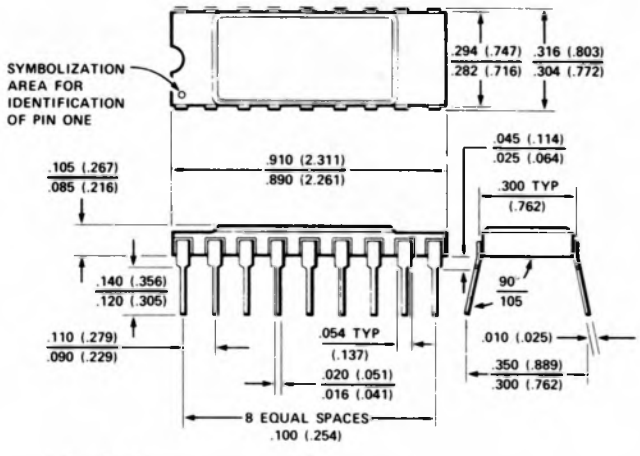
Symbol	Parameter	Min	Max	Units	Conditions
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = –200 µA; $\overline{CE}$ at V_{IL}; $\overline{WE}$ at V_{IH}
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 3.2 mA; $\overline{CE}$ at V_{IL}; $\overline{WE}$ at V_{IH}
V_{IH}	Input High Voltage	+2.0	+6.0	V	
V_{IL}	Input High Voltage	–0.3		V	
I_Z	I/O Leakage Current	–10	+10	µA	$0 < V_{IN} < 5.5$ V

☐ Kapazitäten

(0°C < T_A < +70°C) (V_{CC} = +5.0V ± 10%)

SYMBOL	PARAMETER	MIN	MAX	UNIT
C_{IC}	Input Capacitance ($\overline{CE}$, $\overline{WE}$)		5	pF
C_I	Input Capacitance ($A_0 - A_{11}$, D_{IN})		5	pF
C_O	Output Capacitance		10	pF

☐ Gehäusebauform

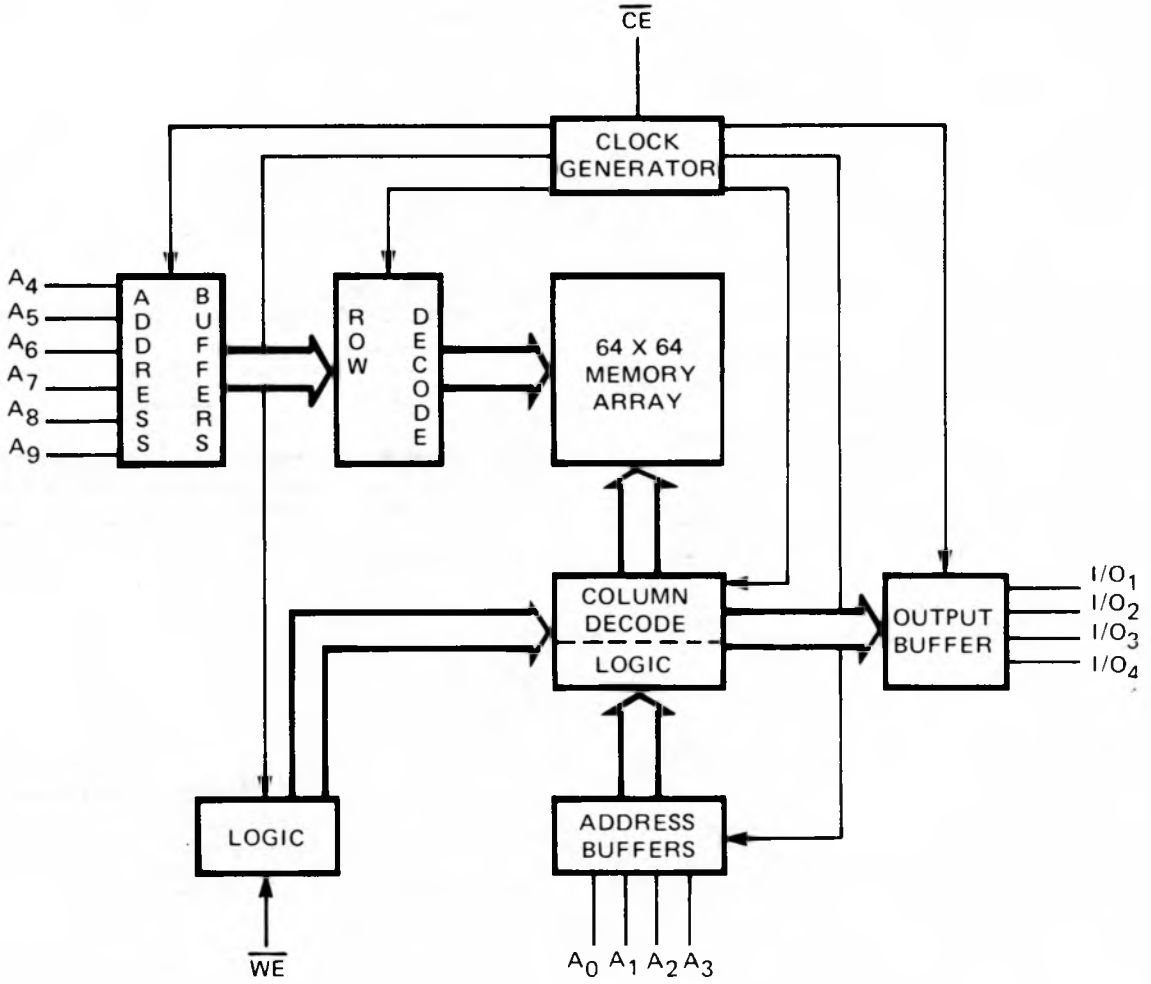


NOTE: Dimensions in parentheses are for metric system (cm).

Z 6114

Zilog

Statischer Schreib/Lesespeicher-Baustein 1024 × 4 Bit



Blockschaltbild

Vorläufige Beschreibung

5. ZILOG-SPEICHERBAUSTEINE

Z6114 1k×4 Bit — Statisches RAM

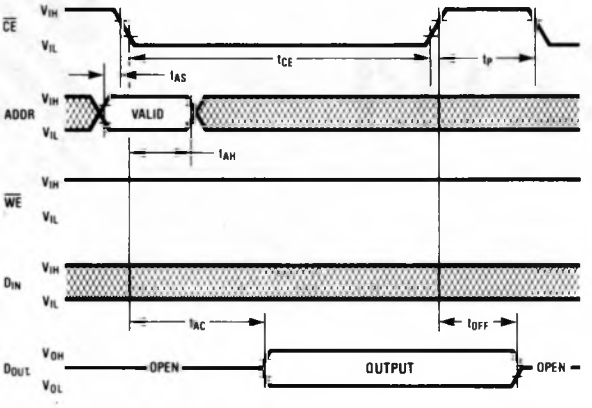
Die Bausteine der Reihe Z6114 sind 1k×4 organisierte Schreib/Lesespeicherbausteine. Durch ihre Organisation und die 4 parallelen Datenanschlüsse (in 8 bit Systemen sind zur Realisierung eines Speicherbereichs nur 2 Bausteine erforderlich) sind sie besonders zum Aufbau kleinerer Mikrocomputersysteme geeignet.

Die Speicherung der Daten ist statisch; die Übernahme der Adreßinformationen erfolgt flankengesteuert und zwar mit der führenden Flanke des Freigabeimpulses (CE).

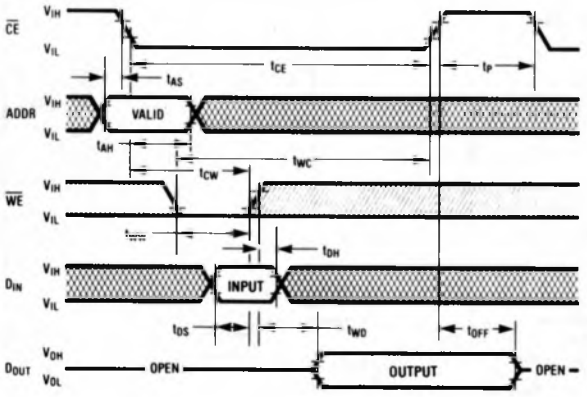
Neben dem Baustein Z6114 ist ein weiteres 1k×4 RAM unter der Bezeichnung Z6142 in Vorbereitung, der mit dem Z6114 funktional identisch ist, jedoch über einen zusätzlichen aktiv-HIGH-arbeitenden Chip-Freigabeeingang und einen Ausgang-Freigabeeingang verfügt.

Beide Bausteine sind in n-Kanal-Depletion-Load Technologie mit Polysilizium-Lasttransistoren aufgebaut.

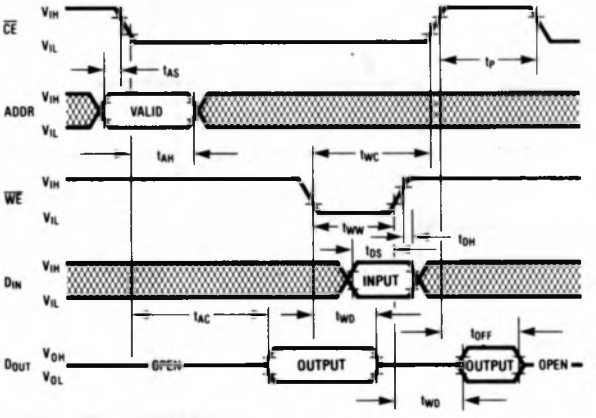
Lesezyklus



Schreibzyklus

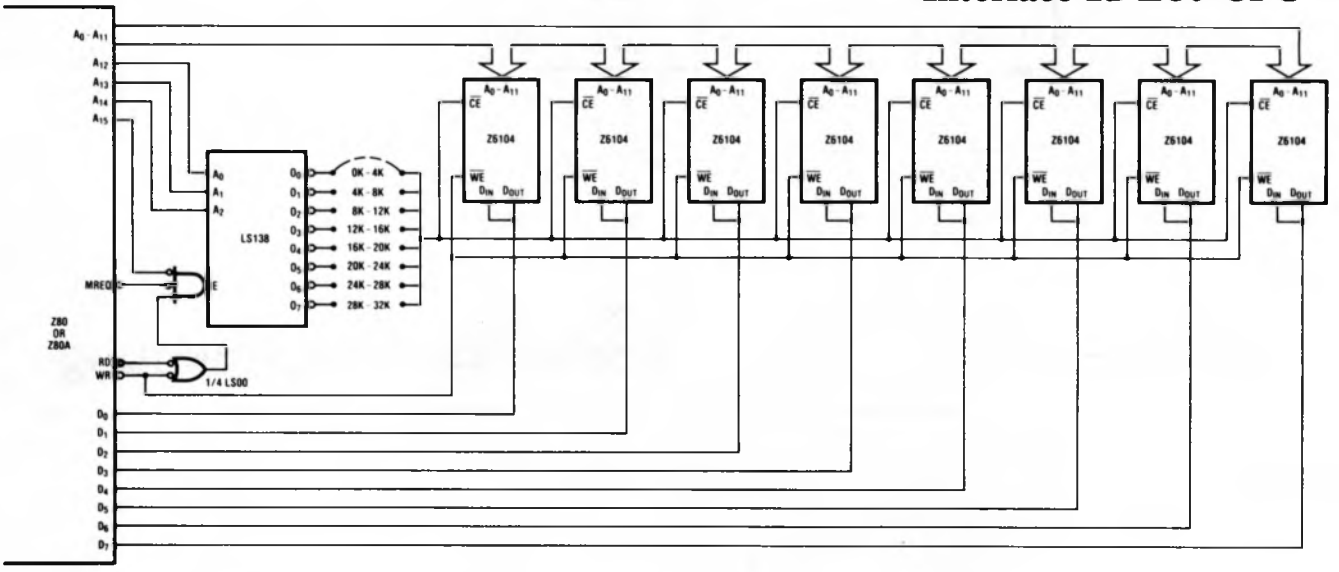


Lese/Schreib-Zyklus

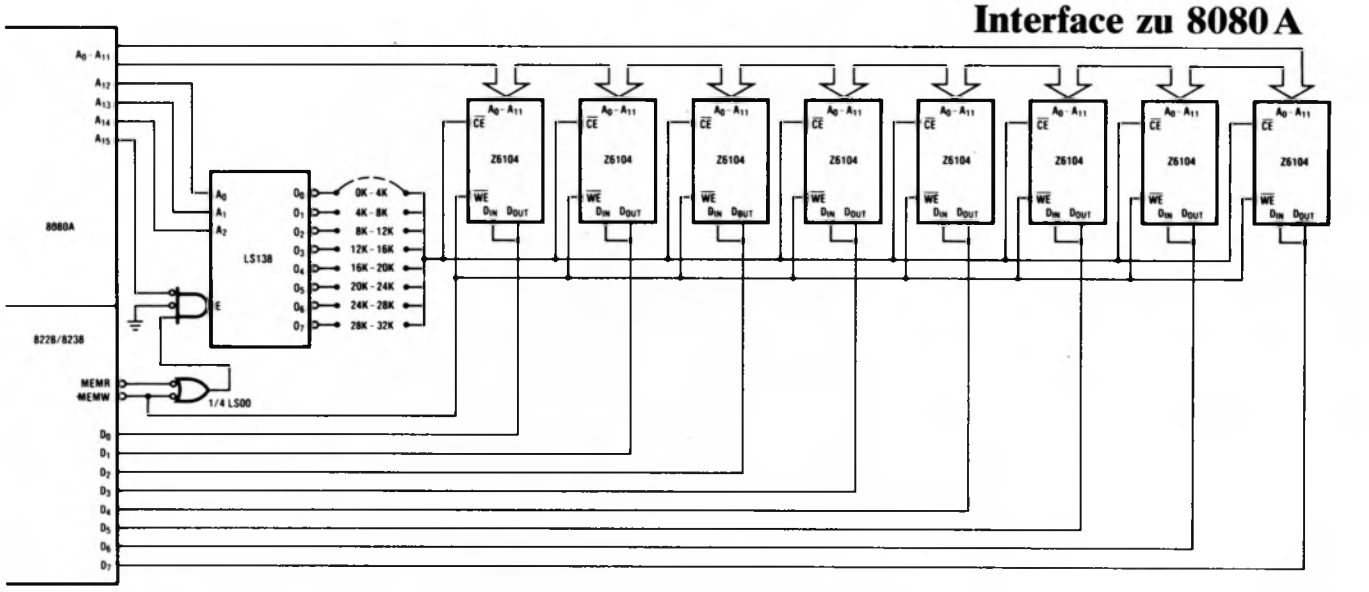


☐ Anschaltung an Mikroprozessoren

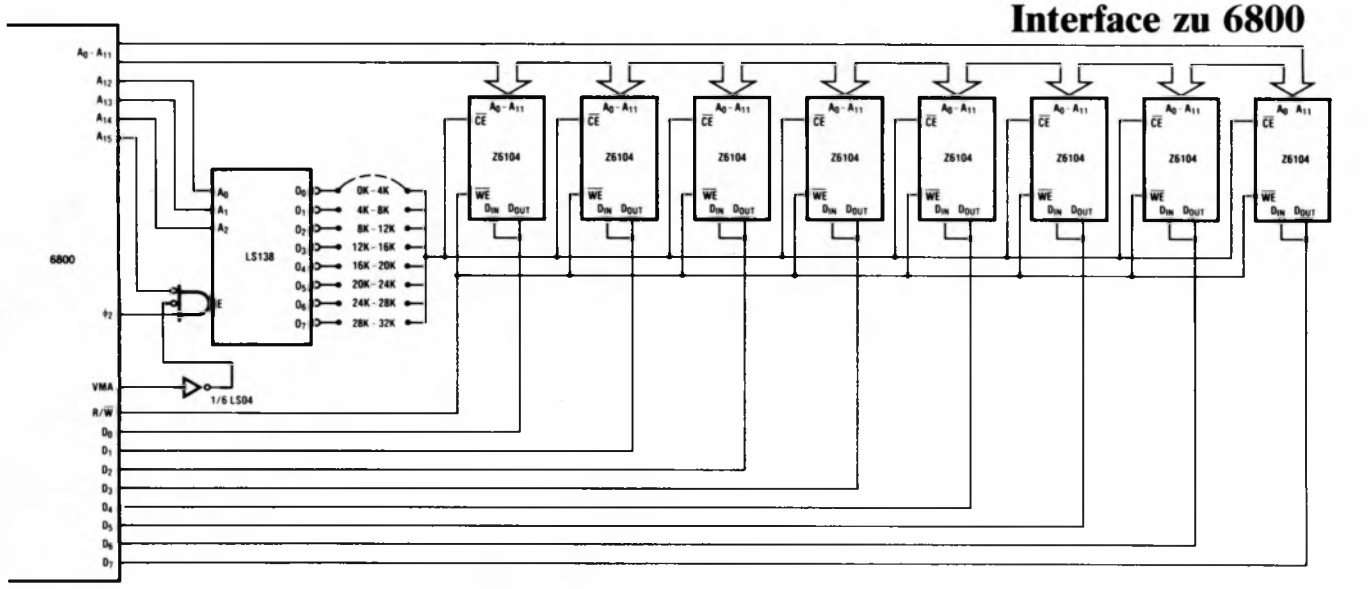
Interface zu Z80-CPU



Interface zu 8080 A



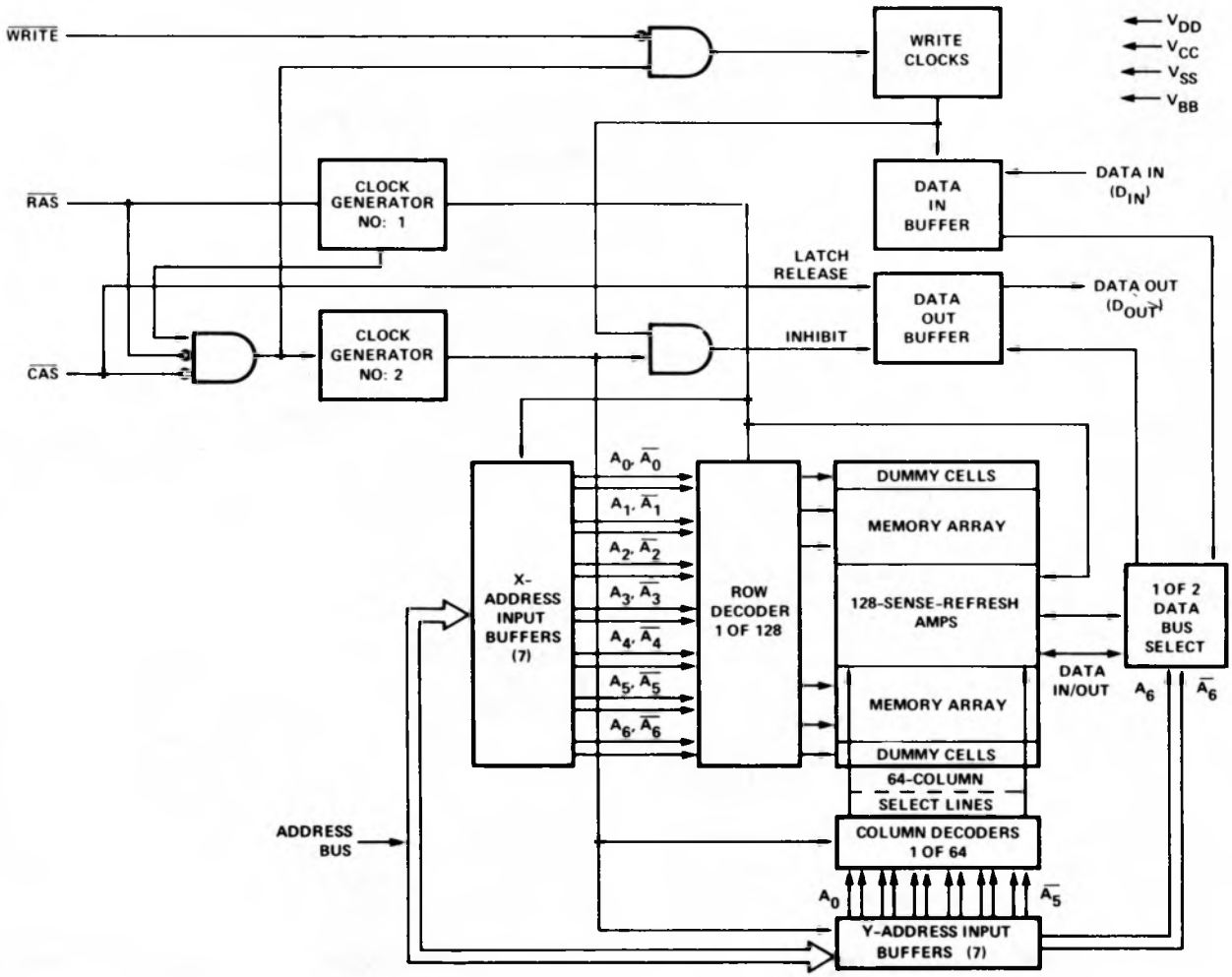
Interface zu 6800



Z6116

Zilog

Dynamischer Schreib/Lesespeicher-Baustein 16384×1 Bit



INTERNE BLOCKSCHALTUNG

5. ZILOG-SPEICHERBAUSTEINE

Eigenschaften

16.384 Worte à 1 bit Organisation, Standard 16 Pin-Gehäuse.

Zugriffszeiten:

150 Nano-Sekunden Zugriffszeit, 375 Nano-Sekunden Zykluszeit (Z6116/2)
200 Nano-Sekunden Zugriffszeit, 375 Nano-Sekunden Zykluszeit (Z6116/3)
250 Nano-Sekunden Zugriffszeit, 410 Nano-Sekunden Zykluszeit (Z6116/4)

- Spannungsversorgung: +12 Volt, ±5 Volt mit ±10% Toleranz auf allen Versorgungsleitungen.
- Niedriger Leistungsverbrauch: 420 mW im aktiven Zustand, 20 mW Standby (maximal)
- Alle Eingänge incl. des Takteingangs TTL kompatibel
- Puffer-Flip-Flop's auf dem Chip für Adreßleitung und Dateneingangsleistung
- Ausgangsdaten werden von $\overline{CAS}$-Signal gesteuert und am Ende eines jeden Zykluses nicht gelatcht
- Gemeinsame Ein/Ausgabe nach dem Early-Write-Verfahren
- Read-Modify-Write, $\overline{RAS}$-only-Refresh und Page-Mode
- Refresh-Periode = 2 msec.
- Gesamt-Refresh in 128 Zyklen
- Betriebstemperatur 0—70°

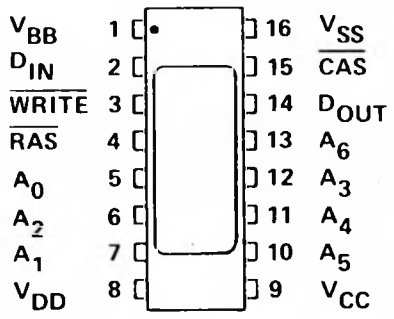
Beschreibung

Die Z6116 dynamische Schreib/Lesespeicher-Bausteinserie umfaßt Bausteine mit 16.384 × 1 bit Wortorganisation in verschiedenen Geschwindigkeitsklassen. Alle Bausteine dieser Serie sind voll pinkompatibel und in einem 16 Pin-Standard DIL-Gehäuse untergebracht. Die verwendete Technologie ist ZILOG's N-Kanal-Ionen-Implantation-Silicongate-Prozeß, mit dem höchste Speicherdichte und Zuverlässigkeit erreicht wird. Neben der hohen Packungsdichte erlaubt dieser Prozeß die Realisierung folgender Bausteineigenschaften:

- ±10% Toleranz auf allen Speisespannungsleitungen
- auf dem Chip integrierte Adreß- und Daten-Puffer-Flip-Flop's machen das externe Zwischenspeichern dieser Information überflüssig
- sämtliche Ein- und Ausgänge sind voll TTL-kompatibel.

Zusätzlich zu den üblichen Lese/Schreib- und Read-Modify-Write-Zyklen können mit den Bausteinen der Z6116-Serie verzögerte Schreibzyklen, Page-Mode-Operationen und Refresh nach dem RAS-Only-Verfahren durchgeführt werden.

Anschlußbelegung

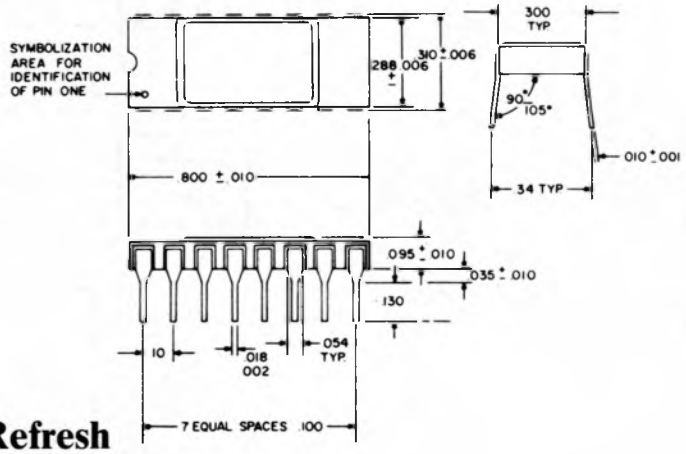


A_0-A_6	ADRESS-EINGÄNGE
$\overline{CAS}$	SPALTEN-ADRESS-FREIGABE
D_{IN}	DATENEINGANG
D_{OUT}	DATENAUSGANG
$\overline{RAS}$	ZEILEN-ADRESS-FREIGABE
$\overline{WRITE}$	SCHREIB/LESE-STEUERUNG
V_{BB}	−5V
V_{CC}	+5V
V_{DD}	+12V
V_{SS}	MASSE

Pin-Belegung

A0 — A6	Adreßleitungen
$\overline{CAS}$	Spalten-Adressenfreigabe (Column Address-Strobe)
D_{in}	Dateneingang (Data in)
D_{out}	Datenausgang (Data out)
$\overline{RAS}$	Zeilen-Adressenfreigabe (Row-Address-Strobe)
WRITE	Schreib/Leseeingang
V_{bb}	minus 5 V Speisespannung
V_{cc}	+5 V Speisespannung
V_{dd}	+12 V Speisespannung
V_{ss}	Masse

Gehäuse-Bauform (hermetisches Keramik-Gehäuse)



Refresh

der dynamischen Speichermatrix erfolgt durch einen Zugriff auf jede der 128 Zeilenadressen innerhalb von 2 msec. Obwohl auch bei jedem normalen Lesezugriff ein „Refreshen" der angesprochenen Speicherzelle erfolgt, wird man das Auffrischen der Information im Allgemeinen mit den vorteilhafteren „RAS-only"-Zyklen durchführen, wodurch eine wesentliche Einsparung an Leistungsverbrauch erzielt wird.

Im Ruhestand (= „standby") darf übrigens der V_{CC}-Anschluß spannungslos sein, ohne daß dabei der Refresh-Vorgang beeinträchtigt wurde.

Anmerkung zum Leistungsverbrauch:

Der Leistungsverbrauch der Z6116-Bausteine ist praktisch proportional zur Arbeitsgeschwindigkeit, da fast seine gesamte interne Schaltung dynamisch aufgebaut ist. Daher kann in Anwendungen, bei denen niedriger Leistungsverbrauch ausschlaggebend ist, dieser durch Herabsetzen der Arbeitsgeschwindigkeit an die jeweiligen Erfordernisse angepaßt werden. Die Bausteine Z6116-2 und 6116-3 können übrigens auch mit Frequenzen >2,66 MHz arbeiten, wenn nur alle Anforderungen an die Zeitverläufe der einzelnen Signale erfüllt sind. Allerdings hat dies einen erhöhten Leistungsverbrauch zu Folge, weswegen eine niedrigere Grenze für die zulässigen Umgebungstemperatur gefordert werden muß.

Obwohl sowohl $\overline{RAS}$ als auch $\overline{CAS}$ im Prinzip dekodiert und als Bausteinfreigabe-Signal für den Z6116 benützt werden können, ergibt sich ein minimaler Leistungsverbrauch bei ausschließlicher Verwendung des $\overline{RAS}$-Signals für diesen Zweck. Alle nicht aktivierten Bausteine (also die, bei denen der RAS-Eingang HIGH ist) bleiben unabhängig vom Zustand des CAS-Signals im Ruhezustand mit minimalem Leistungsverbrauch.

Trotzdem (außer den Toleranzgrenzen der Spannungsversorgungen) keine spezifischen Welligkeitsanforderungen an

die Speisespannungen gestellt sind, sollte eine sorgfältige Ablockung dieser Anschlüsse zur Ausschaltung hochfrequenter Störspannungen vorgenommen werden, die durch interne Schaltvorgänge entstehen können.
Es bestehen keine Vorschriften bezüglich der Reihenfolge, in der die Speisespannungen an den Baustein angebaut werden, soweit nur die statischen Grenzdaten nicht überschritten werden.

Um dies sicherzustellen, empfehlen wir, die Spannung V_{BB} als erste anzulegen und als letzte abzuschalten. V_{BB} sollte nie positiver sein als V_{SS}, solange V_{DD} anliegt.
Nach Anlegen der Speisespannungen braucht der Baustein einige Zyklen zur Aufladung der internen Kapazitäten bzw. Sicherstellung definierter Logikpegel; diese Forderung wird auf jeden Fall durch 8 Refresh-Zyklen erfüllt.

☐ Absolute Grenzdaten*

Voltage on any pin relative to V_{BB}	−0.5 to +20 V
Voltage on V_{DD}, V_{CC} supplies relative to V_{SS}	−1.0 to +15.0V
V_{BB}-V_{SS} (V_{DD}-V_{SS} > 0V)	0V
Operating temperature, TA (Ambient)	0°C to +70°C
Storage temperature (Ambient)	−65°C to +150°C
Short circuit output current	50mA
Power dissipation	1 Watt

*Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Empfohlene Gleichspannungs- Betriebsbedingungen

(0°C ≤ TA ≤ 70°C)[1]

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{DD}	10.8	12.0	13.2	Volts	2
	V_{CC}	4.5	5.0	5.5	Volts	2, 3
	V_{SS}	0	0	0	Volts	2
	V_{BB}	−4.5	−5.0	−5.5	Volts	2
Input High (Logic 1) Voltage all inputs	V_{IH}	2.4	—	7.0	Volts	2
Input Low (Logic 0) Voltage all inputs	V_{IL}	−1.0	—	0.8	Volts	2

DC Electrical Characteristics
(0°C ≤ TA ≤ 70°C)1 (V_{DD} = 12.0V ±10%; V_{CC} = 5.0V ±10% V_{BB} = −5.0 ±10%; V_{SS} = 0V)

PARAMETER	SYMBOL	MIN	MAX	UNITS	NOTES
OPERATING CURRENT	I_{DD1}		35	mA	4
Average power supply operating current	I_{CC1}				5
($\overline{RAS}$, $\overline{CAS}$ cycling; t_{RC} = min. spec.)	I_{BB1}		200	µA	
STANDBY CURRENT	I_{DD2}		1.5	mA	
Power supply standby current ($\overline{RAS}$ = V_{IH}	I_{CC2}	−10	10	µA	
D_{OUT} = High impedance)	I_{BB2}		100	µA	
REFRESH CURRENT	I_{DD3}		27	mA	4
Average power supply current, refresh mode	I_{CC3}	−10	10	µA	
($\overline{RAS}$ cycling, $\overline{CAS}$ = V_{IH} t_{RC} = min. spec.)	I_{BB3}		200	µA	
PAGE MODE CURRENT	I_{DD4}		27	mA	4
Average power supply current,	I_{CC4}				5
page-mode operation ($\overline{RAS}$ =	I_{BB4}		200	µA	
V_{IH}, $\overline{CAS}$ cycling; t_{PC} = min. spec.)					
INPUT LEAKAGE	$I_{I(L)}$	−10	10	µA	
Input leakage current, any input (V_{BB} ±5V, 0V ≤ V_{IN} ≤ +7.0V, all other pins not under test = 0 volts)					
OUTPUT LEAKAGE	$I_{O(L)}$	−10	10	µA	
Output leakage current (D_{OUT} is disabled, 0V ≤ V_{OUT} ≤ +5.5V)					
OUTPUT LEVELS					
Output high (Logic 1) voltage (I_{OUT} = −5mA)	V_{OH}	2.4		Volts	3
Outpt low (Logic 0) voltage (I_{OUT} = 4.2mA)	V_{OL}		0.4	Volts	

NOTES:
1. TA is specified here for operation at frequencies to $t_{RC} \geq t_{RC}$ (min.). Operation at higher cycle rates with reduced ambient temperatures and higher power dissapation is permissible, however, provided AC operating parameters are met.
2. All voltages referenced to V_{SS}.
3. Output voltage will swing from V_{SS} to V_{CC} when activated with no current loading. For purposes of maintaining data in standby mode, V_{CC} may be reduced to V_{SS} without affecting refresh operations or data retention. However, the V_{OH} (min.) specification is not guaranteed in this mode.

Kapazitäten

($0°C \leq TA \leq 70°C$) (V_{DD} = 12.0V ±10%; V_{SS} = 0V; V_{BB} = −5.0V ±10%)

PARAMETER	SYMBOL	TYP	MAX	UNITS	NOTES
Input Capacitance (A_0–A_6), D_{IN}	C_{I1}	4	5	pF	17
Input Capacitance $\overline{RAS}$, $\overline{CAS}$, $\overline{WRITE}$	C_{I2}	8	10	pF	17
Output Capacitance (D_{OUT})	C_O	5	7	pF	17, 18

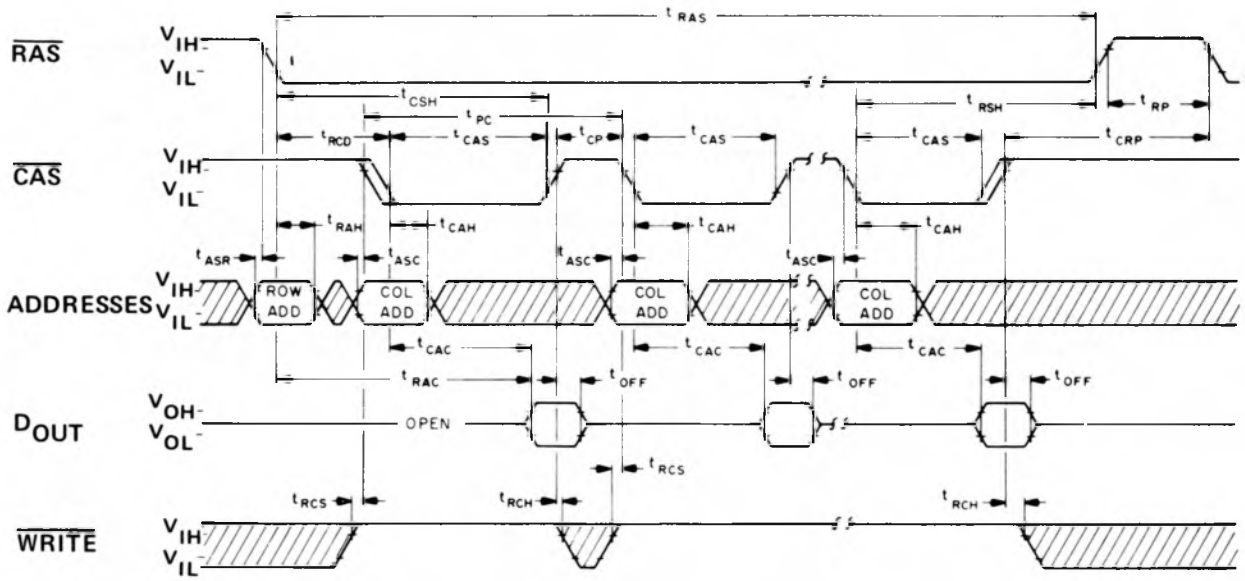
Dynamische Kenndaten und Betriebsbedingungen (s. auch Note 6, 7 u. 8)

($0°C \leq TA \leq 70°C$)[1] (V_{DD} = 12.0V ±10%; V_{CC} = 5.0V ±10%; V_{SS} = 0V, V_{BB} = −5.0V ±10%)

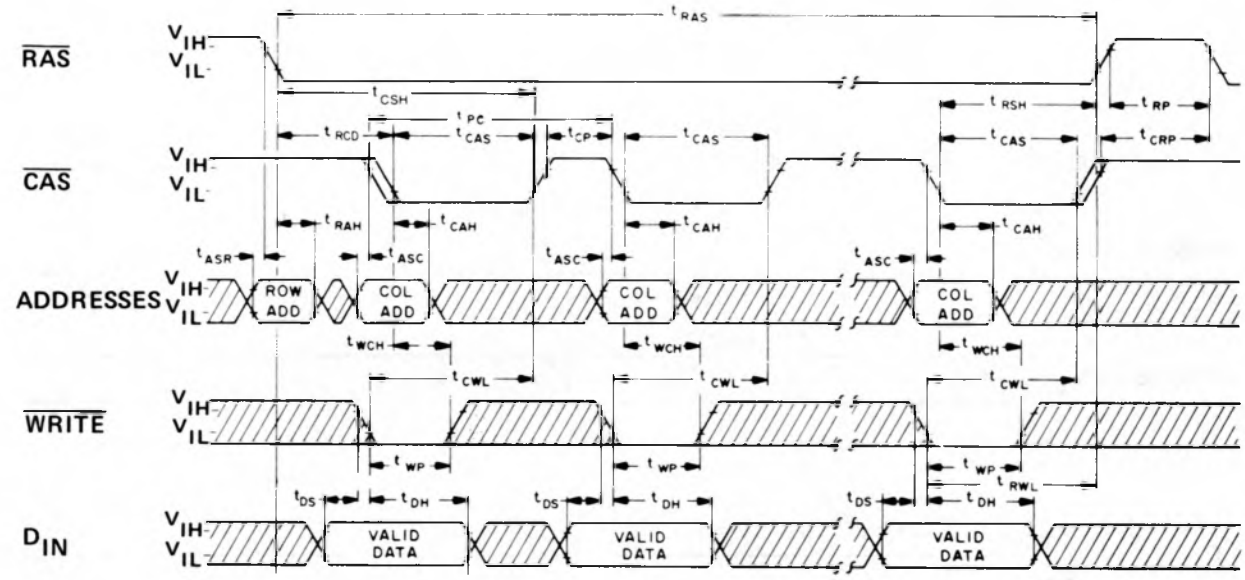
PARAMETER	SYMBOL	Z-6116-2 MIN	Z-6116-2 MAX	Z-6116-3 MIN	Z-6116-3 MAX	Z-6116-4 MIN	Z-6116-4 MAX	UNITS	NOTES
Random read or write cycle time	t_{RC}	375		375		410		ns	9
Read-write cycle time	t_{RWC}	375		375		515		ns	9
Page mode cycle time	t_{PC}	170		225		275		ns	
Access time from $\overline{RAS}$	t_{RAC}		150		200		250	ns	10, 12
Access time from $\overline{CAS}$	t_{CAC}		100		135		165	ns	11, 12
Output buffer turn-off delay	t_{OFF}	0	40	0	50	0	60	ns	13
Transition time (rise and fall)	t_T	3	35	3	50	3	50	ns	8
$\overline{RAS}$ precharge time	t_{RP}	100		120		150		ns	
$\overline{RAS}$ pulse width	t_{RAS}	150	10000	200	10000	250	10000	ns	
$\overline{RAS}$ hold time	t_{RSH}	100		135		165		ns	
$\overline{CAS}$ hold time	t_{CSH}	150		200		250		ns	
$\overline{CAS}$ pulse width	t_{CAS}	100	10000	135	10000	165	10000	ns	
$\overline{RAS}$ to $\overline{CAS}$ delay time	t_{RCD}	20	50	25	65	35	85	ns	14
$\overline{CAS}$ to $\overline{RAS}$ precharge time	t_{CRP}	−20		−20		−20		ns	
Row address set-up time	t_{ASR}	0		0		0		ns	
Row address hold time	t_{RAH}	20		25		35		ns	
Column address set-up time	t_{ASC}	−10		−10		−10		ns	
Column address hold time	t_{CAH}	45		55		75		ns	
Read command set-up time	t_{RCS}	0		0		0		ns	
Read command hold time	t_{RCH}	0		0		0		ns	
Write command hold time	t_{WCH}	45		55		75		ns	
Write command pulse width	t_{WP}	45		55		75		ns	
Write command to $\overline{RAS}$ lead time	t_{RWL}	60		80		100		ns	
Write command to $\overline{CAS}$ lead time	t_{CWL}	60		80		100		ns	
Data in set-up time	t_{DS}	0		0		0		ns	15
Data in hold time	t_{DH}	45		55		75		ns	15
$\overline{CAS}$ precharge time (for page-mode cycle only)	t_{CP}	60		80		100		ns	
Refresh period	t_{REF}		2		2		2	ms	
$\overline{WRITE}$ command set-up time	t_{WCS}	−20		−20		−20		ns	16
$\overline{CAS}$ to $\overline{WRITE}$ delay	t_{CWD}	70		95		125		ns	16
$\overline{RAS}$ to $\overline{WRITE}$ delay	t_{RWD}	120		160		200		ns	16

4. I_{DD1}, I_{DD3} and I_{DD4} depend on cycle rate.
5. I_{CC1} and I_{CC4} depend upon output loading. During readout of high level data V_{CC} is connected through a low impedance (125 Ω typ) to data out. At all other times I_{CC} consists of leakage currents only.
6. Several cycles are required after power-up before proper device operation is achieved. Any 8 cycles which perform refresh are adequate for this purpose.
7. AC measurements assume t_T = 5ns.
8. V_{IH} (min) and V_{IL} (max) are reference levels for measuring timing of input signals. Also, transition times are measured between V_{IH} and V_{IL}.
9. The specifications for t_{RC} (min) and t_{RWC} (min) are used only to indicate cycle time at which proper operation over the full temperature range ($0°C \leq TA \leq 70°C$) is assured.
10. Assumes that $t_{RCD} \leq t_{RCD}$ (max). If t_{RCD} is greater than the maximum recommended value shown in this table, t_{RAC} will increase by the amount that t_{RCD} exceeds the value shown.
11. Assumes that $t_{RCD} \geq t_{RCD}$ (max).
12. Measured with a load equivalent to 2 TTL loads and 100pF.
13. t_{OFF} (max) defines the time at which the output achieves the open circuit condition and is not referenced to output voltage levels.
14. Operation within the t_{RCD} (max) limit insures that t_{RAC} (max) can be met. t_{RCD} (max) is specified as a reference point only; if t_{RCD} is greater than the specified t_{RCD} (max) limit, then access time is controlled exclusively by t_{CAC}.
15. These parameters are referenced to $\overline{CAS}$ leading edge in early write cycles and to $\overline{WRITE}$ leading edge in delayed write or read-modify-write cycles.
16. t_{WCS} and t_{CWD} are not restrictive operating parameters. They are included in the data sheet as electrical characteristics only: If $t_{WCS} \geq t_{WCS}$ (min), the cycle is an early write cycle and the data out pin will remain open circuit (high impedance) throughout the entire cycle; If $t_{CWD} \geq t_{CWD}$ (min), the cycle is a read-write cycle and the data out will contain data read from the selected cell; If neither of the above sets of conditions is satisfied the condition of the data out (at access time) is indeterminate.
17. Effective capacitance calculated from the equation $C = \frac{I \Delta t}{\Delta V}$ with ΔV = 3 volts and power supplies at nominal levels.
18. $\overline{CAS}$ = V_{IH} to disable D_{OUT}.

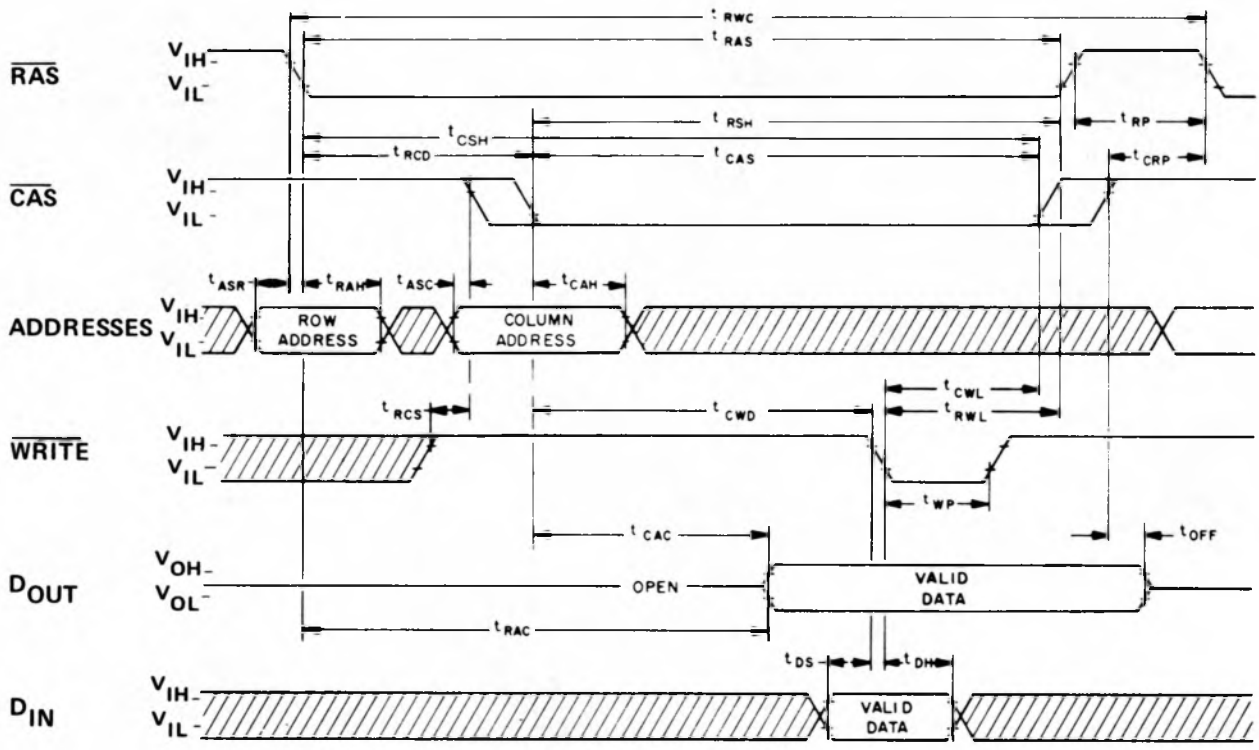
PAGE MODE READ CYCLE



PAGE MODE WRITE CYCLE

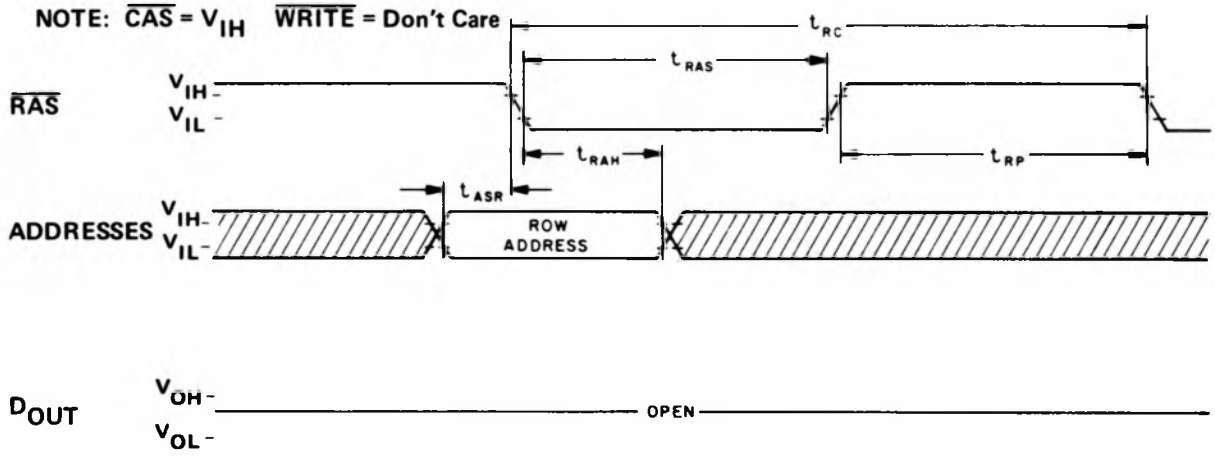


READ-WRITE/READ-MODIFY-WRITE CYCLE

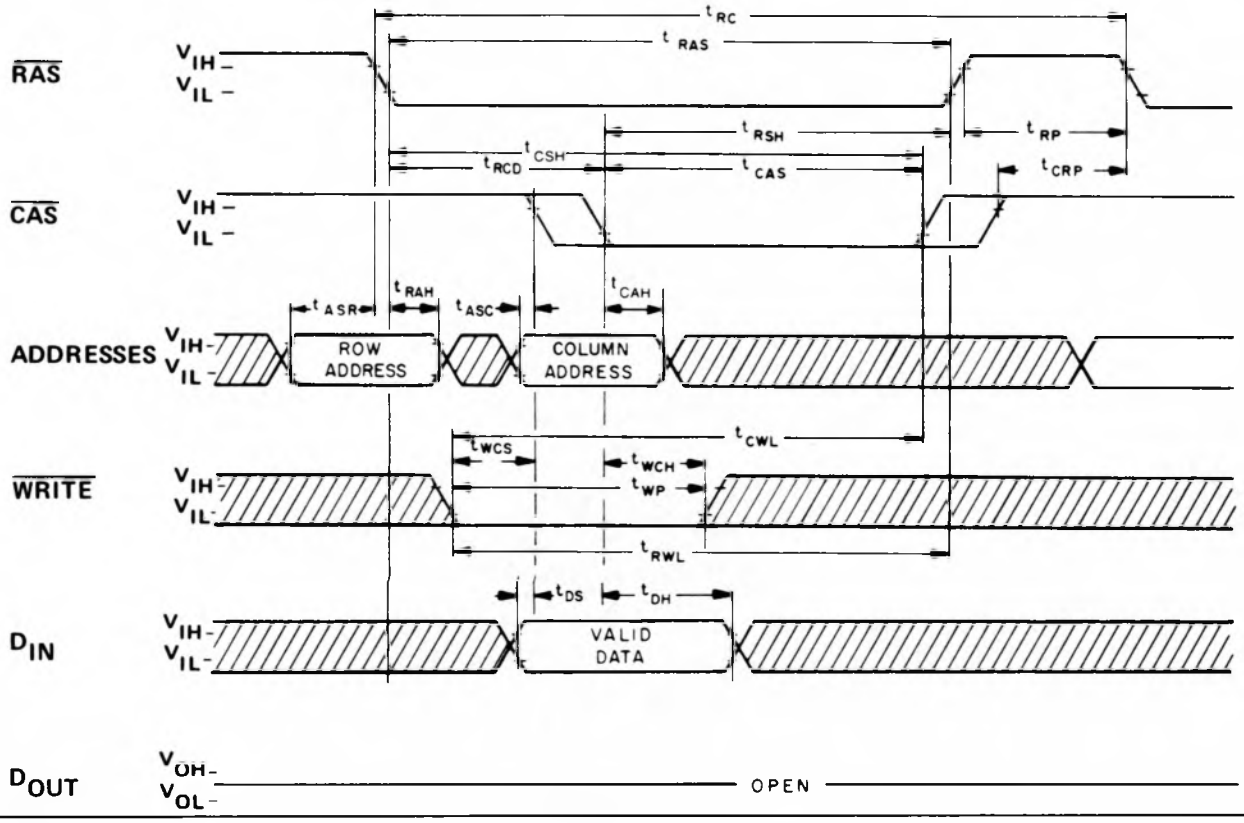


"RAS-ONLY" REFRESH CYCLE

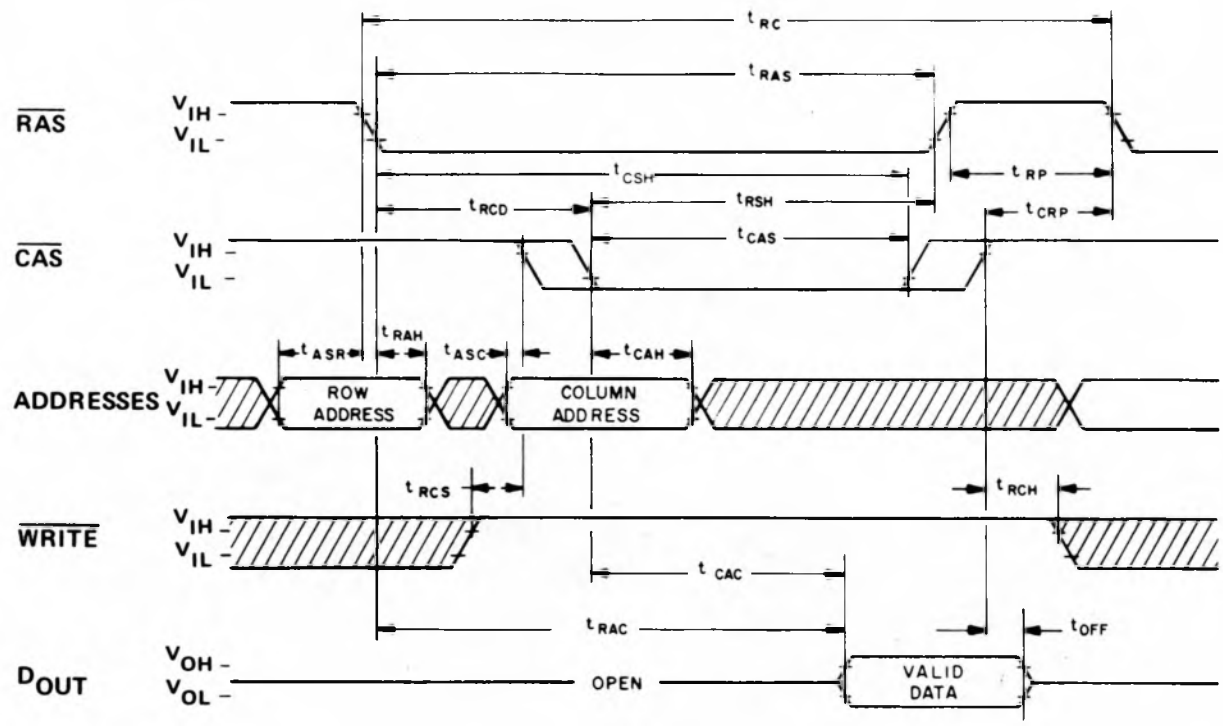
NOTE: $\overline{CAS} = V_{IH}$ $\overline{WRITE}$ = Don't Care



WRITE CYCLE (EARLY WRITE)



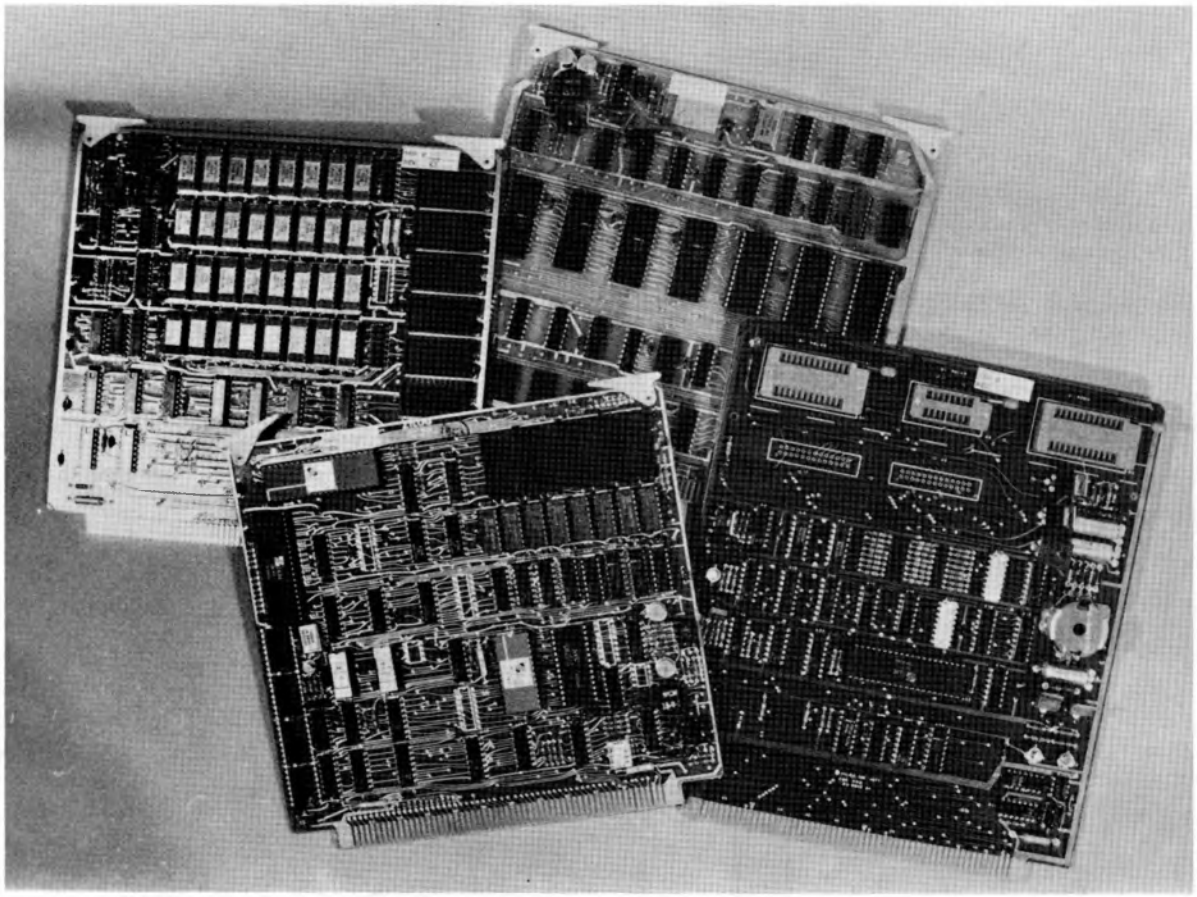
READ CYCLE



Z80-MCB
SERIE

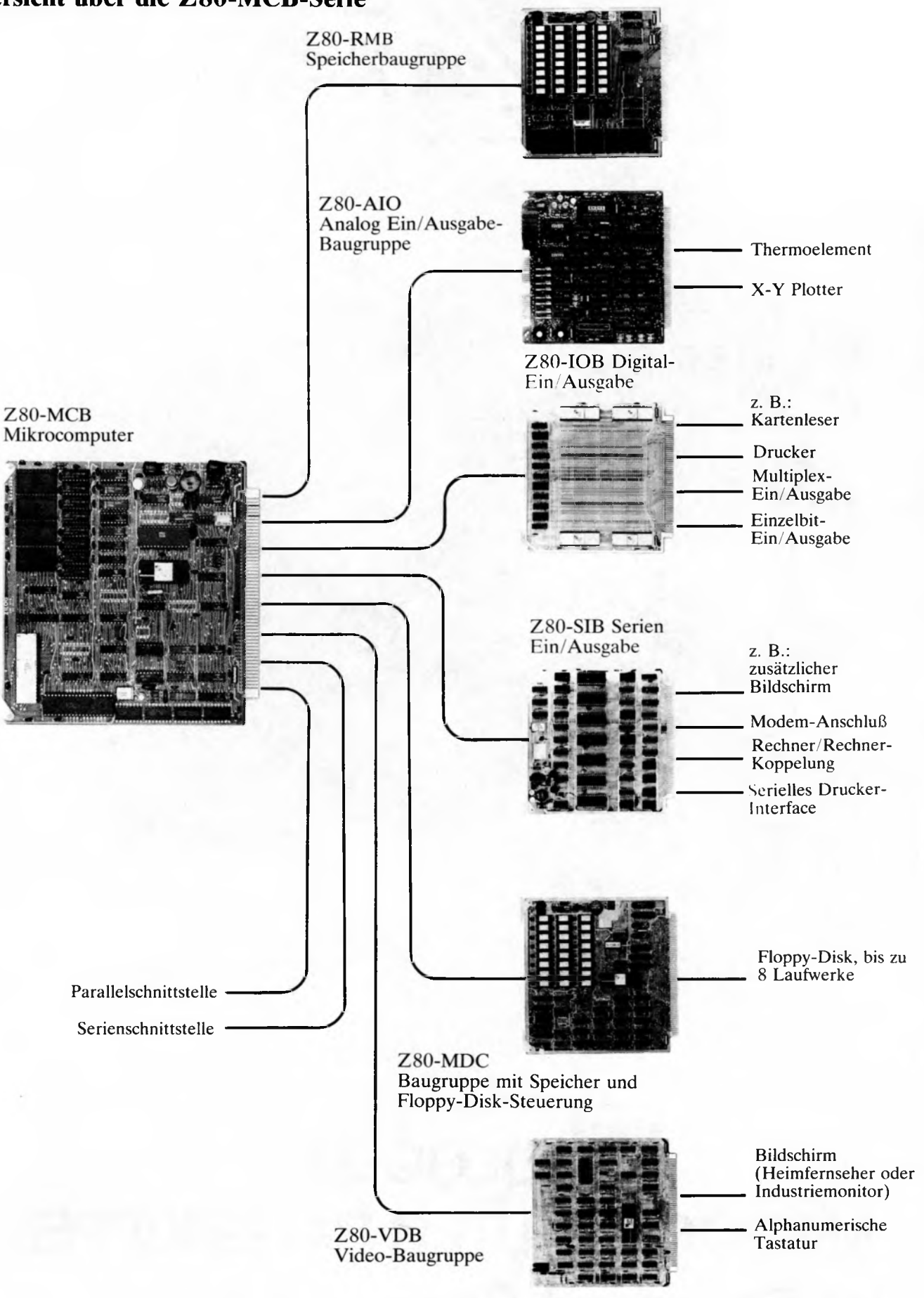
Zilog

Computer Baugruppen im Standard Format 7,7″ × 7,7″



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5” × 7,7”

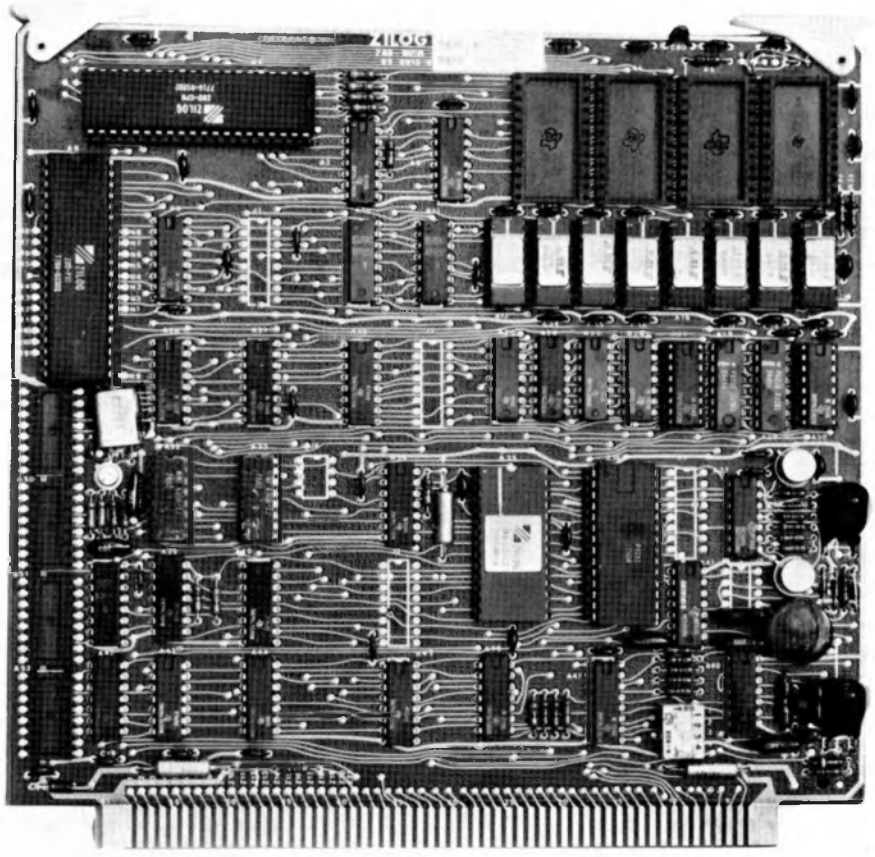
Übersicht über die Z80-MCB-Serie



Z80-MCB

Zilog

Einplatinen-Computer im amerikanischen Format 7,5″ × 7,7″



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5” × 7,7”

Überblick über die Schaltung des Z80-MCB

Bild 1 zeigt das Blockschaltbild. Es ist ersichtlich, daß alle Systembusse (Adreß-, Daten- und Steuer-Bus) zur Sicherstellung voller Erweiterbarkeit gepuffert sind. Auf der hier beschriebenen Baugruppe befinden sich bereits 4 bzw. 16 kByte dynamisches RAM und Fassungen für bis zu 4 kByte Masken-ROM fusible PROM oder EPROM.

Das Adreß-Multiplexen, die Erzeugung der Steuersignale und Adreßcodierung wird durch eine Schaltung auf der Baugruppe besorgt. Normalerweise liegen die 4 kByte Festwertspeicher in Seite 0 (1 Seite bezeichnet einen 4 kByte großen, zusammenhängenden Speicherbereich) und das RAM in Seite 1.

Durch Drahtbrücken lassen sich diese beiden Speicherbausteingruppen in jede beliebige Seite der vom Z80-CPU adressierbaren Speicherbereiche von 64 kByte (= 16 Seiten) legen. Durch weitere Drahtbrücken ist die Baugruppe von 4 kbit-dynamischen RAM's auf 16 kbit-dynamische RAM's (16 pin-Bausteine) umzustellen. Mit diesen Bausteinen beträgt die Schreib/Lesespeicherkapazität der Baugruppe dann 16 kByte! Die für die dynamischen Speicher nötigen Gleichspannungen von —5 V und +12 V werden auf der Platine aus der +5V-Systemversorgung erzeugt, es ist also nur 1 externe Spannungsquelle nötig.

Der Takt des Computers wird von einem 19.6608 MHz-Quarzgenerator abgeleitet; diese Frequenz wurde gewählt, da sie ein ganzzahliges Vielfaches fast aller gebräuchlichen Übertragungsraten in der Nachrichtentechnik ist.

Das Serieninterface der Baugruppe wurde mit einem Standard-USART (Universal Synchronous Asynchronous Receiver Transmitter) realisiert; volle Duplex-Übertragung mit praktisch jedem beliebigen asynchronen und monosync- oder bisync-Übertragungsprotokoll ist möglich; eine RS 232-Schnittstelle mit Kontrollsignalen wie „Request to send" und „Clear to send" ist aufgebaut, außerdem Stromschleifen-

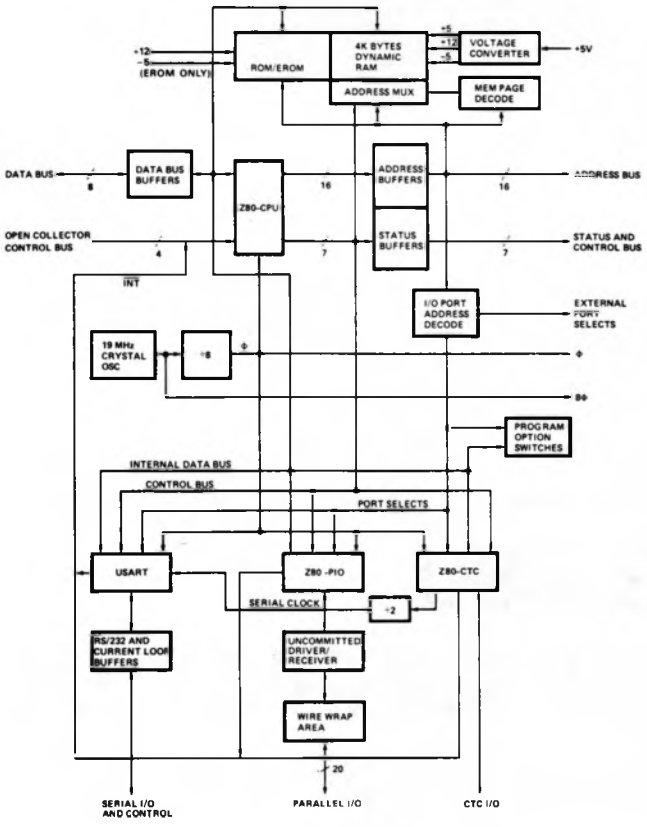


Bild 1: Blockschaltung des MCB

interfaces (current loop) mit einer getrennten „Readercontrol"-Leitung zum Anschluß von Teletype ohne automatische „Reader-control"-Zeichen-Erkennung.

Die Parallelinterfaces der Baugruppe sind mit einem Z80-PIO Baustein realisiert; damit stehen zwei voneinander unabhängige 8 bit-Ports mit sämtlichen Quittungssignalen („Handshaking") und eingebauter Interrupt-Priorisierung („Daisy Chain Priority Interrupt Control") zur Verfügung. Zusätzlich wurden auf der Platine 4 unverdrahtete 16 pin-Schaltkreissockel zum evtl. Einfügen von Leitungspuffern untergebracht und etwas Platinenfläche zum Aufbau anwendungsspezifischer Zusatzschaltung (z. B. Interfaces) in freier Verdrahtung freigelassen.

Zur Speicherplatz- und zeitsparenden Behandlung von Echtzeitaufgaben wurde ein Z80-CTC Baustein eingesetzt, der über 4 von der Z80-CPU softwareprogrammierbare Zähl- bzw. Zeitgeberkanäle verfügt.

Einer dieser 4 Kanäle wurde zur Festlegung der Übertragungsrate des seriellen Übertragungskanals verwendet. Mit einer Anzahl von Schaltern lassen sich eine Reihe von Bedingungen erzeugen, die von der Z80-CPU erkannt und softwaremäßig ausgewertet werden können. So werden beispielsweise über die ½ kB und 1 kB-Monitor-Programme je nach Stellung dieser Schalter die Übertragungsrate des Serienkanals gewählt.

Die Adressdecoder der Platine für Ein/Ausgabe-Ports dekodieren einen zusammenhängenden Bereich von bis zu 32 Portadressen aus, durch die die einzelnen Kanäle der PIO, CTC und des Serieninterfaces aktiviert werden; darüber hinaus sind weitere Auswahlsignale für anwenderspezifische Peripherie am Rand der Baugruppe vorgesehen.

Die MCB-Hardware (Schaltungsbeschreibung)

CPU-Schaltung

umfaßt volle Pufferung von Daten-, Adreß- und Steuerbus, den 19.6608 MHz Quarzoszillator und die Reset-Logik. Dabei ist der CPU-Datenbus direkt mit allen MOS- und Speicherbausteinen auf dieser Platine und mit den bidirektionalen Datenpuffern A28 und A29 verbunden. Die IC's A3, A9 A20 und A21 liegen direkt an Adreß- und Steuerbus der CPU und puffern diese Leitungen sowohl gegenüber externen als auch auf der Platine selbst befindlichen Bausteinen und gehen für DMA in den hochohmigen Zustand, sobald das BUSAK-Signal der CPU aktiv ist.

Beim Einschalten der Speisespannung liefert die Reset-Logik automatisch ein RESET-Signal, ebenso, wenn das RESET-Pin auf Low-Pegel gebracht wird.

Zur Erzeugung eines System-Reset kann man am Steckerpin 31 einen Schalter anschließen (Momentkontakt gegen Masse), der über einen 47 Ω-Widerstand vor Überlastung zu schützen ist.

Das vom Quarzoszillator gelieferte Signal wird durch 8 geteilt, sodaß ein Systemtakt von 2.4576 MHz zur Verfügung steht. Beide Taktsignale und Φ/2 sind gepuffert und über den Systemstecker zur Verwendung auf anderen Platinen von der Platine herausgeführt.

Der RAM-Bereich

ist mit acht 16 pin dynamischen RAM's (A12...A 19) realisiert A 10 und A 11 besorgen das Adreßmultiplexen, A 27 und A 30 entkoppeln die RAM-Datenleitung gegenüber dem internen Datenbus, wodurch die für die RAM's nötige Trennung zwischen einem Ein/Ausgabe- und dem Datenbus realisiert.

Der Baustein A22 ist der Seiten-Decoder, der diesen RAM-Bereich normalerweise auf Seite 1 legt.
Aus seinen Ausgängen leitet A23 die Speicherbereichsauswahl- und das RAS-Signal ab, das von der Low-Power-Inverterkette A26 zur Erzeugung des CAS-Signals verzögert wird; A24 erzeugt das Signal zur Festwertspeicher-Auswahl.
Die folgenden Bilder 2.2 und 2.3 zeigen das Zeitverhalten von Speicherzugriff und Refresh. Näheres hierüber finden Sie in der Applikationsschrift „Anwendung von Standardspeicherbausteinen in Z80-Mikrocomputer-System"

Der Festwertspeicher-Bereich
Die Platine ist mit vier 24 pin-Fassungen (A4, A5, A6 und A7) bestückt, in die Masken ROM's, „Fusible" PROM's oder EPROM's eingesteckt werden können.
Beim Einsatz von EPROM's ist der Anschluß externer Spannungsquellen (—5 V, +12 V) nötig, da der Spannungswandler der Baugruppe nicht die dafür nötige Belastbarkeit hat. Durch Drahtbrücken können Bausteine mit unterschiedlichem Pinning verwendet werden.
Außerdem muß auf genügend kurze Zugriffszeiten der verwendeten EPROM's (bei PROM's, die meist in bipolarer Technik realisiert sind, ist das unkritisch) geachtet werden. Die Z80-CPU (2,5 MHz-Version) erfordert im Worst Case Speicher mit einer Zugriffszeit $\leq$ 380 nsec. Die Bausteine 2708 haben jedoch standardmäßig eine garantierte Zugriffszeit von nur $\leq$ 450 nsec. Die Schaltung kann daher nur durch folgende Maßnahmen mit diesen EPROM's betrieben werden.

a) Selektion von „superschnellen" 2708, deren Zugriffszeit unter 380 nsec liegt.
b) Verringerung der Arbeitsfrequenz des Z80-MCB (anderer Quarz), wobei allerdings unbedingt darauf geachtet werden muß, daß durch eine Änderung der Tatkfrequenz auch die Baud-Rate der Serienschnittstelle beeinflußt wird.
c) Einfügen eines zusätzlichen WAIT-Zyklus bei allen Speicher-Lesezugriffen. Dabei wird natürlich auch die Verarbeitungsgeschwindigkeit des Computers herabgesetzt.
Das Erzeugen dieser WAIT-Zyklen kann durch einen Hardware-Eingriff auf der Z80-MCB-Baugruppe erfolgen (ein zusätzlicher Flip-Flop-Baustein in einer der leeren 16-poligen Fassungen auf dem MCB und Herstellung der entsprechenden Verbindungen und Unterbrechungen. Die zugehörige Schaltung ist im Z80-CPU-Technical Manual angegeben.

Der Z80-CTC-Bereich
umfaßt außer dem Zähler/Zeitgeberbaustein (A35) die I/O-Port-Decoder (A45 und A40) für diese Platine und zusätzliche Platinen. Dabei erzeugt A40 acht Gruppen zusammenhängender Port-Adreßbereiche, die je 4 Portadressen umfassen.
Diese 4 Portadressen werden von den Adreßbits 0 und 1 über A45 decodiert.

Tabelle 2, S. 11 zeigt, wie der ausdecodierte Adreßbereich von insgesamt 32 Adressen zusammenhängend auf jede beliebige Stelle innerhalb der verfügbaren 256 Portadressen gelegt werden kann.
Beispiel: Sollen alle Ports im Bereich zwischen 0...1FH zu liegen kommen, müssen Y mit 0 und Z mit 1 verbunden werden, also die Punkte 4 und 5, 1 und 7, bzw. 6 und 16 von „Jumper" 2.

Von den 4 Kanälen des CTC-Bausteins (technische Einzelheiten siehe Z80-CTC-Produktspezifikation) wird Kanal 1 zur Erzeugung des Synchronisationssignals des Serieninterfaces benutzt. Über Software (Monitor-Programm) ist dessen Frequenz auf das 32fache der gewünschten Übertragungsfrequenz festgelegt. Durch die Nachschaltung des Symmetrierflipflops A2 erfolgt eine Halbierung dieser Frequenz. Kanal 0 ist für Zeitgeberfunktionen mit Floppy Disk, z. B. bei Verwendung der Floppy Disk-Steuerungsplatine Z80-MDC vorgesehen (Drahtbrücken); Kanal 2 ist unbelegt und zur anwendungsspezifischen Verwendung frei verfügbar. Kanal 2 wird vom 1 KB-Monitor zum Zählen des Auftretens der Breakpointbedingungen verwendet.

Der Interrupt-Freigabe-Eingang der Priorisierungskette ist über den Systemstecker herausgeführt, sodaß weitere Peripheriebausteine auf anderen Baugruppen in die Priorisierungskette einbezogen werden können.

Die Parallel-Ein/Ausgabeschaltung
Außer dem Z80-PIO-Baustein (technische Einzelheiten siehe Z80-PIO-Produktspezifikation und Z80-PIO Technical Manual) sind auf der Baugruppe 4 unverdrahtete 16 pin-Fassungen (A49...A52) für Pufferschaltkreise untergebracht.
Die Zwei 6306-PROM's (A33, A32) bestimmen die Übertragungsrichtung der Puffer, die geeignet gewählt werden muß, wenn Buszugriffe von anderen Platinen her erfolgen.

Die Schaltung zur seriellen Ein/Ausgabe
ist mit einem 8251 USART realisiert. Der Baustein arbeitet als serielles asynchrones oder synchrones Port mit Modem-Steuersignalen wie REQUEST TO SEND, CLEAR TO SEND, DATA TERMINAL READY usw. Der Baustein ist auch interrupt-fähig, jedoch wird diese Möglichkeit von unseren Monitorprogrammen nicht benutzt, da hierfür zusätzliche Interrupt-Prioritätslogik nötig wäre.
A47 ist ein Tristate-Puffer für die 4 Funktionsschalter auf der Platine. Bei Verwendung der ½ k und 1 k-Monitorprogramme werden diese Schalter zur Einstellung der Baud-Rate entsprechend der in Bild 3 angegebenen Tabelle verwendet.

BAUD RATE	f_S 16XRATE	JUMPERS S1	S2	S3	S4	N (DIVISOR)
50	800HZ	ON	ON	ON	ON	96X16
75	1.2KHZ	OFF	ON	ON	ON	64X16
(TTY)110	1.745KHZ	ON	OFF	ON	ON	44X16
134.5	2.133KHZ	OFF	OFF	ON	ON	36X16
150	2.4KHZ	ON	ON	OFF	ON	32X16
200	3.2KHZ	OFF	ON	OFF	ON	24X16
300	4.8KHZ	ON	OFF	OFF	ON	16X16
600	9.6KHZ	OFF	OFF	OFF	ON	8X16
1200	19.2KHZ	ON	ON	ON	OFF	4X16
2400	38.4KHZ	OFF	ON	ON	OFF	2X16
4800	76.8KHZ	ON	OFF	ON	OFF	1X16
9600	153.6KHZ	OFF	OFF	ON	OFF	4*
19200	307.2KHZ	ON	ON	OFF	OFF	2*
38400	614.4KHZ	OFF	ON	OFF	OFF	1*
AUTO*	AUTO	OFF	OFF	OFF	OFF	AUTO**

Bild 3: Datenübertragungsrate der Serienschnittstelle

* Drahtbrücke
** Ist standardmäßig nicht in den Z80-Monitorprogrammen implementiert.

Die Interfaceschaltung zur Serienschnittstelle
erlaubt das Betreiben des Serien-Ein/Ausgabekanals als RS 232 oder 20 mA-Strompräge-Schnittstelle:
A41 beinhaltet Pegelwandler, die die Signalspannung von ±12 V oder RS 232 in TTL-Pegel umsetzen. Umgekehrt setzt A37 die Mikrocomputersignale von TTL auf RS 232-Pegel um. Dabei sind ihre beiden Spannungswerte +12 V und —10 V, wenn keine zusätzliche externe —12 V-Spannungsquelle zur Anwendung kommt; der Hub von —10 V ist jedoch im allgemeinen ausreichend und bietet einen genügend großen Störabstand.
Die Transistoren Q 1 und Q 2 dienen der Anpassung der Serienschnittstelle für Stromschleifen-Übertragung; für Teletype ohne „Reader Control" (bei denen also keine ASCII-Start/Stop-Zeichen intern dekodiert werden) wurde eine

eigene Start/Stop-Steuerleitung herausgeführt; das zugehörige Signal wird vom „DTR-Modem-Control"-Signal des 8251-Bausteins abgeleitet.

Spannungswandler

Baustein A12 ist ein DC/DC-Wandler, der aus der Systemstromversorgung +5 V die für die dynamischen RAM's und die RS 232 C nötigen Spannungen +12 und —10 V erzeugt.

Bitte beachten Sie, daß von den Monitor-Programmen die „Reader Control"-Schleife immer aktiviert wird und der Anwender für die Bedienung des TTY-Lochstreifenlesers sorgen muß.

Per Programm kann die Anzahl der pro Zeichen zu übertragenden bits, die Anzahl der Stop-Bits (1, 1½ oder 2) für asynchrone Übertragung, Paritätserzeugung/Prüfung (ungerade, gerade oder keine) und die Übertragungsrate bei Sendung oder Empfang durch Multiplikation mit den Faktoren 1, 16 oder 64 festgelegt werden.

Einer der 4 CTC-Kanäle wird zusammen mit dem USART zur Realisierung des Baud-Raten-Generators verwendet.

Eine Versorgung des USARTS (Anschlüsse TXC und RXC) mit externem Takt anstelle des CTC-Taktsignals ist durch Änderung der Standard-Leiterbahn-Führung möglich.

Interface zwischen PIO und einem Zeilen-Drucker (Bild 4)

Hier ist die Verdrahtung zwischen der Z80-PIO, den Puffer-Schaltkreisen A48...A51 und den Anschlüssen eines Zeilendrucker angegeben.

Die hierzu nötigen 4 Schaltkreise 74 LS 367 werden nicht standardmäßig mit der Computerbaugruppe mitgeliefert.

Bitte beachten Sie, daß von den ½ kByte und 1 kByte Monitor-Programmen die „Reader Control"-Schleife immer aktiviert wird und der Anwender für die Bedienung des TTY-Lochstreifenlesers sorgen muß.

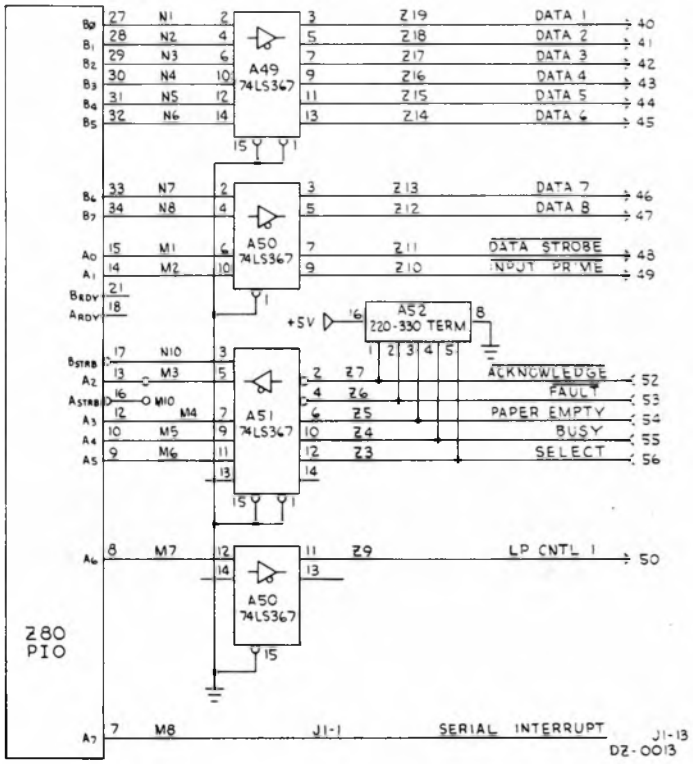


Bild 4: Anschluß eines Schnelldruckers an Z80-MCB (die hierzu nötigen Bauteile A49—A52 sind **nicht** mitgeliefert).

1	+5	34	I/O SUB GROUPΦ–
2	+5	35	RFSH–
3	+5	36	ADDRESS BUS 13
4	IORQ–	37	ADDRESS BUS 11
5	DATA BUS 5	38	OPEN (Z 21) (PULL UP)
6	20 mA DATA	39	OPEN (Z 20)
7	RECEIVE DATA	40	OPEN (Z 19)
8	DATA BUS 3	41	OPEN (Z 18)
9	MASTER RESET	42	OPEN (Z 17)
10	MASTER RESET–	43	OPEN (Z 16)
11	CLEAR TO SEND	44	OPEN (Z 15)
12	DATA BUS 6	45	OPEN (Z 14)
13	DATA BUS Φ	46	OPEN (Z 13)
14	REQ TO SEND	47	OPEN (Z 12)
15	XMITED DATA	48	OPEN (Z 11)
16	MEM SEL IN	49	OPEN (Z 10)
17	DISK C/T	50	OPEN (Z 9)
18	C/T 2	51	OPEN (Z 8)
19	20 mA DATA RET	52	OPEN (Z 7)
20	TTY TAPE CNTL RET	53	OPEN (Z 6)
21	MEMORY SEL OUT	54	OPEN (Z 5)
22	INTE IN CTC	55	OPEN (Z 4)
23	WR–	56	OPEN (Z 3)
24	USER STRB 3	57	OPEN (Z 2)
25	DISK STRB	58	OPEN (Z 1)
26	ADDRESS BUS 7	59	+5
27	ADDRESS BUS 8	60	+5
28	IOWR–	61	+5
29	ADDRESS BUS 5		
30	ADDRESS BUS 6		
31	RESET–		
32	ADDRESS BUS 15		
33	I/O SUB GROUP 2–		

NOTE: Open pins are definable by user. One is pulled up for use as a source for interrupt enable daisy chain.

62	GND	93	8Φ–
63	GND	94	ADDRESS BUS 14
64	GND	95	I/O SUB GROUP 3–
65	TTY TAPE CNTL	96	I/O SUB GROUP 1–
66	–5V EXTERNAL	97	ADDRESS BUS 12
67	–5V EXTERNAL	98	ADDRESS BUS 4
68	DATA BUS 4	99	Φ–
69	+12V EXTERNAL	100	ADDRESS BUS 3
70	+12V EXTERNAL	101	ADDRESS BUS 2
71	DATA BUS 2	102	ADDRESS BUS 1
72	–12V EXTERNAL	103	ADDRESS BUSΦ
73	DATA BUS 7	104	I/O GROUPΦ–
74	DATA SET READY	105	I/O GROUP 1–
75	DATA BUS 1	106	I/O GROUP 2–
76	DATA TERM. RDY/ XMIT CLK	107	I/O GROUP 3–
		108	I/O GROUP 4–
77	20mA RECV. RETN./ RECV. CLK	109	BUS RQ–
		110	NMI–
78	SYNC DETECT	111	PIO INTE OUT
79	INT–	112	SERIAL CLK IN (2X)
80	LINE SIGNAL DET.		
81	20mA RECV.	113	HALT–
82	USER C/T 3	114	INTE IN PIO
83	ROM SEL OUT	115	M 1–
84	USER RTC C/T	116	RD–
85	MRQ–	117	INTE IN SER
86	CTC INTE OUT	118	½Φ
87	2X SERIAL CLK	119	WAIT–
88	BUSAK–	120	GND
89	ADDRESS BUS 9	121	GND
90	IORD–	122	GND
91	ADDRESS BUS 10		
92	ROM SEL IN–		

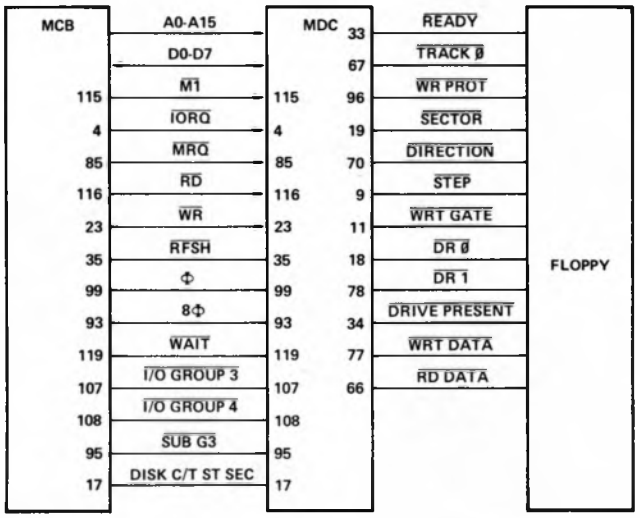
Bild 6: Steckerbelegung der Z80-MCB

Anschluß weiterer Baugruppen

Die MCB-Baugruppe kann als Grundbaustein für umfangreichere Systeme eingesetzt werden.

Von ZILOG wird ein dauernd wachsendes Spektrum von Baugruppen angeboten, die zum Z80-MCB kompatibel sind.

Zur Zeit verfügbar sind Baugruppen zur Aussteuerung von Floppy Disks (Z80-MDC), Bildschirmgeräten (Z80-VDB), RAM-Speichererweiterung (Z80-RMB), Festwertspeichererweiterung (Z80-PMB), PROM-Programmierboard (Z80-CPB), parallele und serielle Ein/Ausgabebaugruppen (Z80-IOB und SIB), Wire-Wrap-Karte (Z80WWB), Verlängerungsplatine (Z80-EXB) und ein Baugruppenreck (Z80-SCC).

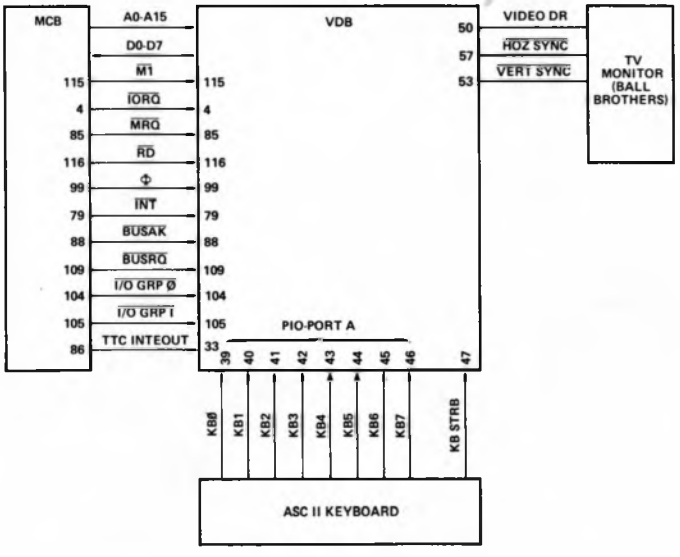


Zusammenschaltung von Z80-MCB, Z80-MDC und einer Floppy Disk

Dabei ermöglicht die MDC-Baugruppe den Datenverkehr zwischen SHUGART-800-Floppy-Disk-Laufwerken und der Z80-CPU (s. Bild 7); die hierfür nötige Software wird in Form eines 3KByte-Monitors angeboten.

Das Video-Board Z80-VDB erlaubt den direkten Anschluß eines beliebigen Fernsehbildschirms (Heimfernseher oder Industriemonitor) und die Abbildung von alphanumerischen oder graphischen Darstellungen (Bild 8).

Schließlich steht auch ein komplettes, in einem Gehäuse betriebsbereit geliefertes Mikrocomputersystem zur Verfügung, das auf diesen Baugruppen basiert. Dieses Z80-MCS (= **Mi**kro-**C**omputer-**S**ystem) umfaßt 2-Floppy-Disk-Laufwerke, 20 KByte Speicher und ein vollständiges Platte-Betriebs-System.



Zusammenschaltung von Z80-MCB, Z80-VDB, Heimfernsehgerät und alphanumerischer Tastatur

Hinweise zur Durchführung von direktem Speicherzugriff (DMA) mit dem Z80-MCB

DMA kann in üblicher Weise durch Aktivierung des BUSRQ-Signals durchgeführt werden. Dabei werden die Adreßbustreiberbausteine A3, A9 und A21, die Datenbustreiberbausteine A28 und A29 und der Steuerbustreiberbaustein A20 in den hochohmigen Zustand gebracht.

Die Busse können nun von externen Schaltungen beaufschlagt werden, solange das BUSRQ-Signal aktiv bleibt.

Die Ausgangsleistungen BUSAK und HALT werden davon nicht betroffen; sie sind immer freigegeben und gehen nie in den hochohmigen Zustand.

Auf dem MCB ist über die zwei Decodier-PROM's 6306 DB CNTL und DB ENA Sorge getragen, daß die Datenbus-Treiber dem DMA-Kanal die Kontrolle über den Datenbus erlauben (Signal IDB07).

Da der Schreib/Lese-Speicher auf dem MCB mit dynamischen RAM's realisiert ist, muß bei DMA per Anwenderprogramm dafür Sorge getragen werden, daß genügend oft Refresh erfolgt und damit die Information im Schreib/Lesespeicher nicht verloren geht.

Dazu sind 64 RAS-Signale im Verlauf von 2 msec erforderlich, was genügend Zeit zur Durchführung der DMA-Operation läßt, ohne daß zusätzliche externe Refresh-Logik nötig wird.

Technische Daten

POWER SUPPLY:	+5V DC ±5%, current: 2 amps max (with 3 PROMs)
CONNECTOR:	122-pin edge (100 mil spacing) Augat PN 14005-19P1
SIZE:	Length, 7.7" Depth, 7.5" Spacing, 0.5" centers
ENVIRONMENTAL:	0° – 50° C temperature range. Up to 90% humidity without condensation.
MEMORY CAPACITY:	4K or 16K bytes dynamic RAM plus up to 4K bytes PROM, ROM or EPROM. Expandable by use of Z-80 RMB RAM board to 64K bytes of main memory.
I/O CHANNELS:	Serial I/O port with RS-232 or 20 mA current loop interface; Two (2) software configurable bidirectional 8-bit parallel I/O ports.

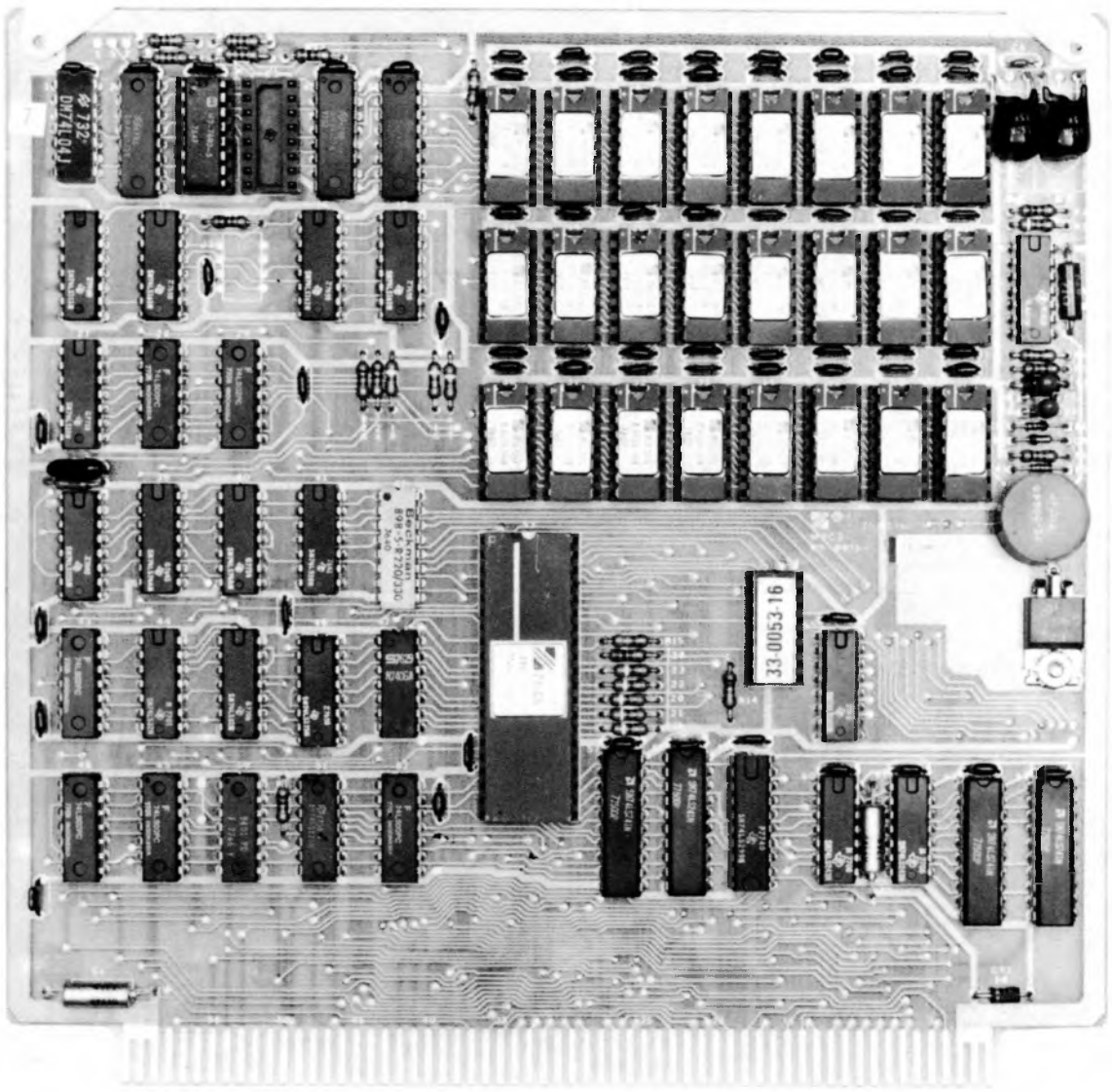
Bestellbezeichnungen:

Z80-MCB/4	Baugruppe MCB mit 4 kByte Schreib/Lesespeicher (4 kbit-dynamische RAM-Bausteine)
Z80-MCB/16	Baugruppe MCB mit 16 kByte Schreib/Lesespeicher (16 kbit-dynamische RAM-Bausteine)

Z80-MDC

Zilog

Floppy-Disk-Controller



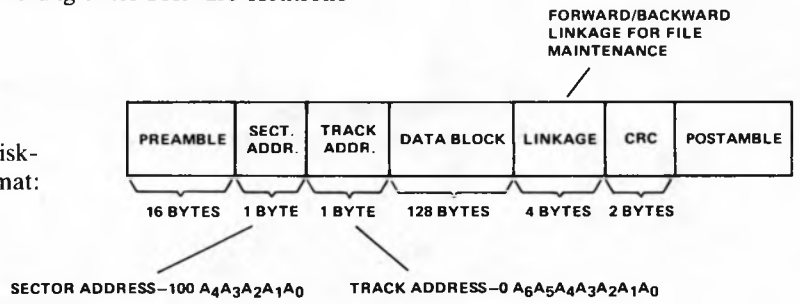
6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standardformat 7,5″ × 7,7″

ZILOG Z80-MDC ("**M**emory/**D**isk **C**ontroller")

Speicher/Floppy-Disk-Steuerung

- 16...48 kByte dynamisches RAM inclusive Decodierung und Pufferung
- 4 FD-Laufwerk-Steuerung mit
  - — 16 Bit CRC-Erzeugung und -Prüfung
  - — Lesen und Steuern des Platten-Status unter Software-Kontrolle über den Z80-PIO
  - — Daten-Separator und -Encoder für einfache Schreibdichte ausgelegt, durch Drahtbrücken auf doppelte Schreibdichte umstellbar.
  - — Parallel/Serien-Wandler und WAIT-Steuerlogik für Softwaregesteuerte Datenübertragung
  - — Disketten-Formatierung unter Software-Kontrolle
- FD-Steuerungs-Grundsoftware im 3 kB-Monitor enthalten!
- Unverdrahtete Sockel für Problemspezifische Treiberschaltungen
- Spannungswandler +5 V→+12 V und —5 V auf der Platine
- 122 PIN-Steckverbindung (100 MIL Spacing)
- Umgebungstemperatur 0 . . . 50 °C

Floppy-Disk-Datenformat:



Bestellbezeichnung:

Z80-MDC/12	Baugruppe mit 12 kByte Speicherkapazität (4 k Chips)
Z80-MDC/16	Baugruppe mit 16 kByte Speicherkapazität (16 k Chips)
Z80-MDC/32	Baugruppe mit 32 kByte Speicherkapazität (16 k Chips)
Z80-MDC/48	Baugruppe mit 48 kByte Speicherkapazität (16 k Chips)

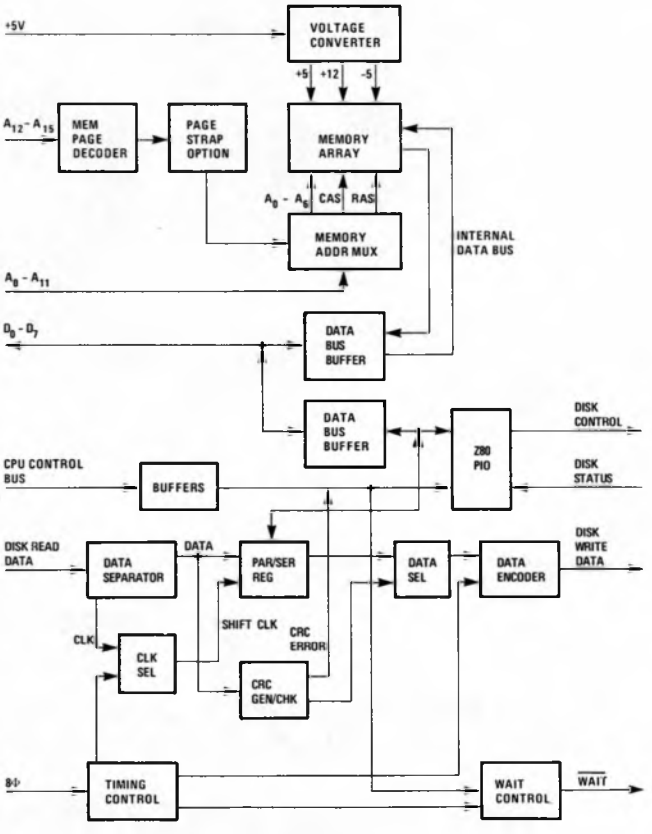
Pin Belegung

Bauelementeseite:

1 +5v	21 OPEN	41 OPEN
2 +5v	22 OPEN	42 OPEN
3 +5v	23 OPEN	43 OPEN
4 IORQ–	24 OPEN	44 OPEN
5 DATA BUS 5	25 OPEN	45 OPEN
6 OPEN	26 ADDRESS BUS 7	46 OPEN
7 DR 7–	27 ADDRESS BUS 8	47 OPEN
8 DATA BUS 3	28 OPEN	48 OPEN
9 STEP–	29 ADDRESS BUS 5	49 OPEN
10 SEL 3–	30 ADDRESS BUS 6	50 OPEN
11 WRITE GATE–	31 OPEN	51 OPEN
12 DATA BUS 6	32 ADDRESS BUS 15	52 OPEN
13 DATA BUS Ø	33 READY–	53 OPEN
14 DR 6–	34 DRIVE PRESENT–	54 OPEN
15 DR 5–	35 REFRESH–	55 OPEN
16 OPEN	36 ADDRESS BUS 13	56 OPEN
17 START SECTOR	37 ADDRESS BUS 11	57 OPEN
18 DR Ø–	38 OPEN	58 OPEN
19 SECTOR–	39 OPEN	59 +5v
20 OPEN	40 OPEN	60 +5v
		61 +5v

Leiterbahnseite:

62 GND	82 SEL Ø–	102 ADDRESS BUS 1
63 GND	83 ROM SEL OUT–	103 ADDRESS BUS Ø
64 GND	84 OPEN	104 OPEN
65 OPEN	85 MEM REQUEST–	105 OPEN
66 READ DATA–	86 SEL 2–	106 OPEN
67 TRACK Ø–	87 OPEN	107 I/O GROUP 3–
68 DATA BUS 4	88 OPEN	108 I/O GROUP 4–
69 SPARE OUT–	89 ADDRESS BUS 9	109 OPEN
70 DIRECTION–	90 OPEN	110 OPEN
71 DATA BUS 2	91 ADDRESS BUS 10	111 OPEN
72 OPEN	92 OPEN	112 OPEN
73 DATA BUS 7	93 8 X CLOCK	113 OPEN
74 DR 3–	94 ADDRESS BUS 14	114 MDC PIO INT IN
75 DATA BUS 1	95 SUBG3–	115 M1–
76 DR 2–	96 WRITE PROTECT–	116 READ–
77 WRITE DATA–	97 ADDRESS BUS 12	117 OPEN
78 DR 1–	98 ADDRESS BUS 4	118 OPEN
79 OPEN	99 CLOCK–	119 WAIT–
80 SEL 1–	100 ADDRESS BUS 3	120 GND
81 DR 4–	101 ADDRESS BUS 2	121 GND
		122 GND

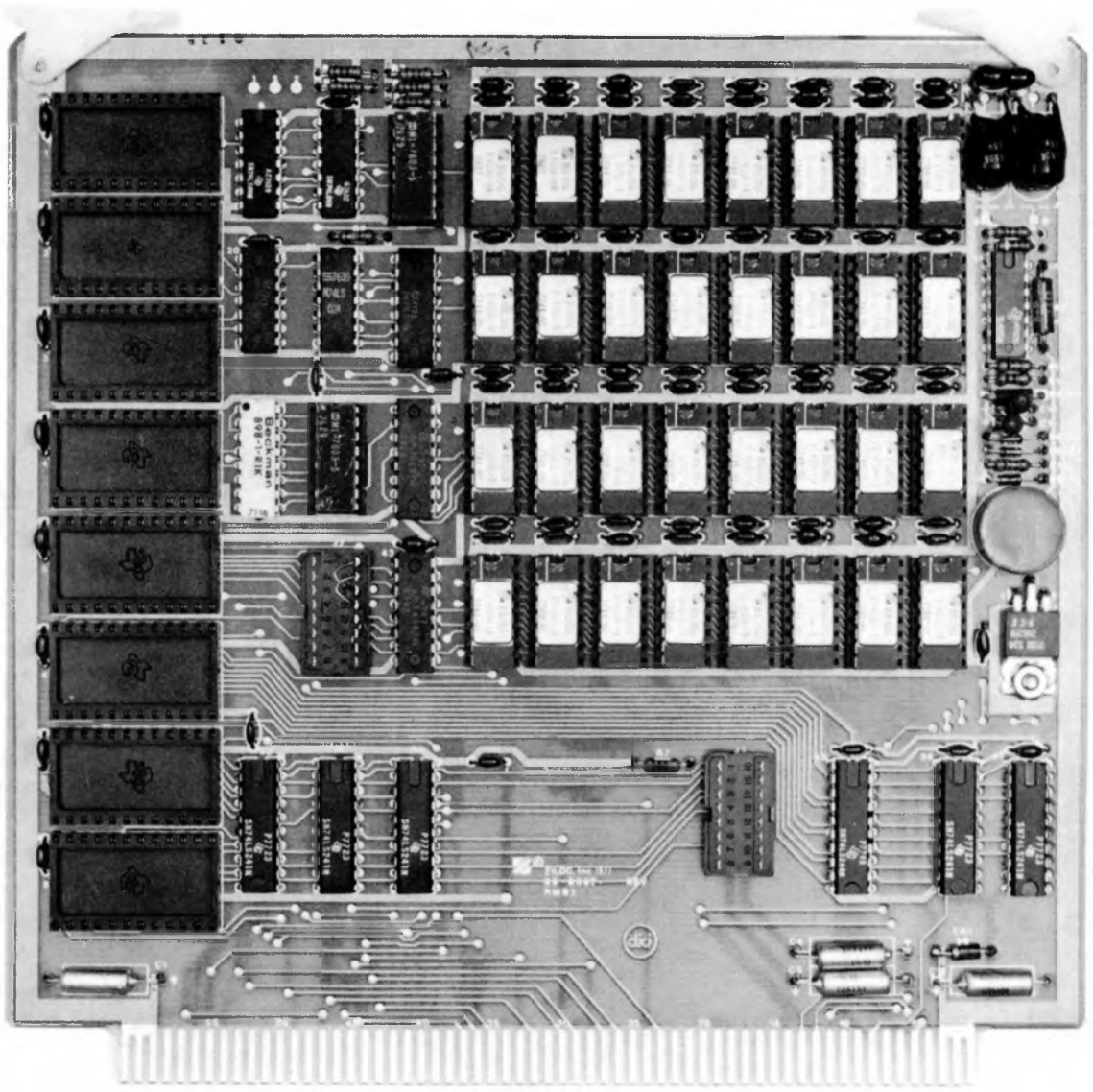


Blockschaltbild

Z80-RMB

Zilog

RAM/ROM-Speicher-erweiterungs-Baugruppen



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5 × 7,7”

ZILOG Z80-RMB ("**R**AM-**M**emory **B**oard")
RAM-Speicherbaugruppe

- 16...64 kByte dynamisches RAM (16 Pin 16 kbit RAM's)
- 4 Fassungen (28 pin) für ½kByte—oder 1kByte PROM's oder EPROM's
- Komplette Multiplexer-, Decodier- und Puffer-Schaltung
- Speicher läßt sich jedem beliebigen Bereich innerhalb der 64 kB zuordnen.
- Gleichspannungswandler +5V→12V und — 5V auf der Platine
- Leistungsverbrauch 10 W
- 122 PIN-Steckverbindung (100 MIL Spacing)
- Umgebungstemperatur 0...50°C

Pin-Belegung

Bauelemente-Seite

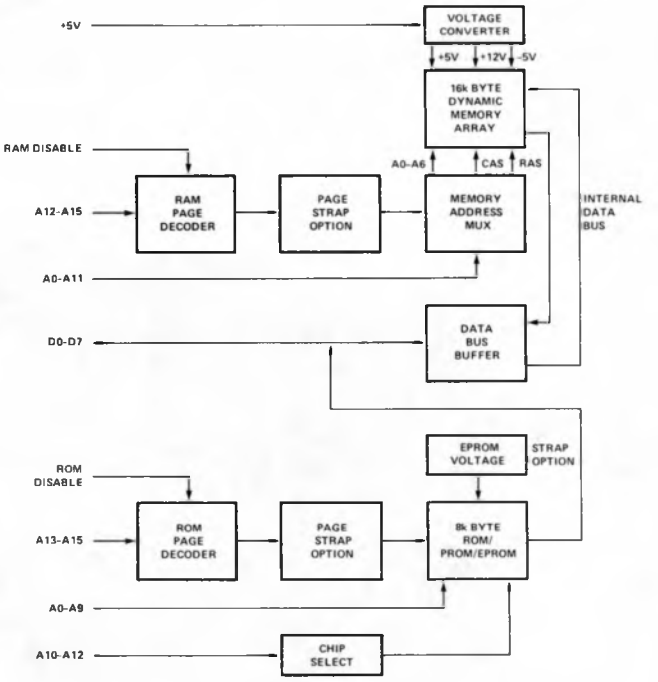
1	+5v	29	ADDRESS BUS 5
2	+5v	30	ADDRESS BUS 6
3	+5v	32	ADDRESS BUS 15
5	DATA BUS 5	35	RFSH —
8	DATA BUS 3	36	ADDRESS BUS 13
12	DATA BUS 6	37	ADDRESS BUS 11
31	DATA BUS Ø	59	+5v
23	WRITE —	60	+5v
26	ADDRESS BUS 7	61	+5v
27	ADDRESS BUS 8		

Leiterbahn-Seite

62	GND	97	ADDRESS BUS 12
63	GND	98	ADDRESS BUS 4
64	GND	100	ADDRESS BUS 3
66	—5v	101	ADDRESS BUS 2
67	—5v	102	ADDRESS BUS 1
68	DATA BUS 4	103	ADDRESS BUS Ø
69	+12v	108	RAM DISABLE —
70	+12v	116	READ —
71	DATA BUS 2	117	ROM DISABLE —
73	DATA BUS 7	120	GND
75	DATA BUS 1	121	GND
85	MEM REQUEST —	122	GND
89	ADDRESS BUS 9		
91	ADDRESS BUS 10		
94	ADDRESS BUS 14		

Bestellbezeichnungen:

Z80-RMB/16	Baugruppe RMB mit 16 kByte-RAM-Kapazität
Z80-RMB/32	Baugruppe RMB mit 16 kByte-RAM-Kapazität
Z80-RMB/48	Baugruppe RMB mit 48 kByte-RAM-Kapazität



Blockschaltung des Z80-RMB

Technische Daten:

POWER SUPPLY: +5 VDC ±5%
1.6 amps maximum current

CONNECTOR: 122-pin edge (100mil spacing)

SIZE:
Length 7.7"
Depth 7.5"
Spacing 0.5" centers

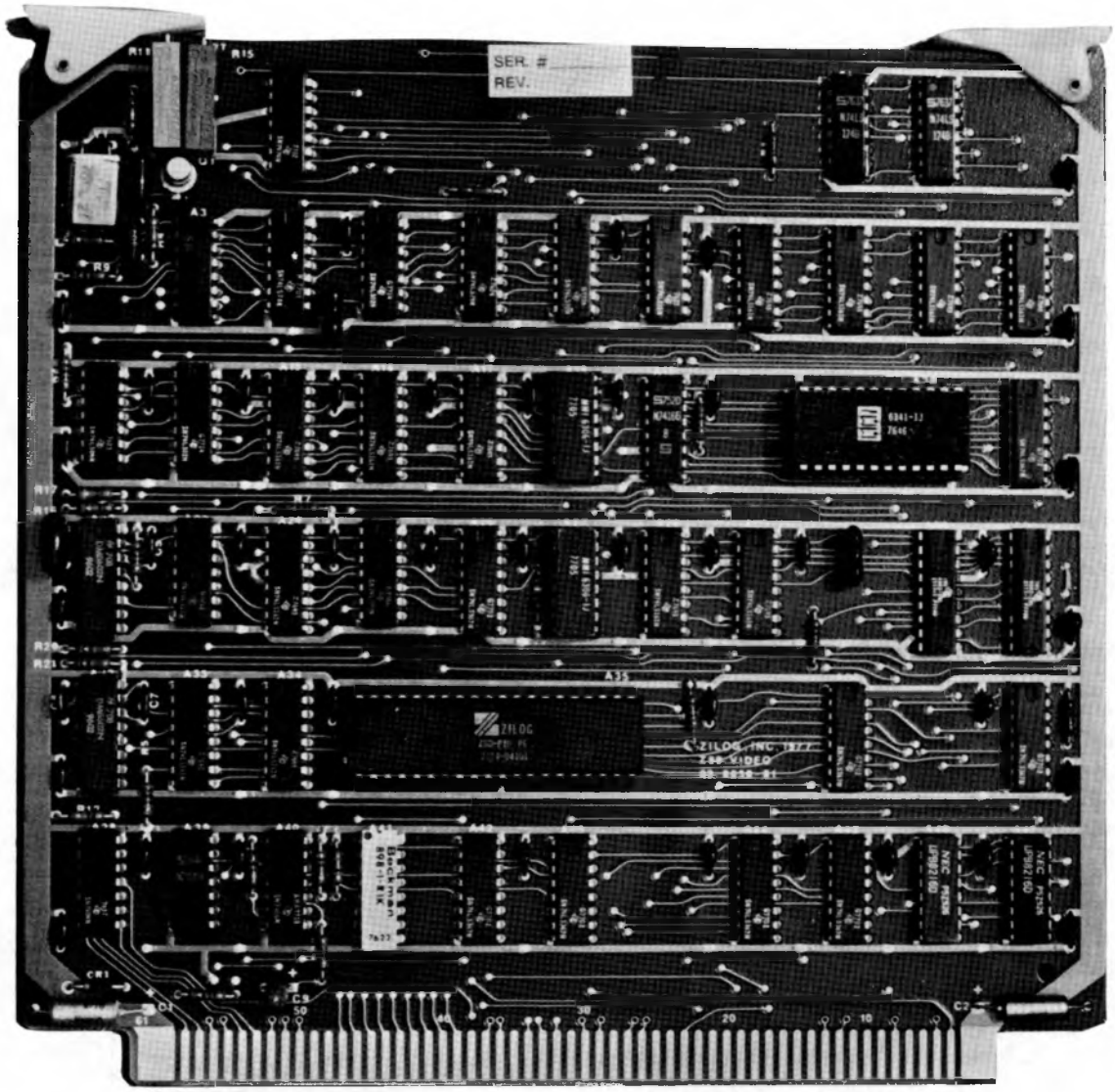
ENVIRONMENTAL: 0° – 50° C temperature range. Up to 90% humidity without condensation.

MEMORY CAPACITY: 16K, 32K, 48K, or 64K bytes dynamic RAM. Each 4K page may have its starting address assigned to any of 16 possible values. Also sockets on board for up to 8K bytes of non-volatile memory (ROM, PROM, or EPROM).

Z80-VDB

Zilog

Baugruppe zur Bildschirmansteuerung



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5" × 7,7"

Z80-VDB

Das Z80-VDB ist eine Baugruppe aus der MCB-Familie zum direkten Anschluß von Industrie-Monitor- oder Heimfernsehgeräten. Darüber hinaus kann an das Z80-VDB direkt eine ASC II-Tastatur angeschlossen werden, sodaß aus VDB-Baugruppe, Tastatur und Bildschirm direkt ein Viedeo-Terminal gebildet wird. Die Architektur des Z80-VDB ermöglicht sowohl alphanumerische als auch graphische Darstellung, ebenso wie die Darstellung von Information, die ein Gemisch von graphischen und alphanumerischen Inhalten darstellen.

Es ist sowohl 50 Hertz als auch 60 Hz-Betrieb möglich; das Umstellen erfolgt über Drahtbrücken. Die Baugruppe ist zum Anschluß von Bildschirmen über Dreidraht TTL Interfaces (Video-Signal, Vertikal-Synchronisation, Horzizontal-Synchronisation) gedacht, es ist jedoch auch auf der Baugruppe ein Composit-Video-Generator aufgebaut, sodaß auch Bildschirmgeräte mit Schnittstelle RS 270 oder Heimfernsehgeräte angeschlossen werden können. Die Bildposition auf dem Bildschirm kann über einstellbare Monoflop's verändert werden. Zum Anschluß der oben erwähnten ASC II-Tastatur steht ein freies Port der auf der Baugruppe untergebrachten Z80-PIO zur Verfügung (Port A), wodurch jede beliebige Tastatur mit bis zu 8 parallelen Leitungen und einer Synchronisationsleitung angeschlossen werden kann. Über Port B dieser PIO wird die Video-Steuerinformation an das VDB übergeben. Diese von Software gesteuerte Information besteht aus 3 Signalen:

- Video sperren
- Bild invertieren und
- Betriebsart

„Video sperren" bewirkt ein Unterbinden der Übertragung der Bildinformation und des DMA-Transfers.

Über „Bild invertieren" kann der gesamte Bildschirm unter Software-Steuerung (Hintergrund) schwarz oder weiß gesteuert werden.

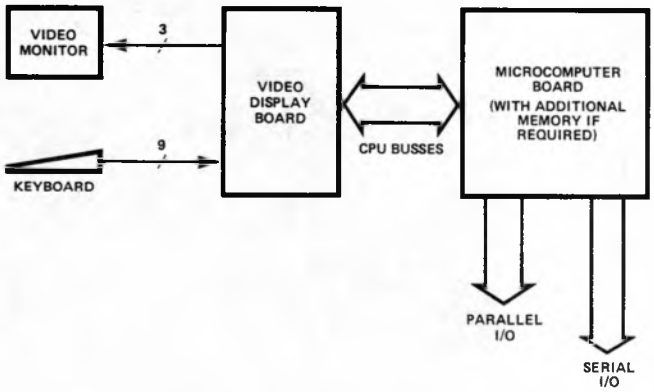
Das Signal „Betriebsart" gibt den Zeichengenerator frei.

Die übrigen, höherwertigen 5 bit's des Port B der Z80-PIO sind zur Festlegung der höchstwertigen Bildwiederholspeicheradressen benützt. Dies impliziert, daß die Anfangsadresse des Bildwiederholspeichers, der Bestandteil des 64 k Byte Computer-Arbeitsspeicher ist, auf ganzen Vielfachen von 2 k Byte liegen muß. Durch die gewählte Adressierweise kann der gesamte Bildinhalt durch eine einzige Ein/Ausgabe-Operation ausgetauscht werden.

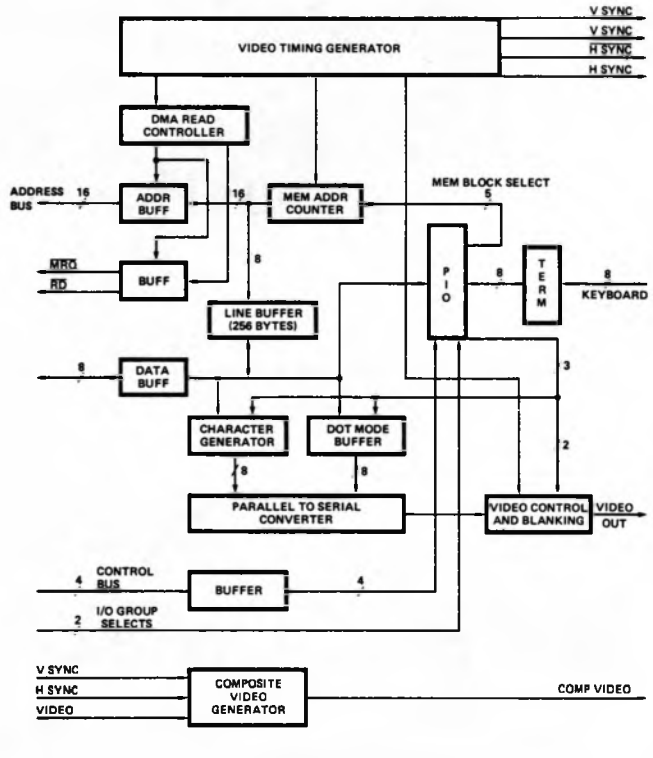
Im laufenden Betrieb werden die Bildinformationen über DMA aus dem Bildwiederholspeicher gelesen. Durch dieses Verfahren ist tatsächlich jeder einzelne Bildpunkt (entsprechend einem Bit des Bildwiederholspeichers) einzeln adressierbar und damit modifizierbar. Das Fortschreiben des Bildwiederholungsspeichers verringert den Informationsdurchsatz der Z80-CPU um nur 10%. In der graphischen Betriebsart wird für jeden Bilddurchlauf ein DMA Transfer durchgeführt; daher ist in diesem Fall der Z80-CPU Informationsdurchsatz um 90% eingeschränkt.

In der alphanumerischen Betriebsart ist das Schirmformat 80 Zeichen pro Zeile und 24 Zeilen auf dem Schirm. Hierfür sind 1980 Bytes Wiederholspeicher erforderlich.

Im graphischen Mode ist das Bildformat 240 Zeilen à 560 Bildpunkte bei Betrieb in 50 Hz-Systemen, wodurch 20160 Byte Bildwiederholspeicher erforderlich sind. In 60 Hz-Systemen ist das Bildschirmformat 240 Zeilen à 560 Bildpunkte, wodurch nur 16800 Byte Bildwiederholspeicher erforderlich sind. In beiden Fällen wird grundsätzlich das höchstwertige Bit eines jeden Bytes zuerst auf den Bildschirm gebracht (von links nach rechts gesehen). Der auf der Baugruppe untergebrachte Zeichengenerator erlaubt die Verwendung von bis zu 128 Zeichen incl. Kleinschreibung, bei der die kleinen Buchstaben Teile haben dürfen, die unter den Zeilenrand reichen (z. B. der Buchstabe g). Da 128 Zeichen-Kombinationen nur 7 bit's zur Darstellung erfordern, ist das achte bit frei und wird zur Information benützt, ob das Video-Signal des entsprechenden Zeichens normal oder invertiert werden soll. Dadurch ist es möglich, jedes einzelne Zeichen wahlweise auf hellem oder dunklem Grund darzustellen. Mit der gleichen Einrichtung läßt sich ein Software gesteuerter Cursor realisieren.



ANWENDUNGSBEISPIEL FÜR Z80-VDB: INTELLIGENTES TERMINAL FÜR ALPHANUMERISCHE UND GRAPHISCHE DARSTELLUNG MIT HEIM- ODER INDUSTRIE-FERNSEH-MONITOR

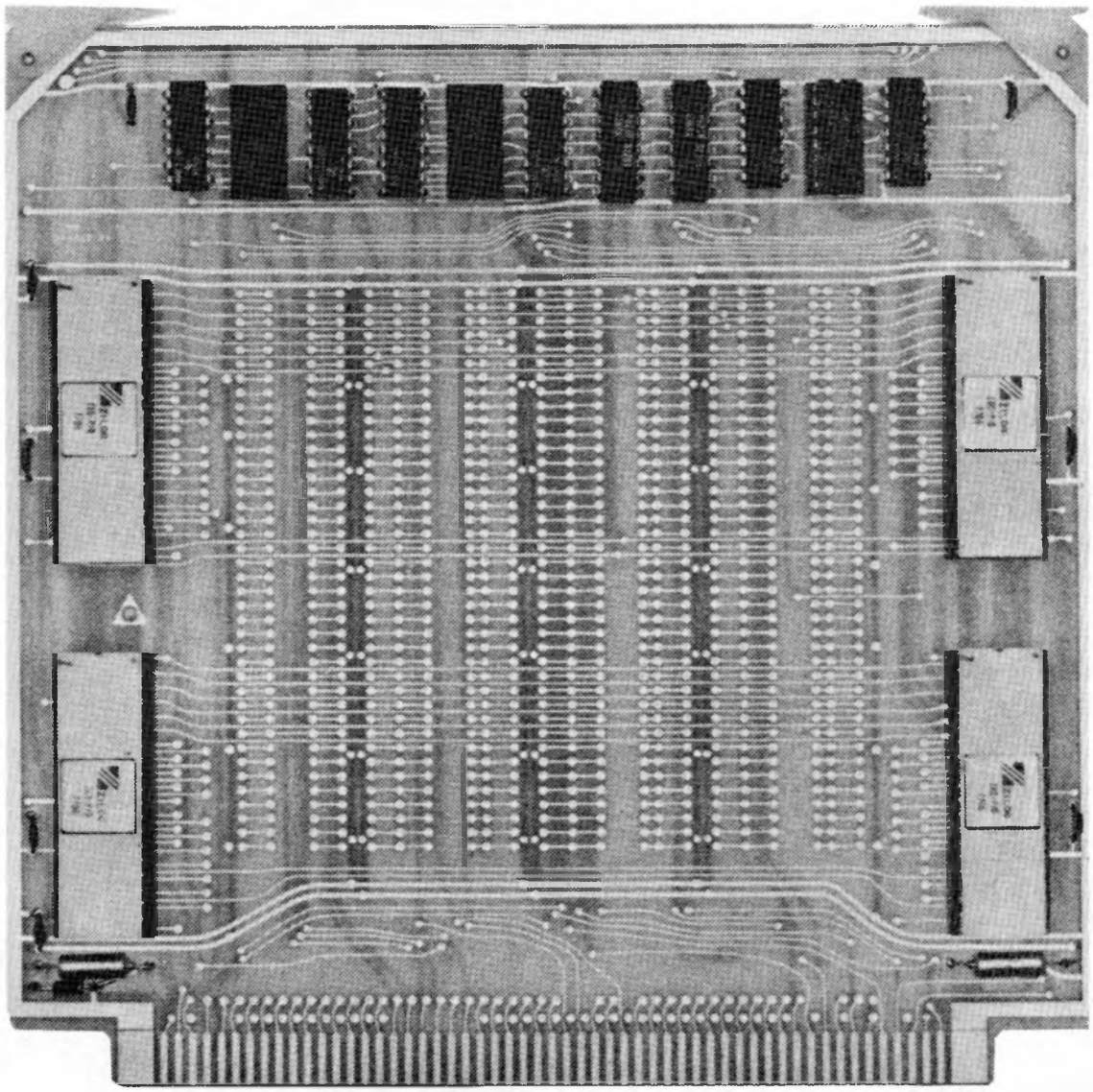


Z80-VDB-BLOCKSCHALTBILD

Z80-IOB

Zilog

**Parallel-
Ein/Ausgabe-
Erweiterung**



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5″ × 7,7″

Beschreibung:

Die Baugruppe dient zur Erweiterung der Parallel-Ein/Ausgabemöglichkeiten der Baugruppen der MCB-Serie.

Sie beinhaltet 4 Z80-PIO-Bausteine und erlaubt dadurch die Behandlung von 64 parallelen Ein- und/oder Ausgabeleitungen zuzüglich 16 Interruptleitungen bzw. bis zu 64 Interruptfähigen Eingängen. Einzelheiten über diese Interrupt-Möglichkeiten finden Sie in der von der KONTRON Elektronik GmbH herausgegebenen Applikations-Sammlung.

Die Baugruppe verfügt außerdem über ein freies Verdrahtungafeld zur Realisierung kundenspezifischer Interfaces. Jeder PIO-Baustein belegt den Z80-CPU-Adreßbereich in folgender Weise:

A0	A1	
0	0	Data for Port A
1	0	Data for Port B
0	1	Control for Port A
1	1	Control for Port B

Die Adressierung der PIO-Bausteine erfolgt nach folgender Tabelle:

ADDRESS RANGE	J2 JUMPERS
00 TO 1F	3-11,5-10,7-9
20 TO 3F	2-11,5-10,7-9
40 TO 5F	3-11,4-10,7-9
60 TO 7F	2-11,4-10,7-9
80 TO 9F	3-11,5-10,6-9
A0 TO BF	2-11,5-10,6-9
C0 TO DF	3-11,4-10,6-9
E0 TO FF	2-11,4-10,6-9

Zur Bereichsauswahl dienen die folgenden Drahtverbindungen:

PIO	JUMPER ON J1	PORT ADDRESS
0	2-13	0,1,2,3
1	3-14	4,5,6,7
2	4-10	8,9,10,11
3	5-11	12,13,14,15

Technische Daten:

POWER SUPPLY: +5 VDC ±5%
Supports up to 10 Watts total power dissipation.

CONNECTOR: 122-pin edge (100 mil spacing)

SIZE: Length' 7.7"
Depth, 7.5"
Spacing, 0.5" centers.

ENVIRONMENTAL: 0° – 50°C temperature range.
Up to 90% humidity without condensation.

I/O CHANNELS: Eight 8-bit parallel I/O channels can be programmed for byte or bit transfer in either direction.

Pin-Belegung

BESTÜCKUNGSSEITE

1	+5 volts	13	DATA BIT Ø
2	+5 volts	26	ADDRESS BIT 7
3	+5 volts	29	ADDRESS BIT 5
4	$\overline{IORQ}$	30	ADDRESS BIT 6
5	DATA BIT 5	55	INT ENABLE IN, PIO2
6	INT ENABLE OUT, PIOØ	58	INT ENABLE OUT, PIO3
7	INT ENABLE IN, PIOØ	59	+5 volts
8	DATA BIT 3	60	+5 volts
12	DATA BIT 6	61	+5 volts

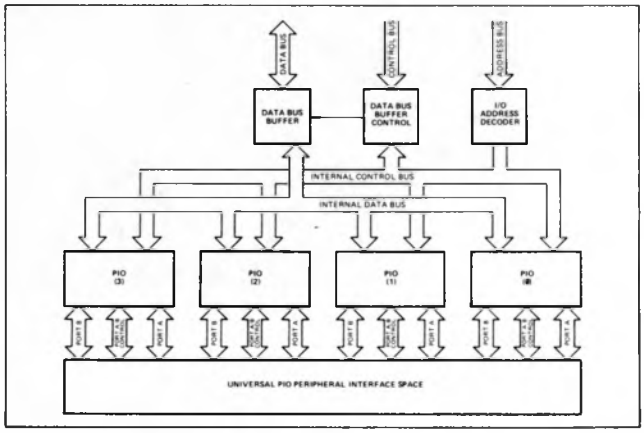
NOTE: PINS 9–11, 14–25, 27, 28, 31–54: UNIVERSAL PIO INTERFACE

LÖTSEITE

62	GND	100	ADDRESS BIT 3
63	GND	101	ADDRESS BIT 2
64	GND	102	ADDRESS BIT 1
65	INT ENABLE OUT PIO1	103	ADDRESS BIT Ø
66	INT ENABLE IN, PIO1	115	$\overline{M1}$
68	DATA BIT 4	116	$\overline{RD}$
71	DATA BIT 2	118	INT ENABLE IN, PIO3
73	DATA BIT 7	119	INT ENABLE OUT, PIO2
75	DATA BIT 1	120	GND
79	$\overline{INTERRUPT}$	121	GND
98	ADDRESS BIT 4	122	GND
99	$\overline{\Phi}$		

NOTE: PINS 56, 57, 67, 69, 70, 72, 74, 76–78, 80–97, 104–114, 117: UNIVERSAL PIO INTERFACE

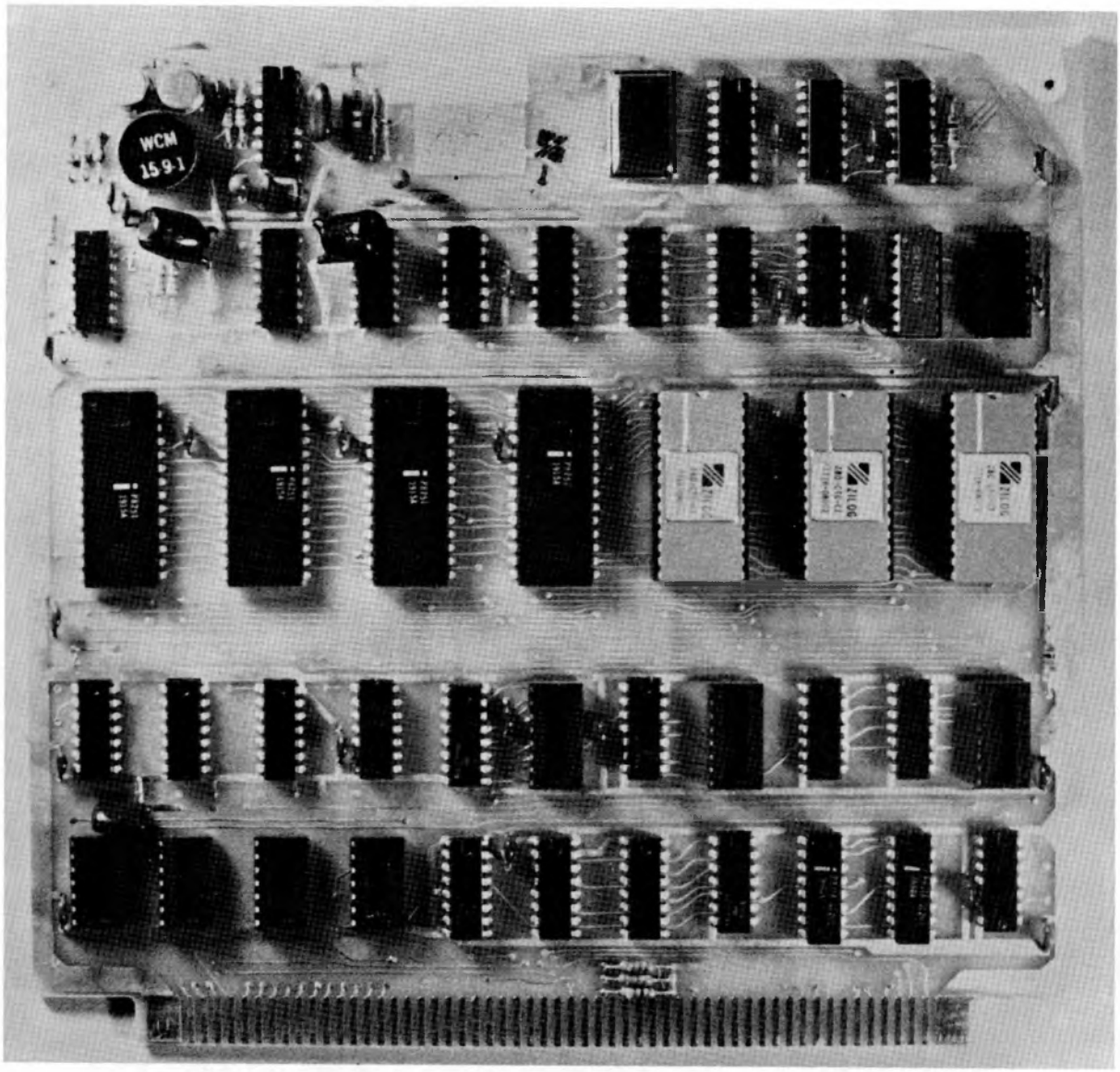
Blockschaltbild



Z80-SIB

Zilog

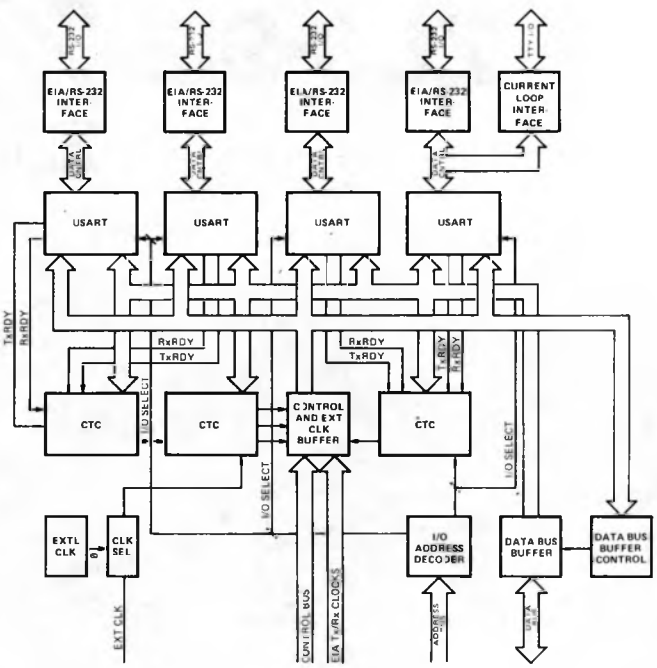
**Serien-
Ein/Ausgabe-
Erweiterung**



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5" × 7,7"

Beschreibung:

Die Baugruppe Z80-SIB erlaubt die Erweiterung eines auf Z80-MCB-Basis aufgebauten Computers um 4 Duplex-Serien-Ein/Ausgabe-Kanäle.
Jeder dieser Kanäle ist von der Z80-CPU über die Anwendersoftware programmierbar.
Die Verwendung von Z80-CTC-Bausteinen, von denen 3 auf der Baugruppe untergebracht sind und damit 12 Zähler/Zeitgeberknaäle zur Verfügung stellen, erlauben volle Ausnutzung der Z80-Interrupt-Fähigkeiten durch jeden der seriellen Duplex-Übertragungskanäle.
Eine der 4 Serienschnittstellen kann über Drahtbrücke vom Anwender als RS-232C oder 20 mA-Stromschleifenschnittstelle festgelget werden. Die Festlegung der Datenübertragungsrate und die Erzeugung der zugehörigen Bezugsfrequenz erfolgt durch die Baugruppe über einen Z80-CTC unter Kontrolle des Anwenderprogramms. Damit lassen sich ohne software-implementierte Zählschleifen-Übertragungsfrequenzen zwischen 50...9600 Baud (asynchron) bzw. 0... 56000 Baud (synchron) vorprogrammieren.
Freie 16 pin-Sockel erlauben das Einfügen anwenderspezifischer Schnittstellen. Die Baugruppe erfordert lediglich eine +5 V-Spannungsversorgung.



Blockschaltung

Technische Daten:

Spannungsversorgung +5 V ± 5%
Umgebungstemperatur 0...50°C
Max. Stromaufnahme 1,5 A

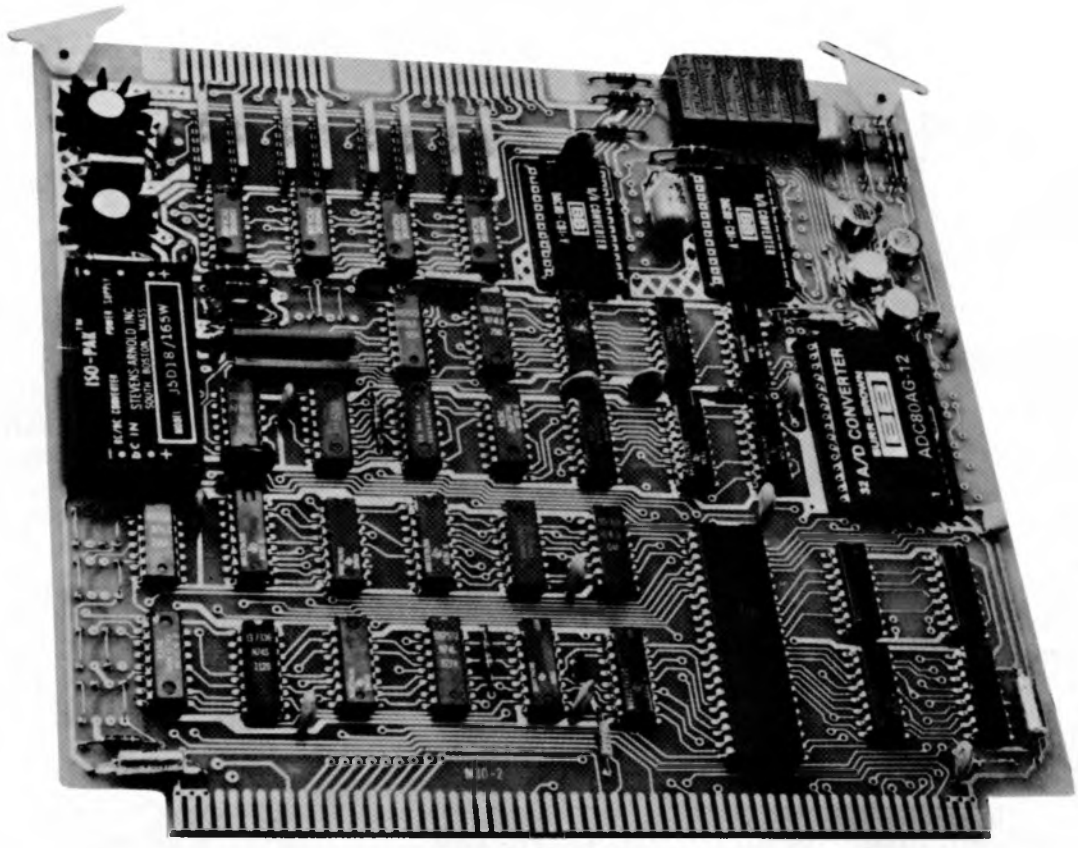
Pin-Belegung

PIN	DESCRIPTION
1, 3, 59–61	+5V P.S.
4	IORQ–
5	DB5
8	DB3
9	MASTER RESET
12	DB6
13	DB0
23	WR–
26	AB7
29	AB5
30	AB6
62–64, 120–122	GND
68	DB4
71	DB2
73	DB7
75	DB1
79	INT–
98	AB4
99	Φ–. (SYSTEM CLOCK–)
100	AB3
101	AB2
102	AB1
103	AB0
115	M1–
116	RD–

Z80-AIO
Z80-AIB

Zilog

Analog – Ein/Ausgabe-Baugruppe



Spezifikationen:

- ☐ 12 bit-Genauigkeit bei Analog Ein- und Ausgabe
- ☐ Eingangs-Spannung: 2.5 mV... 10 V Voller Meßbereich
- ☐ 32 gemultiplexte Analog-Eingänge und 2 Analog-Ausgänge
- ☐ Volle elektromechanische und elektronische Kompatibilität zur Z80-Baugruppen-Serie
- ☐ Maximale Datenrate: 45 μsec pro Kanal
- ☐ Ausgabe unipolar oder bipolar im Spannungsbereich 2,5 mV... 10 V (doppelt gepuffert)
- ☐ Eine einzige +5 V-Versorgungsspannung

6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5"×7,7"

Technische Daten:

Analog Eingabe

Eingangssignale

Anzahl der Kanäle	32 single-ended/16 differential
ADC Gain Ranges (Jumper Selectable)	0–5V, 0–10V, ±2.5V, ±5V, ±10V
Amplifier Gain Ranges (Resistor Programmable)	1 to 1000
Maximum Input Voltage without Damage	±26 volts
Input Impedance	100 mΩ, 10 pF OFF Channel 100 mΩ, 100 pF ON Channel
Bias Current	20 nA
Differential Bias Current	10 nA

Übertragungskenndaten

Genauigkeit	12 bits
Throughput Time (max.) G = 1	45 µsec/channel

Genauigkeit

System Accuracy at +25°C (max.) (1)	±0.025%FSR (2)
Linearity	±1/2 LSB
Differential Linearity	±1/2 LSB
Quantizing Error	±1/2 LSB
Monotonicity (3)	Guaranteed 0°C to +70°C

Ausgabe (nur bei Z80 AIO)

Ausgangssignale

Anzahl der Kanäle	2
Output Voltage Ranges (Strap Selectable)	±10V, 0 to 10V, ±5V, 0 to 5V, ±2.5V at 5 mA
Output Impedance	1Ω

Übertragungskenndaten

Resolution	12 bits
Output Settling Time (max.)	10 µsec

Genauigkeit

Output Accuracy	±0.0125% FSR
Temperature Coefficient of Accuracy	±30 ppm of FSR/°C

NOTES:

1. Includes offset errors, gain errors, linearity errors at gain = 1.
2. FSR mean Full Scale Range.
3. No missing codes guaranteed.
4. Includes offset drift, gain drift and linearity drift.

Typical at 25°C and rated power supplies unless otherwise noted.

Temperaturstabilität

System Accuracy Drift (max.) G = 1	±30 ppm of FSR/°C

Dynamische Genauigkeit

Sample and Hold Aperture Time	30 ns
Aperture Time Uncertainty	±5 ns
Differential Amplifier CMR	74 dB (DC to 1 kHz)
Channel Crosstalk	80 dB down at 1 kHz, for OFF channel to ON channel

Leistungsaufnahme

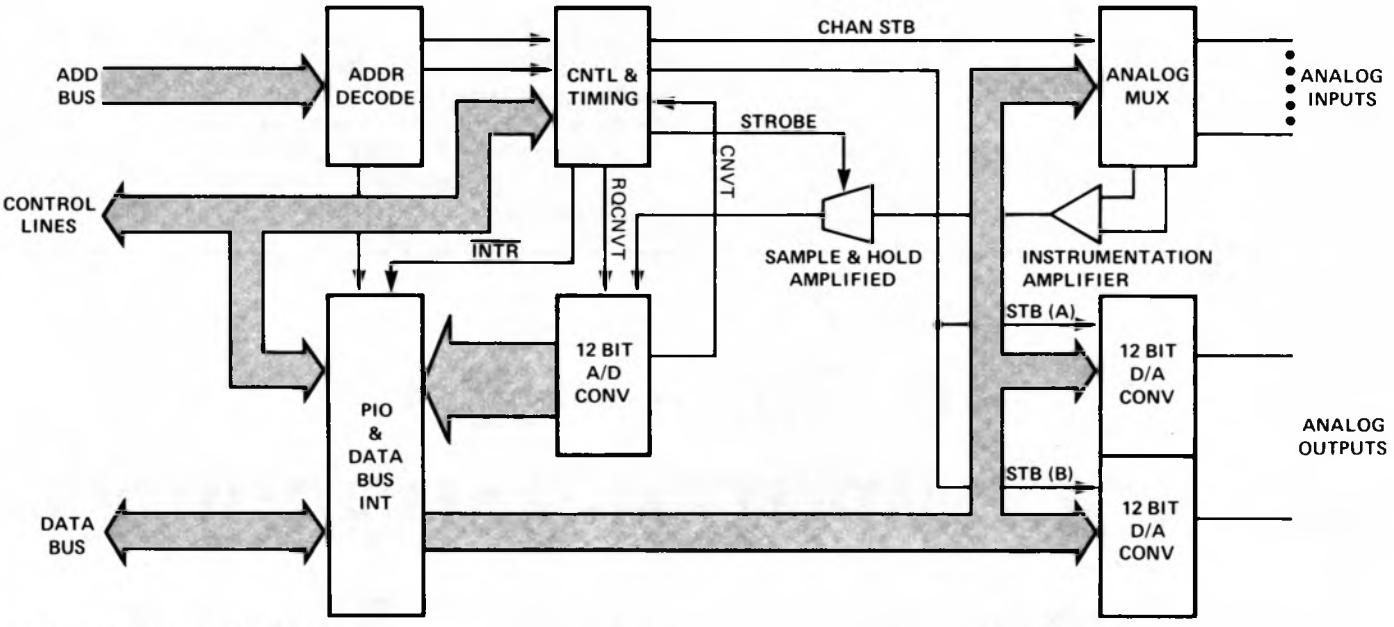
+5 V ± 5%, 1,5 A

Stecker

Ansley: cable. . . . . P/N 171-26
Ansley: socket. . . . . P/N 609-258

Umgebung

Betriebstemperatur	0°C to +70°C
Lagertemperatur	-25°C to +85°C
Feuchtigkeit	95% noncondensing



Blockschaltbild des Z80-AIO

Systembus-Pinbelegung

1	+5V	13	DBØ	63	GND	100	AB3
2	+5V	23	$\overline{WR}$	64	GND	101	AB2
3	+5V	26	AB7	68	DB4	102	AB1
4	$\overline{IORQ}$	29	AB5	71	DB2	103	ABØ
5	DB5	30	AB6	73	DB7	115	$\overline{M1}$
6	IEO	59	+5V	75	DB1	116	$\overline{RD}$
7	IEI	60	+5V	79	$\overline{INT}$	120	GND
8	DB3	61	+5V	98	AB4	121	GND
12	DB6	62	GND	99	$\overline{Ø}$	122	GND

Ein/Ausgabestecker-Pinbelegung

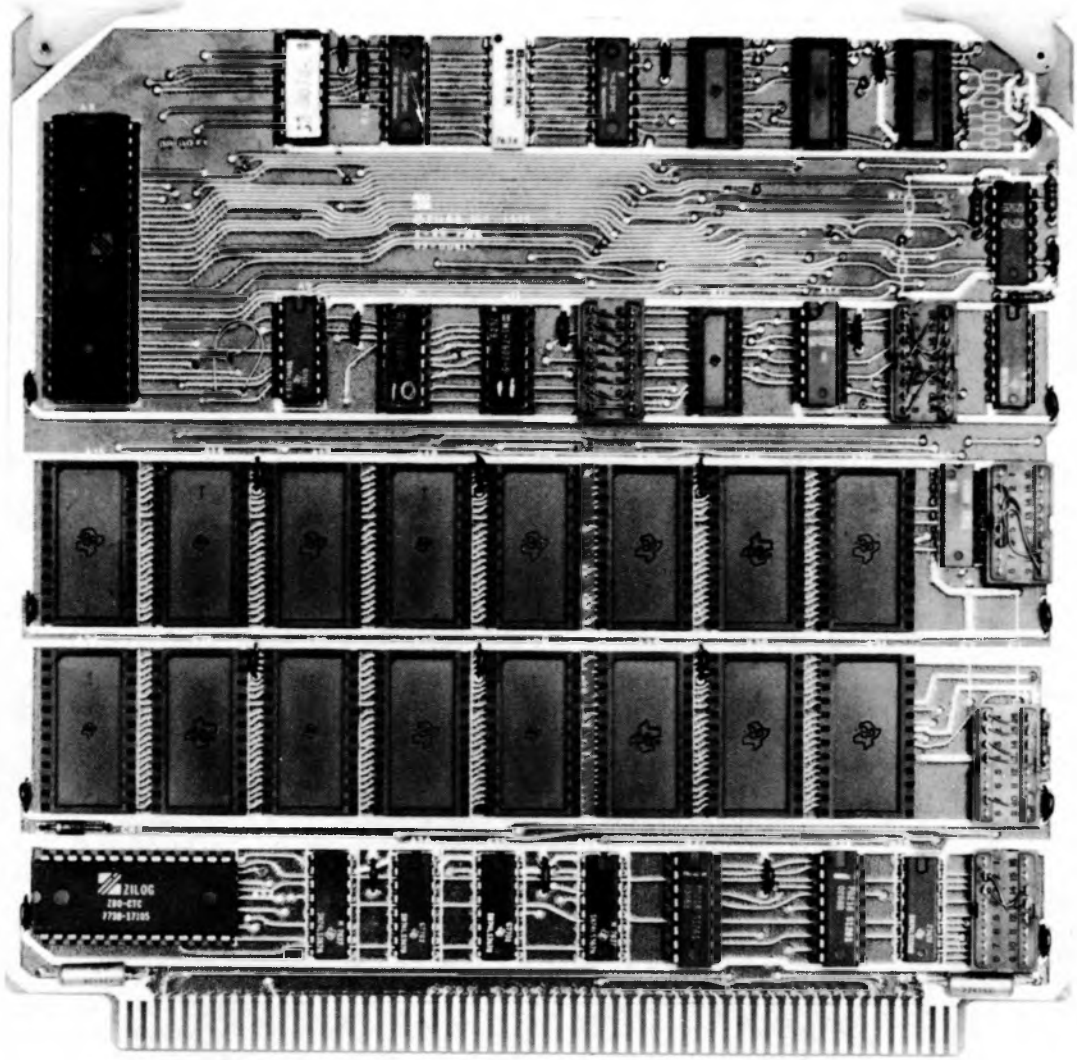
	SIGNAL	P2
ANALOG INPUTS	CH 23, or CH 7 RTN	1, 2
	CH 22, or CH 6 RTN	3
	CH 21, or CH 5 RTN	4
	CH 20, or CH 4 RTN	5
	CH 19, or CH 3 RTN	6
	CH 18, or CH 2 RTN	7
	CH 17, or CH 1 RTN	8
	CH 16, or CH 0 RTN	9
	CH 7	10
	CH 6	11
	CH 5	12
	CH 4	13
	CH 3	14
	CH 2	15
	CH 1	16
	CH 0	17
ANALOG OUTPUTS	DAC–GND	18
	DAC–FB	19
	Analog Common	20
	DAC–GND	21
	DAC–OUT	22
	DAC–OUT	23
	DAC–FB	24
VOLTAGE OUTPUTS	-15V	25
	+15V	26

	SIGNAL	P3
	Remote Common	1, 2
	-15V	3
	+15V	4
	Analog Common	5
	Analog Common	6
	Analog Common	7
	Analog Common	8
	Analog Common	9
	Analog Common	10
ANALOG INPUTS	CH 30, or CH 14 RTN	11
	CH 31, or CH 15 RTN	12
	CH 28, or CH 12 RTN	13
	CH 29, or CH 13 RTN	14
	CH 26, or CH 10 RTN	15
	CH 27, or CH 11 RTN	16
	CH 24, or CH 8 RTN	17
	CH 25, or CH 9 RTN	18
	CH 14	19
	CH 15	20
	CH 12	21
	CH 13	22
	CH 10	23
	CH 11	24
	CH 8	25
	CH 9	26

Z80 PMB

Zilog

Ein/Ausgabe
Speichererweiterung



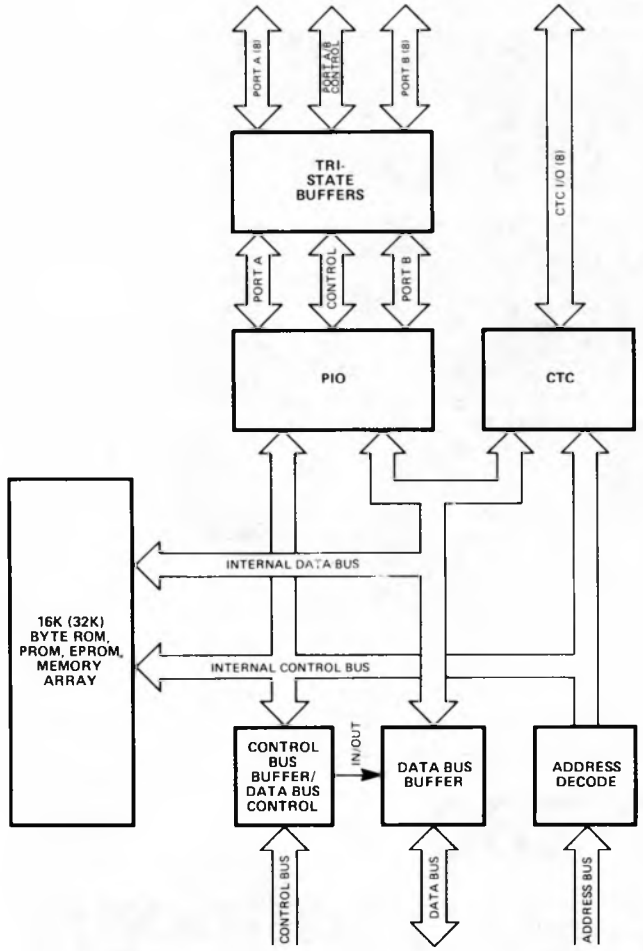
6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5″ × 7,7″

Beschreibung

Die Baugruppe Z80-PMB eignet sich besonders zum Einsatz in Systemen mit umfangreicherem Festwertspeicherbereich. Sie nimmt max. 16 kByte Festwertspeicher bei Verwendung von 2708-EPROMs oder 6381-PROMs auf bzw. 32 kByte bei Verwendung von i2716-EPROM's oder 2316 ROM's.
Bei Verwendung von i2716-Bausteinen ist für den Betrieb lediglich eine +5 V-Versorgungsspannung erforderlich.
Weitere einsetzbare Typen sind 82 S 181, HM 7640, HM 7641, 2516 und 2758 und selbstverständlich deren Äquivalente.
Zusätzlich verfügt die Baugruppe über einen Z80-PIO-Baustein (16 Parallele-Ein/Ausgabe-Leitungen und 4 Handshaking-Anschlüsse) und einen Z80-CTC-Baustein (4 Zähler/Zeitgeberkanäle) zur Ein/Ausgabe-Erweiterung des Systems.

Technische Daten:		**Without Memory**	**2708 Max**	**2716 Max**	**6381 Max**
Stromversorgung	5V	0.60A	.28A	2.28A	3.40A
(ohne Speicher-	−5V	–	.96A	–	–
bausteine)	+12V	–	1.28A	–	–

Umgebungstemperatur: 0°–50°C

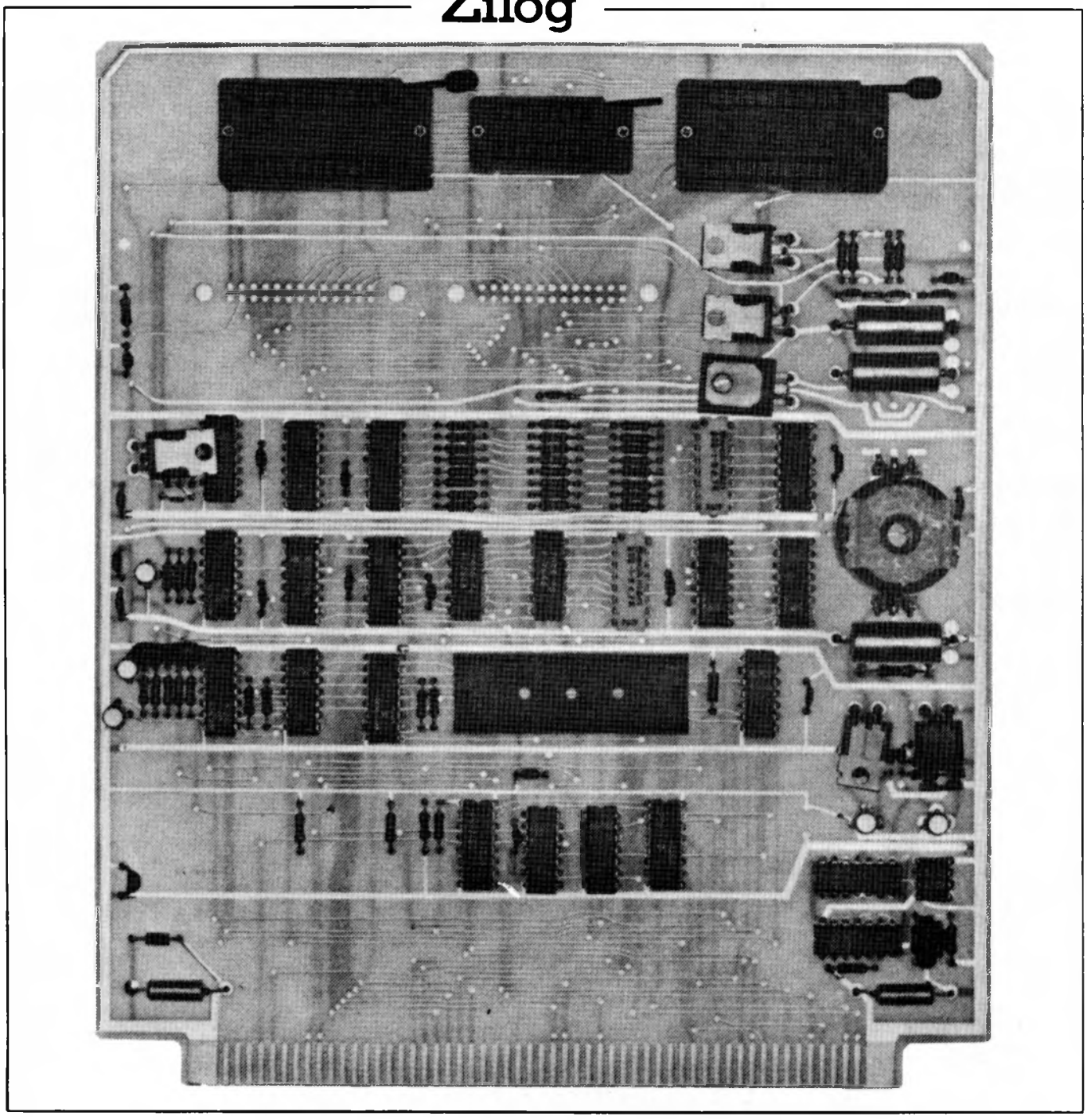


Blockschaltung

Z80-PPB

Zilog

Baugruppe zur PROM-Programmierung



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5"×7,7"

Beschreibung:

Die Schaltung dient zur Programmierung aller modernen, handelsüblichen bipolaren und UV-löschbaren programmierbaren Festwertspeicher-Bausteinen.
Zur Programmierung ist die zugehörige Software für den Betrieb mit einem ZILOG-Computersystem verfügbar. Die Blockschaltung ist im folgenden Bild dargestellt.

Standardsoftware wird für die bipolaren HARRIS-PROM's HM 7610, 7611, 7620, 7621 (16 pin), 7640, 7641 (24 pin) und die UV-löschbaren PROM's 2704 und 2708 mitgeliefert. Für alle PROM's sind in der Software die Funktionen PROGRAM, VERIFY, LIST und DUPLICATE realisiert.
Diese Software (Object-Code) läuft unter dem Betriebssystem RIO der ZILOG-Computersysteme der Serien MCZ bzw. ZDS. Hierfür sind zwei verschiedene Versionen der PPB-Baugruppe lieferbar: MCZ/PPB und ZDS/PPB.

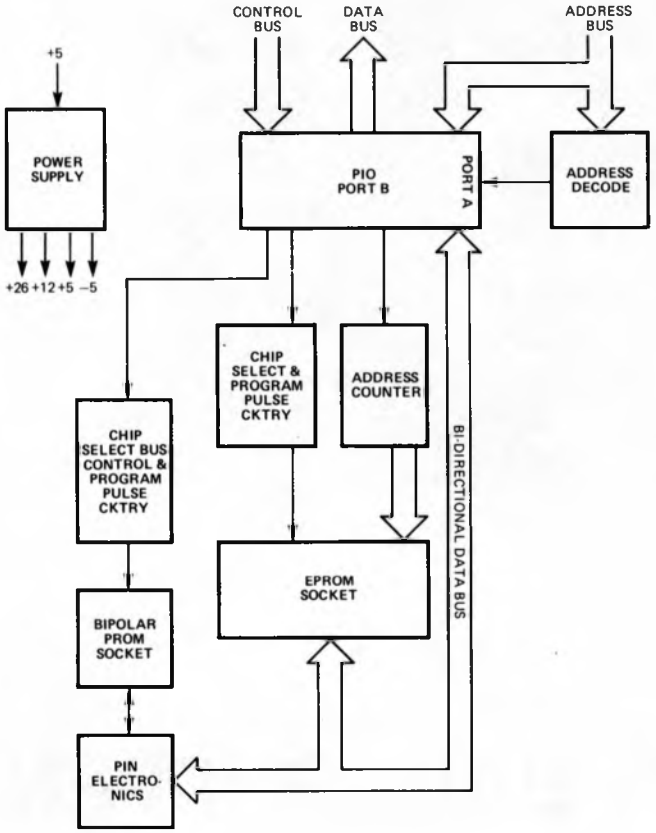
Technische Daten:

Stromversorgung:	+5 V ± 5%
Stromaufnahme:	1,5 A (Standby) 2,5 A (während Programmierung)
Umgebungs-temperatur	0 . . . 50°C

Bestellbezeichnungen

MCZ/PPB: Baugruppe Z80-PPB zum Einsatz zusammen mit MCB-Baugruppen oder in einem Computersystem der MCZ-1-Serie

ZDS/PPB: Baugruppe Z80-PPB zum Einsatz in Mikrocomputer-Entwicklungssystemen der ZDS-1-Serie



Blockschaltbild

Pin Belegung

ZDS/PPB:

9	ADDRESS BIT 6	47	DATA BIT 4
10	ADDRESS BIT 7	48	DATA BIT 5
16	ADDRESS BIT Ø	51	DATA BIT 6
17	ADDRESS BIT 1	52	DATA BIT 7
18	ADDRESS BIT 2	54	DATA BIT Ø
19	ADDRESS BIT 3	55	DATA BIT 1
20	ADDRESS BIT 4	56	DATA BIT 3
21	ADDRESS BIT 5	57	DATA BIT 2
24	$\overline{Ø}$	88	$\overline{IEI}$
28	$\overline{M1}$	109	$\overline{INT}$
30	$\overline{IORQ}$	112	$\overline{IEO}$
45	$\overline{RD}$		

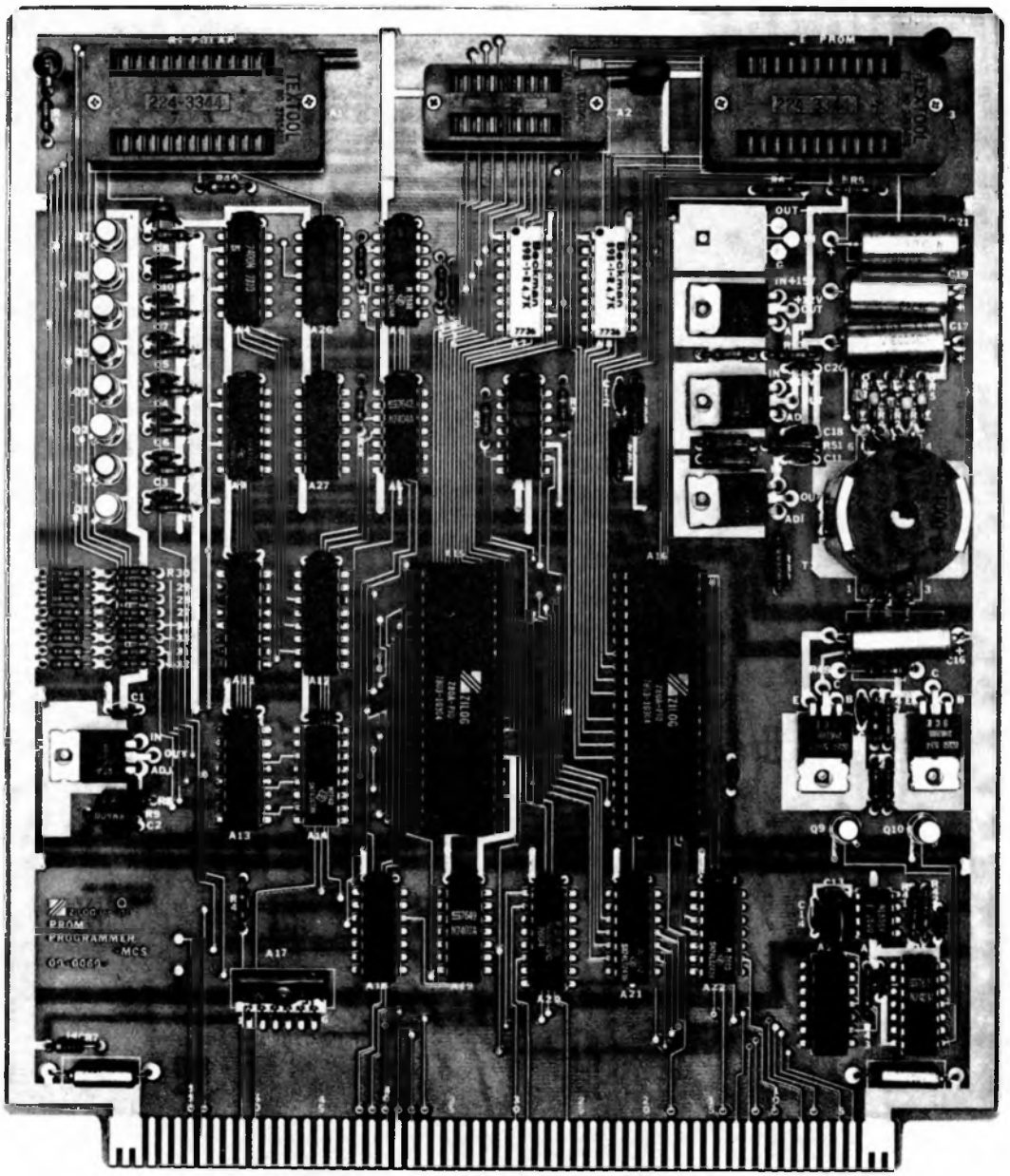
MCZ/PPB:

4	$\overline{IORQ}$	73	DATA BIT 7
5	DATA BIT 5	75	DATA BIT 1
8	DATA BIT 3	79	$\overline{INT}$
12	DATA BIT 6	98	ADDRESS BIT 4
13	DATA BIT Ø	99	$\overline{Ø}$
26	ADDRESS BIT 7	100	ADDRESS BIT 3
29	ADDRESS BIT 5	101	ADDRESS BIT 2
30	ADDRESS BIT 6	102	ADDRESS BIT 1
50	$\overline{IEI}$	103	ADDRESS BIT Ø
51	$\overline{IEO}$	115	$\overline{M1}$
68	DATA BIT 4	116	$\overline{RD}$
71	DATA BIT 2		

Z80-PPB/16

Zilog

Baugruppe zur PROM-Programmierung



6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5"×7,7"

Folgende Festwertspeicherbausteine (PROM's) sind mit dem PPB/16 und der dazu verfügbaren Software zu programmieren:

i2716	(i2758)		
82S126	82S129	82S130	82S131
82S140	82S141	82S180	82S181
82S2708			

Technische Daten:

Zu programmierende PROM-Typen

24-pin EPROM i2716 (2758)

24-pin Bipolar–	82S140	(512 x 8)
	82S141	(512 x 8)
	82S180	(1024 x 8)
	82S181	(1024 x 8)
	82S2708	(1024 x 8)
16-pin Bipolar–	82S126	(256 x 4)
	82S129	(256 x 4)
	82S130	(512 x 4)
	82S131	(512 x 4)

Stromversorgung:	
Spannung:	+5 V ± 5%
Stromaufnahme:	1,5 A (standby)
	2,5 A (während Programmierung)
Umgebungstemperatur:	0 . . . 50°C

Bestellbezeichnungen

MCZ/PPB 16: Baugruppe Z80-PPB-16 zum Einsatz zusammen mit MCB-Baugruppen oder in einem Computersystem der MCZ-1-Serie

ZDS/PPB 16: Baugruppe Z80-PPB 16 zum Einsatz in Mikrocomputerentwicklungssystemen der ZDS-1-Serie

MCZ/PPB-16 Pin-Belegung

1, 3, 59–61	+5V P.S.
4	IORQ–
5	DB5
8	DB3
12	DB6
13	DB0
26	AB7
29	AB5
30	AB6
50	IEI · PPB · PIO
51	IEO · PPB · PIO
62–64, 120–122	GND
68	DB4
71	DB2
73	DB7
75	DB1
79	INT–
98	AB4
99	Φ–. (System Clock–)
100	AB3
101	AB2
102	AB1
103	AB0
115	M1–
116	RD–

ZDS/PPB-16 Pin-Belegung

1–3, 59–61	+5V
9	A6
10	A7
16	A0
17	A1
18	A2
19	A3
20	A4
21	A5
24	Φ (System Clock)
28	M1–
30	IORQ–
45	RD–
47	D4
48	D5
51	D6
52	D7
54	D0
55	D1
56	D3
57	D2
62–64, 120–122	GND
88	IEI
109	INT–
112	IEO

Z80-SCC
Z80-EXB
Z80-WWB
Z80-SCE 4

Zilog

Baugruppenträger und Zubehör für die MCB-Serie

Z80-SCC:	Steckrahmen für bis zu 9 Baugruppen der MCB-Serie
Z80-EXB:	Verlängerungsplatine für Baugruppen der MCB-Serie
Z80-WWB:	Wire-Wrap-Platine zum raschen, kostensparenden Aufbau anwendungsspezifischer Zusammenhang mit den Baugruppen der MCB-Serie eingesetzt werden sollen.
Z80-SCE4:	Tischgehäuse mit 4 Steckplätzen mit 5 V Netzteil und Ventilator. (Foto s. Zentralgerät des MCZ 1-05)

6. ZILOG-Z80 MIKROCOMPUTER-BAUGRUPPEN im Standard-Format 7,5”×7,7”

MO 1
MO 2
OS 2.1

Zilog **Betriebs-Software**

Software-Beschreibung für die Z80-MCB-Mikrocomputer MO 1 bzw. MO 3)

Für die Z80-Baugruppen im 7,5″×7,7″ Standardformat (Z80-MCB) sind Monitor-Programme im Umfang von 1 und 3 kByte und ein komplettes Betriebssystem, das Plattenverwaltung, Editor und Assembler umfaßt, verfügbar.

1. Monitorprogramme

Die Monitor-Programme stellen beide jeweils ein einfaches Betriebssoftwarepaket dar, das als Firmware (also in einem Festwertspeicher gespeichertes Programm) geliefert wird; die PROM's können direkt in die hierfür auf der MCB-Baugruppe montierten Fassungen eingesteckt werden. Der Monitor erlaubt die sofortige Inbetriebnahme des MCB-Computers und erspart dem Anwender die Erstellung gewisser Basis-Routinen, wie Serien-Ein/Ausgabe usw. Darüber hinaus ist mit dem Monitor sogar die Durchführung von Programm-Entwicklungen und -Tests möglich; die verfügbaren Monitor-Kommandos sind im Folgenden beschrieben.

1.1 1 kByte Monitor MO 1

Hardware-Erfordernisse des Monitors

Speicherorganisation

Der 1 kByte Monitor ist in dezimal 1024 (bzw. hexadezimal 400) Festwertspeicherstellen abgelegt. Er benötigt darüber hinaus 35 Byte Schreib/Lesespeicher zuzüglich einem Stack-Bereich im RAM. Dieser Schreib/Lesespeicher liegt am oberen Ende der zweiten Seite, also ab Adresse 2000H (eine Seite entspricht 4 kByte) der darauffolgende Stackspeicher beginnt dann entsprechend bei Speicherstelle 1FDEH und „wächst" in Richtung niederwertigerer Speicheradressen. Über den übrigen Speicherbereich kann frei verfügt werden.

Anschluß eines Bedienungs-Terminals

Über die RS 232- bzw. die 20 mA-Schnittstelle des MCB (bzw. MCB/E) kann das Monitorprogramm mit jedem beliebigen Terminal verkehren, das 8 bit-ASCII-Code asynchron mit oder ohne Parity-Bit seriell überträgt. Es sind 14 verschiedene Übertragungsraten zugelassen, die auf der MCB-Baugruppe über die 4 Schalter einstellbar sind.

Soweit die Fähigkeit des Monitors zur Dateiausgabe bzw. Dateieingabe genutzt werden soll, muß das Terminal über ein Datenspeichermedium verfügen, das softwaremäßig einer Lochstreifen-Leser/Stanzer-Kombination entspricht. Als Start- und Stopsignal muß dieses Kommunikationsmedium die ASCII-Zeichen 11H und 13H erkennen können, oder an geeigneter Stelle manuell zu starten oder zu stoppen sein.

Software-Beschreibung des Monitors

Verhältnis zu evtl. Anwenderprogrammen

Anwenderprogramme, die zusammen mit dem Monitor eingesetzt werden, können auf eine Reihe von dessen Unterprogrammen zugreifen.

Formale Regeln

Sämtliche Kommandos können durch ihren ersten Buchstaben abgekürzt oder auch bis zu jeder beliebigen Länge ausgeschrieben werden, da lediglich der erste Buchstabe in einer neuen Eingabe-Zeile als Schlüsselzeichen für ein empfangenes Kommando benutzt wird. Nicht verstandene Kommando-Eingaben werden vom Monitor durch Angabe des ASCII-Zeichens

?

quittiert, worauf eine neue Kommandoeingabe erwartet wird.
Sämtliche Felder werden durch Leerzeichen begrenzt (= „Delimiters").
Außer den Daten im SET-Kommando können alle Zahlen formatfrei (= "free form") hexadezimal (auch ohne führende Nullen) eingegeben werden.
Die als ein Datum eingebbare Zahl ist auf 4 Zeichen begrenzt; werden mehr als 4 Zeichen eingegeben, werden die letzten 4 Zeichen als gültige Zahl interpretiert.

Beschreibung der einzelnen Kommandos des 1 kB-Monitors

DISPLAY adr n

veranlaßt die Ausgabe des Inhalts der n folgenden Speicherstellen ab (inklusive) der Speicherstelle „adr".

SET adr dat1 dat2 . . . . datm

speichert die angegebenen Datenwörter (jedes als 2 Hexadezimalzeichen einzugeben!!) ab (inklusive) der Speicheradresse „adr" ab.
Unmittelbar auf das letzte Zeichen von datm muß ein Wagenrücklaufzeichen ("carriage return") eingegeben werden.

PUNCH anfadr endadr

gibt den Speicherinhalt ab (inklusiv) „anfadr" bis (inklusiv) „endadr" auf den Lochstreifenstanzer der Kommandokonsole aus, wobei der Stanzvorgang automatisch durch Ausgabe eines „Tape-on-Zeichens" (12H ASCII) gestartet und durch Ausgabe von „Tape-off" (14H ASCII) beendet wird. Nach „Tape-off" wird noch das Stanzen eines Stücks Leerstreifen veranlaßt.

8. ZILOG-Z80 MIKROCOMPUTER-SOFTWARE

LOAD
Einlesen von Daten von der Lochstreifeneinheit der Kommandokonsole in den Speicher, wobei die Anfangsadresse auf dem Lochstreifen eingestanzt sein muß und der Lochstreifen natürlich vorher (und zwar ein kleines Stück vor Vorkommen des ersten lesbaren Zeichens) in den Leser eingelegt werden muß.
Durch ein „Reader On"-Signal wird der Leser vom Programm automatisch gestartet und am Ende der eingelesenen Datei wieder abgeschaltet.
Im Fehlerfall (= Check-Summe nicht richtig) wird der Leser ausgeschaltet und eine Nachricht ausgegeben, da anzunehmen ist, daß Daten verloren gegangen sind.

JUMP adr
veranlaßt einen unbedingten Sprung auf Adresse adr und Ausführung des dort befindlichen Programms.

GO
veranlaßt Ausführung des Programms ab der aktuellen Speicheradresse

BREAK adr
setzt einen Haltepunkt bei Speicheradresse adr, wobei zuvor evtl. vorhandene andere Haltepunkt-Bedingungen außer Kraft gesetzt werden.
Wird der optimale Parameter n als > 1 spezifiziert, so wird die Programmausführung erst dann abgebrochen, wenn die angegebene Programm-Speicheradresse n — mal angesprungen wurde. Im Abschnitt 4.3.4 wird auf die Haltepunkte noch gesondert eingegangen.

REGISTER reg
Mit diesem Kommando können CPU-Registerinhalte ausgegeben und modifiziert werden.
Auf die Angabe des Registernamens hat statt des Wagenrücklaufzeichens ein Leerzeichen (Blank) zu folgen; vom Monitor wird daraufhin der Registerinhalt in der gleichen Zeile ausgegeben, worauf der Benutzer die Ausführung des Kommandos mit der Eingabe eines Wagenrücklaufzeichens abschließen kann.
Soll stattdessen das nächstfolgende Register ausgegeben werden, muß der Benutzer statt des Wagenrücklauf- ein Zeilenvorschub-Zeichen eingeben.
Soll der Inhalt eines eben angezeigten Registers modifiziert werden, ist unmittelbar nach dem Leerzeichen der gewünschte Registerinhalt einzugeben. Danach beendet wieder ein Wagenrücklauf- oder Zeilenvorschub-Zeichen den Vorgang.
Die Registerinhalte werden von RAM-Speicherstellen aus angezeigt, in denen sie vorher vom Monitor-Programm abgelegt werden.
Die Reihenfolge, in der die Register bei fortlaufender Eingabe von Zeilenvorschubzeichen in REGISTER-Kommandos behandelt werden ist folgende:
A, B, C, D, E, F, H, L, I, A', B', C', D', E', F', H', L', PC, SP, IX, IY.

Haltepunkt-Behandlung

Die Software-Haltepunkte des Z80-Monitors dienen zur Fehlersuche und -Beseitigung in Anwenderprogrammen. Wird eine Haltepunktbedingung bei der Ausführung des Anwenderprogramms als gültig erkannt, wird dessen Ausführung abgebrochen, alle Registerinhalte in den hierfür vom Monitor reservierten Speicherbereich kopiert und eine Meldung ausgegeben, daß und bei welcher Speicherstelle der Haltepunkt erreicht wurde. Eine beliebige Anzahl von Haltepunkten kann manuell gesetzt werden, indem als Haltepunktbedingung die Adresse 0FFH angegeben wird.

Besonders zu beachten ist, daß man den Stackpointer initialisieren muß, wenn man ein Anwenderprogramm ablaufen lassen und durch das Kommando BREAKPOINT behandeln will.

Dies kann geschehen
- durch Initialisierung im Anwenderprogramm selbst oder
- durch direktes Setzen des CPU-Stackpointers durch das Kommando REGISTER

Beliebig viele Haltepunkte können manuell durch Angabe der Haltepunktadresse 0FFH gesetzt werden. Dabei muß die Haltepunktadresse auf das erste Byte einer Anweisung zeigen, und die ursprüngliche Anweisung muß von Hand wieder auf diesen Speicherplatz geschrieben werden, sobald dieser Haltepunkt nicht mehr benötigt wird.

Hintergrund dieses Verfahrens ist die Tatsache, daß der Monitor grundsätzlich anhält, wenn er annimmt, daß er auf eine physikalisch nichtexistierende Speicherstelle (wie FF eine darstellt) zugreift.

Der Haltepunkt wird erst dann wirksam, wenn er n-mal angesprungen wurde. Durch diese Einrichtung ergeben sich einige Restriktionen für das Anwender-Programm:

1.2 3 kByte Monitor- und Fehlersuchprogramm MO3

Mit dem Monitor-Programm Z80-MO3 steht ein Firmware-Betriebssystem zur Durchführung von Systemoperationen und Fehlersuche zur Verfüung. Darüberhinaus sind in dieser Firmware grundsätzliche Ein/Ausgaberoutinen und Floppy-Disk-Steuerungsroutinen untergebracht. Als Anfangsadresse des 3 kByte Festwertspeichers ist die Adresse 0 vorgesehen; als Schreib/Lesespeicher ist 1 kByte RAM erforderlich, von dem 256 Byte als Stack-Speicher reserviert sind. Der Monitor MO3 versteht folgende Kommandos:

DISPLAY adr n:
Ausgabe der angegebenen Anzahl Byt's Speicherinhalte ab der angegebenen Adresse adr auf der Systemkonsole. Ist n nicht spezifiziert, so wird nur 1 Byte ausgegeben und zwar das, das den Inhalt der angegebenen Adresse adr darstellt.

Bei jeder dieser Ausgaben wird zunächst die Adresse ausgegeben und darauf folgend der Inhalt dieser Adresse, der von einem Leerzeichen gefolgt wird. Der Benutzer kann den Inhalt dieser Speicherstelle ändern, indem er jetzt einfach den gewünschten Speicherinhalt und daraufhin einen Zeilenvorschub eingibt. Ist kein neuer Inhalt gegeben, bleibt der Speicherinhalt unverändert.

Die Angabe eines jeden Zeilenvorschubzeichens bewirkt das Weiterschalten zur nächstfolgenden Adresse unter Angabe des Speicherstelleninhalts wie eben beschrieben. Soll dieser Vorgang abgebrochen werden, ist ein Zeichen Q („Quit") zu geben, das von einem Wagenrücklaufzeichen gefolgt ist, wodurch das Kommando abgebrochen wird.

Zu bemerken ist noch, daß bei der Ausgabe einer ganzen Zeichenfolge (durch Angabe der Option n) die Speicherinhalte sowohl in hexadezimaler aus auch in ASC II-Darstellung ausgegeben werden. Die ASC II-Darstellung erscheint rechts auf dem Bildschirm zwischen Sternchen, wobei sämtliche nicht abdruckbaren Zeichen als Punkte dargestellt werden.

SET adr daten daten daten .....
Dieses Kommando speichert die angegebenen Datenwörter in aufeinanderfolgende Speicherstellen ab der angegebenen Adresse. Ein Wagenrücklaufzeichen beendet diese Liste.

JUMP adr
bewirkt einen unbedingten Sprung zu der angegebenen Adresse und dort Ausführung der aufgefundenen Befehle. Sämtliche Register werden vor dem Sprung abgespeichert.

GO
bewirkt Fortsetzung bzw. Ausführung des Programms ab der Speicherstelle, die durch den momentanen Inhalt des Befehlszählers definiert ist.

BREAK adr n
setzt auf die angegebene Adresse einen Break-Point, wobei jeglicher vorher angegebener Break-Point rückgesetzt wird, die Angabe von n ist optional. Ist n spezifiziert, so wird die Programmausführung erst nach dem n-maligen Auftreten des angegebenen Break-Points unterbrochen.

REGISTER reg
erlaubt das Ausgeben und Abändern von Registerinhalten. Wird kein Registername spezifiziert, so werden sämtliche CPU-Register in einer Zeile ausgegeben. Ist ein Registername reg angegeben, so wird lediglich der Inhalt des angegebenen Registers ausgegeben und ein darauf folgendes Leerzeichen. Der Inhalt dieses Registers kann durch Angabe eines Datenwortes mit nachfolgendem Registerzeichen geändert werden. Soll keine Änderung erfolgen, ist lediglich ein Zeilenvorschubzeichen zu geben. Nach diesem Vorgang wird vom Betriebsprogramm MO3 automatisch der nächste Registerinhalt ausgegeben, und der Vorgang läuft analog dem eben beschriebenen ab. Sollen keine weiteren Registerinhalte ausgegeben und- oder geändert werden, muß ein Zeichen Q (für QUIT) mit einem darauf folgenden Wagenrücklaufzeichen eingegeben werden, wodurch das gesamte Kommando abgebrochen wird.
Die Abfolge, in der die Register ausgegeben werden, ist A, B, C, D, E, F, H, L, I, A′, B′, C′, D′, E′, F′, H′, L′, EX, EY, PC, SP.

Die Registerinhalte werden durch Speicherstellen repräsentiert, in die die entsprechenden Registerinhalte ausgelagert werden; sie werden in die Register bei jedem JUMP oder GO-Kommando wieder übernommen.

NEXT n
bewirkt die Ausführung des nächsten Maschinenbefehls ab dem Befehl der durch den Momentanstand des Befehlszählers definiert ist, und die Ausgabe sämtlicher Registerinhalte nach jeder Ausführung dieses Kommandos. Die Angabe von n ist optional, Standardwertzuweisung ist n = 1.

MOV zieladr absadr n
transportiert den Inhalt eines Speicherblocks von der Adresse absadr zur Adresse zieladr, wobei n die Anzahl der zu übertragenden Zeichen ist. Es gibt keine Restriktion bezüglich der Angabe der Adressen zieladr und absadr oder n außer daß sie positive ganze Zahlen sein müssen und in den Speicherbereich von 64 kByte der Z80-CPU passen müssen.

COMPARE adr 1 adr 2 n
vergleicht die Inhalte von zwei Speicherblöcken, wobei adr 1 und adr 2 die Anfangsadressen der beiden zu vergleichenden Blöcke definieren und n die Anzahl der zu vergleichenden Speicherinhalte innerhalb der beiden Blöcke voneinander unterschiedlich sind, werden die Adressen und die zugehörigen Speicherstelleninhalte auf der Bedienkonsole ausgegeben.
Sämtliche Kommandos können zu einer beliebigen Länge voll ausgeschrieben werden oder aber bis auf ihren ersten Buchstaben abgekürzt werden.
Wird ein Kommando nicht verstanden, wird ein Fragezeichen ausgegeben und die Eingabe eines neuen Kommandos erwartet. Sämtliche Zahlen werden im freien Format hexadezimal eingegeben, wobei führende Nullen unterdrückt werden. Werden mehr als 4 Hexadezimalzeichen eingegeben, so werden die **letzten** vier als Zahlen interpretiert.

1.3 OEM-Betriebssystem OS 2.1

ZILOG bietet seinen OEM-Kunden, die mit dem Baugruppen der MCB-Serie arbeiten ein komplettes Floppy-Disk-basierendes Betriebssystem an.
Die Lieferung erfolgt auf Floppy-Disk und umfaßt das Plattenbetriebssystem, einen Texteditor und Z80-Assembler.
Zum Betrieb mit einer MCB/MDC-basierenden Anwenderhardware ist zusätzlich der vorher beschriebene Monitor MO3 als Bootstrap und I/O-Handler erforderlich.
Umfangreiche Literatur (OS 2.1 Software Users Manual) ist getrennt zu beziehen.

KONTRON
ELEKTRONIK GMBH

7. MIKROCOMPUTER-BAUGRUPPEN IM EINFACHEUROPA-FORMAT DER ECB-FAMILIE

7.1 Übersicht

Die Gesamtheit dieser Produkte bietet durch ihre Vielfalt für fast alle Mikrocomputer-Anwendungsprobleme fertige μC-Hardware- und Grundsoftware-Lösungen, die einfach durch Zusammenstecken der entsprechenden Einzelprodukte realisiert werden.

Die angesprochenen Produktgruppen sind:

- ☐ Die Serie Z80-ECB.
  Es handelt sich dabei um Computerbaugruppen im Einfacheuropaformat, die mit einer Taktfrequenz von 2.5 MHz arbeiten.
- ☐ Die Serie Z80A-ECB.
  Sie entspricht der Z80-ECB-Serie, ist jedoch für Taktfrequenzen bis 4 MHz ausgelegt.
- ☐ Baugruppen und Zubehör, die sowohl für die Z80-ECB als auch für die Z80A-ECB-Serie geeignet sind.
- ☐ Software für die aufgeführten Serien
- ☐ Komplette betriebsbereite Computersysteme auf Basis der obengenannten Produkte.

Folgende Einheiten sind im Folgenden detailliert beschrieben:

Z80A-Baugruppen im Einfacheuropaformat (100×160 mm) mit Betriebsfrequenz 4 MHz

Z80A-ECB/C8 Zentralbaugruppe
(„**E**infacheuropa-**C**omputer-**B**augruppe/**C**PU-Karte")

- Z80A-CPU
- 16 MHz Quarz
- 1 kByte statischer Schreib/Lesespeicher
- Steckplätze für maximal 8 kByte Festwertspeicher
- Vollständige I/O- und Speicheradreßdekodierung
- 2 Achtbit-Parallelschnittstellen (Z80A-PIO)
- 2 Serienschnittstellen (Z80A-SIO) mit normgerechtem RS 232 C und 20 mA Current-Loop-Interface
- DC/DC-Wandler für RS 232 C Interface
- Einstellbare Baudrate über DIP Schalter
- Zweiter getrennter Quarz für Baudratengenerator
- Schmitt Trigger Pufferung aller Bussignale
- Spannungsversorgung 5 V ± 5%.

Z80A-ECB/D32-P Dynamische Speichererweiterung mit Paritätserzeugung/Prüfung
(„**E**infacheuropa-**C**omputer-**B**augruppe/
Dynamische Speichererweiterung mit **P**aritätsprüfung")

- Dynamische Speichererweiterungsbaugruppe zur Arbeit mit CPU's mit max. Arbeitsgeschwindigkeit von 4 MHz mit 32 kByte Kapazität. Die Erzeugung der Signale RAS und CAS, sowie der Versorgungsspannungen −5 V und +12 V erfolgen auf der Baugruppe.
- Eine einzige +5 V-Versorgung.

Z80A-ECB/D32 Dynamische Speichererweiterung
(„**E**infacheuropa-**C**omputer-**B**augruppe/**D**ynamisches RAM")

- Dynamische Speichererweiterungsbaugruppe zur Arbeit mit CPU's mit max. Arbeitsgeschwindigkeit von 4 MHz mit 32 kByte Kapazität. Die Erzeugung der Signale RAS und CAS, sowie der Versorungsspannungen −5 V und +12 V erfolgen auf der Baugruppe.
- Eine einzige +5 V-Versorgung.
- Auch Version mit 16 kByte Kapazität erhältlich (Z80A-ECB/D16)

Z80A-ECB/E Speichererweiterung

- Fassungen für 4 . . . 8 kByte bipolare PROM's
- 1 kByte stat. RAM's für Z80A-Zugriffszeit
- Adreßwahlschalter, getrennt für RAM/PROM-Bereich

Z80A-ECB/F Serien/Parallel-Ein/Ausgabe
(„**E**infacheuropa-**C**omputer-**B**augruppe/**F**loppy-Disk-Controller")

- Allgemeine Ein/Ausgabeplatine mit 2 Vollduplex-Serien- und zwei 8 bit-Parallel-Schnittstellen
- Anschluß von bis zu 8 Floppy-Disk-Laufwerken (Normal- und Mini-Floppy)
- Spannungsversorgung: Eine einzige 5 V-Spannungsquelle
- Volle RS 232 C-Schnittstelle.

Z80A-ECB/I Ein/Ausgabe-Erweiterung
(„**E**infacheuropa-**C**omputer-**B**augruppe/**I**/O-Erweiterung")

- 4 Parallelschnittstellen (2 Stück Z80A-PIO)
- 4 Zeitgeber/Ereigniszählkanäle (1 Stück Z80A-CTC)
- 9 unverdrahtete Fassungen für Puffer

Z80A-ECB/S
(„**E**infacheuropa-**C**omputer-**B**augruppe/**S**peicher-**I**/O-Erweiterung")

- 8 kByte statische Speichererweiterung
- 40 Parallel-Ein/Ausgabe-Leitungen über zwei Z80A-PIO

Z80A-ECB/V
(„**E**infacheuropa-**C**omputer-**B**augruppen/
nicht volatiles RAM")

- 4 kByte CMOS-RAM
- Decoder und Puffer auf der Platine
- Schalter zur Speicherbereichsfestlegung
- Klein-Akku zur Erhaltung der Information im RAM über mehrere Tage bei Unterbrechung der Stromversorgung.

Z80-Baugruppen im Einfach-Europaformat (100×160 mm) mit Betriebsfrequenz 2.5 MHz

Z80-ECB/C8 Zentralbaugruppe
(„**E**infacheuropa-**C**omputer-**B**augruppe/**C**PU-Karte")

- Z80-CPU
- Quarz
- 1 kByte statischer Schreib/Lesespeicher
- Steckplätze für maximal 8 kByte Festwertspeicher
- Vollständige I/O- und Speicheradreßdekodierung
- 2 Achtbit-Parallelschnittstellen (Z80-PIO)
- 2 Serienschnittstellen (Z80-SIO) mit normgerechtem RS 232 C und 20 mA Current-Loop-Interface
- DC/DC-Wandler für RS 232 C Interface
- Einstellbare Baudrate über DIP Schalter
- Schmitt Triggerung aller Bussignale
- Spannungsversorgung 5 V ± 5%.

Z80-EML/C Zentralbaugruppe
- Z80-CPU
- Z80-PIO
- Z80-CTC
- Programmierbare Echtzeituhr

Z80-ECB/C Zentralbaugruppe
(„Einfacheuropa-Computer-Baugruppe/CPU-Karte")
- Z80-CPU
- Decoder und Puffer für vollen Speicher (60 kByte) und I/O-Ausbau
- max. 2 kByte PROM
- 256 Byte RAM
- 2 Parallelschnittstellen (1 Stück Z80-PIO)
- 1 Serienschnittstelle mit RS 232 und 20 mA Stromschleifen-Interfaces
- Über Schalter 12 Baud-Raten einstellbar.

Z80-ECB/E16 Speichererweiterung bis 16 kByte
(„Einfacheuropa-Computer-Baugruppen/Erweiterungsplatine bis 16 kByte")
- max. 16 kByte PROM oder EPROM einsteckbar (2716)
- 1 kByte statisches RAM, auf max. 4 kByte nachrüstbar
- Decoder und Schmitt-Trigger-Puffer
- Schalter zur Speicherbereichsfestlegung.

Z80-ECB/E Speicher-Erweiterungsplatine im Einfach-Europa-Format
(„Einfacheuropa-Computer-Baugruppen/Erweiterungsplatine")
- max. 8 kByte PROM oder EPROM einsteckbar (2708)
- 1 kByte stat. RAM
- Decoder und Puffer
- Schalter zur Speicherbereichsfestlegung

Z80-ECB/I Ein/Ausgabe-Platine
(„Einfacheuropa-Computer-Baugruppen/I/O-Platine")
- 4 Parallelschnittstellen (2 Stück Z80-PIO)
- 4 Zeitgeber/Ereigniszählkanäle (1 Stück Z80-CTC)
- 9 unverdrahtete Fassungen für Puffer.

Z80-ECB/V CMOS Speichererweiterung
(„Einfacheuropa-Computer-Baugruppen/nicht volatiles RAM")
- 4 kByte CMOS-RAM
- Decoder und Puffer auf der Platine
- Schalter zur Speicherbereichsfestlegung
- Klein-Akku zur Erhaltung der Information im RAM über mehrere Tage bei Unterbrechung der Stromversorgung

Z80-ECB/D32-P Dynamische Speichererweiterung
(„Einfacheuropa-Computer-Baugruppen/Dynamisches RAM mit Paritätsprüfung")
- 32 kWorte dynamisches RAM à 9 bit (9. Bit ist Paritätsbit)
- Hardwaremäßige Paritätsprüfung mit auf der Platine aufgebaut; automatische Korrektur, beispielsweise von „Soft-Errors" unter Computerprogrammkontrolle möglich
- Decoder, Puffer und Ansteuerungsschaltung auf der Platine
- Stromversorgung: Eine einzige 5 V-Spannungsquelle

Z80-ECB/D32 32 kB Dynamische Speichererweiterung
(„Einfacheuropa-Computer-Baugruppen/Dynamisches RAM")
- 32 kByte dynamisches RAM
- Decoder, Puffer und Ansteuerungsschaltung auf der Platine
- Spannungsversorgung: Eine einzige 5 V-Spannungsquelle.

Z80-ECB/D16 16 kB Dynamische Speichererweiterung
Wie Z80-ECB/D32, jedoch nur mit 16 kByte bestückt.

Z80-ECB/D8 8 kB Dynamische Speichererweiterung
Diese Baugruppe ist identisch mit ECB/D32, jedoch nur mit 8 kByte dynamischem Schreib/Lesespeicher bestückt.

Z80-EML/IS Serien-Ein/Ausgabe-Erweiterung
- 3 interruptfähige Duplex-Serienkanäle

Z80-ECB/F Serien/Parallel-Ein/Ausgabe und FD-Controller
(= „Einfacheuropa-Computer-Baugruppe/Floppy-Disk-Controller")
- Allgemeine Ein/Ausgabeplatine mit 2 Vollduplex-Serien- und zwei 8 bit-Parallel-Schnittstellen
- Anschluß von bis zu 8 Floppy-Disk-Laufwerken (Normal, Double-Density und Mini-Floppy)
- Spannungsversorgung: Eine einzige 5 V-Spannungsquelle
- Volle RS 232-Schnittstelle

Z80-ECB/S Speicher-Ein/Ausgabe-Erweiterung
(„Einfacheuropa-Computer-Baugruppe/Speicher-I/O-Erweiterung")
- 8 kByte statische Speichererweiterung
- 40 Parallel-Ein/Ausgabe-Leitungen über zwei PIO's

Z80-KIT/VZ Videozusatz
Zusatz zur Ansteuerung von Fernsehgeräten über Composit-Video
- 16 Zeilen × 64 Zeichen
- 64 ASCII-Zeichen, 5 × 7 Punktematrix
- Übertragungsrate ca. 1200 Baud.

Baugruppen und Ergänzungen zu den beiden Serien Z80-ECB und Z80A-ECB

ANμP80-E16 Analog-Ein/Ausgabe-Baugruppe
- 16 Analoge Eingänge (bzw. 8 Gegentakt-Eingänge)
- 1 Analog Ausgang
- Sämtliche Ein/Ausgabekanäle mit 12 bit Genauigkeit

ANμP80-A4 Analog-Ausgabe-Baugruppe
- 4 Analoge Ausgänge
- 12 bit Genauigkeit
- Spannungsversorgung über eine einzige 5 V-Quelle

EML/SP Subprozessor-Baugruppe
- Z80A-CPU
- 4 kByte bipolares PROM
- 1 kByte statisches RAM
- Z80A-PIO

Z80-KIT/D Druckerzusatz
- Kleindrucker auf Metallpapierbasis
- max. 32 Zeichen/Zeile
- Druckgeschwindigkeit ca. 2 Zeilen/sec.
- Ideal für Informationsausgabe am Arbeitsplatz (z.B. BDE = Betriebsdatenerfassung) und Fertigungssteuerung.

ECB/A Arithmetikprozessor (in Vorbereitung)
(„Einfacheuropa-Computer-Baugruppe/Arithmetikkarte")
- Schneller Arithmetikzusatz für rechenintensive und zeitkritische Anwendungen
- Hardwaremäßige Fest- und Gleitkomma-Rechnung

ECB/B IEC-IEEE-Bus-Controller (in Vorbereitung)
(„Einfacheuropa-Computer-Baugruppe/IEC-Bus-Controller")

ECB/N Netzteil im Europaformat
- Geregeltes Schaltnetzteil
- Kurzschlußfest
- 32-poliger VG-Steckverbinder mit Hochstrom-Kontakten, passend zum Sondersteckplatz im Einschubreck

ECB/R Baugruppenträger
(„**E**infacheuropa-**C**omputer-**B**augruppen-**R**eck")
- Einschubrahmen mit 17 Steckplätzen
- Rückwandverdrahtung (1:1-Durchverdrahtung der Busleitungen) für Taktfrequenzen bis mindestens 4 MHz

ECB/W Wire-Wrap-Karte
(„**E**infacheuropa-**C**omputer-**B**augruppen/**W**ire-Wrap-Karte")
Die Baugruppe erlaubt den raschen, kostensparenden Aufbau von anwenderspezifischen, ECB-kompatiblen Z80-Mikrocomputerschaltungen in Wire-Wrap-Technik. Es können beliebig gemischt 40-, 28-, 20-, 16- und 14-polige Bausteine verdrahtet werden.

ECB/Y Verlängerungsplatine
Die Platine erlaubt das Herausführen von Busanschlüssen aus dem Baugruppenträger ECB/R zu Testzwecken.
Alle Steckeranschlüsse sind 1:1 durchgezogen.

EML/BF Bus/Foundation-Module:
Baugruppe zur raschen Erstellung anwenderspezifischer Schaltungen. Die Baugruppe enthält bereits ein komplettes ECB-Businterface.

EML/TG Tischgehäuse
Tischgehäuse mit Rückwandverdrahtung und Stromversorgung

Grundsoftware für ECB-basierende Anwendersysteme

Z80-ECB/1 1 kByte Monitor-Programm
Grundsoftware für Anwendersysteme mit einfachen Schreib/Lese- und Testfunktionen.

Z80-ECB/2 2 kByte Monitor-Programm
Grundsoftware für Anwendersysteme mit Schreib/Lese-, Test- und Floppy-Disk-Steuerfunktionen.

Z80-MTX Multitask-Echtzeit-Betriebssystem
Das Multitask-Betriebssystem MTX erlaubt die Koordinierung und Verwaltung mehrerer konkurrierender, zeitkritischer Probleme und Interrupts und ist insbesondere für die Verwendung mit Baugruppen der ECB-Serie geeignet.

Das ECB-Grundkonzept

Das ECB-Grundkonzept ersehen Sie aus Bild 1

Sämtliche Bus-Signale wurden so auf die 64-poligen Standard VG-Stecker herausgeführt, daß alle Baugruppen untereinander über „1:1-Verdrahtung" verbunden werden, d.h. daß immer gleiche Steckerpins (ohne Kreuzungen) miteinander verdrahtet sind, wodurch sich eine problemlose Rückwandverdrahtung ergibt.

Die Zentralbaugruppen wurden aus folgenden Gründen bereits mit Speicher- und Ein/Ausgabemöglichkeiten ausgerüstet:

- ☐ einfachere Anwenderprobleme können bereits mit dieser Konfiguration oder mit geringfügiger Erweiterung gelöst werden.

- ☐ eine Serienschnittstelle ist in den meisten Anwendersystemen nur einmal erforderlich; sie wurde daher auf der Zentralplatine untergebracht.

- ☐ kleine Systeme kommen bereits mit den auf der Zentralbaugruppe vorhandenen zwei 8 bit-Ports aus, da diese bereits die gesamte Quittungs- und Interruptsteuerungs-Logik implizieren.
  In größeren Systemen werden dagegen häufig einige wenige parallele Hilfs-Ein/Ausgaben (z.B. für Anzeigen oder Status-Informationen) benötigt, für die die PIO auf der Zentralbaugruppe verwendet werden kann.

- ☐ Das PROM der Zentralplatine wurde bewußt fest durch Programmierung eines Address Decode PROM's auf die Anfangsadresse 0 gelegt (variabel durch Austausch dieses PROM's), da man hier sinnvollerweise die Grundfirmware (= „Monitor) oder aber Programme für Tests und Wartung bzw. zur Interrupt-Behandlung ablegt; die Platine wird mit Sockeln ohne PROM-Bausteine geliefert, da nicht festgelegt ist, ob Monitor- oder Anwenderfirmware verwendet werden soll.

- ☐ Das statische RAM auf der Platine ist als Zwischenspeicher oder Stack einzusetzen.

- ☐ Durch Herausführen der Signale $\overline{\text{BUSRQ}}$ und $\overline{\text{BUSAK}}$ ist der Aufbau von Multiprozessorsystemen und direkter Speicherzugriff (DMA) möglich.

- ☐ Speichererweiterungen werden mit gemischten Festwert-Schreib/Lese-Speicher-Baugruppen durchgeführt, da erfahrungsgemäß bei wachsender Programmgröße auch der Bedarf an Variablen-Speicher proportional wächst. Zum Aufbau des Festwertspeichers können alle Standard-Masken-ROM's, „fusible" PROM's oder EPROM's eingesetzt werden, die den Geschwindigkeitsanforderungen der Z80-CPU genügen. Aus diesem Grund werden Baugruppen nicht mit Festwert-Speicher-Bausteinen geliefert, um dem Anwender freie Wahl der Bausteine zu ermöglichen. Beide Speicherbereiche können jedem beliebigen adressierbaren Speicherbereich der CPU zugeordnet werden. Für Systeme, in denen auch die Anwenderprogramme in Schreib/Lesespeichern abgelegt werden sollen (z.B. bei den meisten Anwender-Systemen, die mit Floppy Disks und der Steuerplatine ECB/F arbeiten), stehen reine RAM-Karten (8, 16 oder 32 kByte dynamisches RAM ECB/D und 4 kByte-CMOS statisches RAM mit Puffer-Batterie und Speicherschutzschalter Z80-ECB/V) und gemischte RAM/I/O-Karten (ECB/S) zur Verfügung. Zur Erweiterung der Ein/Ausgabe-Möglichkeiten der Zentralplatine dient die Z80-ECB/I mit ihren vier 8bit-Ports und vier Zähler/Zeitgeber-Kanälen inklusive kompletter Quittungs- (= "Handshaking") und Interrupt-Steuerungslogik. Zusätzliche Ergänzung ECB-basierender Systeme ist über die Serien/Parallel/Echtzeit-, Ein/Ausgabe und Floppy-Disk-Platine ECB/F sowie über die Baugruppen aus dem Z80-KIT-Konzept und Sonderbaugruppen möglich.

- ☐ Als Bus-Stecker kommt die weitverbreitete 64-polige VG-Steckerleiste zum Einsatz.
  Hier werden sämtliche Signale herausgeführt, die allen Baugruppen gemeinsam sind und einfach durchverdrahtet werden können.

- ☐ Ein/Ausgabe-Signale werden von der Karte an dem dem Bus-Stecker entgegengesetzten Kartenende über 3M-Pfostensteckverbinder verfügbar gemacht, da

  - Ein/Ausgabe-Signale bei sehr vielen Anwendungen an der Frontseite (üblicherweise der „Rückwandverdrahtung" gegenüberliegend) benötigt werden.

  - auch bei aufwendigerer, den Ein/Ausgabebausteinen nachgeschalteter Peripheriehardware eine Rückführung dieser Signale über eine zweite, kundenspezifische Interface-Baugruppe möglich ist (auch „Huckepack"- bzw. „Sandwich"-Verfahren erwägen!!)

  - die Herausführung sämtlicher Signale, also auch der Ein/Ausgabe, auf der Busseite die Verwendung vielpoliger teurer Steckverbinder nötig machen und die komplizierte Leitungsführung auf der Baugruppe zu großem Platzverlust verursachen würde. Um dem Anwender Zeitverlust durch Aufbau- und Verdrahtungsprobleme zu ersparen, wird passend zu diesem System ein Baugruppenträger (ECB/R) angeboten.

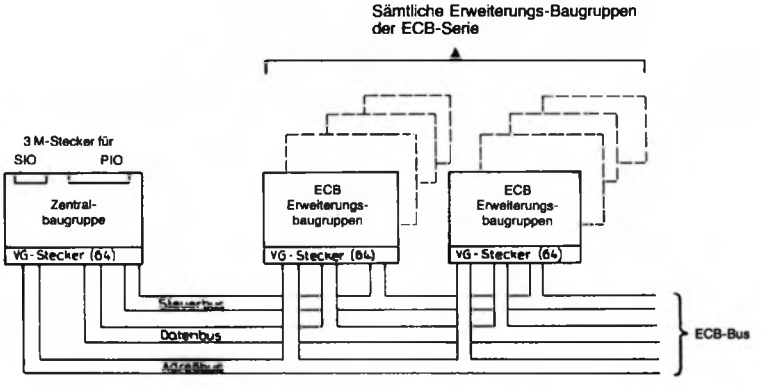
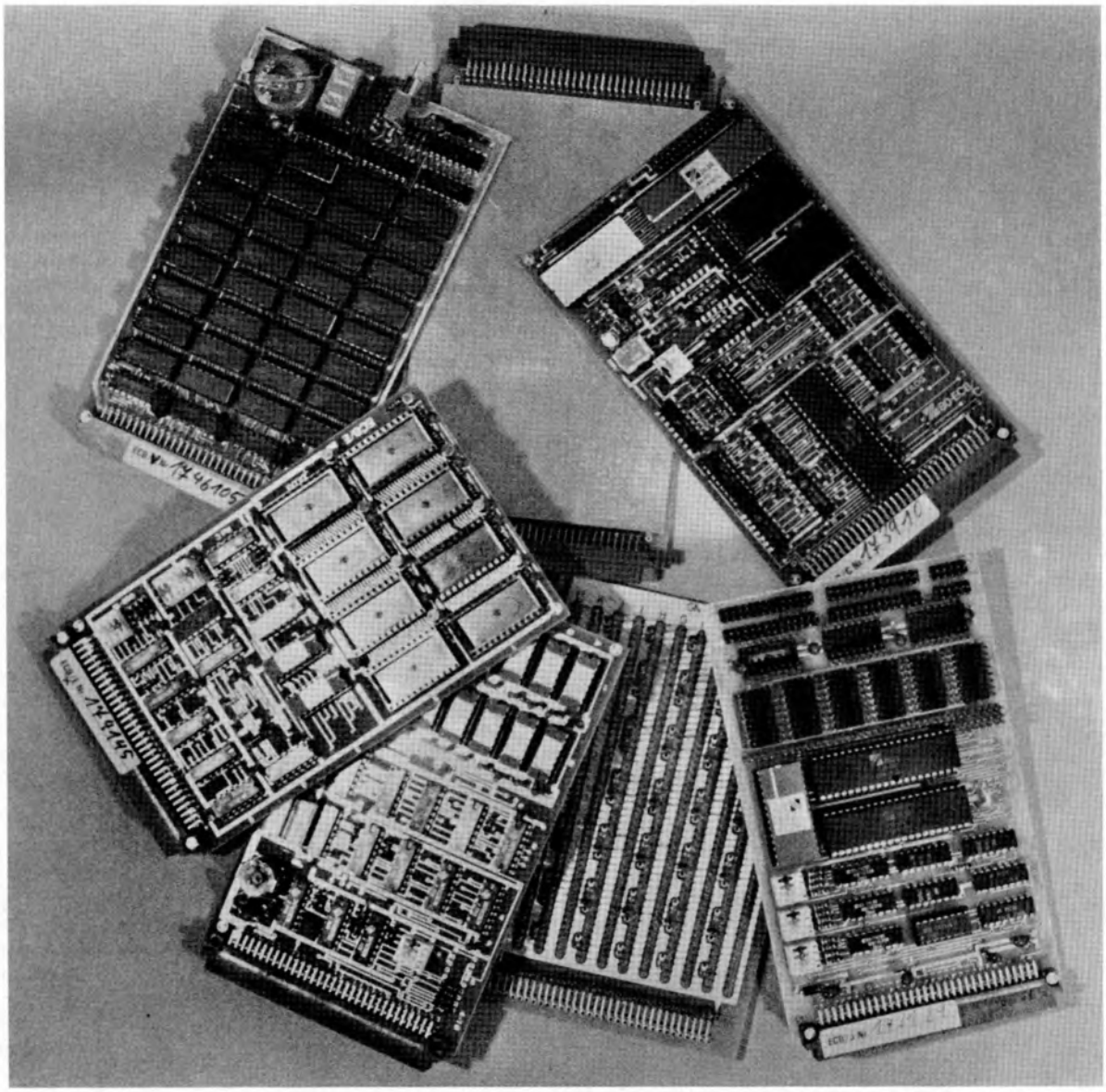


Bild 1: Z80-ECB-Grundkonzept.

Z80A-ECB-SERIE

Mikrocomputer-Baugruppen im Einfach-Europa-Format

KONTRON
ELEKTRONIK GMBH



1. Allgemeines

Die neue Mikrocomputerbaugruppenserie Z80A-ECB von KONTRON stellt eine Weiterentwicklung der bekannten Z80-ECB-Serie dar.

Sie ist für ein Arbeiten mit einer CPU-Taktfrequenz von 4 MHz (Z80A-CPU) ausgelegt, wobei auch sämtliche Speicher- und Ein/Ausgabe-Schaltungen mit der vollen Arbeitsgeschwindigkeit operieren.
Die Serie ist völlig Pin-kompatibel zur Z80-ECB-Serie (Pinning siehe folgende Seite) und wird laufend erweitert.

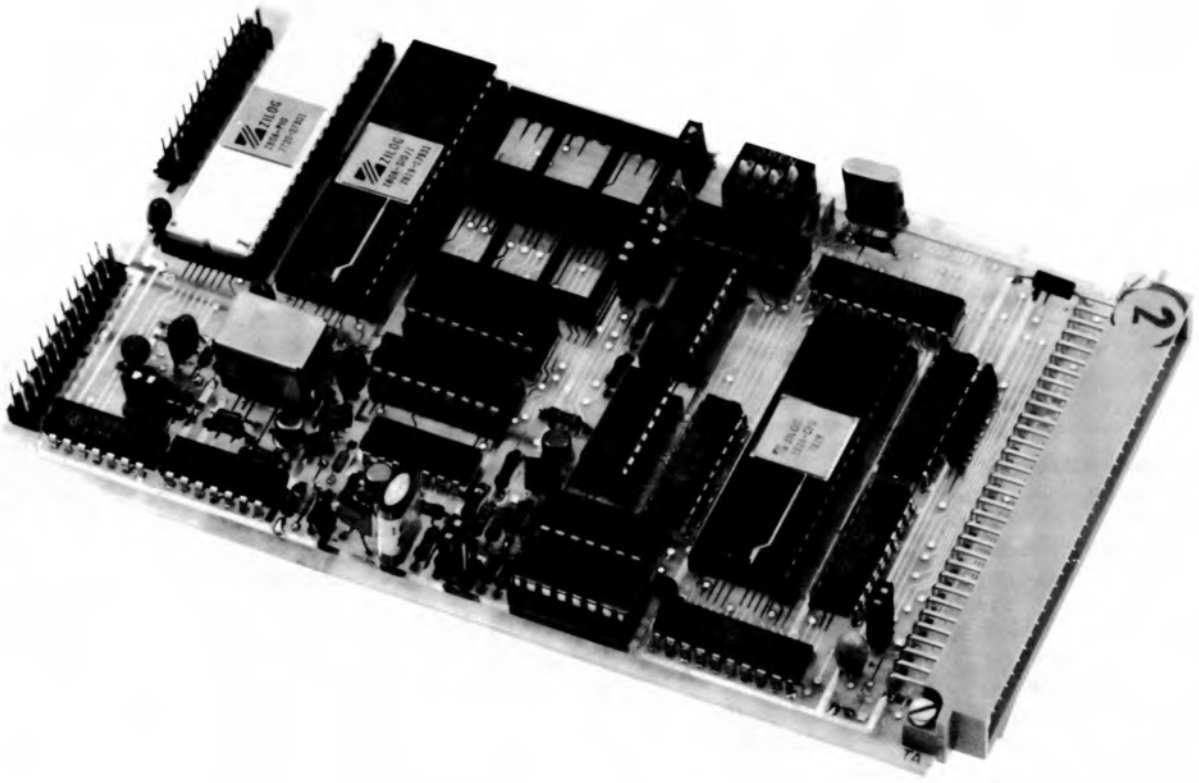
7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

Pin-Belegung der Baugruppenserien Z80A-ECB, Z80-ECB und Z80-EML

Benennung	Stecker Pin	Bezeichnung	Fan Out (TTL)	Fan In	Kommentar
A 0	5c	Adresse 0	9		
A 1	7c	Adresse 1	9		
A 2	6a	Adresse 2	9		
A 3	6c	Adresse 3	9		Adress-Bus
A 4	7a	Adresse 4	9		(Tri-State
A 5	8a	Adresse 5	9		bei $\overline{BUSAK}$)
A 6	9a	Adresse 6	9		
A 7	9c	Adresse 7	9		
A 8	8c	Adresse 8	9		
A 9	30a	Adresse 9	9		Ausgänge
A 10	18c	Adresse 10	9		
A 11	17c	Adresse 11	9		
A 12	27c	Adresse 12	9		
A 13	29a	Adresse 13	9		
A 14	18a	Adresse 14	9		
A 15	28c	Adresse 15	9		
D 0	2c	Data 0	25 mA low	0,25 mA Low	
D 1	14c	Data 1	25 mA low	0,25 mA Low	Daten Bus
D 2	4c	Data 2	25 mA low	0,25 mA Low	Bidirek-
D 3	4a	Data 3	25 mA low	0,25 mA Low	tional;
D 4	5a	Data 4	25 mA low	0,25 mA Low	(Tri-State
D 5	2a	Data 5	25 mA low	0,25 mA Low	bei $\overline{BUSAK}$)
D 6	3a	Data 6	25 mA low	0,25 mA Low	
D 7	3c	Data 7	25 mA low	0,25 mA Low	
D 8	10c	Data 8	reserviert für 16 bit-Computersysteme (z.B. ZILOG Z8000)		
D 9	12c	Data 9			
D 10	13c	Data 10			
D 11	14a	Data 11			
D 12	23c	Data 12			
D 13	19c	Data 13			
D 14	21a	Data 14			
D 15	22a	Data 15			
$\overline{M1}$	20a	Maschinenzyklus 1	9		
$\overline{MRQ}$	30c	Memory Request	9		
$\overline{IORQ}$	27a	IN/OUT Request	9		System
$\overline{RD}$	24c	Read	9		Steuerbus
$\overline{WR}$	22c	Write	9		Ausgänge
$\overline{RFRSH}$	28a	Refresh	9		
$\overline{HLT}$	25c	Halt	9		
$\overline{WAIT}$	10a	Wait		1 mA Low 4,7 kPull up	CPU-
$\overline{INT}$	21c	Interrupt		1 mA Low 4,7 kPull up	Steuer-
$\overline{NMI}$	20c	non Mask. Int.		1 mA Low 4,7 kPull up	Eingänge
$\overline{RESET}$	31c	Reset		1,4 mA Low 4,7 kPull up	
IEI 1	11c	Int. enable in		1 mA Low, 4,7 kPull up	Daisy Chain-
IEO 1	16c	Int. enable out	1		Interrupt-
					Steuerung
$\overline{PWRCL}$	26c	Power on clear	9		
Φ	29c	Clock 2,45 MHz			Ausgänge
2Φ	16a	2 × Clock			
$n\Phi$	25a	n × Clock			
$\overline{BUSRQ}$	11a	Busrequest		1 mA Low 4,7 kPull up	Bus-Steuerung
$\overline{BUSAK}$	31a	Busacknowledge	9		für DMA-
BAI 1	12a	Busprioritätssteuerung Ein			basierende
BAO 1	17a	Busprioritätssteuerung Aus			Systeme
$\overline{WRITE\ EN}$	26a	Write Enable			Schreibfreigabe für ECB/D
$\overline{DPR}$	23a				Synchronisation für ECB/V
+5	1a,c	+5V			Spannungsversor-
GND	32a,c	Ground			gungsleitungen
+12	13a	+12V } für EPROMS			
−5	15a	−5V }			
+15	19a	+15V } für V24 und			
−15	15c	−15V } AD-Wandler			
VC MOS	24a	+5V Batterie-Spannung (Notstrom)			

Z80A-ECB/C8 Zentralbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3 Z80-ECB
MIKROCOMPUTER-BAUGRUPPEN

2. Die Zentralbaugruppe Z80A-ECB/C8

2.1 Übersicht

Die Baugruppe Z80A-ECB/C8 ist eine Mikrocomputerbaugruppe auf der Basis des Mikroprozessors Z80A-CPU im Einfacheuropaformat; sie arbeitet mit dem bekannten ECB-Bus, ist also zu allen Baugruppen der Serien Z80-ECB, Z80A-ECB und Z80-KIT elektronisch und elektromechanisch voll kompatibel.

Die Computerkarte ist sowohl als kompletter, selbständiger Ein-Platinenrechner zu verwenden, als auch als über den vollen Leistungsumfang des Z80-Konzepts erweiterbare Zentralbaugruppe.

Dies wurde durch Verwendung neuester Technologie und höchster Integration (wie Serieninterface Z80A-SIO, 4 kbit-statische RAM's und 32 k/16 k-EPROM's), volle Adreß-Ausdekodierung und Pufferung sämtlicher Bus-Signale erreicht.

Besonders bemerkenswert ist die Eignung des Computers zur Realisierung von Multirechnersystemen, wobei die einzelnen Computer über volle RS 232-Serienschnittstellen gekoppelt werden können, die ihrerseits mit hardwaremäßig auf der Platine implementierter (Z80A-SIO) HDLC/SDLC-Übertragungsprozedur arbeiten. Als Stromversorgung dient eine einzige +5 V Quelle mit typisch 900 mA.

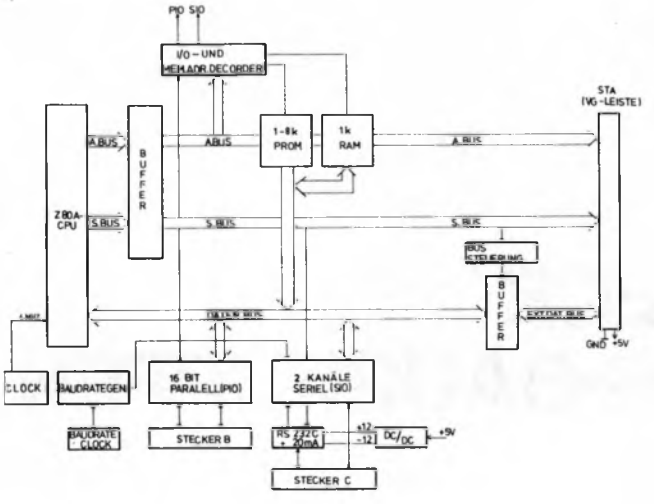
Bestückt mit 3 hochintegrierten ZILOG-Bausteinen (CPU, PIO, SIO) eignet sich die ECB/C8 zum Aufbau von Computersystemen der verschiedensten Komplexitätsstufen. Kleine bis mittlere Systeme können auf Grund der Leistungsfähigkeit der ZILOG-Bausteine und dem großzügig ausgelegten Speicherangebot auf der ECB/C8 (1 k RAM und bis zu 8 k PROM) bereits mit dieser einen Baugruppe realisiert werden.

2.2 Schaltungsbeschreibung

Die Baugruppe enthält folgende Hardware-Komponenten:

- Z80A-CPU
- Dekoder für vollen Speicher- und I/O-Ausbau
- Schmitt-Trigger-Pufferung aller Bussignale
- DMA-fähige Bussteuerung
- Platz für 1–8 kByte Festwertspeicher (zulässige Typen 2758, i2716, i2732, HM 7641, HM 7681)
- 1 kByte statisches RAM (nMOS oder CMOS)
- Zwei 8 bit Parallel-Schnittstellen (1 × Z80A-PIO)
- 2 voll duplex Serienschnittstellen (1 × Z80A-SIO)
- eigner Quarz für Baudrategenerator
- Normgerechtes RS 232 C und 20 mA-Interface
- 14 über Schalter einstellbare Baudraten
- Gleichspannungswandler für Serieninterface (±12 V)
- Busseitig 64-poliger Normstecker mit ECB-Bus-Belegung
- Ausgangsseitig zwei 26-pol. 3M-WWP-Pfostensteckverbinder

Bild 1 zeigt das Blockschaltbild der Baugruppe



Z80A-ECB/C8

Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ± 5%
Stromaufnahme (typisch):	1100 mA (ohne PROM's)
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 150 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II/III

Schaltplan wird mit der Baugruppe mitgeliefert.

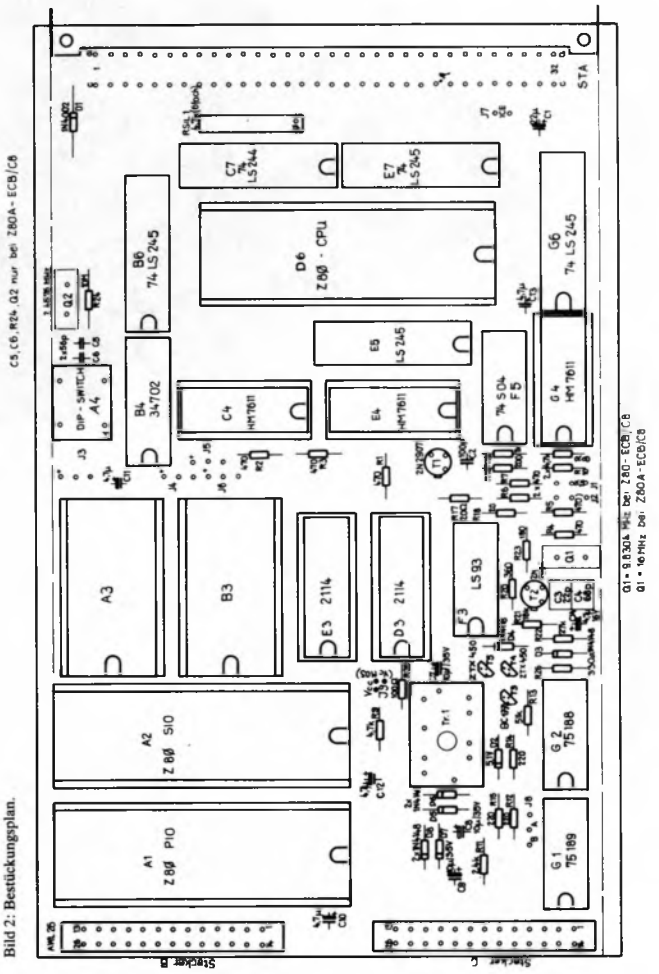
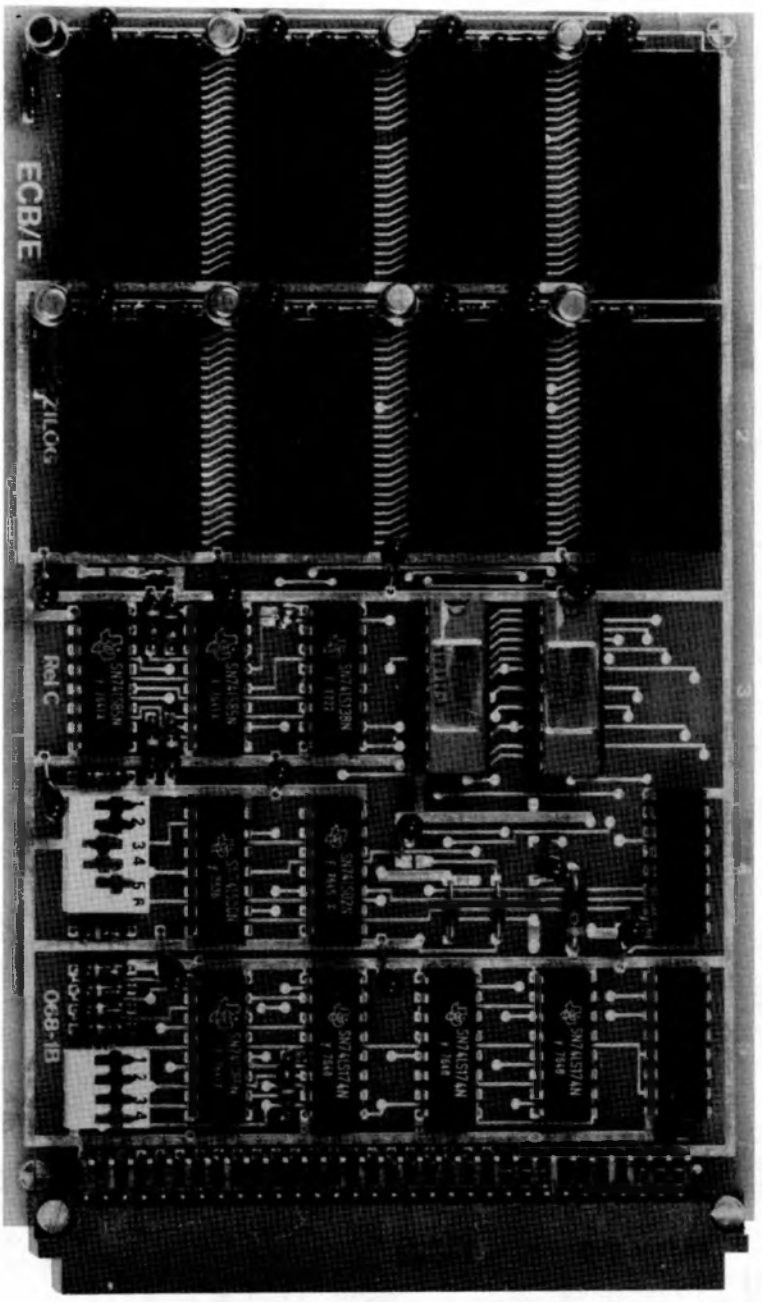


Bild 2: Bestückungsplan.

Z80A-ECB/E

Speicher-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

3. Speichererweiterungs-Baugruppe Z80A-ECB/E

3.1 Schaltungsbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten:

- 1 . . . 8 kByte Festwertspeicher (ROM, PROM oder EPROM)
- 1 kByte statisches RAM
- Decoder
- Schalter zur Speicherbereichsfestlegung
- Busseitiger 64-poliger Stecker nach DIN 41612 (VG 95324).

Bild 1 zeigt das zugehörige Blockschaltbild. Man sieht, daß auch hier die Systembusse gepuffert sind; dadurch ergibt sich zusammen mit den Puffern auf der Zentralplatine Z80A-ECB/C8 eine zweifache Pufferung, was volle Erweiterbarkeit und erhöhte Störsicherheit garantiert.

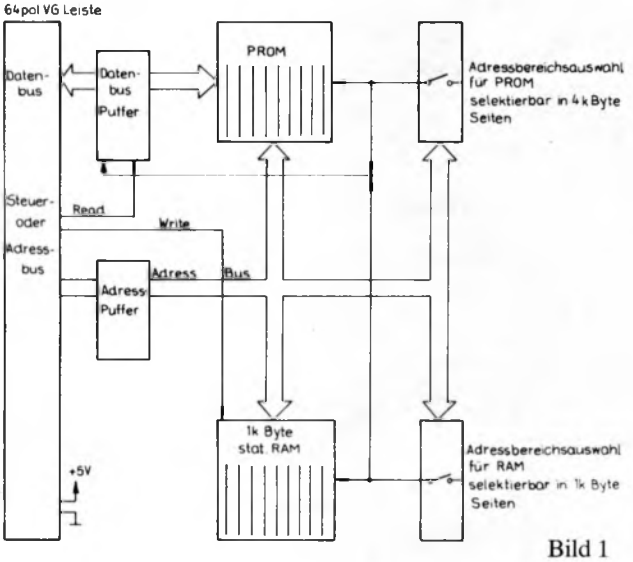
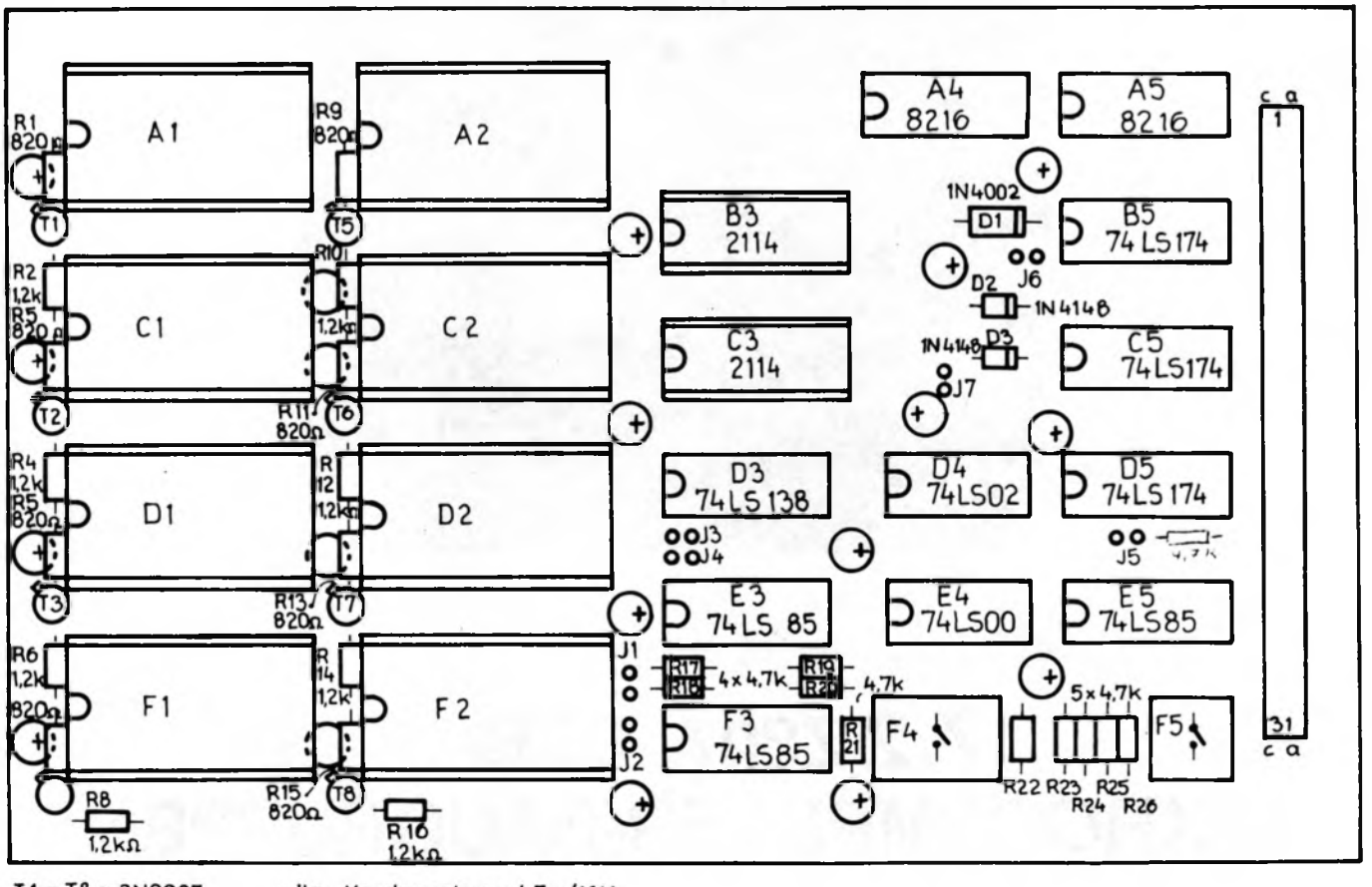


Bild 1

Z80A-ECB/E
Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ±5%, (zuzügl −5 V und +12 V bei Verwendung von 2704/2708-PROM's)
Stromaufnahme (typisch):	360 mA (ohne PROM's)
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 110 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. VI

Schaltplan wird mit der Baugruppe mitgeliefert.

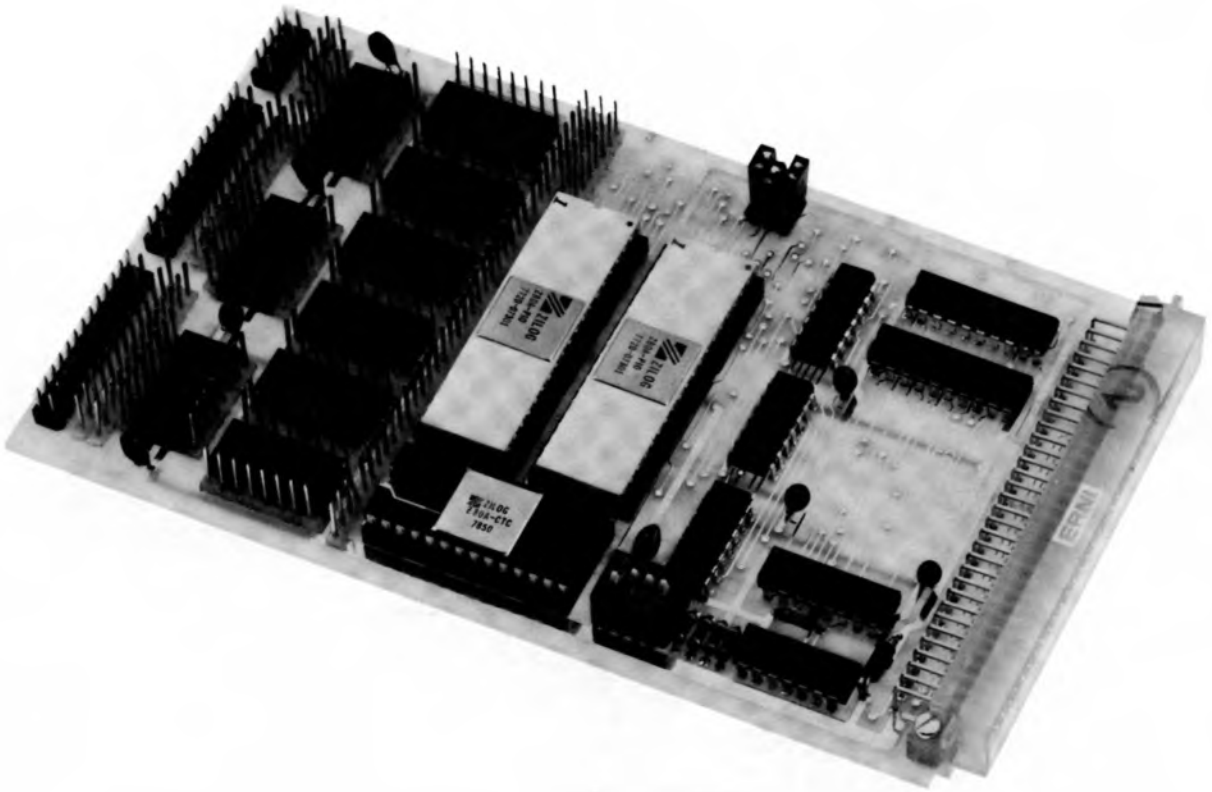


T1 - T8 = 2N2907 alle Kondensatoren 4,7n /16V

Z80A-ECB/I

Ein/Ausgabe-Erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

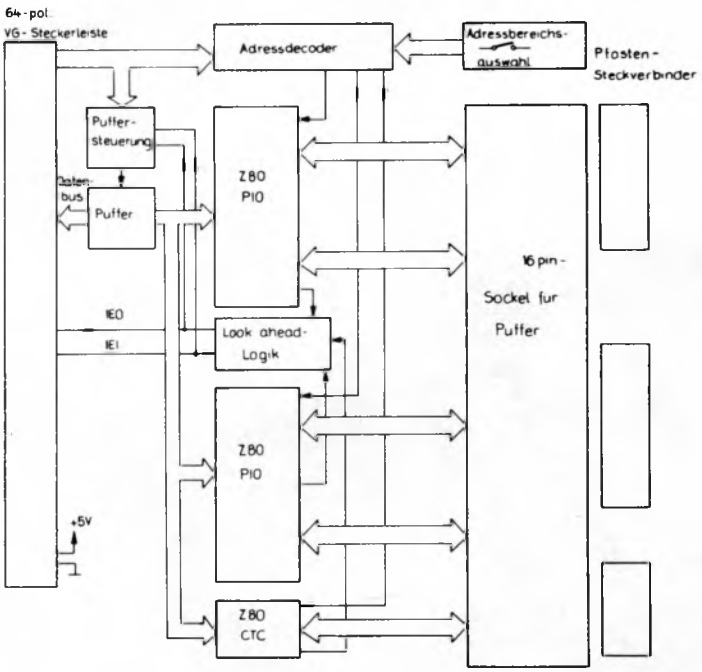
4. Ein/Ausgabebaugruppe Z80A-ECB/I

4.1 Übersicht

Die Baugruppe umfaßt folgende Funktionseinheiten:

- 4 Parallelschnittstellen (2 Stück Z80A-PIO)
- 4 Zeitgeber/Ereigniszählkanäle (1 Stück Z80A-CTC)
- 9 unverdrahtete Fassungen für Standard-Interface-Bausteine
- Volle Buspufferung über Schmitt-Trigger-Puffer für höchste Betriebssicherheit
- Busseitig 64-poliger Stecker nach DIN 41612 (VG 95324)
- I/O-seitig 3 M-WWP-Pfostensteckverbinder.

Bild 1 zeigt das zugehörige Blockschaltbild:



Die Baugruppe behandelt mit ihren 2 Bausteinen Z80A-PIO und dem 1 Baustein Z80A-CTC ingesamt vier 8 bit-Ein/Ausgabekanäle, die jeweils über 2 zusätzliche Leitungen für Quittungsbetrieb verfügen und vier Ereigniszähler/Zeitgeberkanäle.

Einzelheiten über den Baustein Z80A-PIO finden Sie im PIO-Technical-Manual; der Baustein Z80A-CTC ist im CTC-Technical-Manual beschrieben.

Auch bei dieser Baugruppe wurde das Prinzip der doppelten Pufferung weiterverfolgt.

Der bidirektionale Datenbuspuffer (Baustein: 74LS24S) wird nur in den beiden Fällen IO READ und INTA nach außen, sonst dagegen immer nach innen geschaltet.

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	220 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. I

Schaltplan wird mit der Baugruppe mitgeliefert.

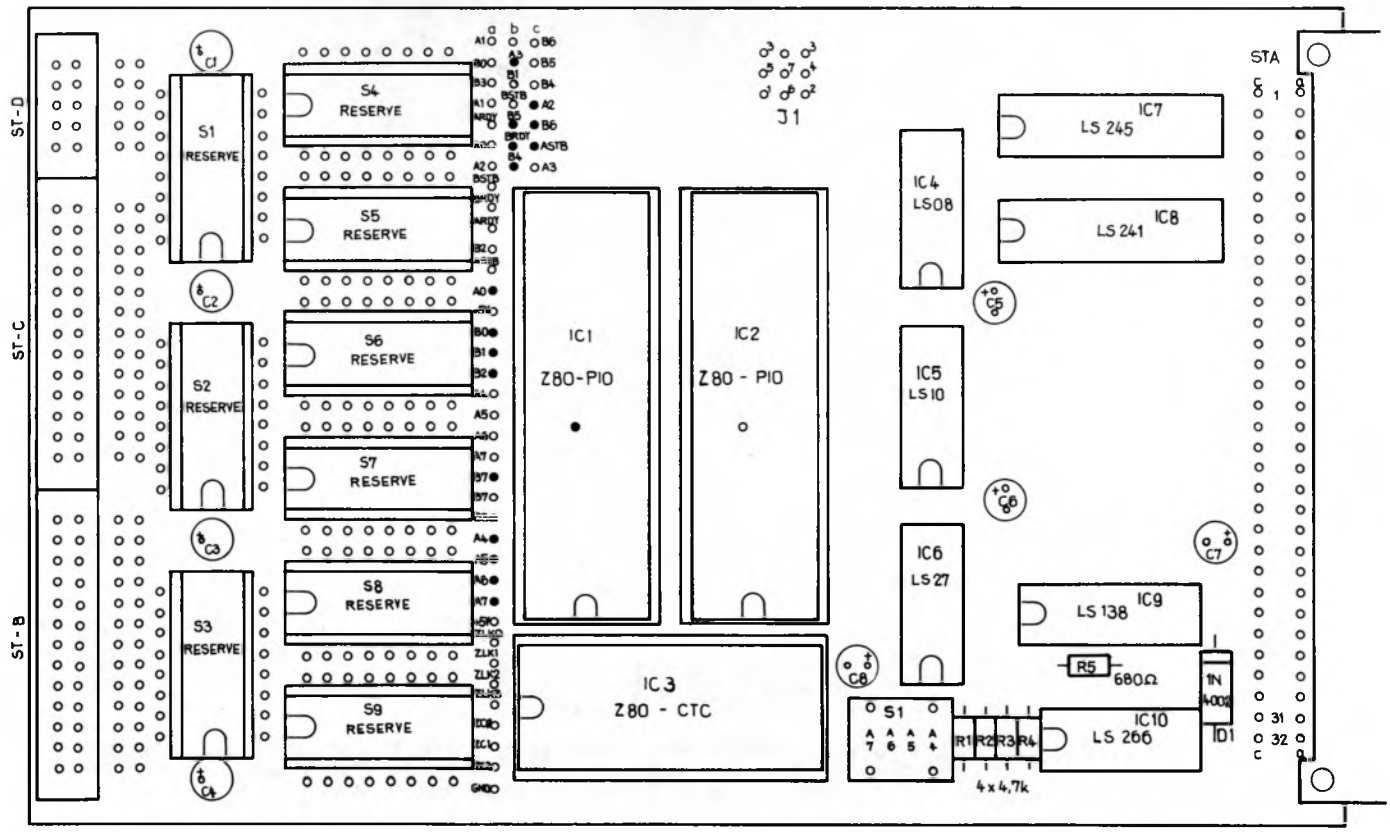
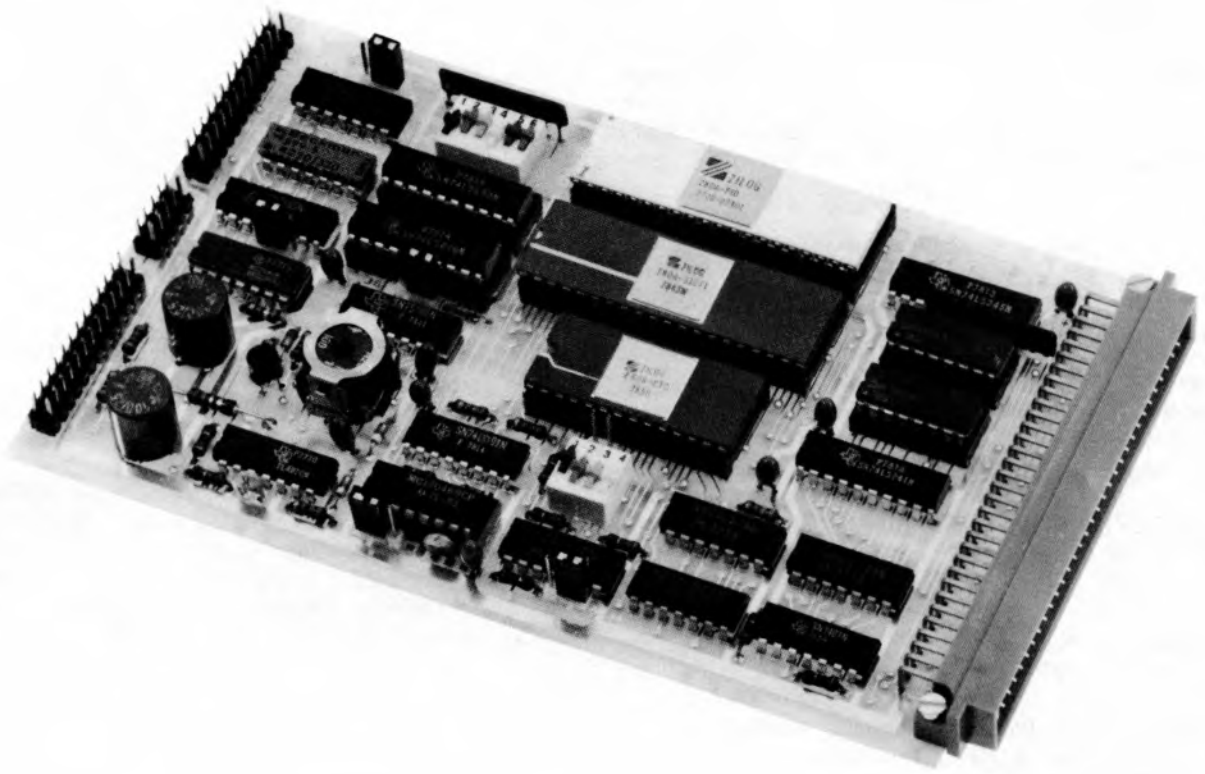


Bild 2: Bestückungsplan

Z80A-ECB/F

Ein/Ausgabe-Baugruppe für allgemeine Aufgaben und Floppy-Disk-Steuerung

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

5. Ein/Ausgabe-Baugruppe für allgemeine Aufgaben und Floppy-Disk-Steuerung Z80A-ECB/F

5.1 Übersicht

Die Baugruppe ECB/F ist eine universelle Serien-Ein- bzw. Ausgabe-Einheit. Sie ist elektrisch und elektromechanisch kompatibel zu allen Baugruppen-Serien Z80A-ECB, Z80-ECB und Z80-KIT. Sie vereint die Leistungsfähigkeit von drei Peripheriebausteinen, nämlich des CTC, des PIO und des SIO. Zwei separate Voll-Duplex-Kanäle erlauben serielle Datenübertragung in nahezu allen bekannten asynchronen und synchronen Verfahren. Über den CTC und einen PLL-Frequenzsynthesizer sind alle gebräuchlichen Baudraten für den Asynchronkanal bzw. alle Übertragungsraten von 0–850 kbit/s für den Synchronkanal programmierbar.

Die Baugruppe ermöglicht den unmittelbaren Anschluß von bis zu acht hardsektorierten Floppy-Disk-Laufwerken verschiedener Größen und Speicherkapazitäten, sowie asynchron arbeitenden Datenendgeräten mit genormten RS 232 C- oder 20 mA Current-Loop-Schnittstellen. Durch den Einsatz des SIO ist die ECB/F in hervorragender Weise auch für Rechner-Rechnerkopplungen geeignet. International genormte Übertragungsprozeduren wie SDLC und HDLC werden vom SIO abgehandelt.

Zum Betrieb der Baugruppe genügt eine einzige Versorgungsspannung von +5 Volt. Die Betriebsspannungen für das RS 232 C-Interface werden durch Gleichspannungswandler auf der ECB/F erzeugt.

Eine leistungsfähige, in zwei Stufen implementierte Floppy-Disk-Betriebssoftware steht ebenfalls zur Verfügung. Zusammen mit der Zentralbaugruppe ECB/C8 stellt die ECB/F somit das Kernstück eines Floppy-Disk basierenden Anwendersystems, oder eines Multi-Rechner-Systems dar; selbstverständlich ist der Einsatz dieser Baugruppe ebenso in allen Anwendungen besonders wirtschaftlich, in denen Serien-(2), Parallel-(2) und Zeitgeber/Zähler-(4) Interfaces erforderlich sind.

5.2 Schaltungsbeschreibung

Die Baugruppe enthält folgende Funktionseinheiten:

a) CPU-Interface
- Schmitt Trigger Pufferung sämtlicher Bussignale
- Adreßdekoder für CTC, PIO und SIO
- Look Ahead Logik für Interruptprioritätssteuerung (Daisy Chain)
- 64-poliger Busstecker nach DIN 41612

b) SIO-Synchronkanal (Floppy-Disk-Interface)
- Voll programmierbarer PLL-Frequenzsynthesizer zur Erzeugung der Übertragungsrate (2 CTC-Kanäle)
- Programmierbarer, rein digitaler Datenseparator mit 16 Zeitkonstanten
- Datenencoder zur Daten-/Clock-Gemischerzeugung
- Diskstatussignale über Z80A-PIO
- Schmitt Trigger Pufferung aller Ein-/Ausgänge

c) SIO-Asynchronkanal
- Voll programmierbare Baudratenerzeugung von 50–38.4000 Baud (1 CTC-Kanal)
- RS 232 C und 20 mA Current-Loop-Interface
- Gleichspannungswandler für ± 12 Volt
- 6 Modem Signale im RS 232 C Pegel

d) Sonstiges
- 6 DIP-Schalter (über PIO) zur Programmierung von Systemvariablen
- Steckbare Jumper zur Umstellung des Synchronkanals an andere Peripherie
- I/O-seitig drei Standardstecker zum Anschluß der Peripheriegeräte

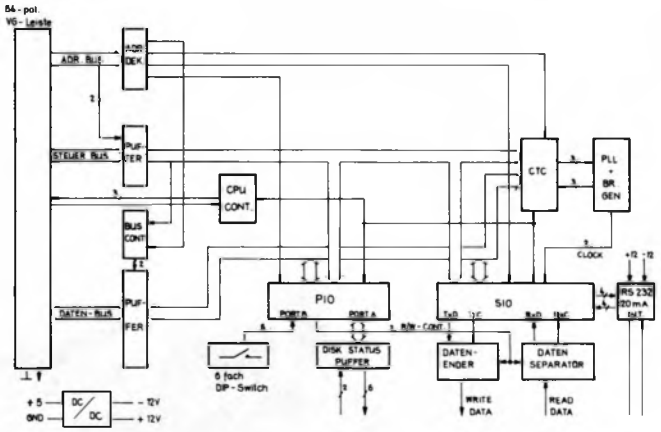


Bild 1: Blockschaltbild der ECB/F

Alle Leitungen sind gepuffert und stellen pro Eingang nicht mehr als eine LS TTL-Last (≙ 0,25 mA bei LOW) dar. Schmitt Trigger Bausteine erhöhen hierbei die Störsicherheit. Alle Ausgänge erlauben den Anschluß von kapazitiv hochbelasteten Bussystemen.

Die Leitungen $\overline{\text{WAIT}}$ und $\overline{\text{INT}}$ sind Open Collector-Ausgänge ohne Pull up Widerstände, da diese bereits auf der Zentralbaugruppe ECB/C8 verhanden sind. Eine „Look Ahead Logik" (Baustein F5: 74LS08) sorgt für die schnelle Durchschaltung der Z80-Interruptprioritätssteuerung (Daisy Chain). Die ECB-F-interne Priorität ist: CTC, SIO, PIO.

5.3 Floppy Disk Interface

Die ECB/F erlaubt den unmittelbaren Anschluß von bis zu 8 Floppy-Disk-Laufwerken (mit hardsektorierten Disketten) in gemultiplexten Systemen entsprechend Bild 2.

Durch die hohe Flexibilität der Hardware sind sowohl Mini- als auch Standardlaufwerke mit einfacher Speicherdichte (single density) anschließbar.

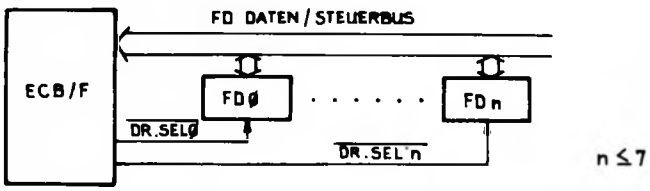


Bild 2: Blockschaltbild eines Systems mit mehreren Laufwerken

Zum Betrieb der unterschiedlichen Laufwerke ist lediglich eine geringfügige Modifikation in der Betriebssoftware nötig.

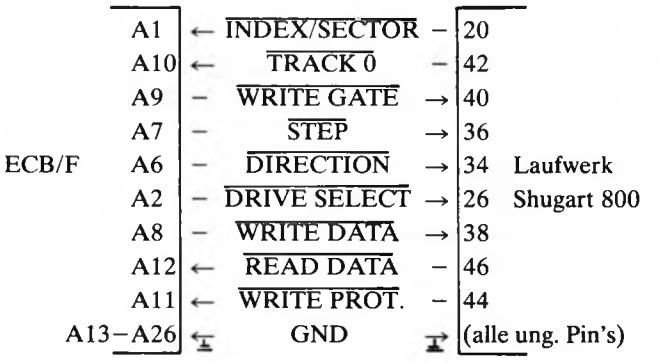


Bild 7: ECB/F-FD-Laufwerk-Verbindungen
(z. B. Laufwerk 800-2)

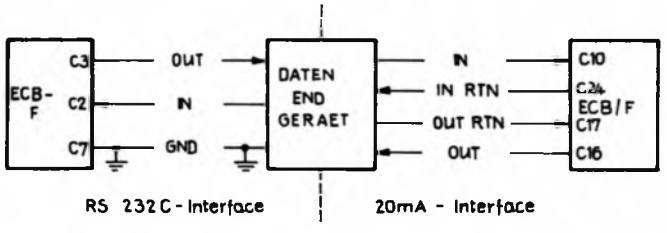
Signalbedeutungen:

INDEX/SECTOR	ist aktiv, wenn das Sektor- oder Indexloch einer Diskette abgetastet wird. Der Indexpuls tritt nur einmal pro Umdrehung auf und kennzeichnet Sektor 0.
TRACK 0	ist aktiv, wenn der Schreib/Lesekopf auf Spur 0 steht.
WRITE GATE	ermöglicht das Schreiben auf die Diskette und verhindert die Ansteuerung des Schrittmotors.
STEP	bewegt den Schreib/Lesekopf um eine Spur weiter.
DIRECTION	bestimmt, in welche Richtung ein STEP erfolgt.
DRIVE SELECT	aktiviert das angesteuerte Laufwerk (Laufwerkadresse)
WRITE DATA	Clock/Datengemisch zum Laufwerk
READ DATA	Clock/Datengemisch vom Laufwerk
MOTOR ON*	startet die Motoren
WRITE PROTECT	die Diskette ist mechanisch schreibgeschützt
READY*	ist 5 Umdrehungen nach Einlegen einer Diskette aktiv.

* Die Signale MOTOR ON und READY sind nicht bei allen Laufwerken vorhanden.

5.4 Anschluß von Terminals

Abhängig von der ausgewählten Schnittstelle (Tabelle 5) erfolgt der Anschluß eines Terminals (Datensichtgerät, Fernschreiber) über eine 3- bzw. 4-Drahtverbindung.



Anschluß von Terminals

5.5 Ein-Ausgabe-Steckerbelegung

Alle Ausgangssignale der ECB/F sind an drei Pfostensteckverbindern herausgeführt.

Stecker A (26polig)	A1	INDEX/SECTOR
	A2	DRIVE 0
	A3	DRIVE 1
	A4	DRIVE 2
	A5	MOTOR ON
	A6	DIRECTION
	A7	STEP
	A8	DISK WRITE DATA
	A9	WRITE GATE
	A10	TRACK 0
	A11	WRITE PROTECT
	A12	DISK READ DATA
	A13—A26	GND
Stecker B (10polig)	B1	DRIVE 3
	B2	DRIVE 4
	B3	DRIVE 5
	B4	DRIVE 6
	B5	DRIVE 7
	B6	EXT. CLOCK
	B7	SPARE OUTPUT
	B8	SPARE INPUT
Stecker C	C1	—
	C2	RS 232 DATA IN
	C3	RS 232 DATA OUT
	C4	DCD
	C5	CTS
	C6	RTS
	C7	COMMON GND
	C8	DTR
	C9	—
	C10	20 mA DATA IN
	C11—C15	—
	C16	20 mA DATA OUT
	C17	20 mA OUT RTN
	C18—C23	—
	C24	20 mA IN RTN
	C25—C26	—

5.6 Jumperstellungen

Die Baugruppe enthält insgesamt 5 steckbare Kurzschlußbügel (Jumper J1–J5), von denen J5 erwähnt wurde (Tabelle 5).

Die Jumper J2 bis J4 dienen dazu, andere synchrone Peripherie an Kanal A des Z80A-SIO anzuschließen. Durch Umstecken der Stecker J2, J3 und J4 auf Stellung B wird die Floppy Disk spezifisches Hardware (Datenencoder, Datenseparator) umgangen. Die SIO-Eingängen RxCA und RxDA führen durch Umstecken von J2 bzw. J3 unmittelbar die an den Steckerpins B6 bzw. A12 anliegenden Signale (EXT CLOCK RECEIVE DATA).

Wird J4 umgesteckt, so ist der Anschluß TxDA über den Puffer B2 mit Stecker A, Pin 8 (TRANSMIT DATA) verbunden. Mit Jumper J1 wird der WRDY-Ausgang des SIO-Kanals B mit dem entsprechenden Ausgang von Kanal A verbunden (beides Open Drain Anschlüsse) und über einen Open-Collector-Puffer (F6:7407) an die CPU-WAIT-Leitung angeschlossen. Damit läßt sich eine Synchronisation zwischen CPU und SIO erreichen (siehe SIO-Produktspezifikation).

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%	Gewicht:	ca. 150 g
Stromaufnahme (typisch):	900 mA	Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Umgebungstemperatur:	0 . . . 50 °C		
Relative Feuchte:	0 . . . 95% (nicht kondensierend)	Beschriebene Version:	Rev. II
Abmessungen:	160 × 100 × 20 mm	Schaltplan wird mit der Baugruppe mitgeliefert.	

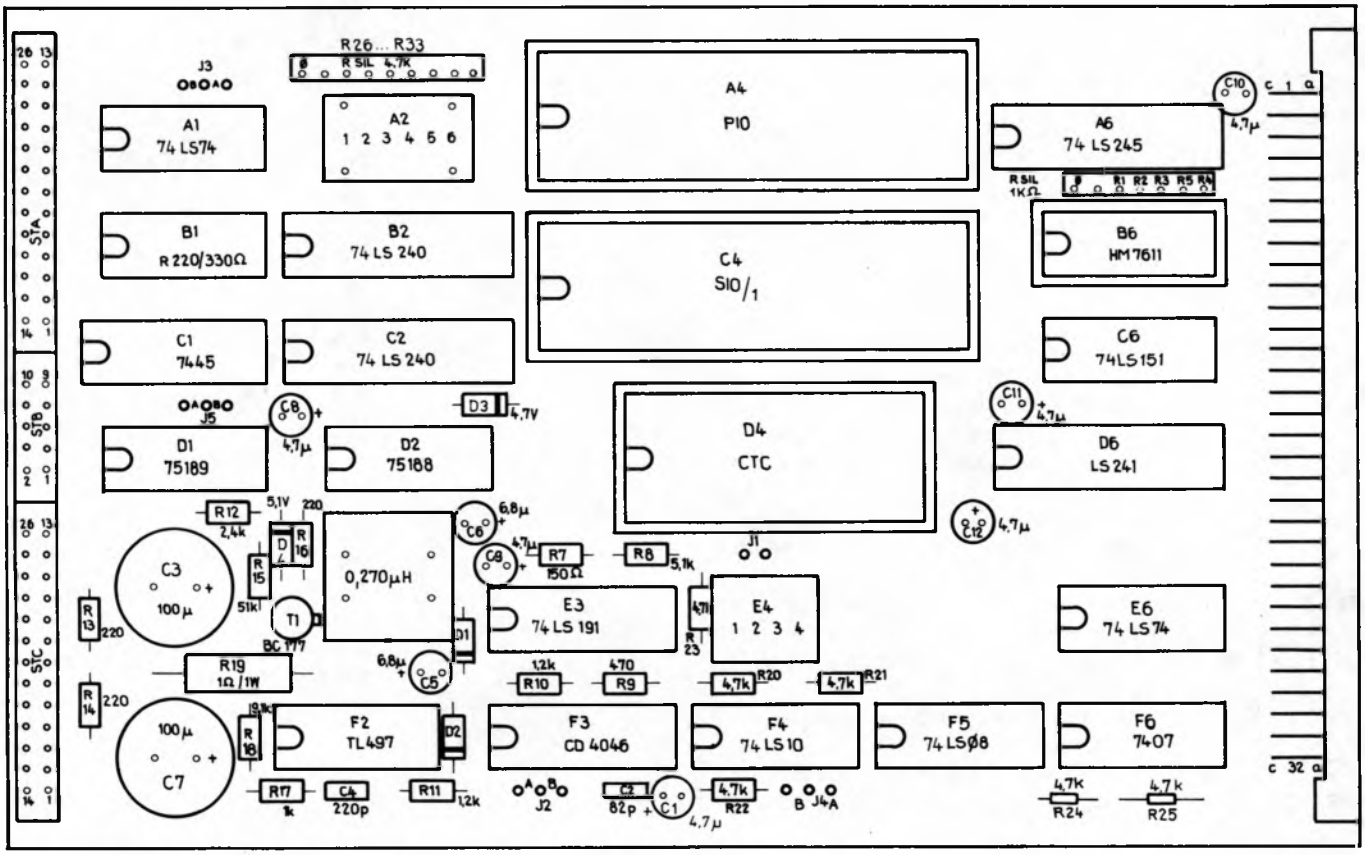
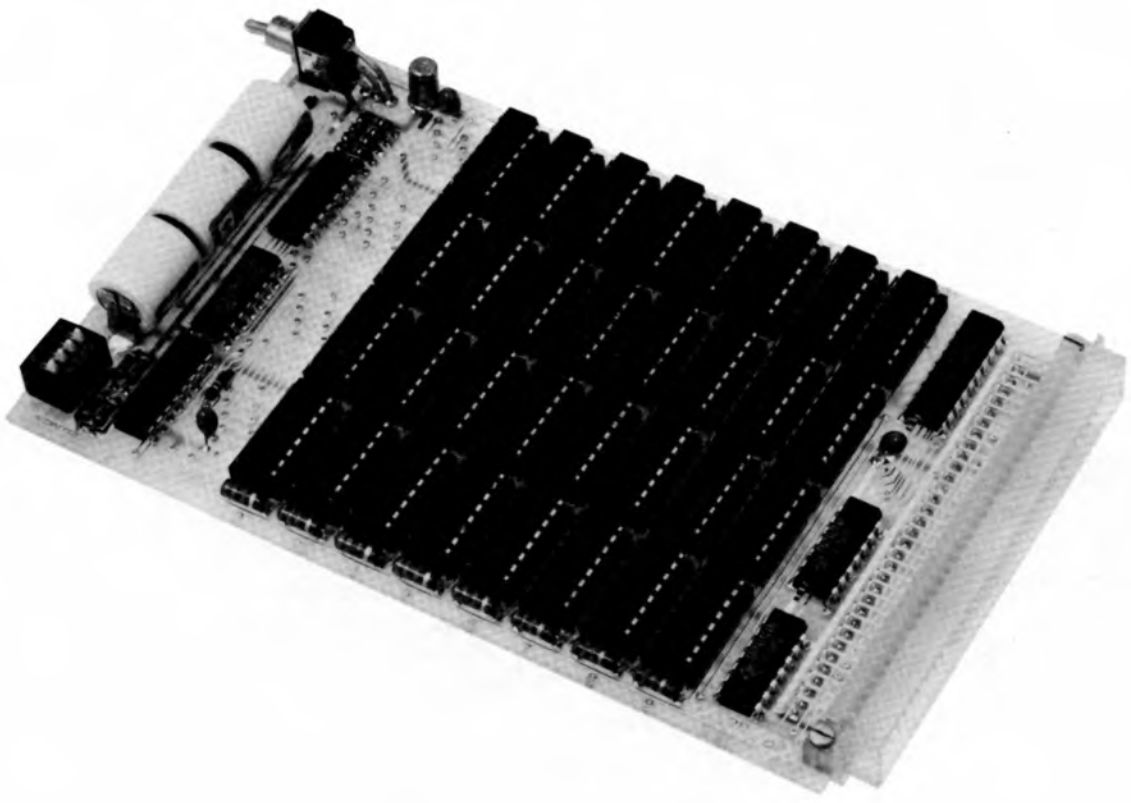


Bild 10: Bestückungsplan der ECB/F

Z80A-ECB/V

CMOS-Speicher-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

Vorläufige Daten

6. CMOS-Speichererweiterungsbaugruppe Z80A-ECB/V

6.1 Schaltungsbeschreibung

Die Einfach-Europakarte ECB/V ist eine voll statische 4 kByte-CMOS-Speicherbaugruppe hoher Flexibilität und Packungsdichte. Wegen der extrem niedrigen Stromaufnahme ist sie besonders für folgende zwei Anwendungen geeignet:

- ☐ Große und schnelle Arbeitsspeicher: z.B. 64 kByte = 16 Karten mit 2,5 A-Netzteil in einem 19''-Einschub evtl. über Zentral-Akku zu puffern: 20 mA Ruhestrom.
  Diese Packungsdichte ist nur durch die vernachlässigbare Wärmeentwicklung der CMOS-RAMS möglich.
- ☐ Nicht-flüchtige Speicher.
  Programme und Daten können beim Austesten gegen Überschreiben geschützt werden (Speicher-Schutz-Schalter). Beim Abschalten des Systems oder bei Stromausfall bleiben die eingegebenen Daten erhalten.

Ferner kann man bei der System-Entwicklung auf die Verwendung von löschbaren PROM's verzichten,weil z.B. Programme bis zu ihrer endgültigen Festlegung auf der Karte erhalten bleiben (Speicherschutzschalter in Stellung "ROM"). Löschungen und Modifikationen können selektiv durch die normalen WRITE-Operationen vorgenommen werden, dazu muß lediglich der Speicherschutzschalter auf "RAM" geschaltet werden.
In gewissem Sinne bietet also diese CMOS-RAM-Karte eine preisgünstige, schnelle und platzsparende Alternative zu Kernspeicher-Systemen.
Eine oft willkommene Erleichterung für die Microcomputer-Entwickler bieten auch die auf der Karte angebrachten Basis-Adreß-Umschalter. Durch Betätigung eines 4 bit-DIL-Schalters kann so der Ansprechbereich der Karte z.B. im 64 kByte Adreßraum beliebig gewählt werden.
Die Blockschaltung zeigt Bild 1.
Die Pinbelegung ist identisch mit der Zentral-Baugruppe, so daß 1:1-Verdrahtung möglich ist.

6.2 Bedienungsanleitung

a) Speicherschutzschalter (Write-Protect).
Steht der Schalter S1 in Stellung Bestückungsseite, so ergibt sich die Betriebsart 'Quasi-ROM', steht dieser Schalter in Richtung Leiterbahnseite, kann auf den Speicher der Karte wie auf jeden Schreib/Lese-Speicher geschrieben und aus ihm gelesen werden.

b) Adreßeinstellung
Die Adressen der ECB/V können mit Hilfe des DIP-Schalters S2 seitenweise (in 4 kByte Schritten) im Speicherbereich der Z80-CPU eingestellt werden.
Den Schaltern S2−1 . . . 4 sind die Adreßleitungen A12 . . . A15 zugeordnet. Daraus ergeben sich folgende Adreßzuweisungen.

c) DPR Signal.
Das DPR Signal ist low aktiv. Ist das Signal logisch "0", so kann auf die Karte weder geschrieben, noch von ihr gelesen werden.
Wird dieses Signal nicht benötigt, so es mit Jumper J2 auf high zu legen.

d) Jumper J1
Mit Jumper J1 wird die Pufferbatterie abgeklemmt, damit sich bei längerer Lagerung keine Tiefentladung der Akkus ergibt.

☐ **Zur Beachtung:**

Die Baugruppe ECB/V wird mit geladenem Akkumulator und aufgetrennter Verbindung zum Akkumulator geliefert, um seine Entladung während der Lagerungszeit zu vermeiden.

Vom Anwender ist dafür Sorge zu tragen, daß der Akkumulator vor Einsatz der Baugruppe als Netzausfallsicherung aufgeladen ist und zu berücksichtigen, daß sich der Akkumulator bei ausgeschaltetem Gerät entlädt.
Die Information im Speicher der Baugruppe bleibt nach Stromausfall oder Abschaltung mindestens 7 Tage erhalten, vorausgesetzt, daß der Akkumulator zu diesem Zeitpunkt voll aufgeladen war.

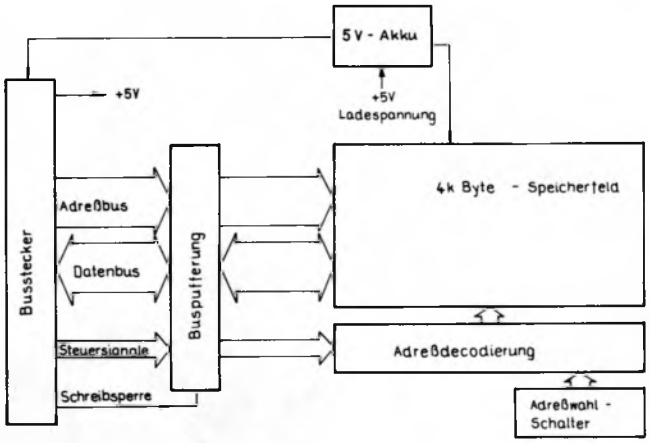
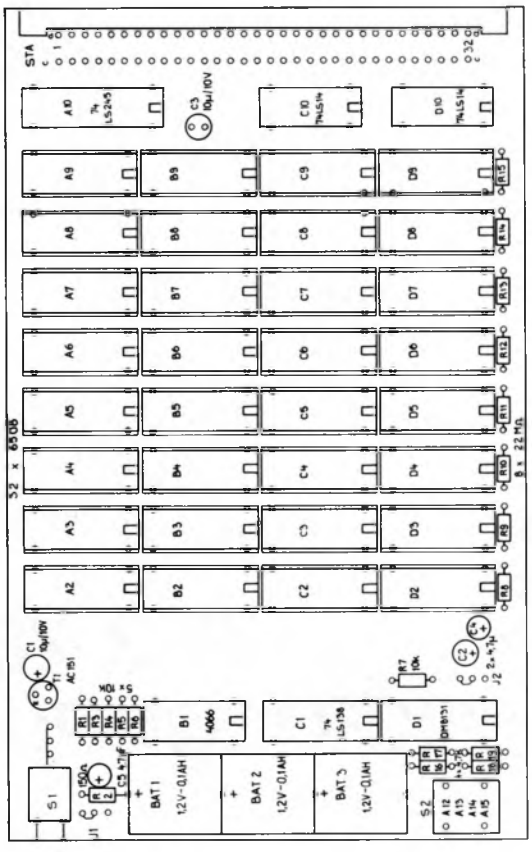


Bild 1

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	160 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 170 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

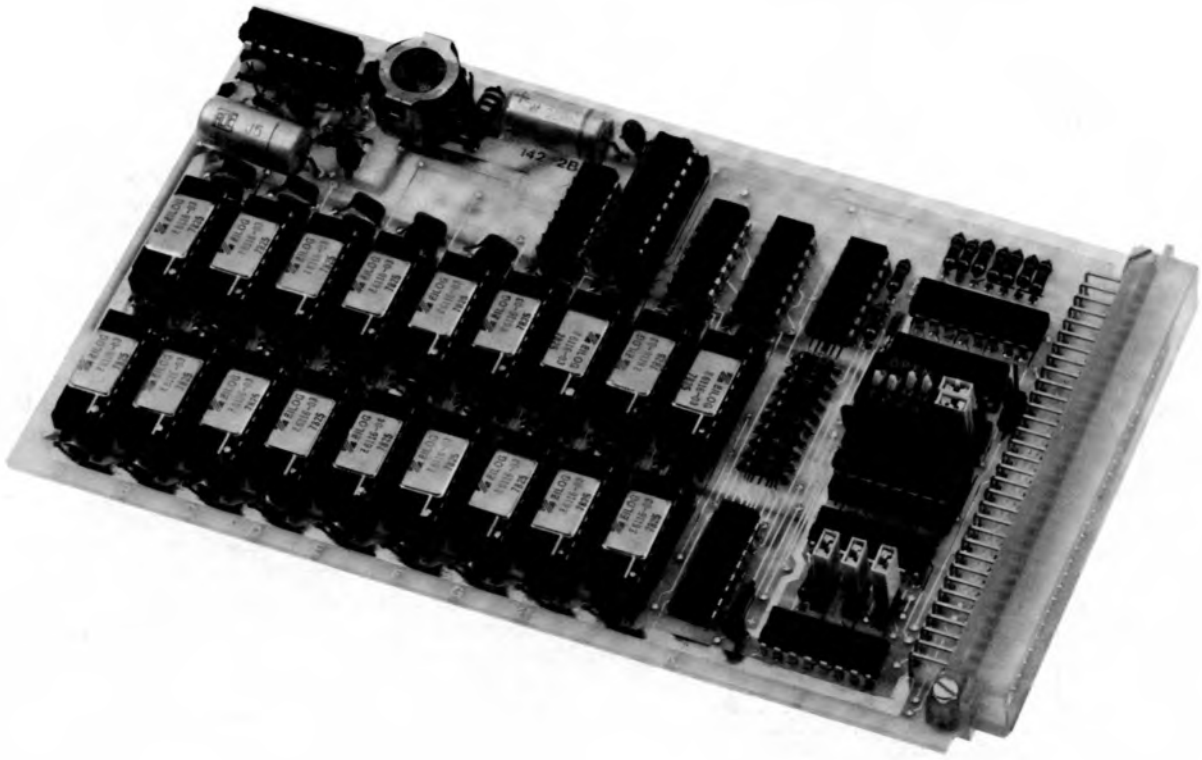
Schaltplan wird mit der Baugruppe mitgeliefert.



Z80A-ECB/D

Dynamische Speicher-erweiterungsbaugruppen

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

7. Dynamische Speichererweiterungsbaugruppen der Serie Z80A-ECB/D

Die Baugruppen Z80A-ECB/D sind dynamische Speichererweiterungsbaugruppen zur Arbeit mit CPU's mit max. Arbeitsgeschwindigkeit von 4 MHz.
Die Erzeugung der Signale RAS und CAS, sowie der Versorgungsspannungen −5 V und +12 V erfolgen auf der Baugruppe, so daß sie mit einer einzige +5 V-Versorgung betrieben werden können.
Die Baugruppen mit der Kennzeichnung -P verfügen darüber hinaus über einen hardwaremäßigen Paritätsgenerator und -Prüfer und einer neunten Speicherzelle pro Wort.
Innerhalb der Baugruppenserie Z80A-ECB/D sind Kapazitäten von 32 kByte und 16 kByte pro Platine verfügbar.

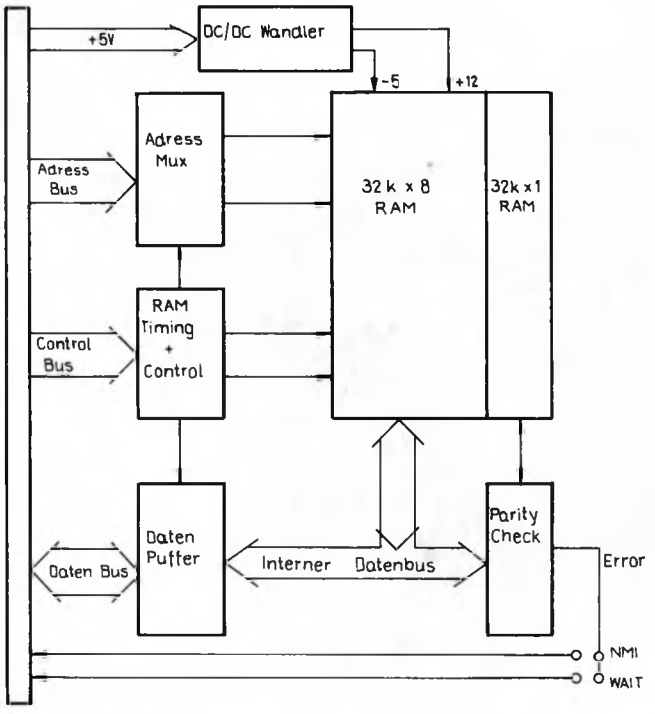


Bild 1: Blockschaltung

7.1 Speicherbaugruppe Z80A-ECB/D32-P

Mit der Z80A-ECB/D32-P lassen sich Systeme im Einfacheuropaformat realisieren, die entweder eine große Menge an Datenspeicher oder aber das Programm im Schreib/Lesespeicher haben müssen.
In solchen Systemen ist es wichtig, daß evtl. auftretende Speicherfehler erkannnt werden können. Hierfür verfügt die ECB/D32-P über eine Paritätsprüfeinrichtung. Diese legt bei jedem Schreibvorgang zusätzlich ein Parity-Prüfbit ab. Dieses Prüfbit wird bei jedem Lesevorgang geprüft, falls ein Paritätsfehler auftritt, kann ein nichtmaskierbarer Interrupt ausgeführt werden.
In der zugehörigen Interrupt-Service-Routine können dann entsprechende Alternativmaßnahmen oder sogar automatische Fehlerkorrektur softwaremäßig durchgeführt werden.

Die Z80A-ECB/D32-P besteht aus 4 Funktionsblöcken:
- Timing und Steuerteil
- Speicherbank
- Parity-Erzeugung und -Prüfung
- Stromversorgung

7.1.1 Schaltungseinzelheiten

7.1.1.1 Steuerteil

Kernstück dieses Funktionsblocks ist das Adreß-PROM 7621 mit den Koordinaten L3. Als Eingänge sind in ihm miteinander verknüpft die Systemadressen A12 bis A15, die Schalterstellungen S1 bis S4 und die Signale REFRESH und MEMORY REQUEST.

Die Systemadressen A12 bis A15 werden durch ein vorgeschaltetes 4-bit-Latch zum Zeitpunkt von Memory-Request festgehalten. Der Schalter S4 unterscheidet zwischen den Bestückungsvarianten mit 4 kbit dyn. RAM's oder 16 kbit dyn. RAM's. Je nach Schalterstellung von S4 sind dann mit S1 bis S3 die verschiedenen Adreßbereiche, die mit dieser Karte erreicht werden können, einstellbar (Bild 1).

An seinen Ausgängen stellt das AdreßDecoder-PROM die Signale

Read-Adress-Strobe 0 und
Read-Adress-Strobe 1.

zur Verfügung. Diese dienen zum Einspeichern der Spaltenadresse in die jweils angewählte Speicherbank bzw. beim Refresh-Zyklus zum refreshen des gesamten Speicher. Das Signal Memory Select gibt den Datenpuffer frei. Das Signal zum Umschalten der Adreßmultiplexer bzw. zur Generation des Column-Adress-Strobe wird über eine Kette von Low-Power-TTL-Invertern erzeugt. Die Verzögerungskette hat mehrere Abgriffe um flexibel gegenüber unterschiedlichen Anforderungen im Timing zu sein. Die Standardeinstellung dieser Jumper passend für RAM-Chips verschiedener Hersteller ist in Bild 1 wiedergegeben.

7.1.1.2 Speicherblock

Bei der Auslegung des Speicherbereichs wurde auf die Besonderheiten der dynamischen Speicher Rücksicht genommen. So sind eine genügende Anzahl von induktionsarmen Blockkondensatoren für alle Versorgungsspannungen vorgesehen. Die kritischen Signalleitungen sind durch ausreichende Masseabdeckung gesichert. Schließlich wurde die Quellimpedanz der Adreß- und Kontroll-Signaltreiber durch Längswiderstände an die Leitungsimpedanz der gedruckten Schaltung angepaßt. Dadurch werden Überschwinger vermieden, die bei negativem Vorzeichen zu den berüchtigten Latch-up-Effekten der Adreßdecoder des dyn. RAM's führen könnten.

7.1.1.3 Paritäts-Erzeugung und Überwachung

Ein 8 bit Parity-Generator ist mit dem internen Datenbus an ECB/D32-P verbunden. Sein Ergebnis wird in das neunte bit bei jedem Schreibzyklus eingespeichert. Bei jedem Lesezyklus wird das Ergebnis des Parity-Generators und das eingespeicherte neunte bit exclusiv oder miteinander verknüpft. Ein D Flipflop speichert eine eventuelle Diskrepanz bis zum Auftreten eines weiteren Schreibzyklus. Wenn, wie durch Jumper 11 einstellbar, ein Parity-Fehler zu einem nicht-maskierbaren Interrupt führt, führt der dann auftretende Schreibzyklus (schreiben der Return-Adresse auf den Stack) zum Löschen des Fehler-Flipflops führen. Die Service-Routine für den non-maskable-Interrupt kann dann zum Veranlassen weiterer Maßnahmen per Software führen.

7.1.1.4 Stromversorgung

Da gegenwärtig erhältliche dynamische RAM's Versorgungsspannungen von +12 und ±5 Volt benötigen, ist auf der ECB/D32-P ein Gleichspannungswandler untergebracht, der mit dieser Karte nur eine einzige Versorgungsspannung von 5 Volt benötigt. Der Gleichspannungswandler wurde durch einen extern hinzugefügten Schalttransistor verstärkt. Dadurch steht genügend Leistung zur Verfügung, um eine sichere Versorgung auch bei ungünstigen Refresh-Zeiten (Prozessor im HALT) zu gewährleisten.

7.1.1.5 Anwendungshinweise

Die gesamte Logik zum Demultiplexen und zur Erzeugung des Speicherrefresh in Z80-CPU-gesteuerten Systemen ist mit auf der Baugruppe untergebracht. Der Refresh erfolgt abhängig vom laufenden Programm max. alle 0,7 msec unter der Kontrolle der Z80-CPU vollautomatisch ohne zusätzliche Software und ohne Verlust von Verarbeitungszeit, d.h. für den Anwender verhält sich die Baugruppe nach außen hin genauso wie jede statische Speichererweiterung.
Folgende Punkte sind jedoch zur Vermeidung von Informationsverlusten im dynamischen Speicher zu beachten:

- ☐ Bei Verwendung des Signals RESET ist dafür Sorge zu tragen, daß die Dauer dieses Signals unter 2 msec liegt, da die CPU kein Refresh-Signal erzeugt, solange ihr RESET-Eingang auf LOW liegt. Ein einfacher Schalter genügt an dieser Stelle also nicht, falls Informationsverluste im dynamischen RAM beim Systemrückstellvorgang unerwünscht ist; die Problematik kann z. B. mit einem Differenzierglied umgangen werden, das der RESET-Leitung der CPU vorgeschaltet wird.

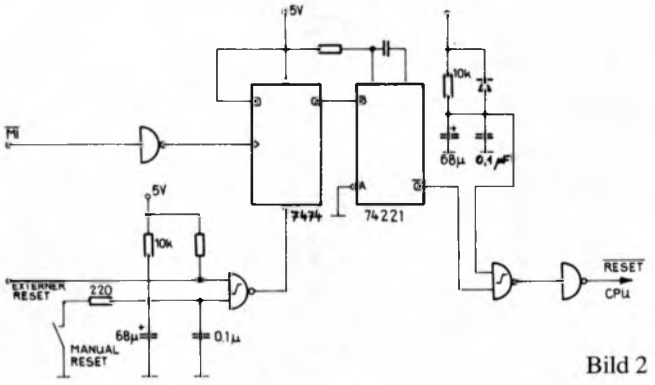


Bild 2

- ☐ Bei direktem Speicherzugriff (DMA) ist die Länge der DMA-Zugriffe so kurz zu wählen, daß immer noch alle 2 msec sämtliche Speicherstellen des dynamischen RAM's „aufgefrischt" werden.
  Die Z80-CPU schaltet bei jedem M1-Zyklus die Refresh Adresse um 1weiter. Der Zeitraum zwischen zwei Refreshvorgängen ist abhängig von der Art der Befehle, die in einem Programm vorkommen. Geht man von den längsten Befehlen (23 Taktzyklen) aus, so beinhalten diese zwei M1-Zyklen.
  Damit ergibt sich:

$$1\ \text{Refreshzyklus} = \frac{23}{2}\ \text{Taktzyklen} \cdot 128$$
$$\approx 0{,}7\ \text{ms bei einer Taktfrequenz von } 2{,}5\ \text{MHz}.$$

  Da die RAMS alle 2 ms einen vollständigen Refreshzyklus durchlaufen haben müssen, ergibt sich eine Restzeit von 1,3 msec für andere Aktivitäten.
- ☐ Das eben über DMA Gesagte ist sinngemäß auch auf die Benützung des WAIT-Eingangs der Z80-CPU anzuwenden, da auch WAIT-Zyklen den Refresh-Vorgang vorübergehend unterbinden (s. Z80-CPU-Technical Manual).
  Die Dauer eines periodisch anliegenden WAIT-Signals muß also unter 1,3 msec liegen.

7.2 Speicherbaugruppe Z80 A-ECB/D 32

Die Baugruppe ist mit 16 kbit dynamischen Speicherbausteinen aufgebaut und 32 k × 8 bit-weise organisiert.
Ein Paritätsgenerator/Prüfer ist auf dieser Variante **nicht** vorgesehen.
Die übrigen technischen Daten entsprechen denen der Baugruppe Z80 A-ECB/D 32-P.

7.3 Speicherbaugruppe Z80 A-ECB/D 16

Die Baugruppe ist mit 16 kbit dynamischen Speicherbausteinen aufgebaut und 16 k × 8 bit-weise organisiert.
Ein Paritätsgenerator/Prüfer ist auf dieser Variante **nicht** vorgesehen.
die übrigen technischen Daten entsprechen denen der Baugruppe E 80 A-ECB/D 32-P.

Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	350 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 160 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80 A-ECB/D 32

Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	330 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 160 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und s nach ECB-Standard
Beschriebene Version:	Rev. II

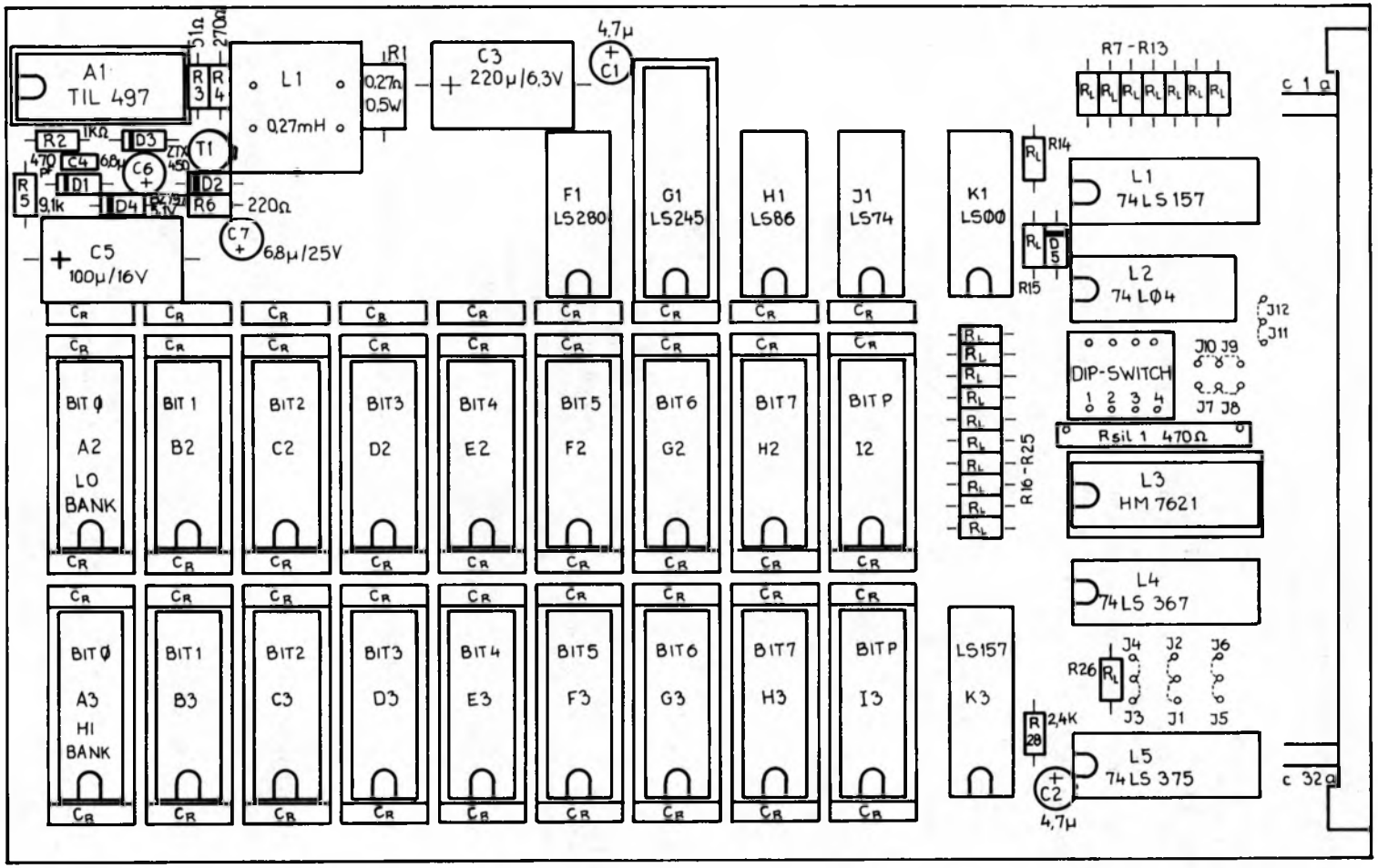
Schaltplan wird mit der Baugruppe mitgeliefert.

Z80 A-ECB/D 16

Technische Daten:

Spannungsversorgung (Gleichspannung):	+5V, ±5%
Stromaufnahme (typisch):	280 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.



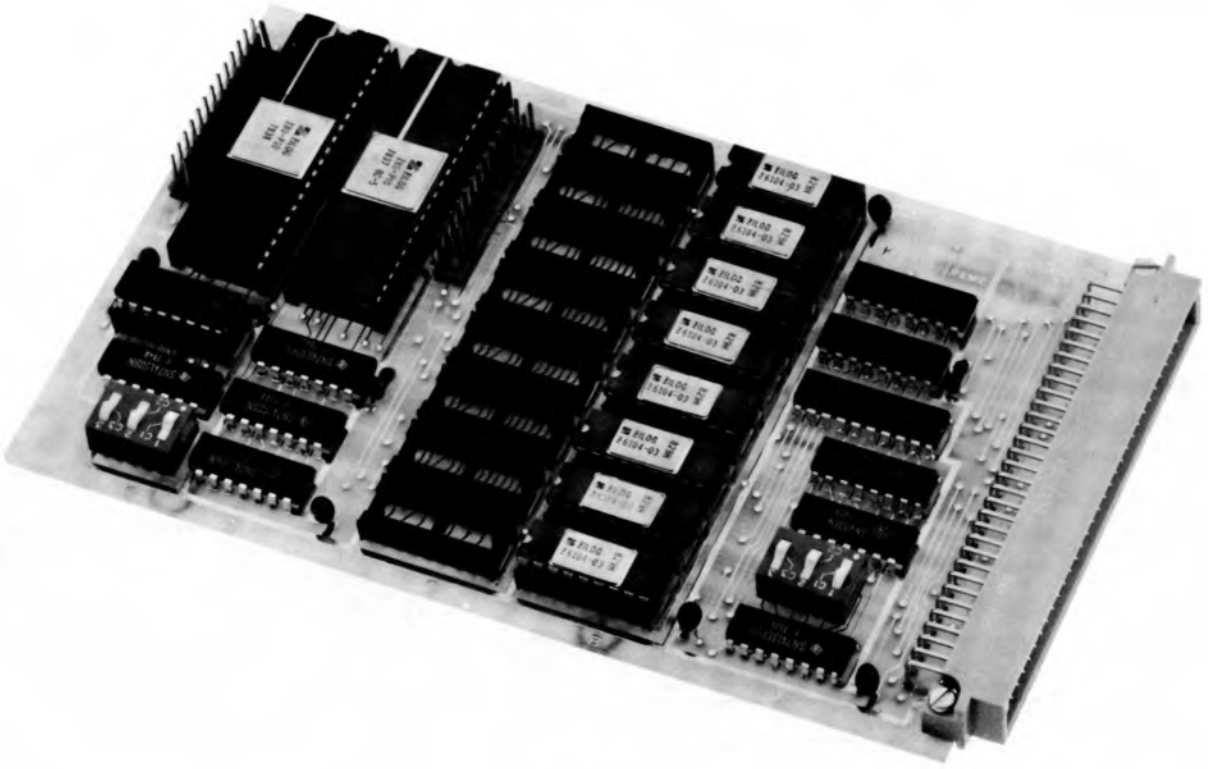
D1 - D3, D5 = 1N914

R_L = 51Ω C_R = 0,1µ

Z80A-ECB/S

Speicher-Ein/Ausgabe-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.2 Z80A-ECB-MIKROCOMPUTER-BAUGRUPPEN

8. Speicher-I/O-Erweiterung Z80A-ECB/S

Europakarte steckbar auf einen der fünf Steckplätze des KIT,

- ☐ 2 × PIO ≙ 4 parallel 8 bit I/O Ports mit Quittungsleitungen leitungen
- ☐ 4 kByte statisches RAM (austauschbar auf 8 kB)
- ☐ Einstellung der Speicheranfangsadresse über Kippschalter in Schritten von 8 k
- ☐ Einstellung der Portadressen über Kippschalter
- ☐ sämtliche Signale mit Busanschlußwert von einer TTL-LS-Last
- ☐ alle Adressen zwischengespeichert
- ☐ Daten über bidirektionalen Schmitt-Trigger-Tri-State-Buffer 74LS245 geführt
- ☐ Ausgangsseitig (= PIO Ein/Ausgänge) zwei 26-polige Stiftleisten

8.1 Adressenbelegung

Die beiden PIO's liegen adressenmäßig fest hintereinander und können über den dreifachen Umschalter U2 in Schritten zu je 4 Ports eingestellt werden.

8.2.1 Adreßeinstellung des RAM-Bereiches

Der RAM-Bereich kann mit einem dreifachen Umschalter U1 in Schritten von 8k verändert werden.

8.2 Pinbelegung

Stift-Nr.	26-pol. WWP-Steckerfeld
1	A_0
2	A_1
3	A_2
4	A_3
5	A_4
6	A_5
7	A_6
8	A_7
9	B_0
10	B_1
11	B_2
12	B_3
13	B_4
14	B_5
15	B_6
16	B_7
17	ASTB
18	BSTB
19	ARDY
20	BRDY
21	NC
22	NC
23	+5V
24	+5V
25	GND
26	GND

Bus-Stecker STA

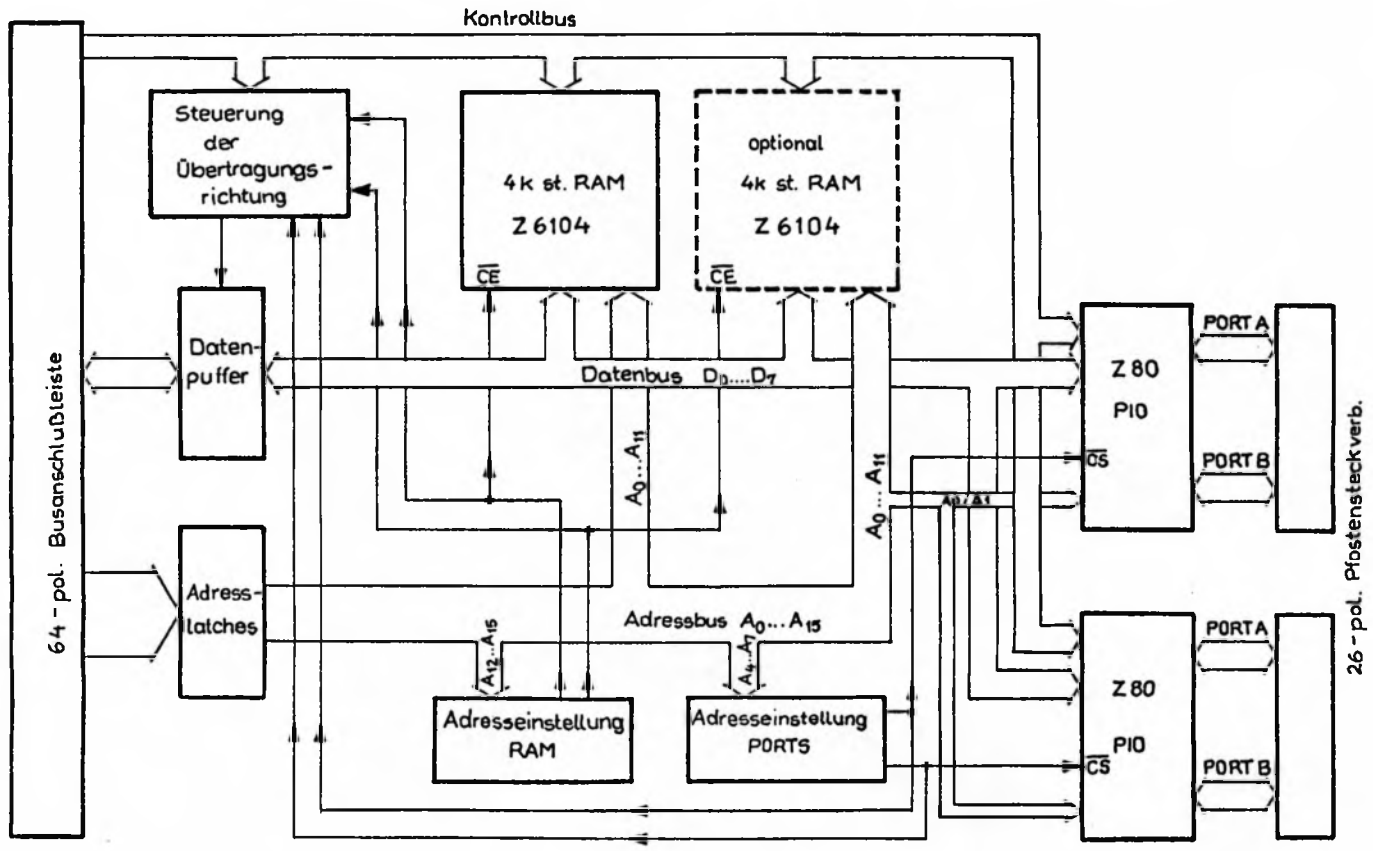
64-poliger Steckverbinder VG 95324
Es handelt sich hier um eine Belegung gemäß der ECB-BUS-NORM, wobei selbstverständlich nur die erforderlichen PIN's angeschlossen sind.

Benennung	Stecker PIN	Bezeichnung
A 0	5c	Adresse 0
A 1	7c	Adresse 1
A 2	6a	Adresse 2
A 3	6c	Adresse 3
A 4	7a	Adresse 4
A 5	8a	Adresse 5
A 6	9a	Adresse 6
A 7	9c	Adresse 7
A 8	8c	Adresse 8
A 9	30a	Adresse 9
A 10	18c	Adresse 10
A 11	17c	Adresse 11
A 12	27c	Adresse 12
A 13	29a	Adresse 13
A 14	18a	Adresse 14
A 15	28c	Adresse 15
D 0	2c	Daten 0
D 1	14c	Daten 1
D 2	4c	Daten 2
D 3	4a	Daten 3
D 4	5a	Daten 4
D 5	2a	Daten 5
D 6	3a	Daten 6
D 7	3c	Daten 7
M 1	20a	Maschinenzyklus 1
MRQ	30c	Memory Request
IORQ	27a	IN/OUT Request
RD	24c	Read
WR	22c	Write
INT	21c	Interrupt
IEI 1	11c	Int. enable in
IEO 1	16c	Int. enable out
0	29c	Clock 2,45 MHz
+5	1a, c	
GND	32 a, c	

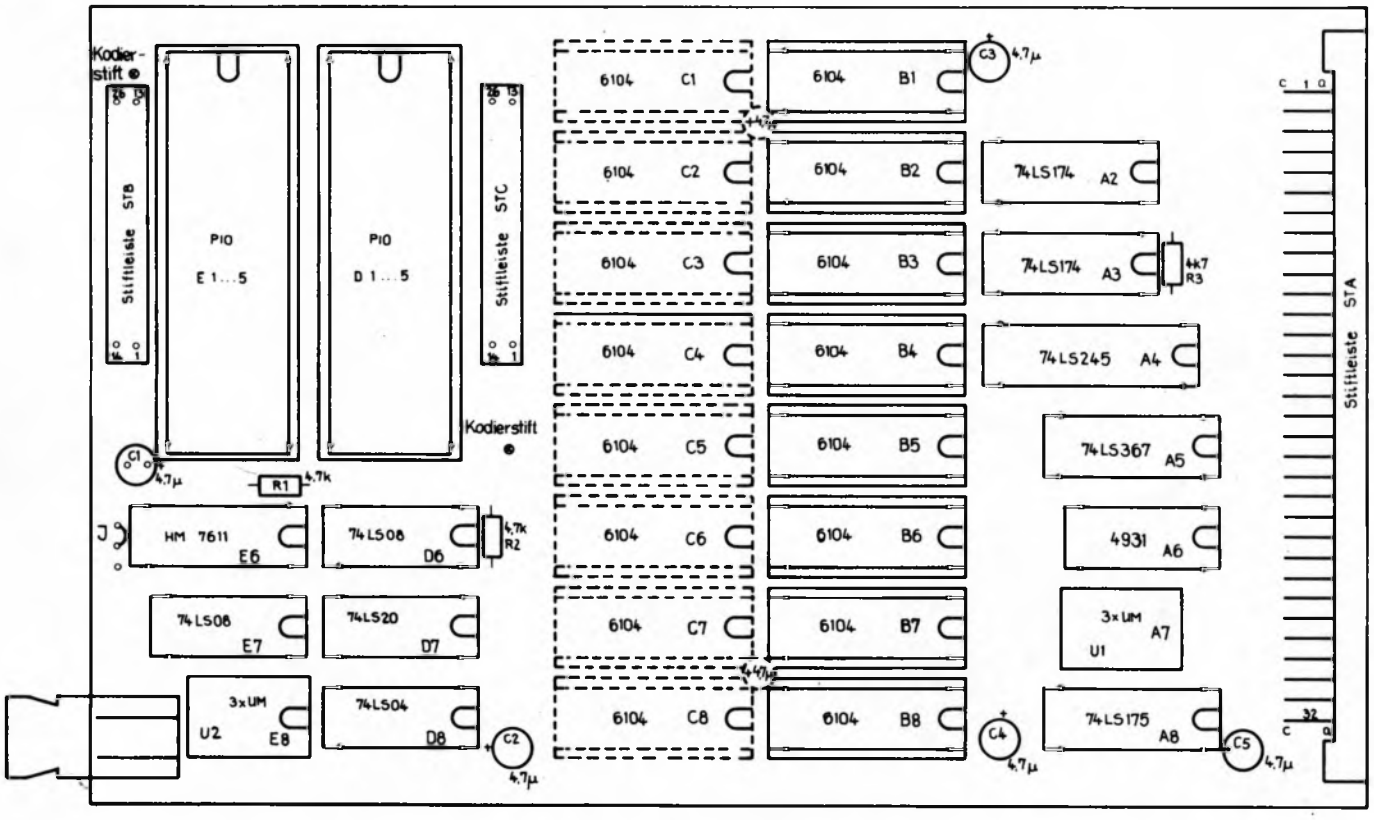
Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	300 mA (bei 4 kByte-RAM-Bestückung)
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 140 g (bei 4 kByte-RAM-Bestückung)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.



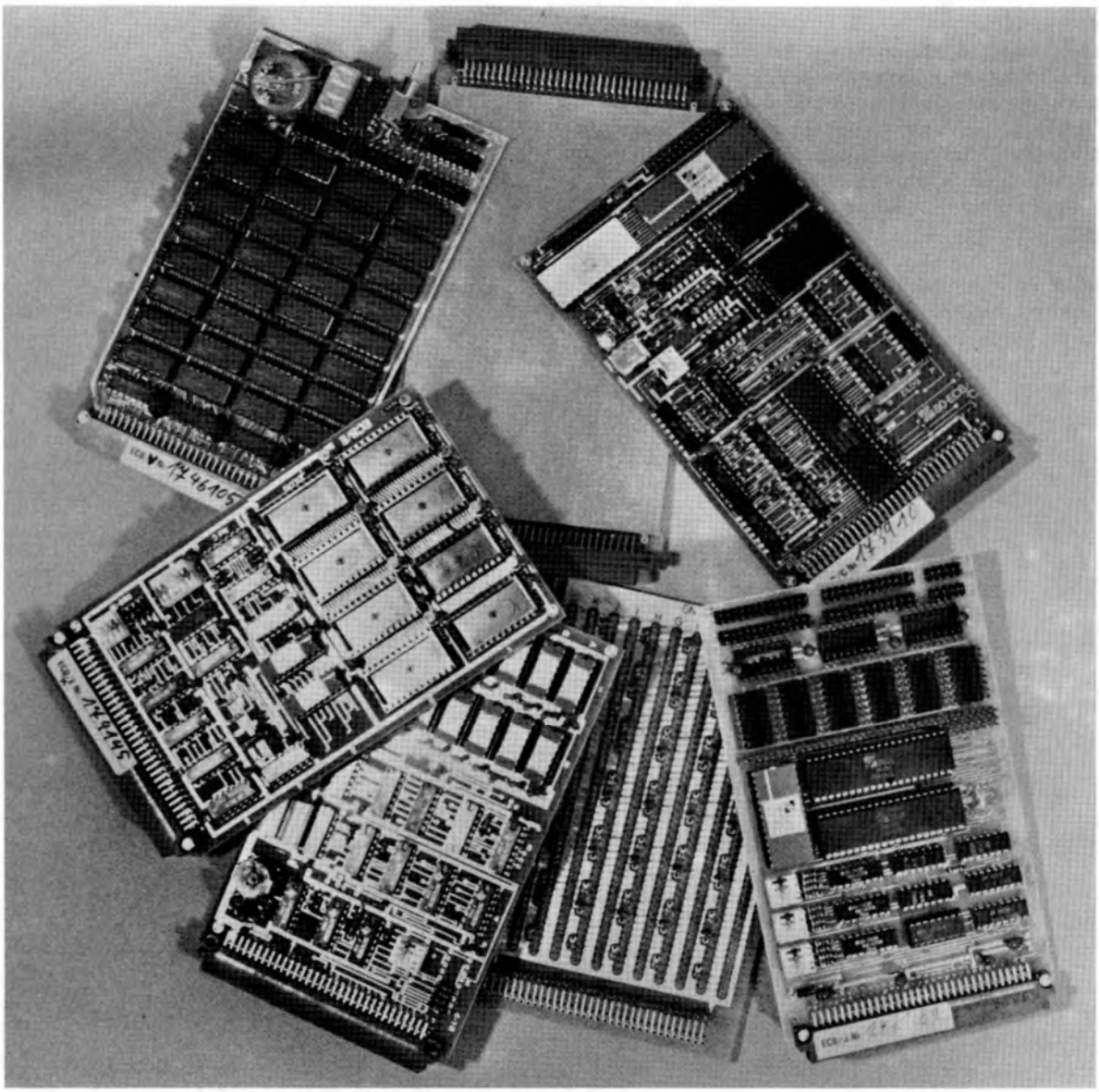
Blockschaltbild des ECB/S



Z80-ECB-SERIE

Mikrocomputer-Baugruppen im Einfach-Europa-Format

KONTRON
ELEKTRONIK GMBH



7.3 Z80-ECB MIKROCOMPUTER-BAUGRUPPEN

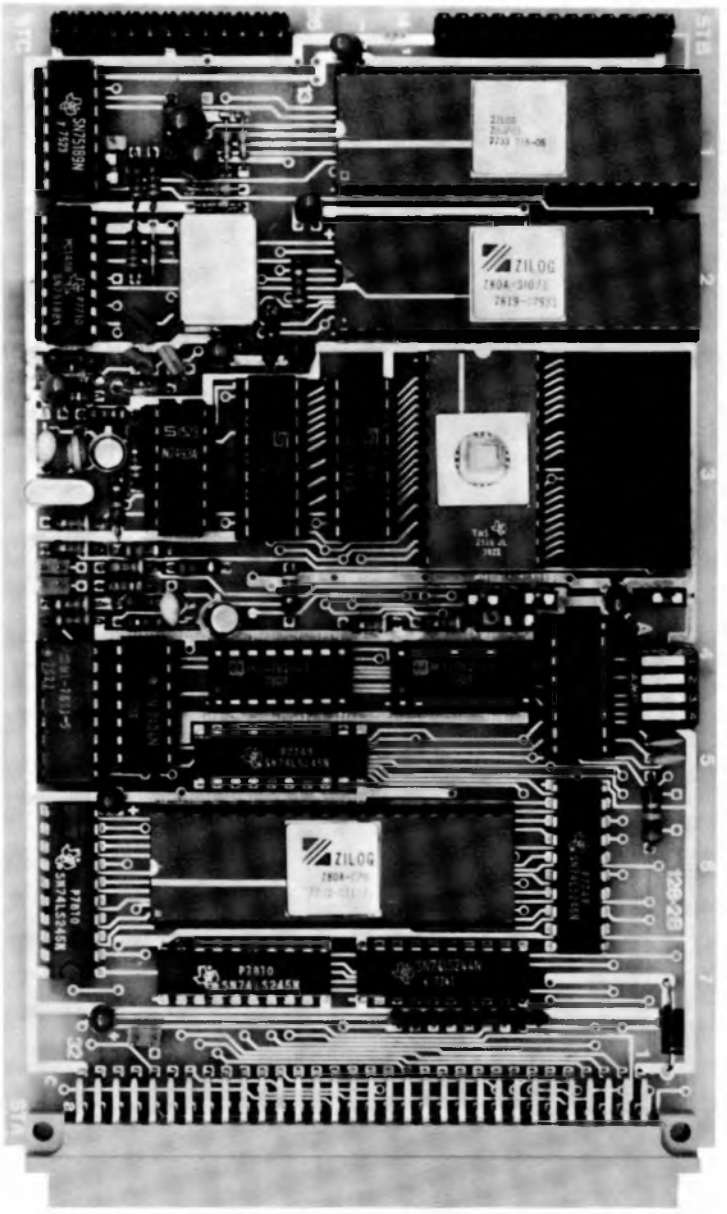
1. Allgemeines

Die seit mehreren Jahren bewährte Baugruppenserie Z80-ECB basiert auf dem Mikroprozessor Z80-CPU, der mit einer Taktfrequenz von 2.5 MHz arbeitet.
Die Z80-ECB-Serie stellt eine besonders wirtschaftliche Realisierungsmöglichkeit von Anwendermikrocomputer-Systemen auf der Basis von Standardhardware dar; die Verwendung dieser fertigen Standards ist in den meisten Fällen bis zu etwa 1000 Stück pro Typ wirtschaftlich.
Bei höheren Stückzahlen besteht die Möglichkeit, die Fertigungsrechte und -Unterlagen von der KONTRON Elektronik GmbH zu erwerben.
Die Tatsache, das mehrere Hersteller zur KONTRON-Z80-ECB-Serie kompatible Baugruppen anbieten und das ungewöhnlich umfangreiche Spektrum von der KONTRON Elektronik GmbH hergestellter Karten machen diese Serie zu einem universellen, attraktiven Hilfsmittel zur Bewältigung von Problemen auf der Basis modernster Technologie.
Die Baugruppen sind zu den Serien Z80 A-ECB und Z80-KIT kompatibel.

Die folgende Anschlußbelegung ist für alle Baugruppen der ECB-Serien verschiedener Hersteller genormt. Die einzelnen Platinen sind durch 64-polige Steckverbinder (VG 95324) über 1:1 Verdrahtung miteinander verbunden. Die Angaben über Fan In und Fan Out beziehen sich auf die Zentralplatine.

Z80-ECB/C8 Zentralbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3 Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

2. Die Zentralbaugruppe Z80-ECB/C8

2.1 Übersicht

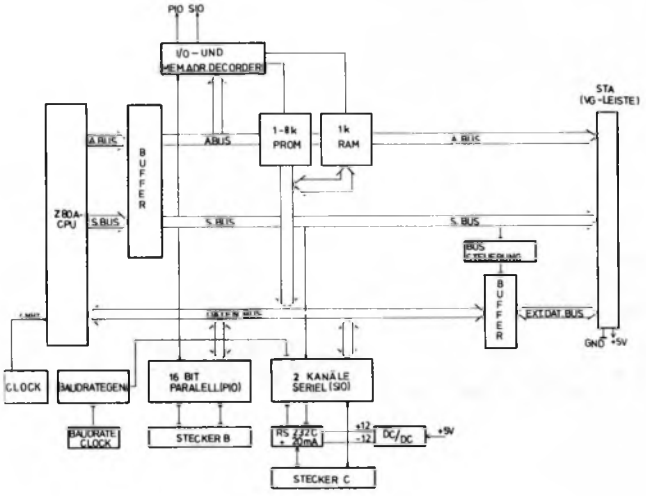
Die Baugruppe Z80-ECB/C8 ist eine Mikrocomputerbaugruppe auf der Basis des Mikroprozessors Z80-CPU im Einfacheuropaformat; sie arbeitet mit dem bekannten ECB-Bus, ist also zu allen Baugruppen der Serien Z80-ECB, Z80A-ECB und Z80-KIT elektronisch und elektromechanisch voll kompatibel.
Die Computerkarte ist sowohl als kompletter, selbständiger Ein-Platinenrechner zu verwenden, als auch als über den vollen Leistungsumfang des Z80-Konzepts erweiterbare Zentralbaugruppe.
Dies wurde durch Verwendung neuester Technologie und höchster Integration (wie Serieninterface Z80-SIO, 4 kbit-statische RAM's und 32 k/16 k-EPROM's), volle Adreß-Ausdekodierung und Pufferung sämtlicher Bus-Signale erreicht.
Besonders bemerkenswert ist die Eignung des Computers zur Realisierung von Multirechnersystemen, wobei die einzelnen Computer über volle RS 232-Serienschnittstellen gekoppelt werden können, die ihrerseits mit hardwaremäßig auf der Platine implementierter (Z80-SIO) HDLC/SDLC-Übertragungsprozedur arbeiten. Als Stromversorgung dient eine einzige +5 V Quelle mit typisch 900 mA.
Bestückt mit 3 hochintegrierten ZILOG-Bausteinen (CPU, PIO, SIO) eignet sich die ECB/C8 zum Aufbau von Computersystemen der verschiedensten Komplexitätsstufen. Kleine bis mittlere Systeme können auf Grund der Leistungsfähigkeit der ZILOG-Bausteine und dem großzügig ausgelegten Speicherangebot auf der ECB/C8 (1 k RAM und bis zu 8 k PROM) bereits mit dieser einen Baugruppe realisiert werden.

2.2 Schaltungsbeschreibung

Die Baugruppe enthält folgende Hardware-Komponenten:

- Z80-CPU
- Dekoder für vollen Speicher- und I/O-Ausbau
- Schmitt-Trigger-Pufferung aller Bussignale
- DMA-fähige Bussteuerung
- Platz für 1–8 kByte Festwertspeicher (zulässige Typen 2758, i2716, i2732, HM 7641, HM 7681)
- 1 kByte statisches RAM (nMOS oder CMOS)
- Zwei 8 bit Parallel-Schnittstellen (1 × Z80-PIO)
- 2 voll duplex Serienschnittstellen (1 × Z80-SIO)
- Normgerechtes RS 232 C und 20 mA-Interface
- 14 über Schalter einstellbare Baudraten
- Gleichspannungswandler für Serieninterface (±12 V)
- Busseitig 64-poliger Normstecker mit ECB-Bus-Belegung
- Ausgangsseitig zwei 26-pol. 3M-WWP-Pfostensteckverbinder

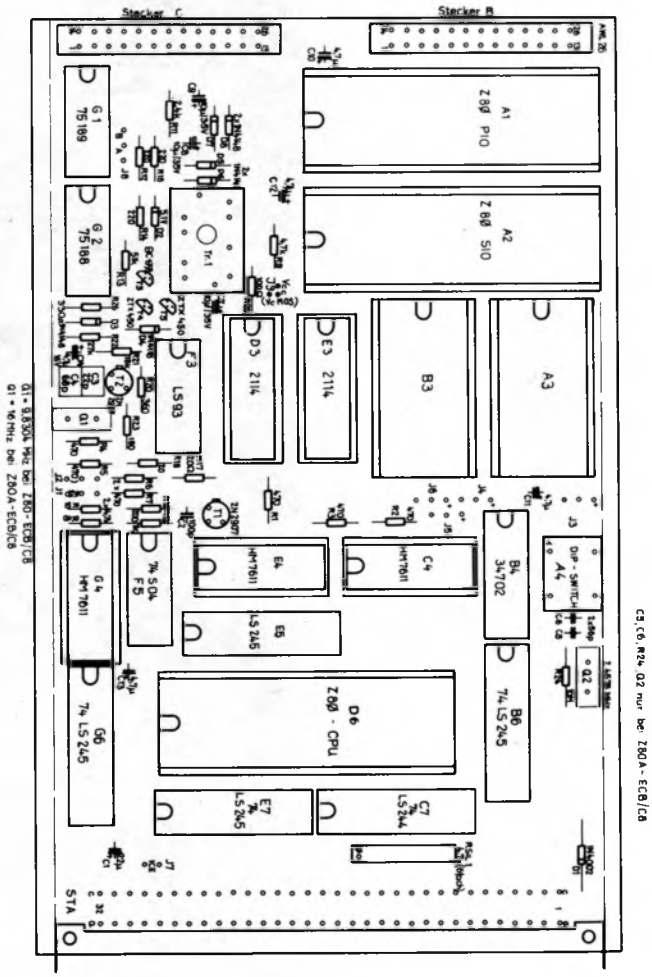
Bild 1 zeigt das Blockschaltbild der Baugruppe



Technische Daten:

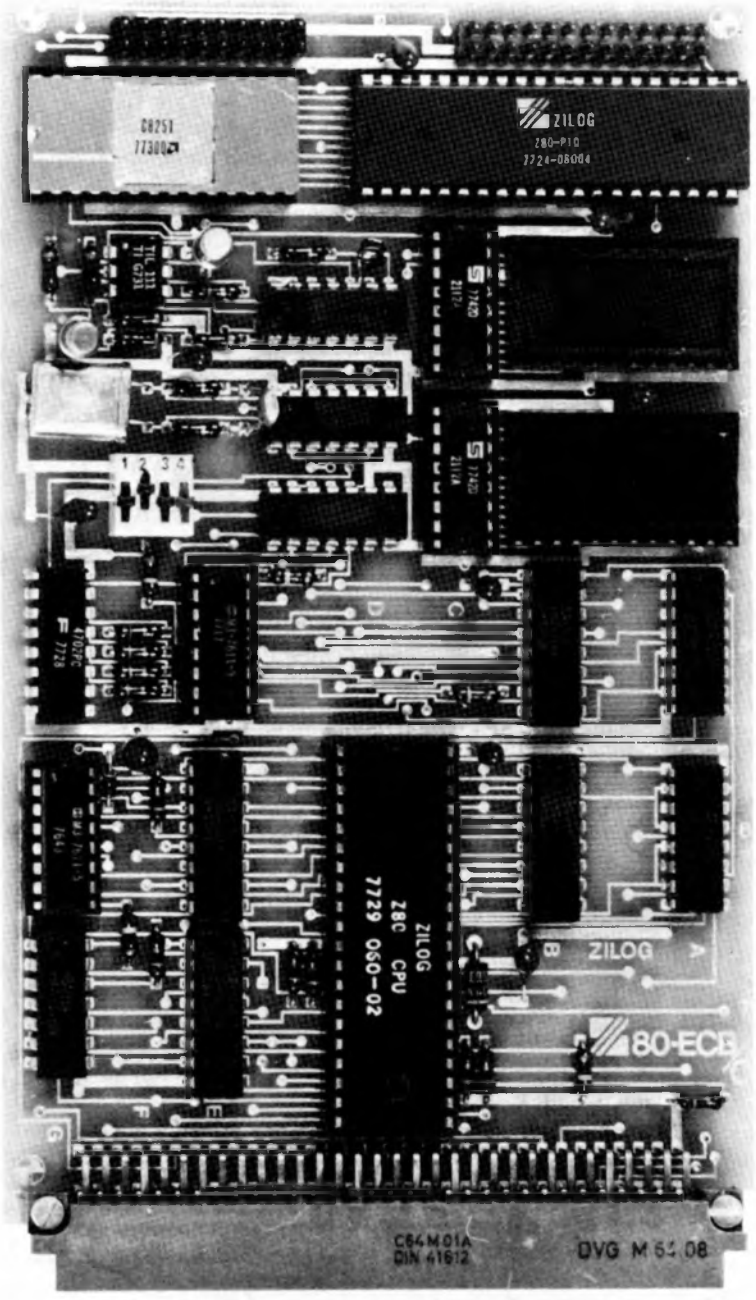
Spannungsversorgung (Gleichspannung):	+5 V, ± 5%
Stromaufnahme (typisch):	1000 mA (ohne PROM's)
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 150 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.



Z80-ECB/C Zentralbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3 Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

3. Die Zentralbaugruppe Z80-ECB/C

3.1 Schaltungbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten

- Z80-CPU
- Decoder und Puffer für vollen Speicher-(64 kByte) und I/O-Ausbau

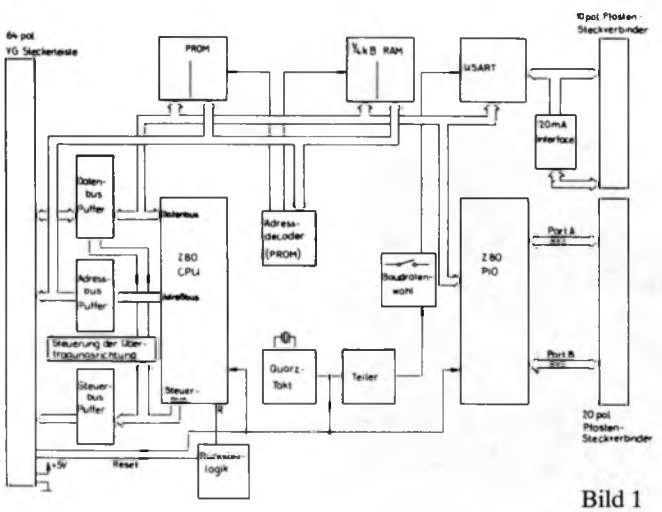


Bild 1

- Platz für 2 kByte PROM
- 256 Byte RAM
- 2 Parallelschnittstellen (1 Stück Z80-PIO)
- 1 Serienschnittstelle mit RS 232 und 20 mA Stromschleifen-Interfaces
- Über Schalter 14 Baud-Raten einstellbar
- Busseitiger 64-poliger Stecker nach DIN 41612 (VG 95324)
- I/O-seitig 3M-WWP-Pfostensteckverbinder

Aus der Blockschaltung (Bild 1) ist die Verknüpfung der einzelnen Funktionseinheiten zu erkennen; die Busleitungen sind gepuffert. Die Adreßeingänge der auf der Z80-ECB/C untergebrachten Speicherbausteine werden über diese Puffer, die Datenleitungen vom CPU-Datenbus direkt angesprochen.

Festwert- und Schreib/Lese-Speicherbereich auf der Platine haben durch den Adreßdecoder vorgegebene Anfangsadressen. Die Baudrate für die Serienschnittstelle wird hardwaremäßig vom Quarz-Systemtakt abgeleitet.

Hinweise zur Interrupt-Behandlung

Bei Mehrbaugruppen-Systemen, die unter Interrupt arbeiten, existiert grundsätzlich das Problem, die RETI-Anweisung über die Datenbuspuffer zu bringen.

Da einerseits Interrupt auf der Zentralplatine möglich sein muß, andererseits das Anwenderprogramm vom Entwicklungssystem auf beliebige (also auch außerhalb des 1kB-Bereichs der ECB/C) Speicherbereiche gelegt werden kann, existieren zur Testbarkeit der Anwender-Programme über den Echtzeit-Testadapter des Z80-Entwicklungssystems nur folgende mögliche Programmstrukturen:

aa) Im gesamten Programm kommt nur ein einziger RETI-Befehl vor, der im Speicherbereich 0 ... 1 kByte stehen muß (adressiert über eine Marke IRET).

ab) Der Rücksprung aus weiteren Interrupt-Bedienroutinen erfolgt dann über den Befehl

JP IRET.

oder

bb) alle Interrupt-Service-Routinen liegen im RAM/ROM-Speicherbereich der ECB/C-Baugruppe.

Wird in der Testphase mit dem Z80-Entwicklungssystem-internen Speicher gearbeitet, so ist außerdem sicher zu stellen, daß **keine** Prom's in den Promsockeln der ECB/C stecken. Ist dies nicht der Fall, kommt es bei jeder Ausführung eines RETI-Befehls zu einem Buskonflikt, der in der Regel verhindert, daß die I/O-Bausteine den RETI-Befehl richtig decodieren.

Adressen-Belegung des Z80-ECB/C

Speicher

Der Speicher ist in 4 kByte-Seiten eingeteilt, bei Verwendung des ¼ kByte-Speichers auf der ECB/C-Platine bleibt der Rest der Seite, in der das RAM liegt, unbenutzt.

Die ausgelieferte Standardversion hat folgende Zuweisung (Bild 4):

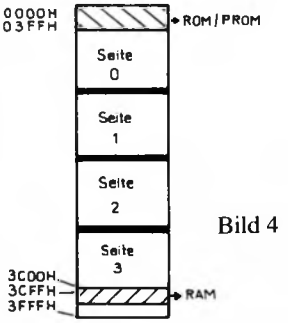


Bild 4

ROM/PROM

Adresse: 0 0 0 0 ... 0 3 F F } 1 kByte

RAM

3 C 0 0 ... 3 C F F } ¼ kByte

Der RAM-Bereich wurde an das obere Ende der Seite 3 gelegt, da ECB-Systeme dann problemlos mit der Minimal-Ausrüstung des Z80-Entwicklungssystems hard- und software-mäßig auszutesten sind.

Ein/Ausgabe

Die Ein/Ausgabe-Bausteine auf dem Z80-ECB/C sind in 1 aus 8 Code der Speicheradress-Bits 0—3 dekodiert.

Z80-PIO:

PORT B/A	A0	LOW = A; HI = B
CONTROL DATA	A1	LOW = DATA; HI = CONTROL
CHIP SEL	A2	LOW = CHIP SELECTED

USART:

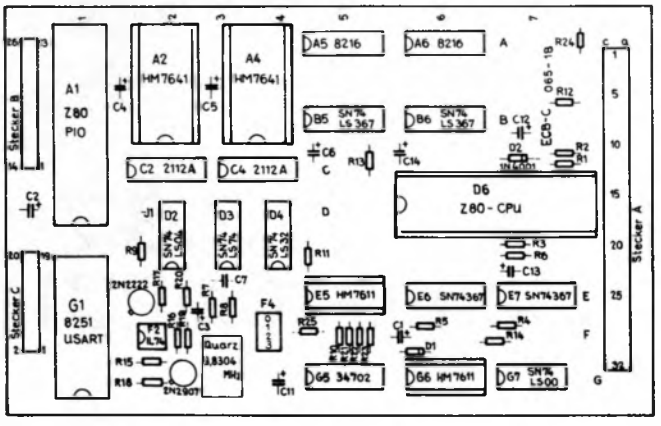
CONTROL/DATA	A1	LOW = DATA; HI = CONTROL
CHIP SEL	A3	LOW = CHIP SELECTED

Es ist zu beachten, daß bei Erweiterung um weitere I/O-Bausteine die Adreß-Bits 2 und 3 immer HI sind!

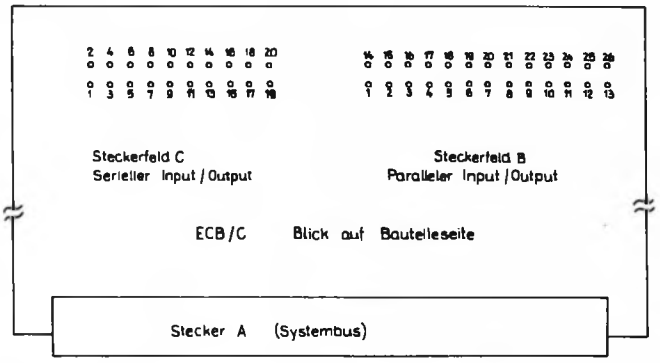
Die Adressen für die Ein/Ausgabe-Bausteine ergeben sich damit wie folgt:

PIO:	PORT B:	DATA 09H
	PORT B:	CONTROL 0BH
	PORT A:	DATA 08H
	PORT A:	CONTROL 0AH
USART:	CONTROL:	06H
	DATA:	04H

Grundsätzlich nicht erlaubt sind die I/O Adressen X0 . . .X3H, da dann sowohl USART als auch PIO ausgewählt sind (X = 0 . . . . .F).



Ein/Ausgabe-Stecker des ECB/C



Stift-Nr.	Steckerfeld C	20 pol. WWP Steckerfeld
1	$\overline{\text{DTR}}$ (Data Terminal Ready; 8251)	
2	Current loop Out +	
3	$\overline{\text{RTS}}$ (Request to send Data; 8251)	
4	Current loop Out — (GND)	
5	$\overline{\text{DSR}}$ (Data Set Ready; 8251)	
6	NC	
7	RS 232 Out +	
8	GND	
9	+5 V	
10	+5 V	
11	NC	
12	Tx EMPTY (Transmitter Empty; 8251)	
13	RS 232 IN +	
14	GND	
15	$\overline{\text{CTS}}$ (Clear to send Data; 8251)	
16	Current loop in +	
17	RxRDY (Receiver Ready; 8251)	
18	Current loop in —	
19	Tx Ready (Transmitter Ready; 8251)	
20	GND	

passendes Gegenstück: 20 pol. Pfosten Verbinder
z. B. Scotch 3421—0000

Stift-Nr.	Steckerfeld B	26 pol. WWP-Steckerfeld
1	GND	
2	A7	⎫
3	A6	⎪
4	A5	⎪
5	A4	⎬ PIO PORTA
6	A3	⎪
7	A2	⎪
8	A1	⎪
9	A0	⎭

Stift-Nr.	Steckerfeld C	20 pol. WWP Steckerfeld
10	$\overline{\text{ASTB}}$	A Strobe
11	$\overline{\text{BSTB}}$	B Strobe
12	ARDY	A Ready
13	+5 V	
14	GND	
15	B7	⎫
16	B6	⎪
17	B5	⎪
18	B4	⎬ PIO
19	B3	⎪ PORT B
20	B2	⎪
21	B1	⎪
22	B0	⎭
23	NC	
24	NC	
25	BRDY B	Ready
26	+5 V	

Gegenstück: 26 pol. Pfosten-Verbinder
z. B. Scotch 3399—0000

Technische Daten:

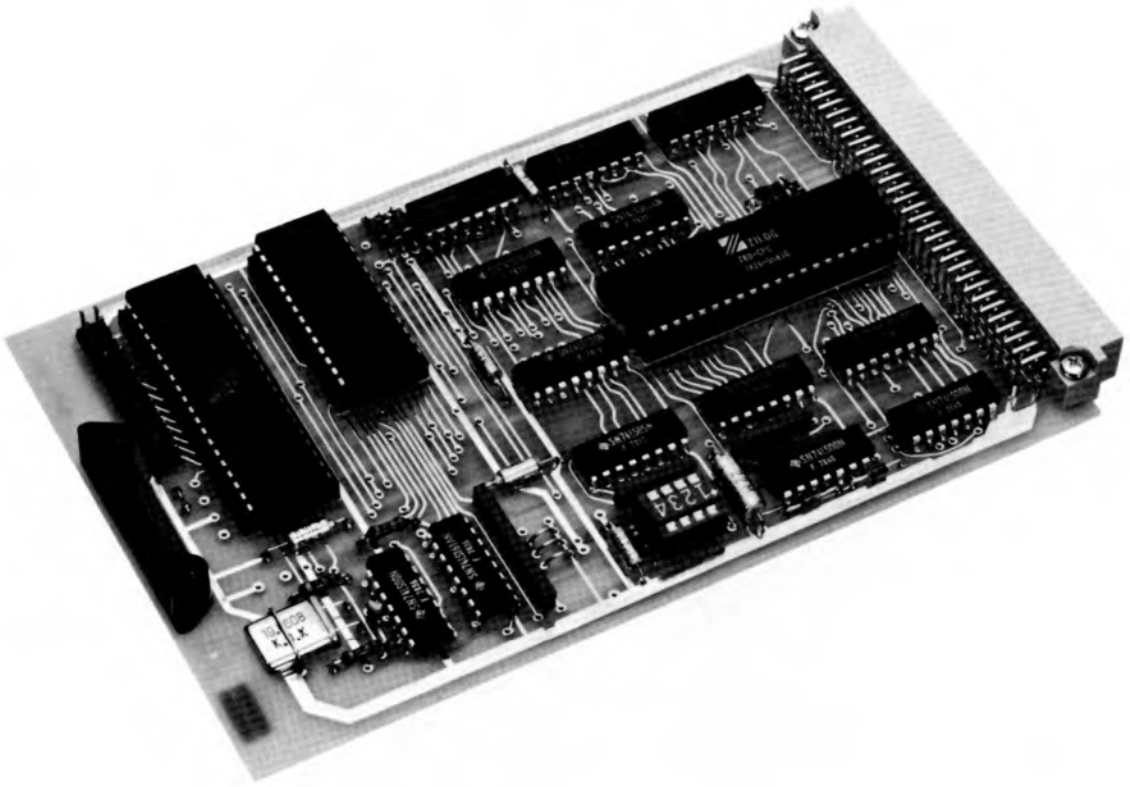
Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	650 mA (ohne PROM's)
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 130 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. III

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80-EML/C

STOLZ AG

Zentralbaugruppe



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

4. Z80-EML/C

4.1 Übersicht

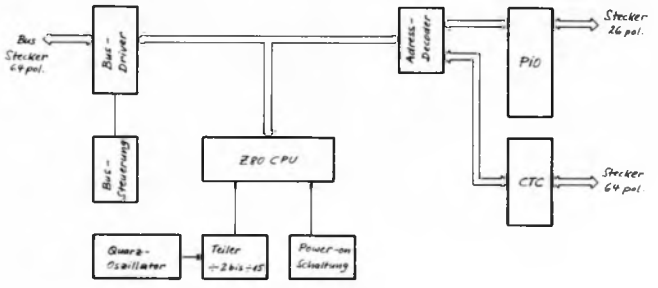
- Z80 Zentraleinheit, Quarzoszillator mit einstellbarem Nachteiler
- Power-on Reset Schaltung
- Programmierbarer Überwachungszeitgeber (Watchdog-timer) mit automatischer Restart-Möglichkeit
- Programmierbare Echtzeituhr mit Eingang auf NMI
- 2 programmierbare Zeitintervallgeber (20 msec . . . 5,12 sec)
- 16 programmierbare Input/Output Leitungen mit Interruptmöglichkeit (PIO-Schaltkreis)
- Bus-Signale gepuffert
- PIO und CTC voll adreßdekodiert

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%, (zuzügl. −5 V, +12 V bei Verwendung von 2704/2708-PROM's)
Stromaufnahme (typisch):	600 mA bei +5 V (ohne PROM's)
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 120 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard

4.2 Schaltungsbeschreibung

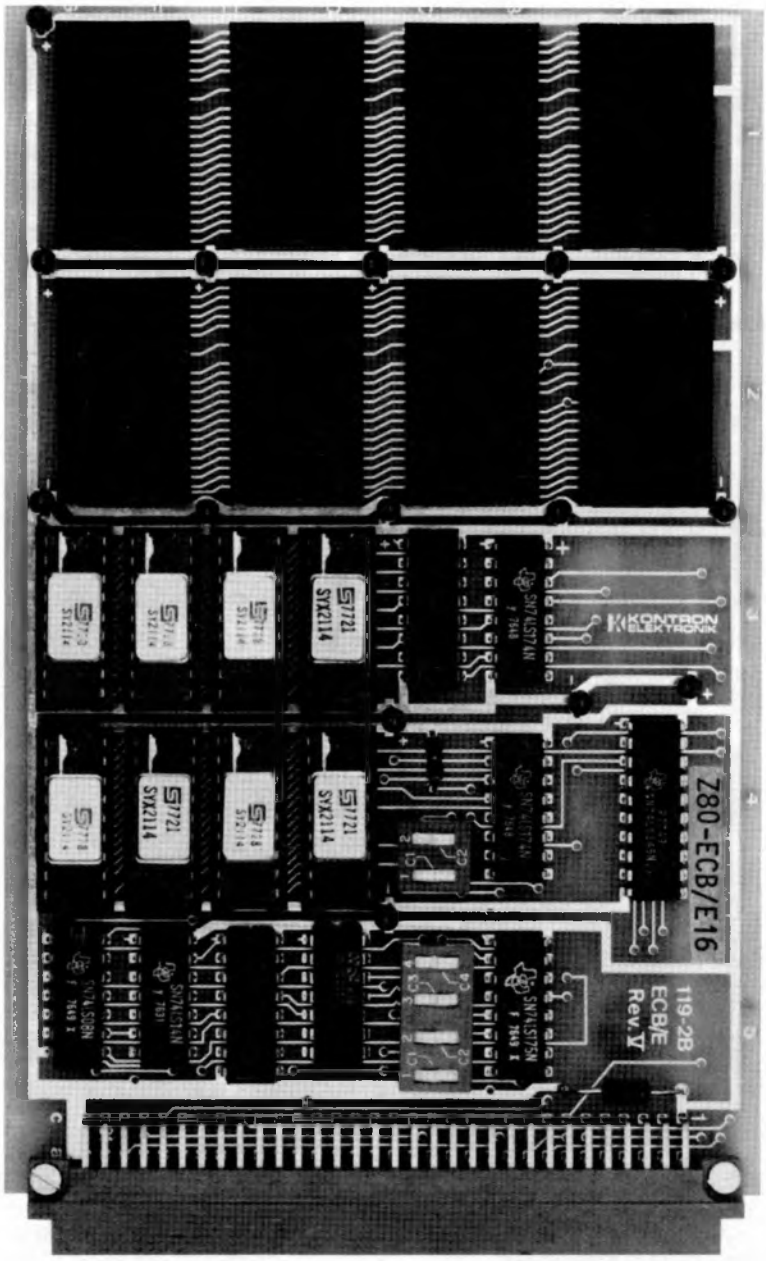
Diese Zentral-Platine ist alternativ in **den** Fällen zu verwenden, die anstelle von Speicher Ein/Ausgabemöglichkeiten in stärkerem Maße erfordern.
Sie verfügt über die Z80-CPU, einen volldecodierten 4-Kanal-Z80-CTC (Zähler/Zeitgeber-Baustein), der durch Lötbrücken als Überwachungstimer (Watchdog) mit automatischem Hardware-Reset oder als Echtzeituhr verwendet werden kann und über eine ebenfalls voll decodierte Z80-PIO mit 16 Ein/Ausgabe-Leitungen und 4 Handshake-Signalen mit Interruptmöglichkeit.



Z80-ECB/E16

Speicher-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

5. Die Erweiterungsbaugruppe Z80-ECB/E16

Schaltungsbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten:

- Bis zu 16 kByte Festwertspeicher (ROM, PROM oder EPROM)
- 1 kByte statisches RAM, bis zu 4 kByte nachrüstbar.
- Decoder
- Schalter zur Speicherbereichsfestlegung
- Busseitiger 64-poliger Stecker nach DIN 41612 (VG 95324)

Bild 1 ist das zugehörige Blockschaltbild. Man sieht, daß auch hier die Systembusse gepuffert sind; dadurch ergibt sich zusammen mit den Puffern auf der Zentralplatine Z80-ECB/C eine zweifache Pufferung, was volle Erweiterbarkeit und erhöhte Störsicherheit garantiert.

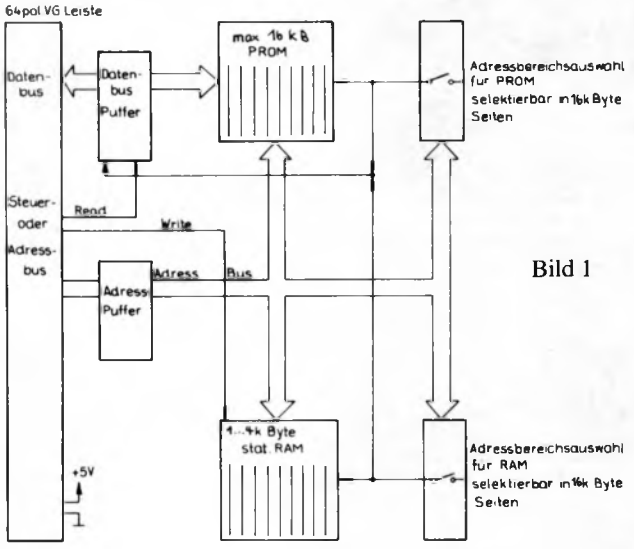


Bild 1

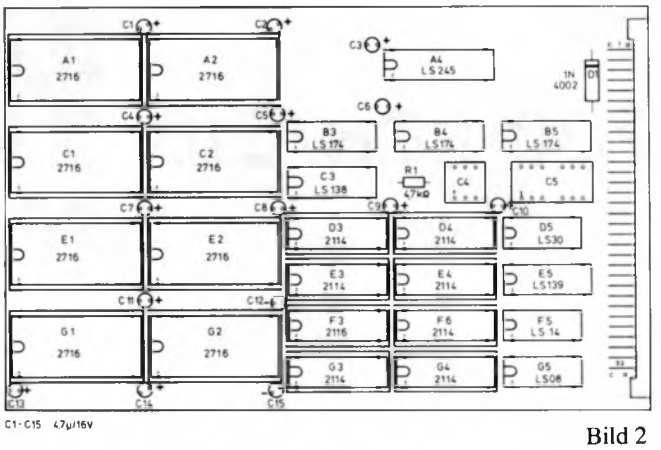


Bild 2

Technische Daten:

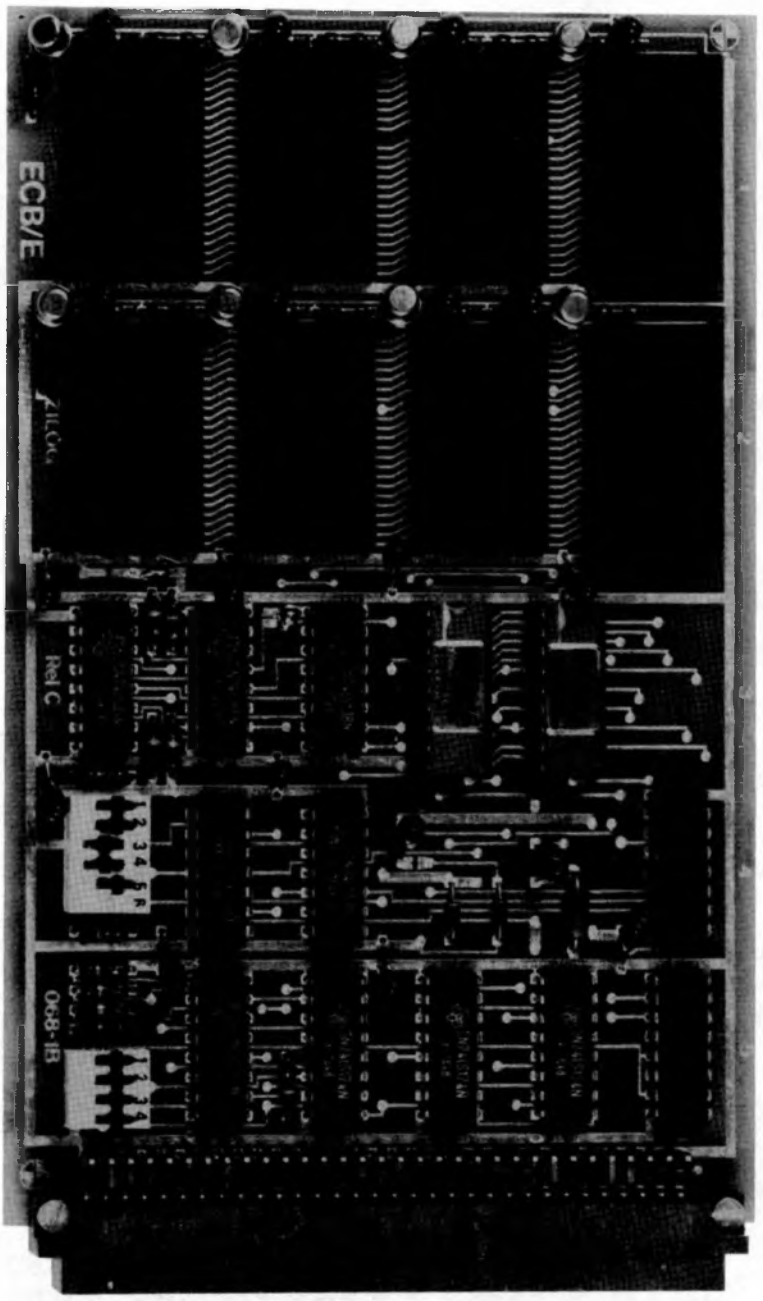
Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	250 mA (ohne PROM's mit 1 kByte-RAM-Bestückung)
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 110 g (ohne PROM's mit 1 kByte-RAM-Bestückung)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. V

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80-ECB/E

Speicher-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

6. Speichererweiterungs-Baugruppe Z80-ECB/E

6.1 Schaltungsbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten:

- 1 . . . 8 kByte Festwertspeicher (ROM, PROM oder EPROM)
- 1 kByte statisches RAM
- Decoder
- Schalter zur Speicherbereichsfestlegung
- Busseitiger 64-poliger Stecker nach DIN 41612 (VG 95324).

Bild 1 ist das zugehörige Blockschaltbild. Man sieht, daß auch hier die Systembusse gepuffert sind; dadurch ergibt sich zusammen mit den Puffern auf der Zentralplatine Z80-ECB0 C8 eine zweifache Pufferung, was volle Erweiterbarkeit und erhöhte Störsicherheit garantiert.

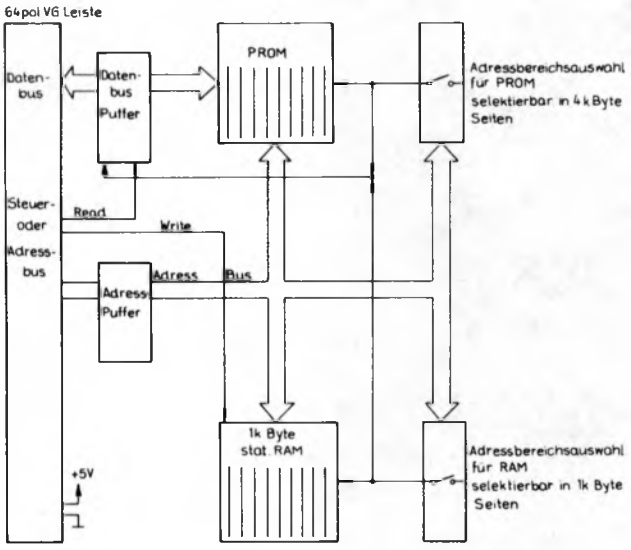


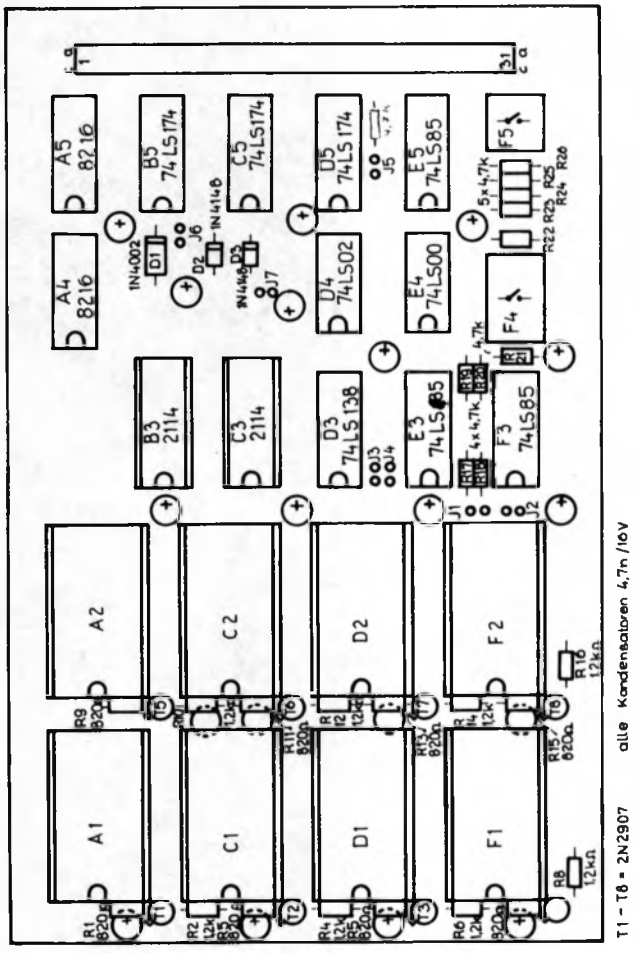
Tabelle 1

½ kB Bipolare PROM's	J1 geschlossen
	J2 geschlossen
	J5 geschlossen
1 kB Bipolare PROM's	J1 geschlossen
	J4 geschlossen
1 kB 2708	J2 geschlossen
	J4 geschlossen

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%, (zuzügl − 5 V und + 12 V bei Verwendung von 2704/2708-PROM's)
Stromaufnahme (typisch):	360 mA (ohne PROM's)
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 110 g (ohne PROM's)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. IV

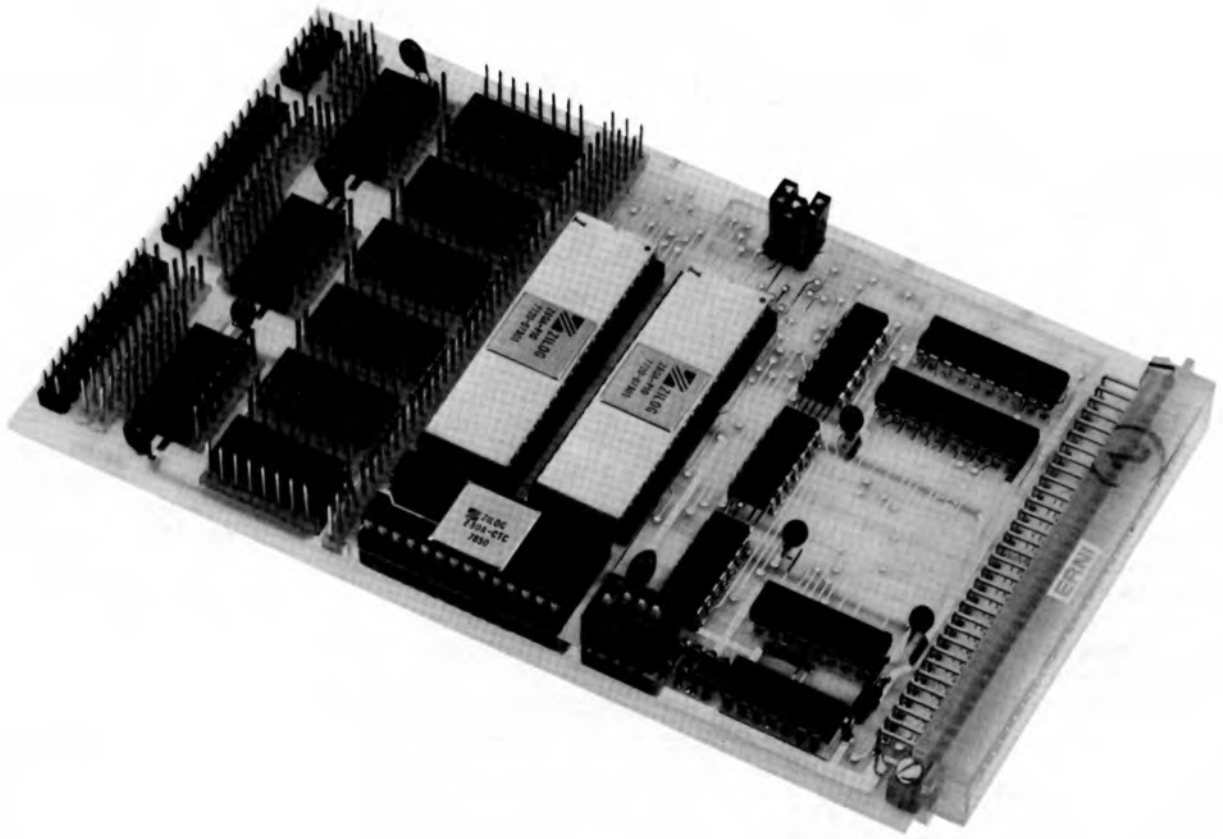
Schaltplan wird mit der Baugruppe mitgeliefert.



Z80-ECB/I

Ein/Ausgabe-Erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



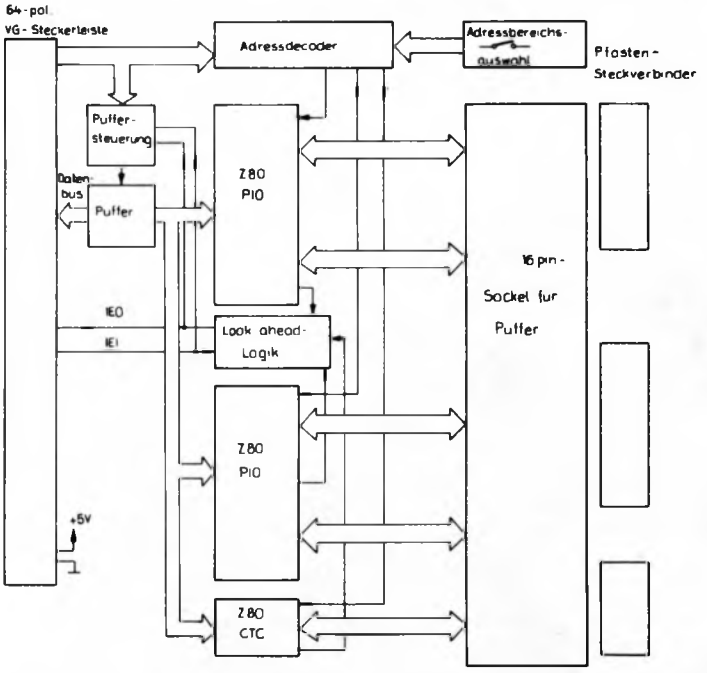
7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

7. Ein/Ausgabebaugruppe Z80-ECB/I

Die Baugruppe umfaßt folgende Funktionseinheiten:

- 4 Parallelschnittstellen (2 Stück Z80-PIO)
- 4 Zeitgeber/Ereigniszählkanäle (1 Stück Z80-CTC)
- 9 Unverdrahtete Fassungen für Standard-Interface-Bausteine
- Volle Buspufferung über Schmitt-Trigger-Puffer für höchste Betriebssicherheit
- Busseitig 64-poliger Stecker nach DIN 41612 (VG 95324)
- I/O-seitig 3M-WWP-Pfostensteckverbinder.

Bild 1 zeigt das zugehörige Blockschaltbild:



Die Baugruppe behandelt mit ihren 2 Bausteinen Z80-PIO und dem 1 Baustein Z80-CTC insgesamt vier 8bit-Ein/Ausgabekanäle, die jeweils über 2 zusätzliche Leitungen für Quittungsbetrieb verfügen, und vier Ereigniszähler/Zeitgeberkanäle. Einzelheiten über den Baustein Z80-PIO finden Sie im Z80-PIO-Technical Manual; der Baustein Z80-CTC ist im Z80-CTC-Technical Manual beschrieben.

Zur Realisierung anwenderspezifischer Interfaces wurden auf der Baugruppe neun 16-polige Fassungen untergebracht, deren Ausgänge über Wire-Wrap-Stifte entsprechend der im Anwendungssystem gewünschten Steckerbelegung mit den 3 Pfostensteckverbindern verdrahtet werden.

Für den Fall, daß komplexere Interface-Schaltungen oder Verbindungen zur Busseite der Platinen hergestellt werden sollen, können diese Pfostensteckverbinder als Verbindungen zu „Huckepack"-(="Sandwich")-Platinen verwendet werden. Sollen die Ausgänge der PIO- bzw. CTC-Bausteine ohne zwischengeschaltete Puffer benutzt werden, sind die Grenzwerte bzw. Charakteristika zu beachten:

Z80-PIO: I_{OL} = 1,8 mA bei V_{OL} = max. 0,4 V (Port A)
I_{OHD} = 1,5 . . . 3,8 mA bei V_{OH} = 1,5 V und Rext = 390 Ω
(Darlington-Ausgänge der Ports B)

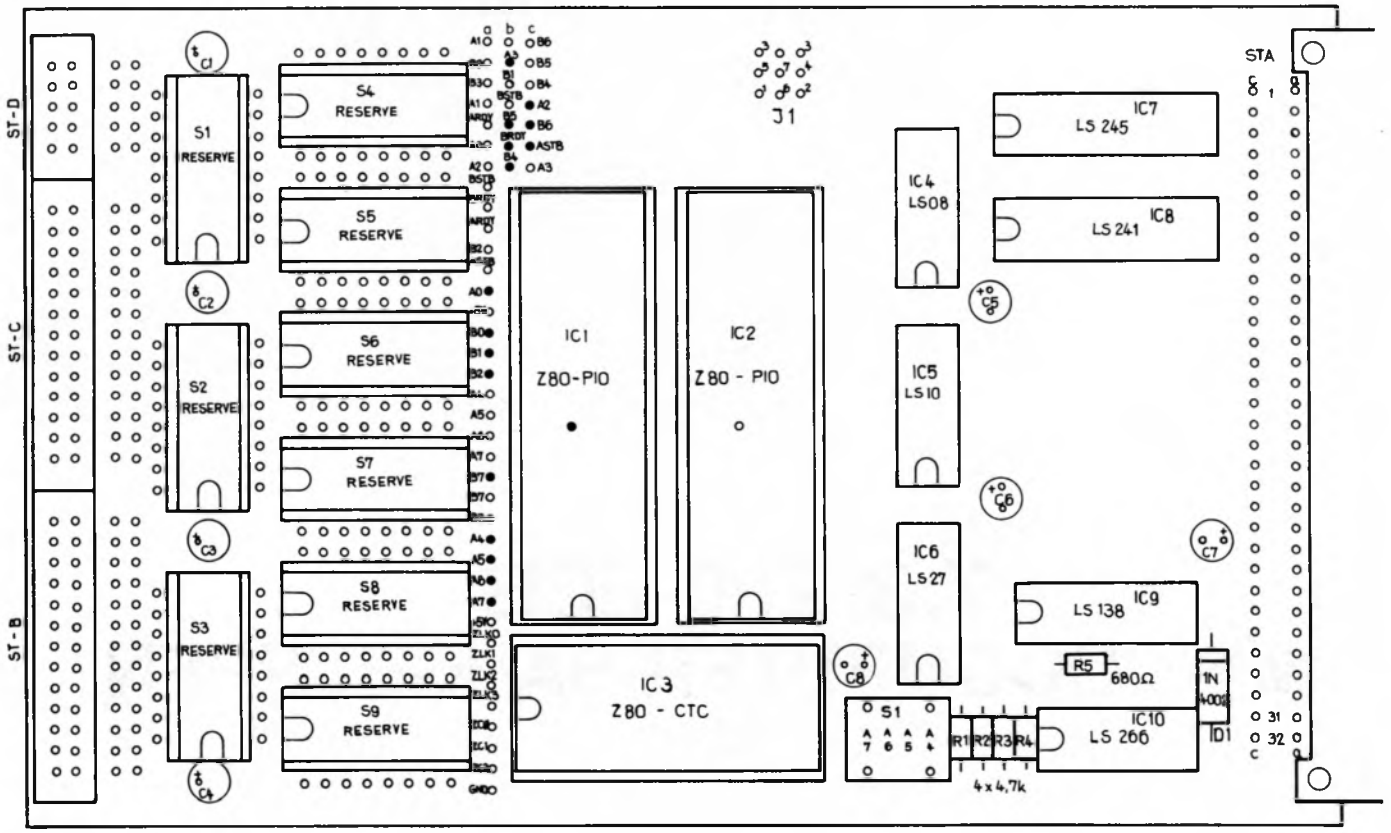
Z80-CTC I_{OL} = 1,8 mA bei V_{OL} = max. 0,4 V
I_{OHD} = 1,5 . . . 3,8 mA bei V_{OH} = max. 1,5 V und Rext = 390 Ω

Z80-ECB/I

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	220 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

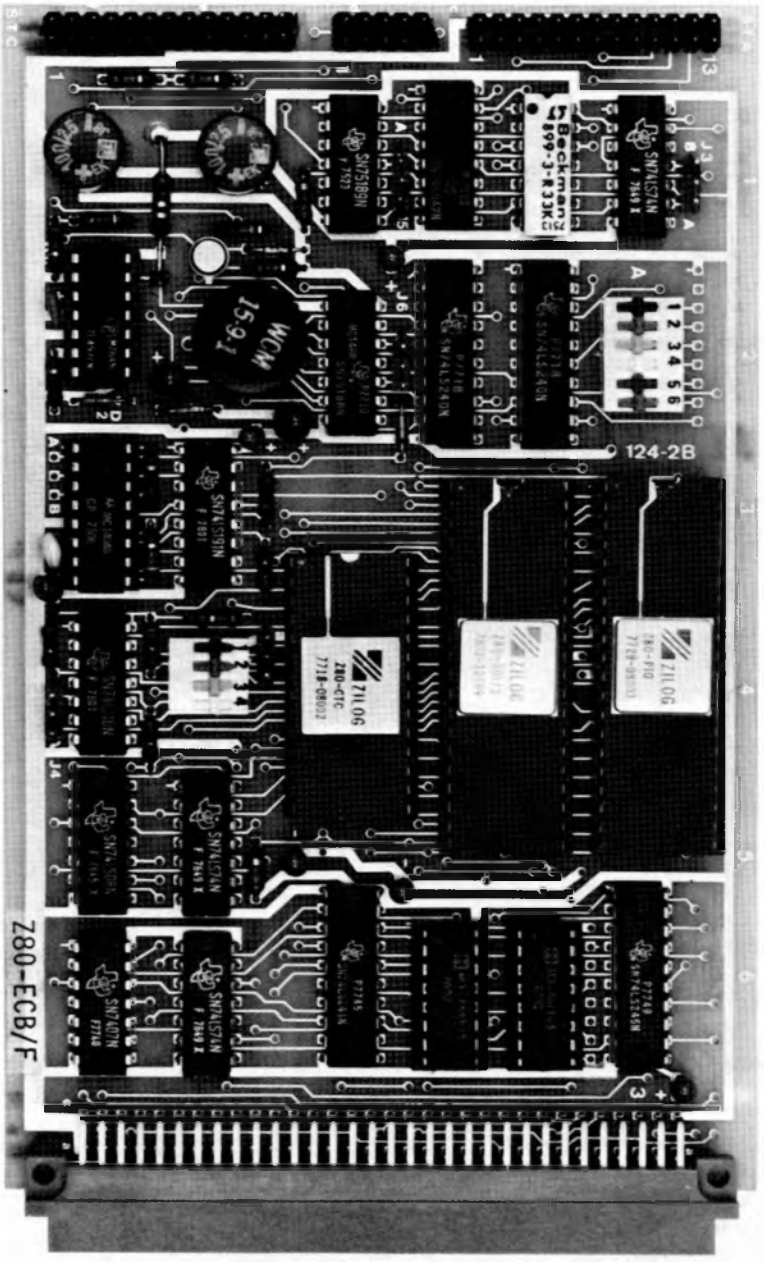
Schaltplan wird mit der Baugruppe mitgeliefert.



Z80-ECB/F

Ein/Ausgabe-Baugruppe für allgemeine Aufgaben und Floppy-Disk-Steuerung

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

8. Ein/Ausgabe-Baugruppe für allgemeine Aufgaben und Floppy-Disk-Steuerung Z80-ECB/F

8.1 Übersicht

Die Baugruppe ECB/F ist eine universelle Serien-Ein- bzw. Ausgabe-Einheit. Sie ist elektrisch und elektromechanisch kompatibel zu allen Baugruppen-Serien Z80A-ECB, Z80-ECB und Z80-KIT. Sie vereint die Leistungsfähigkeit von drei Peripheriebausteinen, nämlich des CTC, des PIO und des SIO. Zwei separate Voll-Duplex-Kanäle erlauben serielle Datenübertragung in nahezu allen bekannten asynchronen und synchronen Verfahren. Über den CTC und einen PLL-Frequenzsynthesizer sind alle gebräuchlichen Baudraten für den Asynchronkanal bzw. alle Übertragungsraten von 0–850 kbit/s für den Synchronkanal programmierbar.
Die Baugruppe ermöglicht den unmittelbaren Anschluß von bis zu acht hardsektorierten Floppy-Disk-Laufwerken verschiedener Größen und Speicherkapazitäten, sowie asynchron arbeitenden Datenendgeräten mit genormten RS 232 C- oder 20 mA Current-Loop-Schnittstellen. Durch den Einsatz des SIO ist die ECB/F in hervorragender Weise auch für Rechner-Rechnerkopplungen geeignet. International genormte Übertragungsprozeduren wie SDLC und HDLC werden vom SIO abgehandelt.
Zum Betrieb der Baugruppe genügt eine einzige Versorgungsspannung von +5 Volt. Die Betriebsspannungen für das RS 232 C-Interface werden durch Gleichspannungswandler auf der ECB/F erzeugt.
Eine leistungsfähige, in zwei Stufen implementierte Floppy-Disk-Betriebssoftware steht ebenfalls zur Verfügung. Zusammen mit der Zentralbaugruppe ECB/C 8 stellt die ECB/F somit das Kernstück eines Floppy-Disk basierenden Anwendersystems, oder eines Multi-Rechner-Systems dar; selbstverständlich ist der Einsatz dieser Baugruppe ebenso in allen Anwendungen besonders wirtschaftlich, in denen Serien-(2), Parallel-(2) und Zeitgeber/Zähler-(4) Interfaces erforderlich sind.

8.2 Schaltungsbeschreibung

Die Baugruppe enthält folgende Funktionseinheiten:

a) CPU-Interface
- Schmitt Trigger Pufferung sämtlicher Bussignale
- Adreßdekoder für CTC, PIO und SIO
- Look Ahead Logik für Interruptprioritätssteuerung (Daisy Chain)
- 64-poliger Busstecker nach DIN 41612

b) SIO-Synchronkanal (Floppy-Disk-Interface)
- Voll programmierbarer PLL-Frequenzsynthesizer zur Erzeugung der Übertragungsrate (2 CTC-Kanäle)
- Programmierbarer, rein digitaler Datenseparator mit 16 Zeitkonstanten
- Datenencoder zur Daten-/Clock-Gemischerzeugung
- Diskstatussignale über Z80-PIO
- Schmitt Trigger Pufferung aller Ein-/Ausgänge

c) SIO-Asynchronkanal
- Voll programmierbare Baudratenerzeugung von 50–38.4000 Baud (1 CTC-Kanal)
- RS 232 C und 20 mA Current-Loop-Interface
- Gleichspannungswandler für ± 12 Volt
- 6 Modem Signale im RS 232 C Pegel

d) Sonstiges
- 6 DIP-Schalter (über PIO) zur Programmierung von Systemvariablen
- Steckbare Jumper zur Umstellung des Synchronkanals an andere Peripherie
- I/O-seitig drei Standardstecker zum Anschluß der Peripheriegeräte

Bild 1 zeigt das Blockschaltbild der Baugruppen

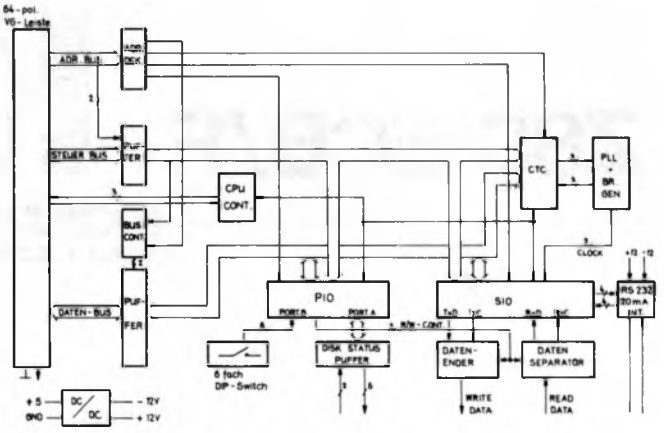


Bild 1: Blockschaltbild der ECB/F

Alle Leitungen sind gepuffert und stellen pro Eingang nicht mehr als eine LS TTL-Last (≙ 0,25 mA bei LOW) dar. Schmitt Trigger Bausteine erhöhen hierbei die Störsicherheit. Alle Ausgänge erlauben den Anschluß von kapazitiv hochbelasteten Bussystemen.
Die Leitungen $\overline{\text{WAIT}}$ und $\overline{\text{INT}}$ sind Open Collector-Ausgänge ohne Pull up Widerstände, da diese bereits auf der Zentralbaugruppe ECB/C 8 verhanden sind. Eine „Look Ahead Logik" (Baustein F5: 74LS08) sorgt für die schnelle Durchschaltung der Z80-Interruptprioritätssteuerung (Daisy Chain). Die ECB-F-interne Priorität ist: CTC, SIO, PIO.

8.3 Floppy Disk Interface

Die ECB/F erlaubt den unmittelbaren Anschluß von bis zu 8 Floppy-Disk-Laufwerken (mit hardsektorierten Disketten) in gemultiplexten Systemen entsprechend Bild 2.
Durch die hohe Flexibilität der Hardware sind sowohl Mini- als auch Standardlaufwerke mit einfacher Speicherdichte (single density) anschließbar.

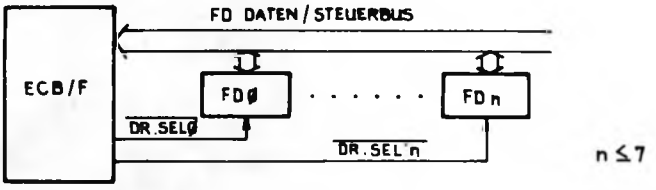


Bild 2: Blockschaltbild eines Systems mit mehreren Laufwerken

Zum Betrieb der unterschiedlichen Laufwerke ist lediglich eine geringfügige Modifikation in der Betriebssoftware nötig.

Anschaltung von Floppy-Disk-Laufwerken

Der Anschluß eines Laufwerkes erfolgt über die im Bild 7 dargestellten Leitungen.

	A1	←	INDEX/SECTOR	– 20	
	A10	←	TRACK 0	– 42	
	A9	–	WRITE GATE	→ 40	
	A7	–	STEP	→ 36	
ECB/F	A6	–	DIRECTION	→ 34	Laufwerk
	A2	–	DRIVE SELECT	→ 26	Shugart 800
	A8	–	WRITE DATA	→ 38	
	A12	←	READ DATA	– 46	
	A11	←	WRITE PROT.	– 44	
	A13–A26		GND		(alle ung. Pin's)

Bild 7: ECB/F-FD-Laufwerk-Verbindungen
(z.B. Laufwerk 800-2)

Signalbedeutungen:

INDEX/SECTOR	ist aktiv, wenn das Sektor- oder Indexloch einer Diskette abgetastet wird. Der Indexpuls tritt nur einmal pro Umdrehung auf und kennzeichnet Sektor 0.
TRACK 0	ist aktiv, wenn der Schreib/Lesekopf auf Spur 0 steht.
WRITE GATE	ermöglicht das Schreiben auf die Diskette und verhindert die Ansteuerung des Schrittmotors.
STEP	bewegt den Schreib/Lesekopf um eine Spur weiter.
DIRECTION	bestimmt, in welche Richtung ein STEP erfolgt.
DRIVE SELECT	aktiviert das angesteuerte Laufwerk (Laufwerkadresse)
WRITE DATA	Clock/Datengemisch zum Laufwerk
READ DATA	Clock/Datengemisch vom Laufwerk
MOTOR ON*	startet die Motoren
WRITE PROTECT	die Diskette ist mechanisch schreibgeschützt
READY*	ist 5 Umdrehungen nach Einlegen einer Diskette aktiv.

* Die Signale MOTOR ON und READY sind nicht bei allen Laufwerken vorhanden.

8.4 Anschluß von Terminals

Abhängig von der ausgewählten Schnittstelle (Tabelle 5) erfolgt der Anschluß eines Terminals (Datensichtgerät, Fernschreiber) über eine 3- bzw. 4-Drahtverbindung.

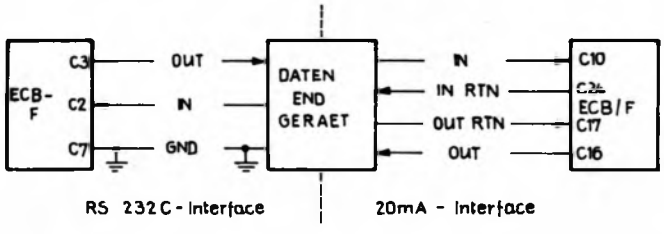


Bild 8: Anschluß von Terminals

8.5 Ein-Ausgabe-Steckerbelegung

Alle Ausgangssignale der ECB/F sind an drei Pfostensteckverbindern herausgeführt.

Stecker A (26polig)	A1	INDEX/SECTOR
	A2	DRIVE0
	A3	DRIVE 1
	A4	DRIVE2
	A5	MOTOR ON
	A6	DIRECTION
	A7	STEP
	A8	DISK WRITE DATA
	A9	WRITE GATE
	A10	TRACK0
	A11	WRITE PROTECT
	A12	DISK READ DATA
	A13—A26	GND
Stecker B (10polig)	B1	DRIVE3
	B2	DRIVE4
	B3	DRIVE5
	B4	DRIVE6
	B5	DRIVE7
	B6	EXT. CLOCK
	B7	SPARE OUTPUT
	B8	SPARE INPUT
Stecker C	C1	—
	C2	RS 232 DATA IN
	C3	RS 232 DATA OUT
	C4	DCD
	C5	CTS
	C6	RTS
	C7	COMMON GND
	C8	DTR
	C9	—
	C10	20 mA DATA IN
	C11—C15	—
	C16	20 mA DATA OUT
	C17	20 mA OUT RTN
	C18—C23	—
	C24	20 mA IN RTN
	C25—C26	—

8.6 Jumperstellungen

Die Baugruppe enthält insgesamt 5 steckbare Kurzschlußbügel (Jumper J1–J5), von denen J5 erwähnt wurde (Tabelle 5).

Die Jumper J2 bis J4 dienen dazu, andere synchrone Peripherie an Kanal A des Z80A-SIO anzuschließen. Durch Umstecken der Stecker J2, J3 und J4 auf Stellung B wird die Floppy Disk spezifisches Hardware (Datenencoder, Datenseparator) umgangen. Die SIO-Eingängen RxCA und RxDA führen durch Umstecken von J2 bzw. J3 unmittelbar die an den Steckerpins B6 bzw. A12 anliegenden Signale (EXT CLOCK RECEIVE DATA).

Wird J4 umgesteckt, so ist der Anschluß TxDA über den Puffer B2 mit Stecker A, Pin 8 (TRANSMIT DATA) verbunden. Mit Jumper J1 wird der WRDY-Ausgang des SIO-Kanals B mit dem entsprechenden Ausgang von Kanal A verbunden (beides Open Drain Anschlüsse) und über einen Open-Collector-Puffer (F6:7407) an die CPU-WAIT-Leitung angeschlossen. Damit läßt sich eine Synchronisation zwischen CPU und SIO erreichen (siehe SIO-Produktspezifikation).

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	850 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 150 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. III

Schaltplan wird mit der Baugruppe mitgeliefert.

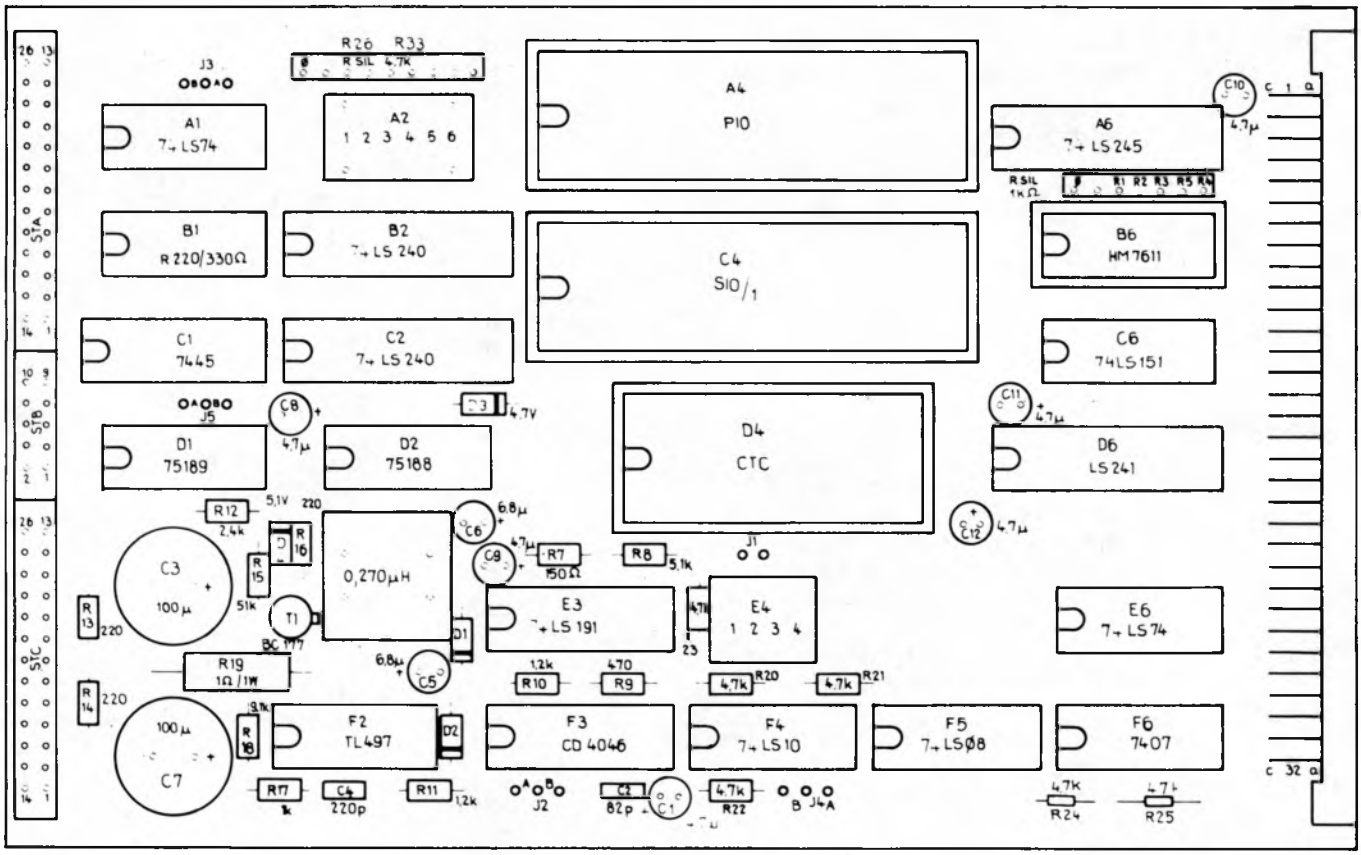
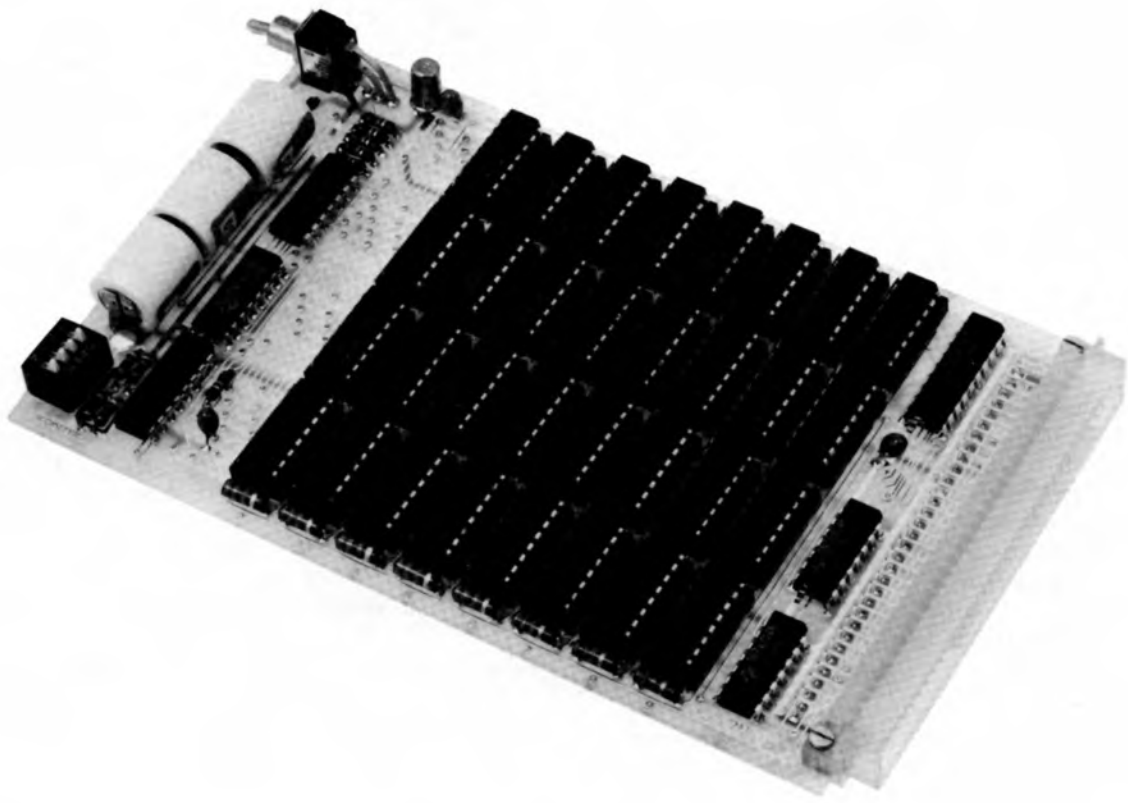


Bild 10

Z80-ECB/V

CMOS-Speicher-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

9. CMOS-Speichererweiterungsbaugruppe Z80-ECB/V

9.1 Schaltungsbeschreibung

Die Einfach-Europakarte ECB/V ist eine voll statische 4 kByte-CMOS-Speicherbaugruppe hoher Flexibilität und Packungsdichte. Wegen der extrem niedrigen Stromaufnahme ist sie besonders für folgende zwei Anwendungen geeignet:

- ☐ Große und schnelle Arbeitsspeicher: z.B. 64 kByte = 16 Karten mit 2,5 A-Netzteil in einem 19''-Einschub evtl. über Zentral-Akku zu puffern: 20 mA Ruhestrom.
  Diese Packungsdichte ist nur durch die vernachlässigbare Wärmeentwicklung der CMOS-RAMS möglich.
- ☐ Nicht-flüchtige Speicher.
  Programme und Daten können beim Austesten gegen Überschreiben geschützt werden (Speicher-Schutz-Schalter). Beim Abschalten des Systems oder bei Stromausfall bleiben die eingegebenen Daten erhalten.

Ferner kann man bei der System-Entwicklung auf die Verwendung von löschbaren PROM's verzichten,weil z.B. Programme bis zu ihrer endgültigen Festlegung auf der Karte erhalten bleiben (Speicherschutzschalter in Stellung "ROM"). Löschungen und Modifikationen können selektiv durch die normalen WRITE-Operationen vorgenommen werden, dazu muß lediglich der Speicherschutzschalter auf "RAM" geschaltet werden.
In gewissem Sinne bietet also diese CMOS-RAM-Karte eine preisgünstige, schnelle und platzsparende Alternative zu Kernspeicher-Systemen.
Eine oft willkommene Erleichterung für die Microcomputer-Entwickler bieten auch die auf der Karte angebrachten Basis-Adreß-Umschalter. Durch Betätigung eines 4 bit-DIL-Schalters kann so der Ansprechbereich der Karte z.B. im 64 kByte Adreßraum beliebig gewählt werden.
Die Blockschaltung zeigt Bild 1.
Die Pinbelegung ist identisch mit der Zentral-Baugruppe, so daß 1:1-Verdrahtung möglich ist.

9.2 Bedienungsanleitung

a) Speicherschutzschalter (Write-Protect).
Steht der Schalter S1 in Stellung Bestückungsseite, so ergibt sich die Betriebsart 'Quasi-ROM', steht dieser Schalter in Richtung Leiterbahnseite, kann auf den Speicher der Karte wie auf jeden Schreib/Lese-Speicher geschrieben und aus ihm gelesen werden.

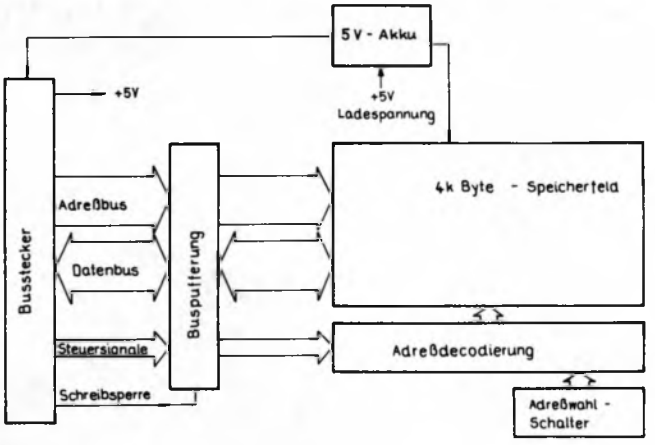
b) Adreßeinstellung
Die Adressen der ECB/V können mit Hilfe des DIP-Schalters S2 seitenweise (in 4 kByte Schritten) im Speicherbereich der Z80-CPU eingestellt werden.
Den Schaltern S2−1 . . . 4 sind die Adreßleitungen A12 . . . A15 zugeordnet. Daraus ergeben sich folgende Adreßzuweisungen.

c) DPR Signal.
Das DPR Signal ist low aktiv. Ist das Signal logisch "0", so kann auf die Karte weder geschrieben, noch von ihr gelesen werden.
Wird dieses Signal nicht benötigt, so es mit Jumper J2 auf high zu legen.

d) Jumper J1
Mit Jumper J1 wird die Pufferbatterie abgeklemmt, damit sich bei längerer Lagerung keine Tiefentladung der Akkus ergibt.

☐ **Zur Beachtung:**

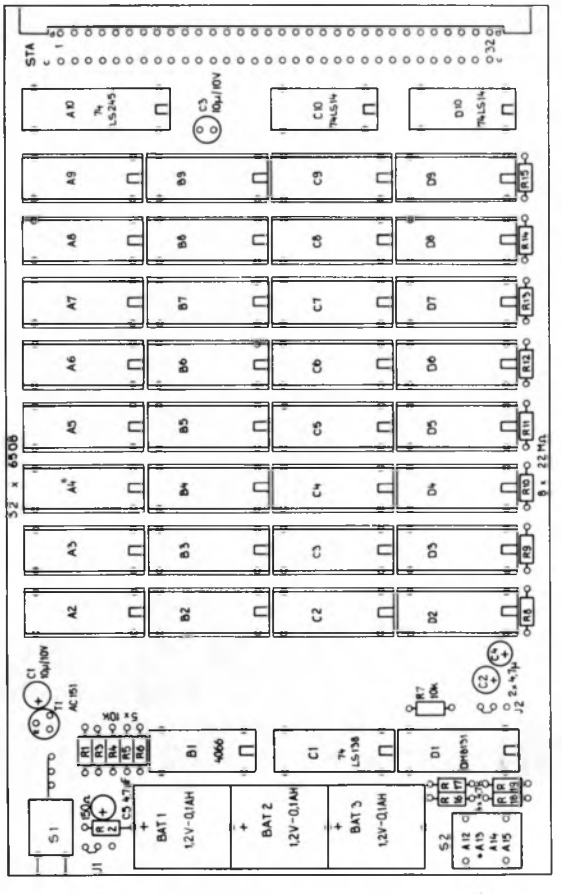
Die Baugruppe ECB/V wird mit geladenem Akkumulator und aufgetrennter Verbindung zum Akkumulator geliefert, um seine Entladung während der Lagerungszeit zu vermeiden.
Vom Anwender ist dafür Sorge zu tragen, daß der Akkumulator vor Einsatz der Baugruppe als Netzausfallsicherung aufgeladen ist und zu berücksichtigen, daß sich der Akkumulator bei ausgeschaltetem Gerät entlädt.
Die Information im Speicher der Baugruppe bleibt nach Stromausfall oder Abschaltung mindestens 7 Tage erhalten, vorausgesetzt, daß der Akkumulator zu diesem Zeitpunkt voll aufgeladen war.



Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	150 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 170 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

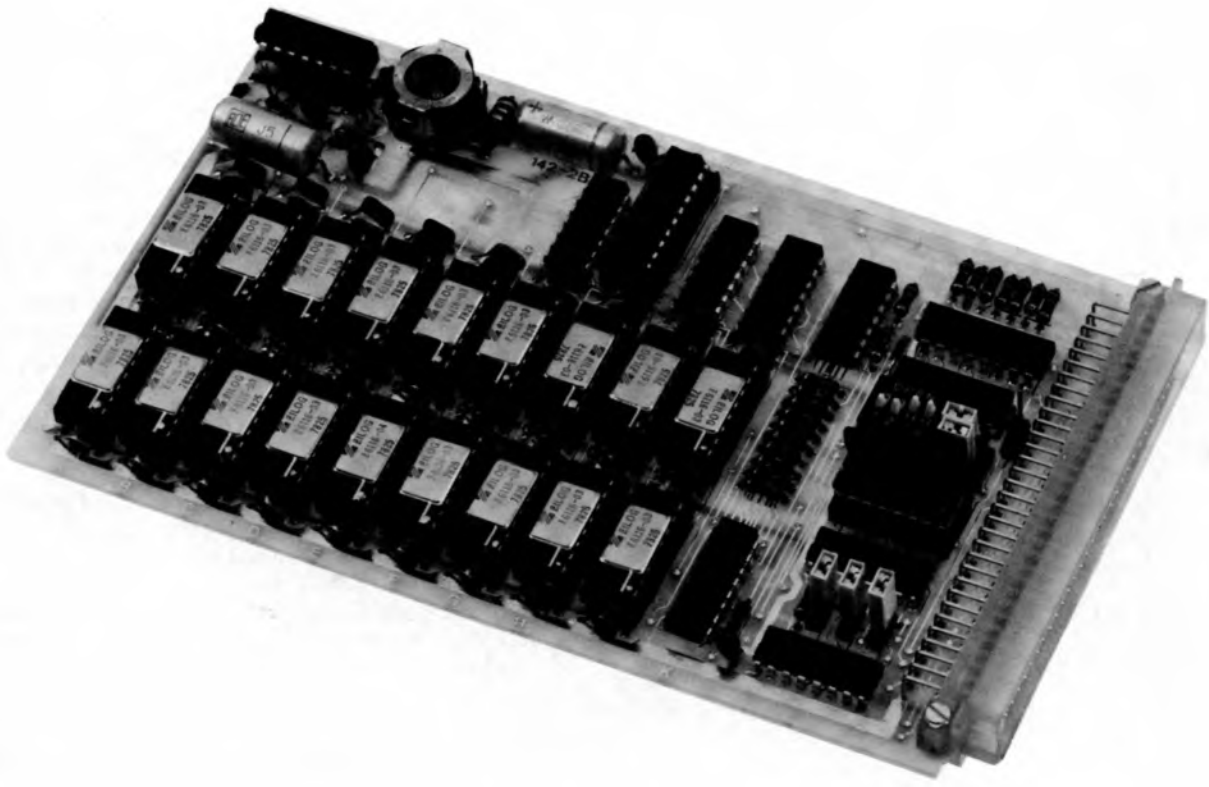
Schaltplan wird mit der Baugruppe mitgeliefert.



Z80-ECB/D

Dynamische Speicher-erweiterungsbaugruppen

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

10. Dynamische Speichererweiterungs-baugruppen der Serie Z80-ECB/D

Die Baugruppen Z80-ECB/D sind dynamische Speichererweiterungsbaugruppen zur Arbeit mit CPU's mit max. Arbeitsgeschwindigkeit von 2.5 MHz.
Die Erzeugung der Signale RAS und CAS, sowie der Versorgungsspannungen −5 V und +12 V erfolgen auf der Baugruppe, so daß sie mit einer einzige +5 V-Versorgung betrieben werden können.
Die Baugruppen mit der Kennzeichnung -P verfügen darüber hinaus über einen hardwaremäßigen Paritätsgenerator und -Prüfer und einer neunten Speicherzelle pro Wort.
Innerhalb der Baugruppenserie Z80-ECB/D sind Kapazitäten von 32 kByte, 16 und 8 kByte pro Platine verfügbar.

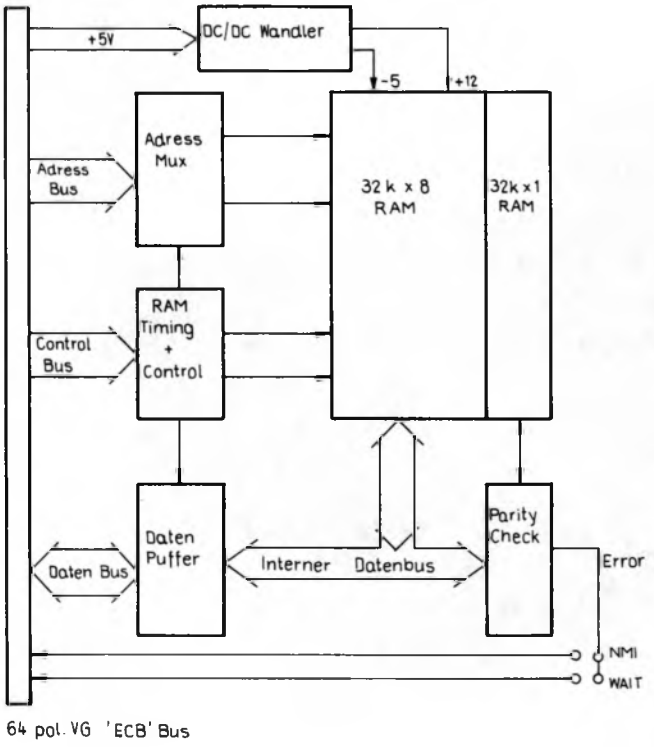


Bild 1: Blockschaltung

10.1 Speicherbaugruppe Z80-ECB/D32-P

Mit der Z80-ECB/D32-P lassen sich Systeme im Einfacheuropaformat realisieren, die entweder eine große Menge an Datenspeicher oder aber das Programm im Schreib/Lesespeicher haben müssen.
In solchen Systemen ist es wichtig, daß evtl. auftretende Speicherfehler erkannnt werden können. Hierfür verfügt die ECB/D32-P über eine Paritätsprüfeinrichtung. Diese legt bei jedem Schreibvorgang zusätzlich ein Parity-Prüfbit ab. Dieses Prüfbit wird bei jedem Lesevorgang geprüft, falls ein Paritätsfehler auftritt, kann ein nichtmaskierbarer Interrupt ausgeführt werden.
In der zugehörigen Interrupt-Service-Routine können dann entsprechende Alternativmaßnahmen oder sogar automatische Fehlerkorrektur softwaremäßig durchgeführt werden.

Die Z80-ECB/D32-P besteht aus 4 Funktionsblöcken:
- Timing und Steuerteil
- Speicherbank
- Parity-Erzeugung und -Prüfung
- Stromversorgung

10.1.1 Schaltungseinzelheiten

10.1.1.1 Steuerteil

Kernstück dieses Funktionsblocks ist das Adreß-PROM 7621 mit den Koordinaten L3. Als Eingänge sind in ihm miteinander verknüpft die Systemadressen A12 bis A15, die Schalterstellungen S1 bis S4 und die Signale REFRESH und MEMORY REQUEST.

Die Systemadressen A12 bis A15 werden durch ein vorgeschaltetes 4-bit-Latch zum Zeitpunkt von Memory-Request festgehalten. Der Schalter S4 unterscheidet zwischen den Bestückungsvarianten mit 4 kbit dyn. RAM's oder 16 kbit dyn. RAM's. Je nach Schalterstellung von S4 sind dann mit S1 bis S3 die verschiedenen Adreßbereiche, die mit dieser Karte erreicht werden können, einstellbar (Bild 1).

An seinen Ausgängen stellt das AdreßDecoder-PROM die Signale

Read-Adress-Strobe 0 und
Read-Adress-Strobe 1.

zur Verfügung. Diese dienen zum Einspeichern der Spaltenadresse in die jweils angewählte Speicherbank bzw. beim Refresh-Zyklus zum refreshen des gesamten Speicher. Das Signal Memory Select gibt den Datenpuffer frei. Das Signal zum Umschalten der Adreßmultiplexer bzw. zur Generation des Column-Adress-Strobe wird über eine Kette von Low-Power-TTL-Invertern erzeugt. Die Verzögerungskette hat mehrere Abgriffe um flexibel gegenüber unterschiedlichen Anforderungen im Timing zu sein. Die Standardeinstellung dieser Jumper passend für RAM-Chips verschiedener Hersteller ist in Bild 1 wiedergegeben.

10.1.1.2 Speicherblock

Bei der Auslegung des Speicherbereichs wurde auf die Besonderheiten der dynamischen Speicher Rücksicht genommen. So sind eine genügende Anzahl von induktionsarmen Blockkondensatoren für alle Versorgungsspannungen vorgesehen. Die kritischen Signalleitungen sind durch ausreichende Masseabdeckung gesichert. Schließlich wurde die Quellimpedanz der Adreß- und Kontroll-Signaltreiber durch Längswiderstände an die Leitungsimpedanz der gedruckten Schaltung angepaßt. Dadurch werden Überschwinger vermieden, die bei negativem Vorzeichen zu den berüchtigten Latch-up-Effekten der Adreßdecoder des dyn. RAM's führen könnten.

10.1.1.3 Paritäts-Erzeugung und Überwachung

Ein 8 bit Parity-Generator ist mit dem internen Datenbus an ECB/D32-P verbunden. Sein Ergebnis wird in das neunte bit bei jedem Schreibzyklus eingespeichert. Bei jedem Lesezyklus wird das Ergebnis des Parity-Generators und das eingespeicherte neunte bit exclusiv oder miteinander verknüpft. Ein D Flipflop speichert eine eventuelle Diskrepanz bis zum Auftreten eines weiteren Schreibzyklus. Wenn, wie durch Jumper 11 einstellbar, ein Parity-Fehler zu einem nicht-maskierbaren Interrupt führt, führt der dann auftretende Schreibzyklus (schreiben der Return-Adresse auf den Stack) zum Löschen des Fehler-Flipflops führen. Die Service-Routine für den non-maskable-Interrupt kann dann zum Veranlassen weiterer Maßnahmen per Software führen.

10.1.1.4 Stromversorgung

Da gegenwärtig erhältliche dynamische RAM's Versorgungsspannungen von +12 und ±5 Volt benötigen, ist auf der ECB/D32-P ein Gleichspannungswandler untergebracht, der mit dieser Karte nur eine einzige Versorgungsspannung von 5 Volt benötigt. Der Gleichspannungswandler wurde durch einen extern hinzugefügten Schalttransistor verstärkt. Dadurch steht genügend Leistung zur Verfügung, um eine sichere Versorgung auch bei ungünstigen Refresh-Zeiten (Prozessor im HALT) zu gewährleisten.

10.1.1.5 Anwendungshinweise

Die gesamte Logik zum Demultiplexen und zur Erzeugung des Speicherrefresh in Z80-CPU-gesteuerten Systemen ist mit auf der Baugruppe untergebracht. Der Refresh erfolgt abhängig vom laufenden Programm max. alle 0,7 msec unter der Kontrolle der Z80-CPU vollautomatisch ohne zusätzliche Software und ohne Verlust von Verarbeitungszeit, d.h. für den Anwender verhält sich die Baugruppe nach außen hin genauso wie jede statische Speichererweiterung.
Folgende Punkte sind jedoch zur Vermeidung von Informationsverlusten im dynamischen Speicher zu beachten:

- ☐ Bei Verwendung des Signals RESET ist dafür Sorge zu tragen, daß die Dauer dieses Signals unter 2 msec liegt, da die CPU kein Refresh-Signal erzeugt, solange ihr RESET-Eingang auf LOW liegt. Ein einfacher Schalter genügt an dieser Stelle also nicht, falls Informationsverluste im dynamischen RAM beim Systemrückstellvorgang unerwünscht ist; die Problematik kann z. B. mit einem Differenzierglied umgangen werden, das der RESET-Leitung der CPU vorgeschaltet wird.

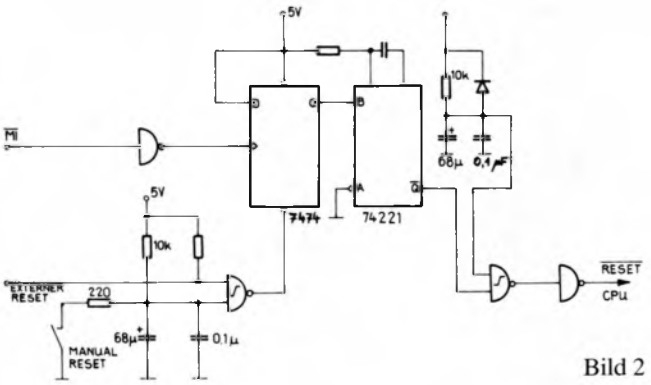


Bild 2

- ☐ Bei direktem Speicherzugriff (DMA) ist die Länge der DMA-Zugriffe so kurz zu wählen, daß immer noch alle 2 msec sämtliche Speicherstellen des dynamischen RAM's „aufgefrischt" werden.
  Die Z80-CPU schaltet bei jedem M1-Zyklus die Refresh Adresse um 1weiter. Der Zeitraum zwischen zwei Refreshvorgängen ist abhängig von der Art der Befehle, die in einem Programm vorkommen. Geht man von den längsten Befehlen (23 Taktzyklen) aus, so beinhalten diese zwei M1-Zyklen.
  Damit ergibt sich:

$$1\ \text{Refreshzyklus} = \frac{23}{2}\ \text{Taktzyklen} \cdot 128$$
$$\approx 0{,}7\ \text{ms bei einer Taktfrequenz von } 2{,}5\ \text{MHz}.$$

  Da die RAMS alle 2 ms einen vollständigen Refreshzyklus durchlaufen haben müssen, ergibt sich eine Restzeit von 1,3 msec für andere Aktivitäten.
- ☐ Das eben über DMA Gesagte ist sinngemäß auch auf die Benützung des WAIT-Eingangs der Z80-CPU anzuwenden, da auch WAIT-Zyklen den Refresh-Vorgang vorübergehend unterbinden (s. Z80-CPU-Technical Manual).
  Die Dauer eines periodisch anliegenden WAIT-Signals muß also unter 1,3 msec liegen.

10.2 Speicherbaugruppe Z80-ECB/D32

Die Baugruppe ist mit 16 kbit dynamischen Speicherbausteinen aufgebaut und 32 k × 8 bit-weise organisiert.
Ein Paritätsgenerator/Prüfer ist auf dieser Variante **nicht** vorgesehen.
Die übrigen technischen Daten entsprechen denen der Baugruppe Z80A-ECB/D32-P.

10.3 Speicherbaugruppe Z80-ECB/D16

Die Baugruppe ist mit 16 kbit dynamischen Speicherbausteinen aufgebaut und 16 k × 8 bit-weise organisiert.
Ein Paritätsgenerator/Prüfer ist auf dieser Variante **nicht** vorgesehen.
Die übrigen technischen Daten entsprechen denen der Baugruppe Z80-ECB/D32-P.

10.4 Speichererweiterungsbaugruppe Z80-ECB/D8

Die Baugruppe ist mit 4 kbit dynamischen Speicherbausteinen aufgebaut und 8 k × 8 bit-weise organisiert. Ein Paritätsgenerator/Prüfer ist in dieser Variante **nicht** vorgesehen. Die übrigen technischen Daten entsprechen denen der Baugruppe Z80-ECB/D32-P.

Z80-ECB/D32-P
Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	350 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 160 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80-ECB/D32
Technische Daten:

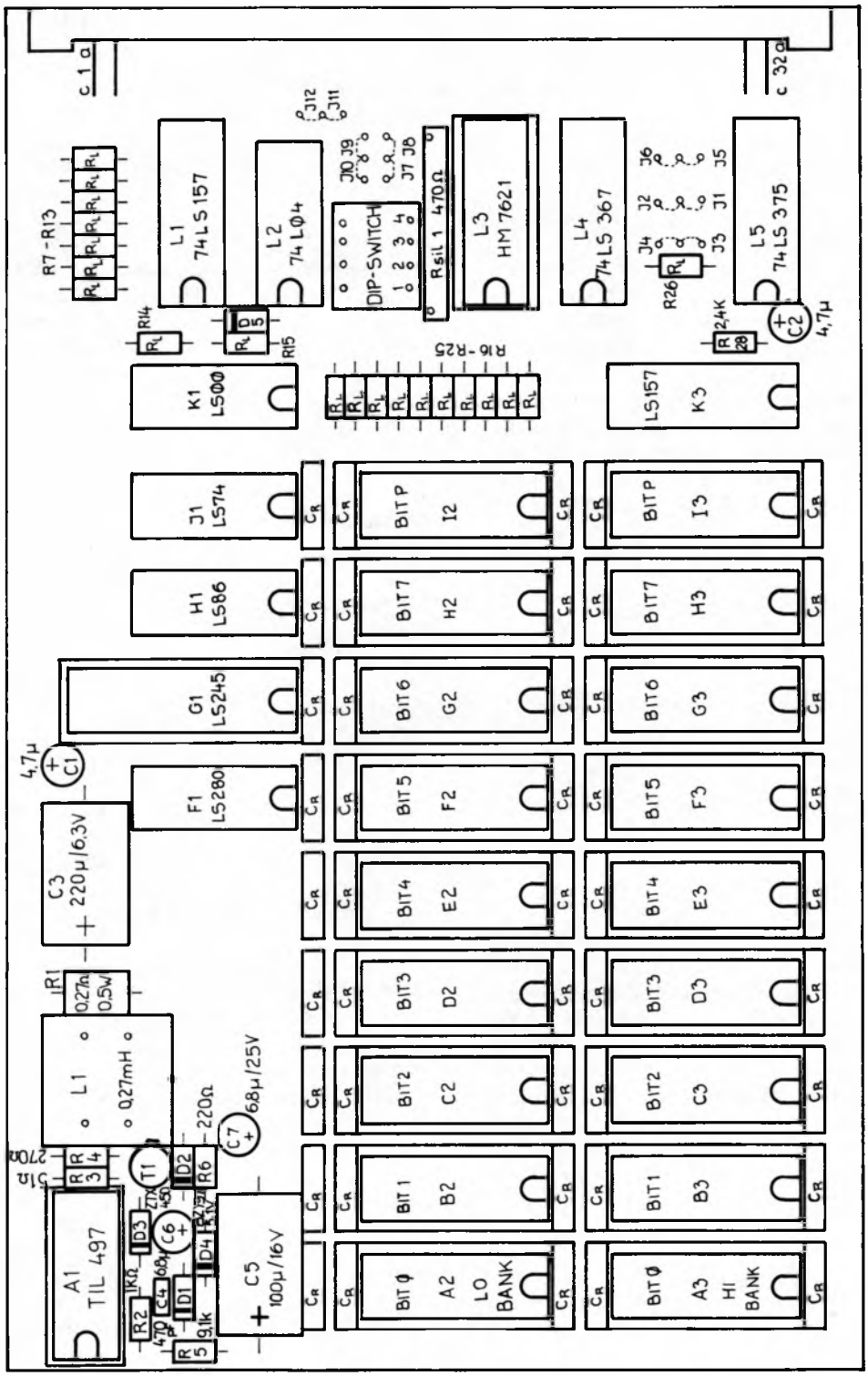
Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	330 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 160 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und s nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80-ECB/D16
Technische Daten:

Spannungsversorgung (Gleichspannung):	+5V, ±5%
Stromaufnahme (typisch):	280 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.



Z80-ECB/D8

Technische Daten:

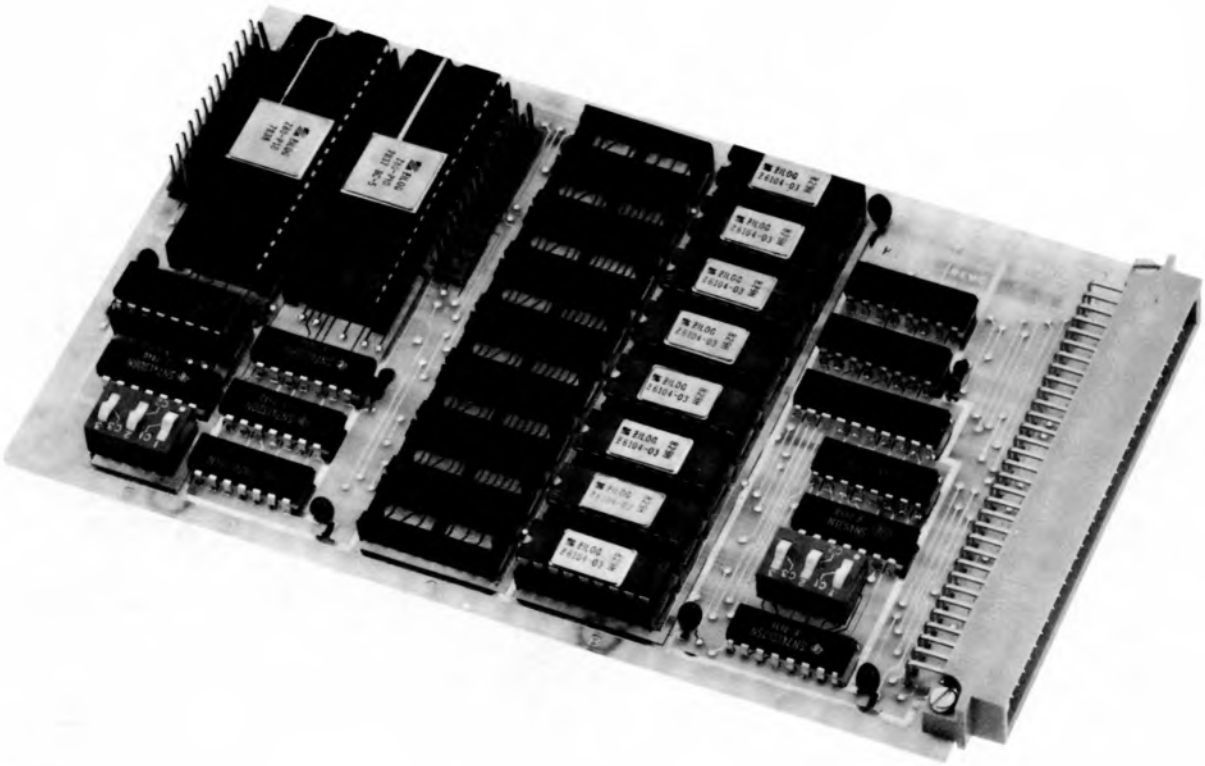
Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	420 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 160 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.

Z80-ECB/S

Speicher-Ein/Ausgabe-erweiterungsbaugruppe

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

11. Speicher-I/O-Erweiterung Z80-ECB/S

- ☐ 2×Z80-PIO ≙ 4 parallele 8bit I/O Ports mit Quittungsleitungen
- ☐ 4 kByte statisches RAM (austauschbar auf 8 kB)
- ☐ Einstellung der Speicheranfangsadresse über Kippschalter in Schritten von 8 k
- ☐ Einstellung der Portadressen über Kippschalter
- ☐ sämtliche Signale mit Busanschlußwert von einer TTL-LS-Last
- ☐ alle Adressen zwischengespeichert
- ☐ Daten über bidirektionalen Schmitt-Trigger-Tri-State-Buffer 74LS245 geführt
- ☐ Ausgangsseitig (= PIO Ein/Ausgänge) zwei 26-polige Stiftleisten

11.3 Pinbelegung

Stift-Nr.	26-pol. WWP-Steckerfeld
1	A_0
2	A_1
3	A_2
4	A_3
5	A_4
6	A_5
7	A_6
8	A_7
9	B_0
10	B_1
11	B_2
12	B_3
13	B_4
14	B_5
15	B_6
16	B_7
17	ASTB
18	BSTB
19	ARDY
20	BRDY
21	NC
22	NC
23	+5V
24	+5V
25	GND
26	GND

Bus-Stecker STA

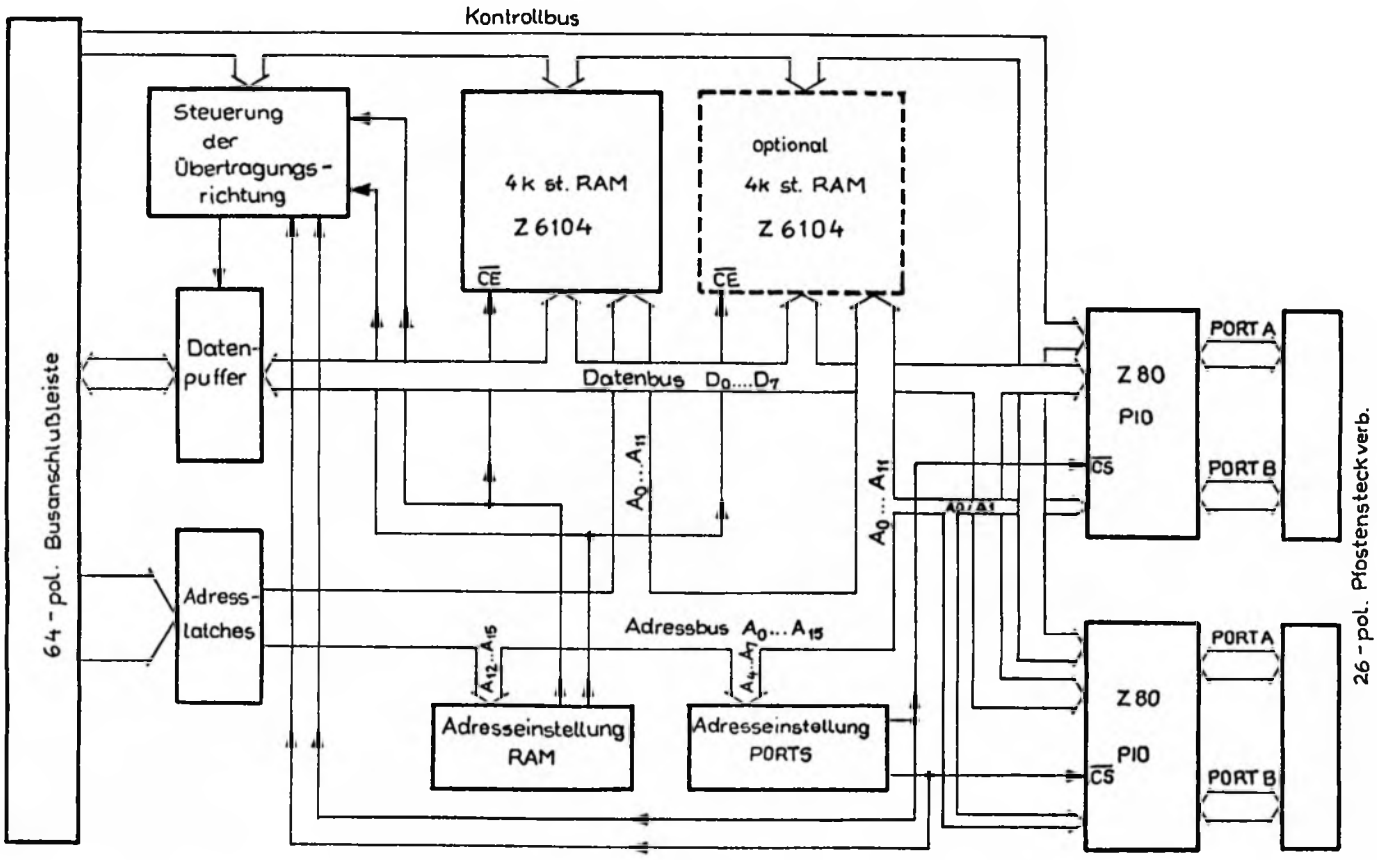
64-poliger Steckverbinder VG 95324
Es handelt sich hier um eine Belegung gemäß der ECB-BUS-NORM, wobei selbstverständlich nur die erforderlichen PIN's angeschlossen sind.

Benennung	Stecker PIN	Bezeichnung
A 0	5c	Adresse 0
A 1	7c	Adresse 1
A 2	6a	Adresse 2
A 3	6c	Adresse 3
A 4	7a	Adresse 4
A 5	8a	Adresse 5
A 6	9a	Adresse 6
A 7	9c	Adresse 7
A 8	8c	Adresse 8
A 9	30a	Adresse 9
A 10	18c	Adresse 10
A 11	17c	Adresse 11
A 12	27c	Adresse 12
A 13	29a	Adresse 13
A 14	18a	Adresse 14
A 15	28c	Adresse 15
D 0	2c	Daten 0
D 1	14c	Daten 1
D 2	4c	Daten 2
D 3	4a	Daten 3
D 4	5a	Daten 4
D 5	2a	Daten 5
D 6	3a	Daten 6
D 7	3c	Daten 7
M 1	20a	Maschinenzyklus 1
MRQ	30c	Memory Request
IORQ	27a	IN/OUT Request
RD	24c	Read
WR	22c	Write
INT	21c	Interrupt
IEI 1	11c	Int. enable in
IEO 1	16c	Int. enable out
0	29c	Clock 2,45 MHz
+5	1a, c	
GND	32 a, c	

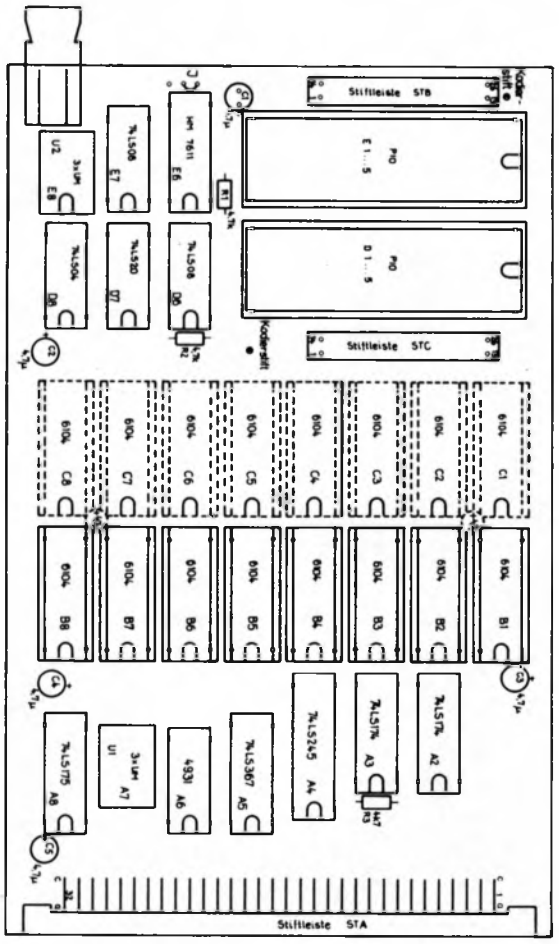
Technische Daten:

Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	300 mA (bei 4 kByte-RAM-Bestückung)
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160×100×15 mm
Gewicht:	ca. 140 g (bei 4 kByte-RAM-Bestückung)
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.



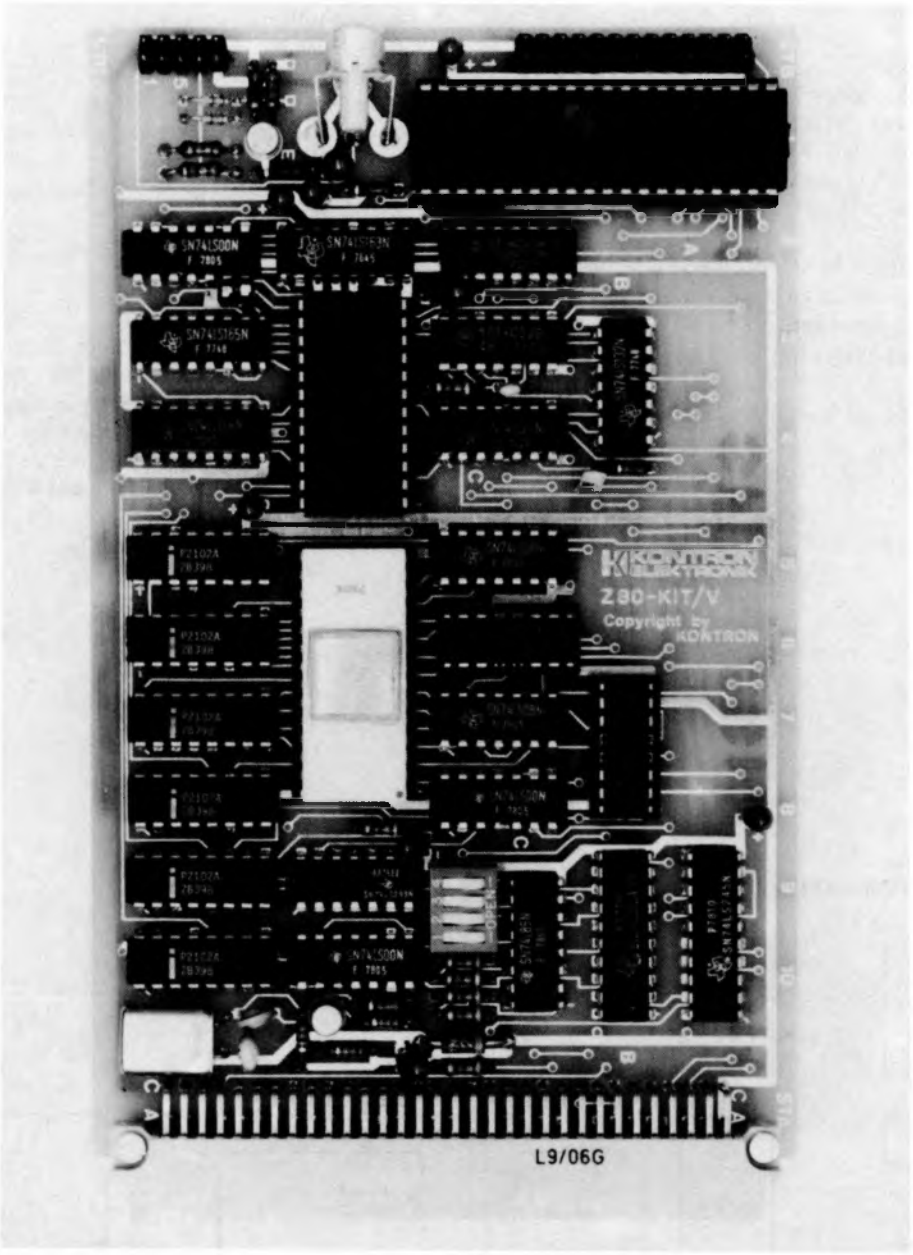
Blockschaltbild des ECB/S



Z80-KIT/VZ

Baugruppe für Bildschirmanschluß

KONTRON
ELEKTRONIK GMBH



7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

12. Baugruppe zum Bildschirmanschluß Z80-KIT/VZ

12.1 Schaltungsbeschreibung Z80-KIT/VZ

Die Baugruppe umfaßt folgende Funktionseinheiten:

Zwei Parallelschnittstellen (1 Stück Z80-PIO)
Decoder
Bus-Pufferung
Bildprozessor (Video-Controller)
1 K RAM-Bildspeicher
Zeichengenerator
Parallel-Serienwandler
Video-Mischer und Ausgangsverstärker
Anschlußseitig 26-pol. PIO-Stecker, 10-pol. Videostecker für getrennte Video- und Synchronisationssignale, Koaxstecker für Composite-Video-Out Signale.
Busseitig 64-pol. Stecker nach DIN 41612 (VG 95324)

Aus dem Blockschaltbild ist die Verknüpfung der einzelnen Funktionseinheiten zu erkennen.
Die Busleitungen sind gepuffert.

Die Adresse des Z80-PIO kann über DIL-Schalter S1 festgelegt werden.
Es stehen 16 Adressen zur Verfügung. Die Schalterstellung ist invertiert zu betrachten.

12.3 Ansteuerung der Z80-KIT/V bzw. Z80-KIT-VZ Baugruppe

Maximal werden von der Bildschirmansteuerung 120 Zeichen pro Sekunde verarbeitet (entspricht 8,3 ms/Zeichen). Das Löschen des Bildschirmes benötigt ca. 132 ms.

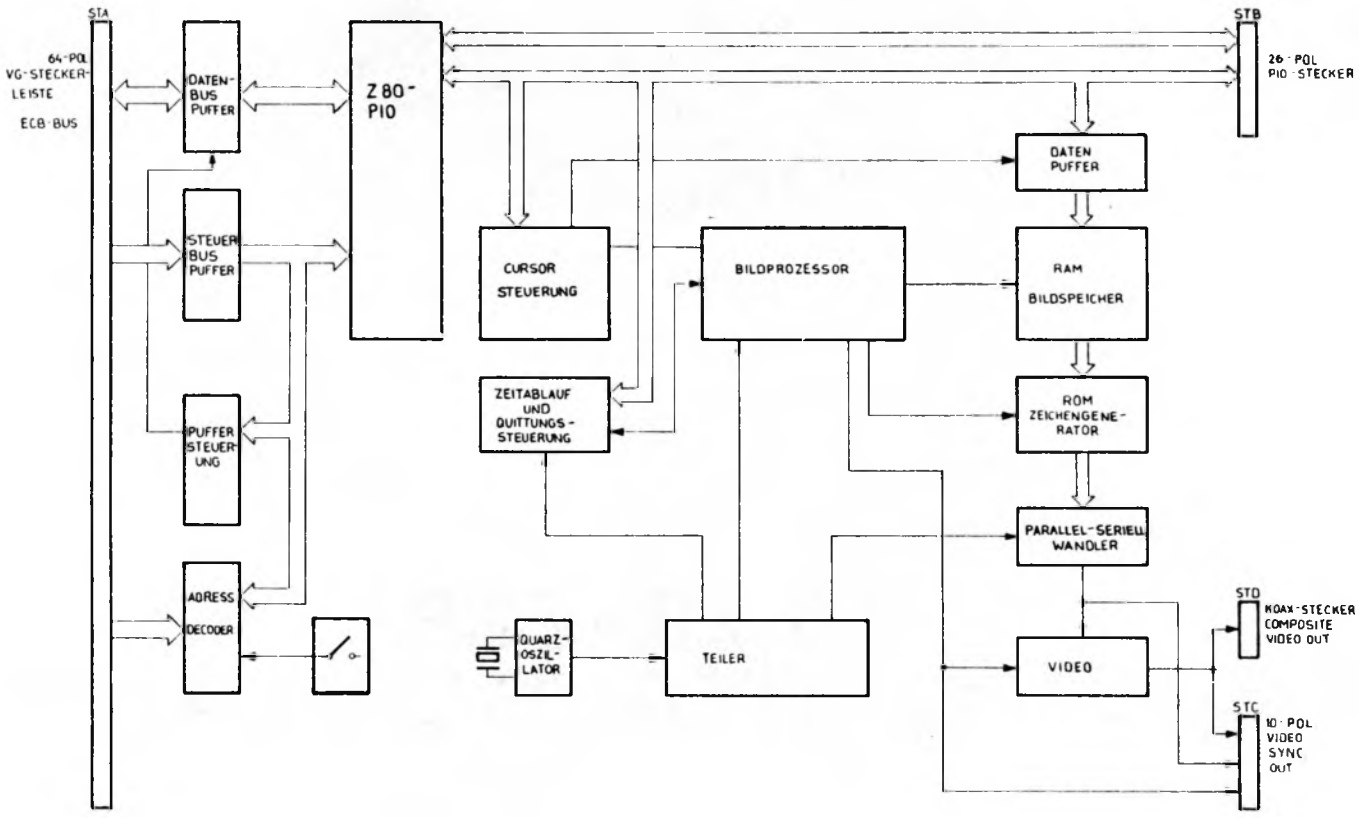
Eine entsprechende Zeitablaufsteuerung erfolgt automatisch über die Handshakeleitungen BRDY und BSTB des Z80-PIO in der Betriebsart Output.
(Mode 0, siehe Z80-PIO-Manual)

Arbeitsweise:
Bei Ausgabe eines Zeichens geht die BRDY-Leitung auf LOG1.
Nach Abarbeitung (8,3 Ms, bzw. 132 Ms) durch das Videoboard wird BSTB aktiv und löst einen Interrupt aus. Bei entsprechender softwaremäßiger Bearbeitung kann somit die maximale Datenausgabegeschwindigkeit der Z80-KIT/VZ Baugruppe genutzt werden.

12.4 Darstellungsweise

Die Zeichendarstellung erfolgt in 16 Zeilen zu 64 Zeichen. Der Zeichenumfang beträgt 64 Zeichen (ASCII-UPPER CASE), die Darstellung erfolgt in einer 5 × 7 Punktmatrix.
Die augenblickliche Schreibposition wird durch den Cursor angezeigt. Durch Umstecken des Jumpers J1 kann zwischen Darstellung schwarze Zeichen auf weißem Grund oder Invers gewählt werden.

12.2 Blockschaltbild Z80-KIT/VZ



12.5 Video-Schnittstelle

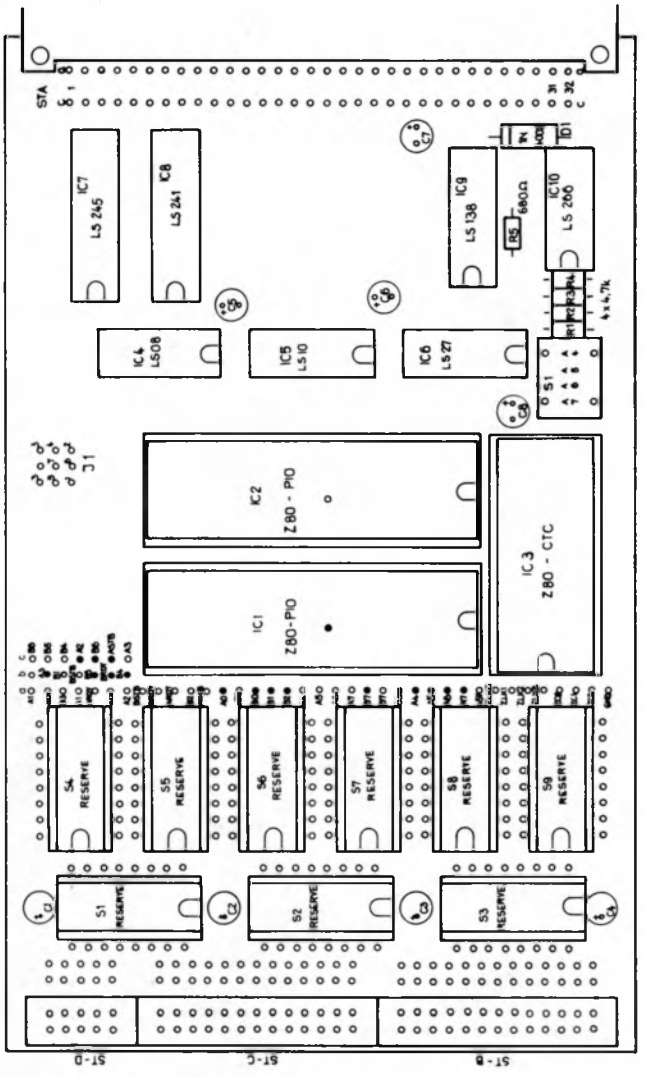
Die Z80-KIT/VZ-Baugruppe ist für den Anschluß eines Schwarz-weiß-Monitors oder eines handelsüblichen SW-Fernsehers mit Video-Eingang ausgelegt. Dadurch erreicht man wesentlich höhere Bildqualität und Störsicherheit gegenüber der Verwendung des Antenneneingangs des Fernsehers. Bei Verwendung eines Fernsehgerätes ohne Video-Eingang ist ein einfacher Eingriff im Gerät notwendig.

12.5.1 Monitoranschluß

Der Anschluß eines SW-Monitors ist wahlweise über das Composite-Video-Signal (STC oder STD) oder über Getrennte Video- und Composite-Sync.-Signale (STC) möglich.
Die Impedanz des Ausgangstreibers für das Videosignal (T1) beträgt 75 Ohm ± 5%.
Koaxleitung wird empfohlen, die Verwendung ist aber bei Leitungslängen unter 1,80 Meter nicht zwingend.
Das Kabel soll am Leitungsende mit 75 Ohm reflektionsfrei abgeschlossen werden. Die Widerstände R7 und R16 bestimmen die Amplitude des Composite-Video-Out-Signals. Eine Anpassung an den Sichtschirm kann durch ändern dieser Widerstände vorgenommen werden.

12.5.2 SW-Fernseher mit Videoanschluß

Der Anschluß ist über STD (Composite-Video-Out) vorgesehen. Hierzu gilt sinngemäß das vorher Gesagte.



Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	700 mA
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 20 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. I

Schaltplan wird mit der Baugruppe mitgeliefert.

13. Serien-I/O-Erweiterungsbaugruppe Z80-EML/IS (vorläufige Beschreibung)

13.1 Schaltung

Die Kommunikationspatine Z80-EML/IS ist für den Einsatz in Datenübertragungsanlagen vorgesehen. Sie besitzt als wichtigstes Element ZILOG's SIO-Baustein (SIO = Serial Input/Output) und eröffnet somit ein weites Feld von Anwendungen. Die Platine enthält zwei autonome, vollständig unabhängige Datenkanäle mit den folgenden Betriebsmöglichkeiten:

- Betrieb von Terminals
- Betrieb von Modems (Datenstrecken)
- Vollduplex- und Halbduplex-Betrieb
- Asychrone oder synchrone Schnittstellen.

Sämtliche Ein/Ausgabe-Adressen sind voll decodiert und können mittels DIL-Schaltern angewählt werden. Peripherieseitig sind standardisierte V 24-Schnittstellen mit allen Steuersignalen realisiert. Auf der Platine selbst lassen sich alle genormten Datenübertragungs-Raten von 50 bis 9600 Baud mittels DIL-Schaltern oder über die Software einstellen.
Die Z80-EML/IS wird über eine einzige 5 Volt-Speisespannung betrieben. Die erforderlichen ±12 Volt Spannungen für die V 24-Schnittstelle werden auf der Platine mit einem Gleichspannungswandler erzeugt. Die SIO-Kommunikationsplatine unterstützt hardwaremäßig alle für eine FTZ-Zulassung in der BRD erforderlichen Kontrollsignale.

Steuersignale

Die folgenden Schnittstellensignale werden über die SIO erzeugt:

- Request to send (Output)
- Clear to send (Input)
- Data Termianl ready (Output)
- Carrier detect (Input)

zusätzlich werden die folgenden Stuersignale über die PIO erzeugt:

- Modem ready (Input)
- Ring Indicator (Input): zur Bedienung von automatischen Antworteinrichtungen
- 1 weiteres, frei wählbares Signal (Output)
- 1 weiteres, frei wählbares Signal (Input).

Die Zuordnung **aller** Signale zu den Steckerstiften (Pin-Belegung) kann willkürlich über Wire-Wrap-Technik hergestellt werden. Damit läßt sich wahlweise eine Terminalschnittstelle, eine Modemschnittstelle oder eine benützerdefinierte Schnittstelle realisieren, die mit genormten Flachbandkabeln herausgeführt werden kann.
Werden die Signale „Modem ready" und „Ring Indicator" nicht verwendet und soll von der Möglichkeit, Datenübertragungsraten softwaremäßig einstellen zu können kein Gebrauch gemacht werden, ist die PIO frei für andere Anwenderzwecke.

13.2 Takt

Für die Taktsignale (RX clock, TX clock), die für **beide** Kanäle empfänger- und sender-seitig unabhängig sind, bestehen zwei Möglichkeiten:

1. Erzeugung der Taktsignale mittels des Quarz-Oszillators auf der Platte selbst. (Takt-Raten genormt von 50 bis 9600 Baud (× 16) einstellbar). Die Einstellung kann wahlweise über die DIL-Schalter (feste Einstellung) oder über die Ausgänge der PIO (softwaremäßige Einstellung, während des Betriebes durch Programmbefehle veränderbar) eingestellt werden. Für jeden der 2 Kanäle der SIO kann eine eigene Datenübertragungs-Rate eingestellt werden; außerdem ist „Split-Speed" Modus möglich.
2. Zuführung der Takte (Sende- und Empfangsschritt-Takt) von außen über die V 24-Schnittstelle.

13.3 Anwendung

Durch die Möglichkeit, die Datenübertragungsschnittstelle SIO anwenderprogrammseitig zu programmieren, sind folgende Einsatzmöglichkeiten von besonderem Interesse:

Kanal 1: Bedienung einer synchronen Vollduplex-Leitung mit externen Sende- und Empfangsschrittaktsignalen. Setzen und Überwachen aller Kontrollsignale der Schnittstelle durch Software.

Kanal 2: Bedienung eines asynchronen Terminals im Halb-Duplex-Betrieb. Softwaremäßige Überwachung der Terminal-Kontrollsignale. Umschaltmöglichkeit der Datenübertragungs-Rate durch Programmbefehle.

7.3. Z80-ECB-MIKROCOMPUTER-BAUGRUPPEN

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	600 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 140 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard

Schaltplan wird mit der Baugruppe mitgeliefert.

2. SUBPROZESSOR EML/SP (vorläufige Beschreibung)

2.1 Übersicht

Das Subprozessor-Modul EML/SP ist als intelligente periphere Einheit für Z80-ECB-Systeme konzipiert worden. Es enthält auf einer Einfach-Europakarte ein autonomes Zilog-Z80-Mikrocomputersystem mit 4 MHz Betriebsfrequenz. Das Subprozessor-Modul ist direkt über den Datenbus über zwei gegeneinandergeschaltete PIO's) gekoppelt.
Das Subprozessor-Modul arbeitet in einem Z80-System als Untersystem, d.h. es führt gewisse rechenintensive Aufgaben hauptsystem-unabhängig aus und zeigt die Beendigung der Verarbeitung mittels eines Interrupts an. Da das Subprozessor-Modul nach einem eigenen Programm unabhängig vom Hauptsystem arbeitet, belastet es während seiner Verarbeitungszeit das Hauptsystem nicht. In einem Hauptsystem lassen sich bei Bedarf mehrere Subprozessor-Module integrieren.

2.2 Anwendungen

Bei den meisten Anwendungen sind die Datenübertragungszeiten zwischen Hauptsystem und Subprozessor-Moduln kurz im Vergleich zu den eigentlichen Verarbeitungszeiten. Typische Beispiele für den zweckmäßigen Einsatz dieses Subprozessor-Modules sind:

- ☐ Realisierung von komplexen Chiffrieralgorithmen in Echtzeit für Datenübertragungs- oder Datenspeicherungssystemen (Datenverschlüsselung)
- ☐ Durchführung von fehlerkorrigierenden Decodieroperationen (automatische Fehlerkorrektur) in Übertragungs- oder Speichersystemen
- ☐ Korrelation (Auto- und Kreuzkorrelationen) von digitalen Signalen oder Bitsequenzen in Signalverarbeitungssystemen (Synchronisation, Radartechnik)
- ☐ Berechnung komplexer mathematischer oder kombinatorischer Funktionswerte.

Das Subprozessor-Modul wird mit konventionellen Z80-Programmiertechniken programmiert. Nach sorgfältiger Ausprüfung des Programmes wird es in die PROM's des Subprozessor-Modules eingebrannt. Durch den Einsatz des Subprozessor-Modules können auf relativ einfache Weise komplexe Hochleistungs-Gesamtsysteme realisiert werden, da die Aufgaben jedes Subprozessor-Modules leicht zu überschauen und im Griff zu halten sind.

2.3 Datenaustausch zwischen Haupt- und Subprozessor

Das Interface des Subprozessor-Moduls gegenüber einem Z80-Hauptsystem besteht aus einer PIO (Zilog Parallel Input/Output-Baustein), d.h.: das Subprozessor-Modul wird programmtechnisch vom Hauptsystem wie eine PIO (vgl. Blockschaltbild) gesehen. Der Datenaustausch findet mittels Ein/Ausgabe-Befehlen auf die PIO-Register statt. Auf diese Art und Weise können Daten **byteweise** vom Hauptsystem ins Subprozessor-Modul und umgekehrt übermittelt werden. Zusätzlich kann die volle Interrupt-Möglichkeit des PIO-Bausteines ausgenützt werden. Der eigentliche Datenaustausch-Mechanismus kann vom Benützer selbst definiert (und programmiert) werden. Ein typischer Ablauf für Verarbeitungsblöcke von 8 Bytes Länge kann zum Beispiel die folgende Form haben:

1. Initialisierung des Subprozessor-Modules:
   durch RESET beim Einschalten der Betriebsspannnung – das Subprozessor-Modul ist bereit für die Annahme eines Befehles vom Hauptsystem.
2. Initialisierung der Hauptsystem-seitigen PIO:
   durch das Initialisierungsprogramm im Hauptsystem wird die Hauptsystem-seitige PIO auf dem Subprozessor-Modul initialisiert (d.h. mit Mode, Werten und Interrupt-Vektor geladen).
3. Übergabe von Verarbeitungsdaten an das Subprozessor-Modul durch 9 aufeinanderfolgende OUT-Befehle des Hauptsystemes auf die Adresse der PIO, Kanal A:
   a) OUT: Befehlwort (8 Bit = Wahl der Funktion des Subprozessor-Moduls)
   b) 8 × OUT: Datentransfer (jeweils 1 Byte).

Nach der Annahme des 8. Datenbytes beginnt das Subprozessor-Modul die selbständige Verarbeitung gemäß seinem im eigenen ROM gespeicherten Programm. Während dieser Zeit wird das Hauptsystem **nicht belastet** und ist frei für andere Aufgaben. Ist die selbständige Verarbeitung des Subprozessor-Modules beendigt, erzeugt dieses über die Hauptsystem-seitige PIO einen Interrupt. Sobald der Interrupt vom Hauptsystem akzeptiert wird, können die **verarbeiteten** Daten zum Beispiel nach folgendem Mechanismus übernommen werden:

4. Übernahme von verarbeitenden Daten vom Subprozessor-Modul durch 9 aufeinanderfolgende IN-Befehle des Hauptsystemes auf die Adresse der PIO, Kanal B:
   a) IN: Status (evtl. aufgetretene Verarbeitungsfehler, etc.)
   b) 8 × IN: Datentransfer (jeweils 1 Byte).

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

2.4 Programmierung des Subprozessor-Modules

Die Erstellung von Verarbeitungsprogrammen für das Subprozessor-Modul kann mit den normalen Mitteln der Z80-Programmierung geschehen. Subprozessor-Modul-Programme können bis zu ihrer Reife auf einem gewöhnlichen Z80-System erstellt und ausgetestet werden.
Die in einem Subprozessor-Modul vorhandene Software läßt sich in 3 getrennte Teile zerlegen:

1. Initialisierungsteil:
   Durchlaufen bei Einschalten der Betriebsspannung des Hauptsystemes (Power-On-Schaltung der CPU-Platine) oder bei Übergabe eines speziellen Befehles vom Hauptsystem.
2. Kommunikationsprogramm zwischen Hauptsystem und Subprozessor-Modul:
   Übernahme von Befehlen und Daten über die PIO vom Hauptsystem und Ablage in reserviertem RAM-Bereich des Subprozessor-Modules. Übergabe von Zustand und Daten an das Hauptsystem aus einem reservierten RAM-Bereich im Subprozessor-Modul.
3. Verarbeitung:
   vom Anwender geschriebenes, spezifisches Verarbeitungsprogramm. Bezieht seine Daten und Parameter aus dem reservierten RAM-Bereich im Subprozessor-Modul und legt seine Meldungen und verarbeiteten Daten wieder in einem reservierten RAM-Bereich ab (von wo aus sie durch das Kommunikationsprogramm an das Hauptsystem übergeben werden).

2.5 Lieferumfang:

Z80A-CPU mit 4 MHz Quarzoszillator
4096 Bytes PROM (bipolar)
1024 Bytes statisches RAM
Parallelschnittstelle zum Hauptsystem-Bus (8 Bit Input, 8 Bit Output mit gegenseitiger Interrupt-Möglichkeit).
Voll decodierte Subprozessor-Adresse (I/O) mit DIL-Schaltern einstellbar.

Technische Daten:

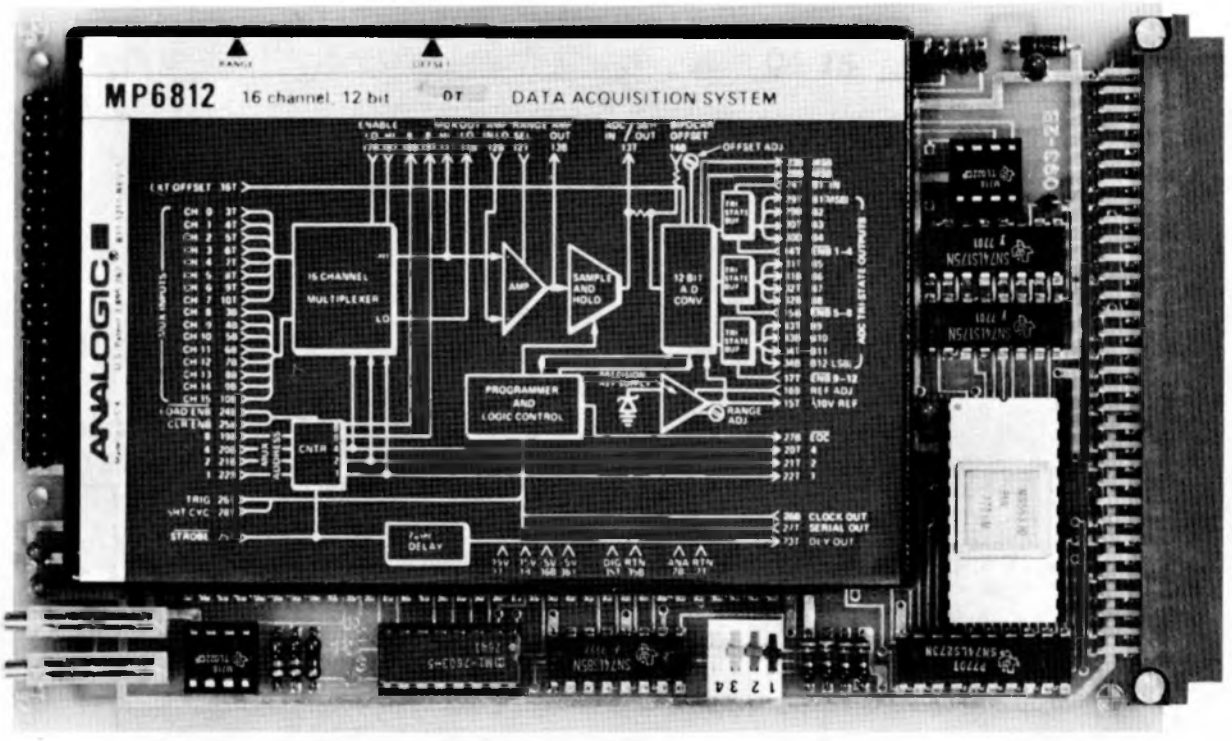
Spannungsversorgung (Gleichspannung):	+5 V, ±5%
Stromaufnahme (typisch):	1200 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 110 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und C nach ECB-Standard

Schaltplan wird mit der Baugruppe mitgeliefert.

AN-μP80-E16

Analog-Ein/Ausgabe-Baugruppe

KONTRON
ELEKTRONIK GMBH



7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

3.1 Übersicht

Die Baugruppe AN-μP80/E16 ermöglicht sowohl Erfassung, als auch Erzeugung von analogen Signalen in einem Mikrocomputersystem. Neben 16 A/D-Wandlerkanälen zur Verarbeitung analoger Signale steht auch ein D/A-Kanal zur Erzeugung analoger Signale zur Verfügung. Die Auflösung beträgt in jedem Fall 12 bit. Ein 10-fach DIP-Schalter erlaubt das bequeme Einstellen von Meßbereich, Betriebsart usw. Die Baugruppe enthält folgende Funktionseinheiten:

- 1 12 bit A/D-Wandler mit 16 Kanälen (MP 6812)
- 1 12 bit D/A-Wandlerkanal (MN 563)
- Adreßdekordereinheit zur Selektion von ADC bzw. DAC
- Schalter zur Adreßbereichsfestlegung der gesamten KONTRON AN-μP80/E16
- Busseitig 64-poliger Stecker nach DIN 41612 (VG 95324)
- I/O-seitig 40-poliger Normstecker.

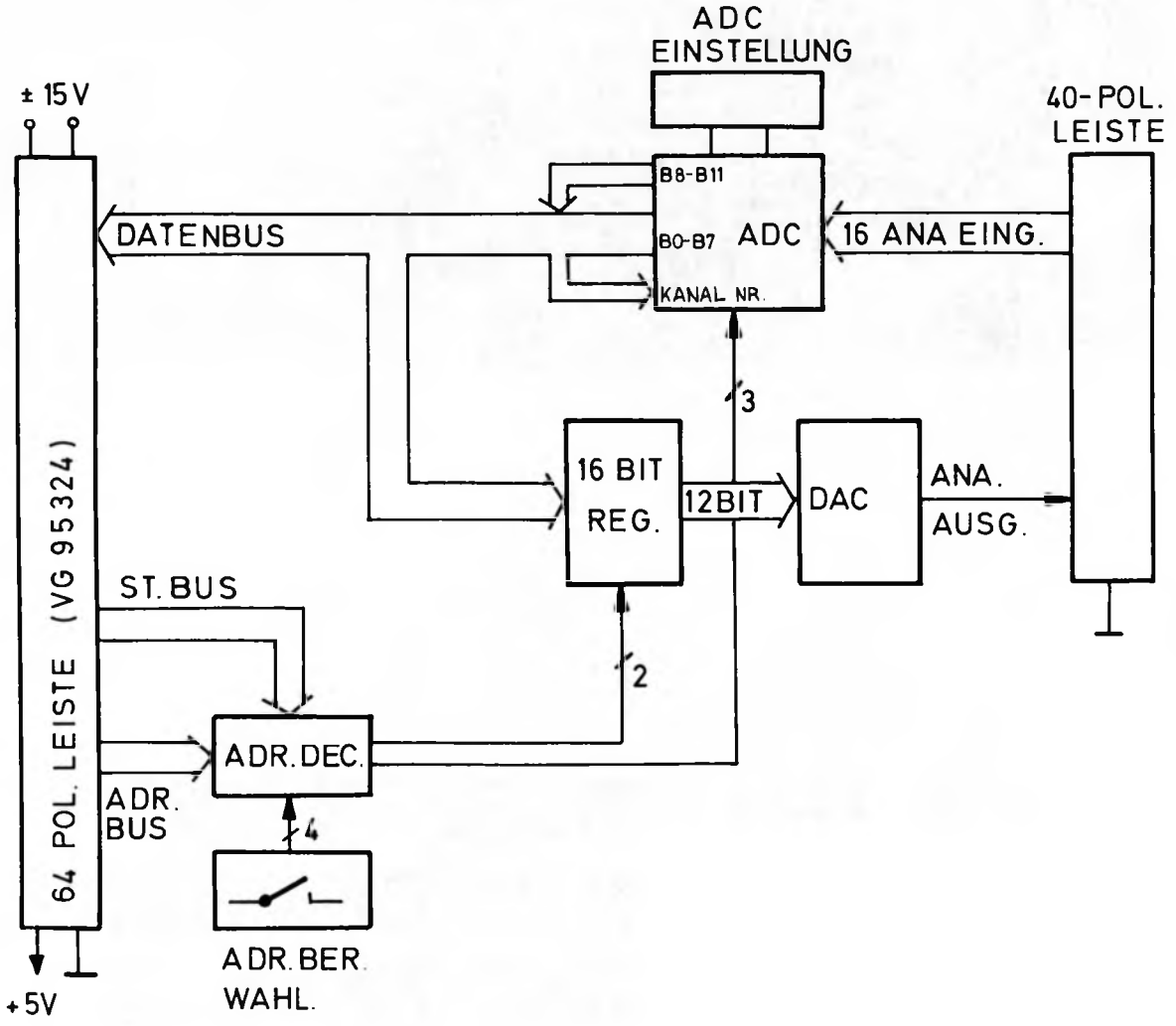
Die Baugruppe wurde so ausgelegt, daß sie zu Z80-ECB und Z80A-ECB pinkompatibel ist.

Bild 1 zeigt das entsprechende Blockschaltbild.

3.2 Schaltungsbeschreibung

3.2.1 Adressierung

Innerhalb eines Mikrocomputersystems wird die Baugruppe als gewöhnliche Ein-/Ausgabeeinheit betrachtet. Ihre Adresse ist eindeutig dekodiert und wird aus dem Vergleich des Adreßbits A2–A7 mit dem eingestellten Wert eines 6fach DIP-Schalters gewonnen. Damit lassen sich 64 verschiedene Baugruppenadressen (Card Select Adressen) erzeugen, wobei der Adreßvergleich bei der Schalterstellung „OPEN“ auf log. 1 erfolgt.



3.2.2 A/D-Wandler

Der Analog-Digitalwandler Teil der AN-μP80/E16 ist mit dem 16 Kanal 12 bit Wandlermodul MP 6812 von Analogic realisiert. Dieses Modul bietet vielfältige Möglichkeiten zur Meßbereichseinstellung und zur Festlegung der Eingangskonfiguration. Sämtliche Einstellmöglichkeiten sind mit einem 10-poligen DIP-Schalter möglich, entsprechend der folgenden Tabellen.

a) Wahl der Eingangskonfiguration.

Drei Eingangs-Pin-Konfigurationen sind möglich:

- ☐ 16 Eintakteingänge (Betriebsart A)
- ☐ 8 Echte Differenzeingänge (Betriebsart B)
- ☐ 16 Pseudodifferenzeingänge (Betriebsart C).

Tabelle 1 gibt an, welche Schalterstellung für welche Betriebsart notwendig ist. Die Analoganschlüsse sind ebenfalls aufgeführt.

Ebenfalls per Schalter können vier verschiedene Eingangsspannungsbereiche eingestellt werden. In jedem Fall steht unabhängig vom eingestellten Wert die volle 12 bit Auflösung im Zweierkompliment zur Verfügung. Bei Auslieferung ist Bereich 3 und Betriebsart A (— 10 bis + 10 V) eingestellt.

3.2.3 Kanal und Moduladressierung

Dem Wandlermodul sind drei Ein-/Ausgabeadressen zugeordnet. Diese Adressen erzeugt Baustein JC2 (ein 3 zu 8 Dekoder) aus den Adreßleitungen A0, A1 sowie dem Signal $\overline{WR}$ (WRITE). Die Freigabe des Bausteins (74LS138) erfolgt durch das CPU-Signal $\overline{IORQ}$ und dem Ist-Gleich-Signal des 6fach Vergleichers. Es gilt folgende Adreßzuordnung:
(XXXXXX ≙ Card Select Adresse).

- ☐ die Adresse XXXXXXII ermöglicht den Start eines Wandelvorgangs. Hierzu ist ein OUT-Befehl notwendig, wobei die Kanaladresse als binäre Information auf den Datenleitungen D0−D3 liegen muß.
- ☐ die Adresse XXXXXX00 dient zum Einlesen der Datenbits D0−D7 (IN-Befehl)
- ☐ die Adresse XXXXXX01 dient schließlich zum Einlesen des höherwertigen Halbbytes (D8−D11).

Da das Wandlermodul bereits intern mit TTL-Tristate Puffern ausgerüstet ist, sind die Wandlerausgänge unmittelbar mit dem Mikrocomputersystembus verbunden. Diese Puffer sind in 3 Gruppen zu je vier Bit getrennt schaltbar. Sie werden durch die letzten beiden der oben erwähnten Adressen freigegeben.

Beachten Sie bitte, daß zwischen dem Start eines Wandlerzyklus und em Einlesen der Daten, die Wandelzeit des Moduls von ca. 30 μs abgewartet werden muß (siehe auch Programmbeispiele).

3.2.4 D/A-Wandler

Neben der A/D-Wandler-Einheit enthält die Baugruppe KONTRON AN -μP80/E16 auch einen 12 bit D/A-Wandler. Dieser ist unabhängig von der A/D-Einheit zu adressieren. Da der D/A-Wandler immer 12 bit parallel zur Verfügung haben muß, ist ein Registerzusatz von insgesamt 16 bit vorgeschaltet. Dieser Registersatz ist in zwei getrennt adressierbar Blöcke (4+12 bit) eingeteilt, so daß zunächst das höherwertige Halbbyte (Bit 8-11) und anschließend das niederwertige Byte (Bit 0-7) übertragen werden muß. Dem Wandler JC ist ein Operationsverstärker nachgeschaltet, der eine Ausgangsspannung von ±10 V liefern kann. Sowohl Verstärkung, als auch Nullpunkt können über Potentiometer eingestellt werden. (P1 für Verstärkung, P2 für Nullpunkt). Die zulässige Ausgangs-Belastung ist < ±6 mA, bei Verwendung des Verstärkertyps 72L022. Eine größere Ausgangsbelastung ist möglich, wird anstelle des 72L022 der pinkompatible Typ RC 4558 eingesetzt.

3.4 Pinbelegung

Aus Tabelle 4 ist die Belegung des „analogseitigen" 40-poligen Steckers zu ersehen:

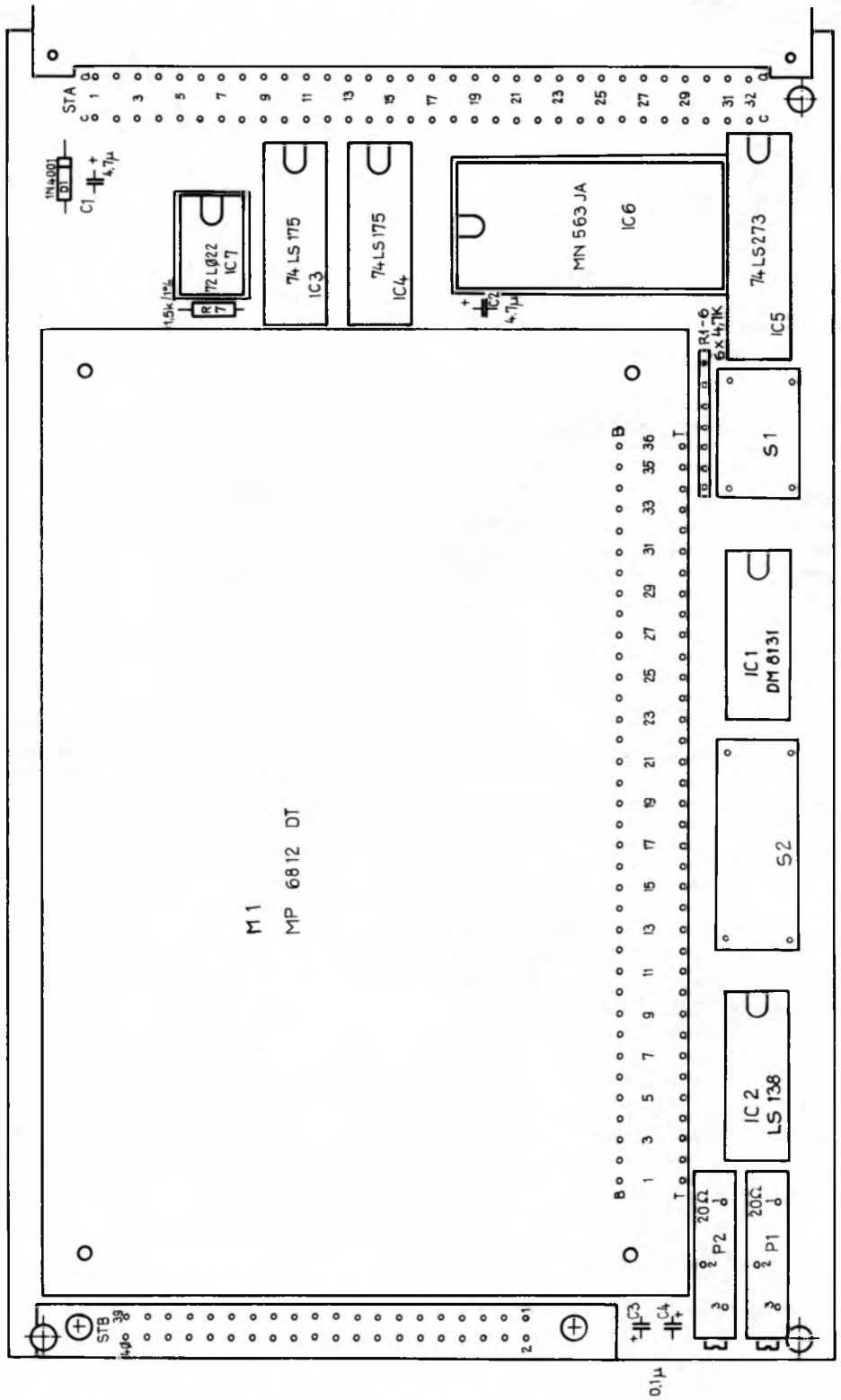
1	Kanal	0
3	Kanal	1
5	Kanal	2
7	Kanal	3
9	Kanal	4
11	Kanal	5
13	Kanal	6
15	Kanal	7
17	Kanal	8 oder (Kanal 0 RTN)
19	Kanal	9 oder (Kanal 1 RTN)
21	Kanal	10 oder (Kanal 2 RTN)
23	Kanal	11 oder (Kanal 3 RTN)
25	Kanal	12 oder (Kanal 4 RTN)
27	Kanal	13 oder (Kanal 5 RTN)
29	Kanal	14 oder (Kanal 6 RTN)
31	Kanal	15 oder (Kanal 7 RTN)
35	DA Ausgang	

alle übrigen Pins liegen auf GND (Masse).

Technische Daten:

Spannungsversorgung	+ 5 V, ± 5%
(Gleichspannung):	±15 V, ± 5%
Stromaufnahme (typisch):	450 mA bei 5 V, 60 mA bei ±15 V (je)
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 230 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

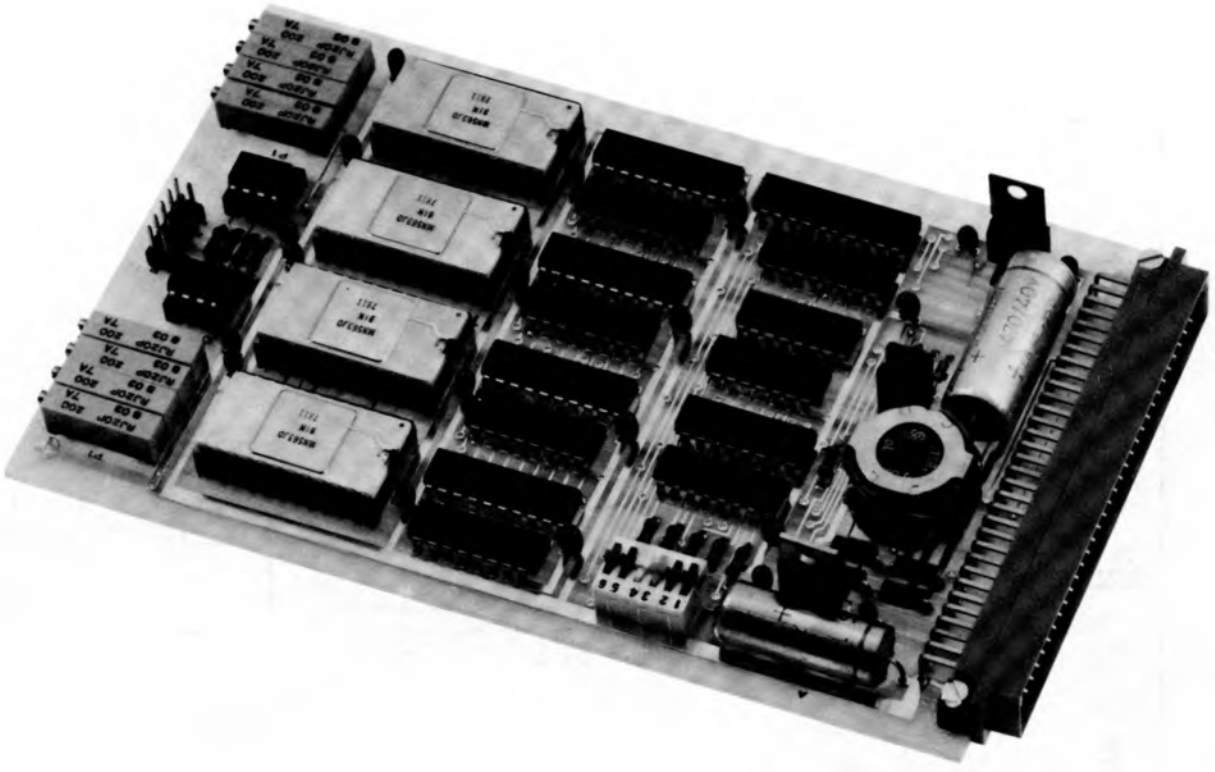
Schaltplan wird mit der Baugruppe mitgeliefert.


STA
STB
IN4001
D1
C1
4,7µ
72 L922
IC7
1,5K 1%
R7
74 LS 175
IC3
74 LS 175
IC4
MN 563 JA
IC6
IC2
4,7µ
74 LS 273
IC5
R1-6
6x4,7K
S1
S2
IC1
DM 8131
IC 2
LS 138
P2
P1
20Ω
20Ω
M1
MP 6812 DT
IC3
IC4
0,1µ

AN-μP80-A4

4 Kanal-Analogausgabe-Baugruppe

KONTRON
ELEKTRONIK GMBH



7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

4. D/A-Wandler Baugruppe AN-μP80/A4

4.1 Schaltungsbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten:

- Vier 12 Bit D/A-Kanäle
- Pufferung und Adreßdekoder
- Schalter zur Adreßbereichsfestlegung
- Betriebsartenschalter (I/O und Memory mapped I/O)
- DC/DC-Wandler für ± 15 V
- Busseitiger 64-poliger Stecker nach DIN 41612 (VG 95324)
- Ausgangsseitig 10-poliger 3M Stecker

Bild 1 zeigt das Blockschaltbild der Baugruppe. Es ist ersichtlich, daß der Systemdatenbus gepuffert ist, so daß volle Erweiterbarkeit und erhöhte Störsicherheit gewährleistet ist.

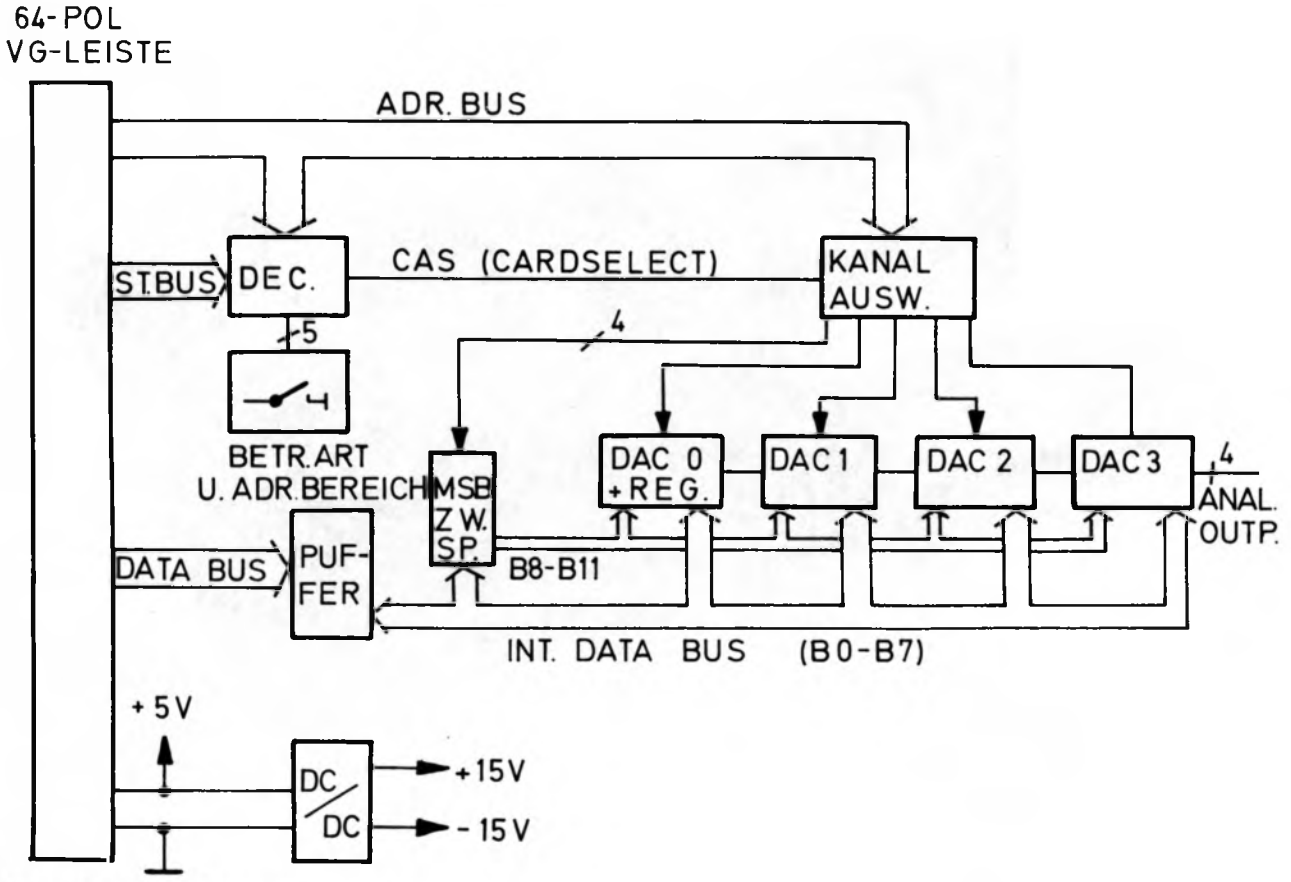


Bild 1: Blockschaltbild der Baugruppe

Die Baugruppe enthält vier voneinander unabhängige 12 Bit D/A-Kanäle. Jeder dieser Kanäle besteht aus drei Komponenten:

- dem eigentlichen D/A-Wandler
- einem vorgeschalteten 12 Bit Registersatz
- einem nachgeschalteten Operationsverstärker.

Außerdem ist bei jedem Kanal Nullpunkt und Verstärkung über je ein Potentiometer einzustellen (siehe auch Inbetriebnahme).

Standardmäßig ist die Baugruppe mit dem Wandler-IC MN563JA bestückt, das zur Wandlung von 12 Bit Binärzahlen geeignet ist. Auf Wunsch kann jedoch auch der Typ MN563JB mit 3 Digit-BCD-Eingang eingesetzt werden. Am Ausgang steht eine Spannung von max. ± 10 Volt zur Verfügung. Die zulässige Ausgangsbelastung kann wesentlich erhöht werden, wenn der OP 74L022 durch den pinkompatiblen Typ RC 4558 ersetzt wird. Die Belastbarkeit der Gleichspannungswandler auf der Baugruppe ist entsprechend dimensioniert.

Da dem Wandler-IC immer statisch 12 Bit zur Verfügung. stehen müssen, ist ein Registersatz vorgeschaltet. Aus dem Blockschaltbild (Bild 1) ist ersichtlich, daß die Bits B0-B7 direkt vom internen Datenbus abgeleitet werden, während die Bits B8-B11 vom MSB-Zwischenspeicher stammen. Daraus ist unmittelbar der Ablauf der 12 Bit Datenübertragung zur Baugruppe ersichtlich.

In einem ersten Schritt wird das MS-Halbbyte (B8-B11) in den MSB-Zwischenspeicher übertragen. Anschließend erfolgt in einem zweiten Schritt die Übertragung des LS-Bytes. Gleichzeitig wird das zwischengespeicherte MSB an den selektierten Kanal weitergeschoben.

Steckerbelegung

Systemstecker (STA)

A0	5 c	Adreßbus
A1	7 c	
A4	7 a	
A5	8 a	
A6	9 a	
A7	9 c	
D0	2 c	Datenbus
D1	14 c	
D2	4 c	
D3	4 a	
D4	5 a	
D5	2 a	
D6	3 a	
D7	3 c	
$\overline{\text{IORQ}}$	27 a	Ein/Ausgabeanforderung
$\overline{\text{RD}}$	24 c	Read
$\overline{\text{WR}}$	22 c	Write
$\overline{\text{PWRCL}}$	26 c	Reset
+5 V	1 a, c	Spannungsversorgung
GND	32 a, c	

Pfostensteckverbinder (STB)

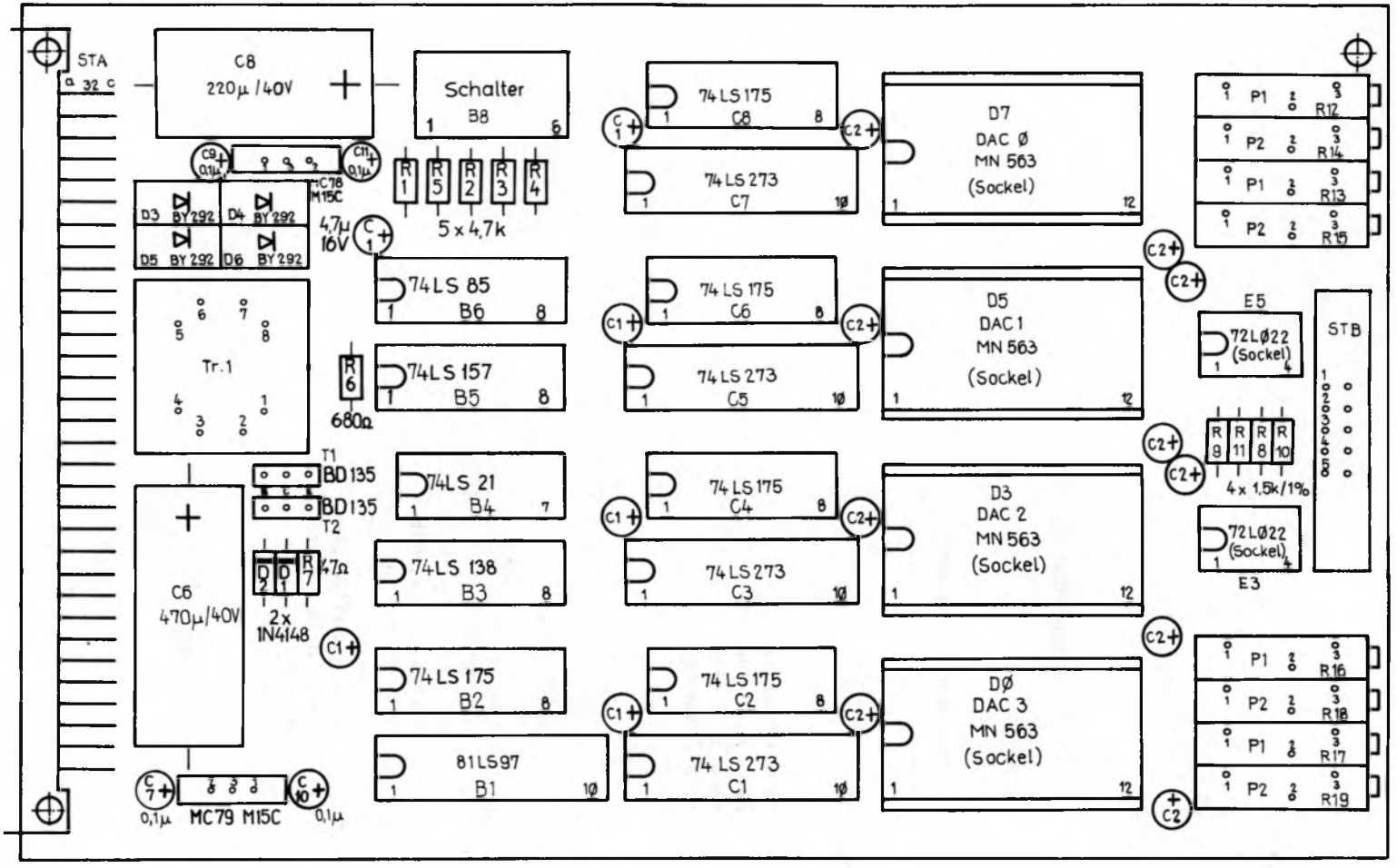
Pin	
1	Kanal 0 Ausgang
2	Kanal 1 Ausgang
4	Kanal 2 Ausgang
5	Kanal 3 Ausgang
3, 6–10	Masse

AN-μP80/A4

Technische Daten:

Spannungsversorgung (Gleichspannung):	+ 5 V, ± 5%
Stromaufnahme (typisch):	750 mA
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 25 mm
Gewicht:	ca. 150 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. II

Schaltplan wird mit der Baugruppe mitgeliefert.

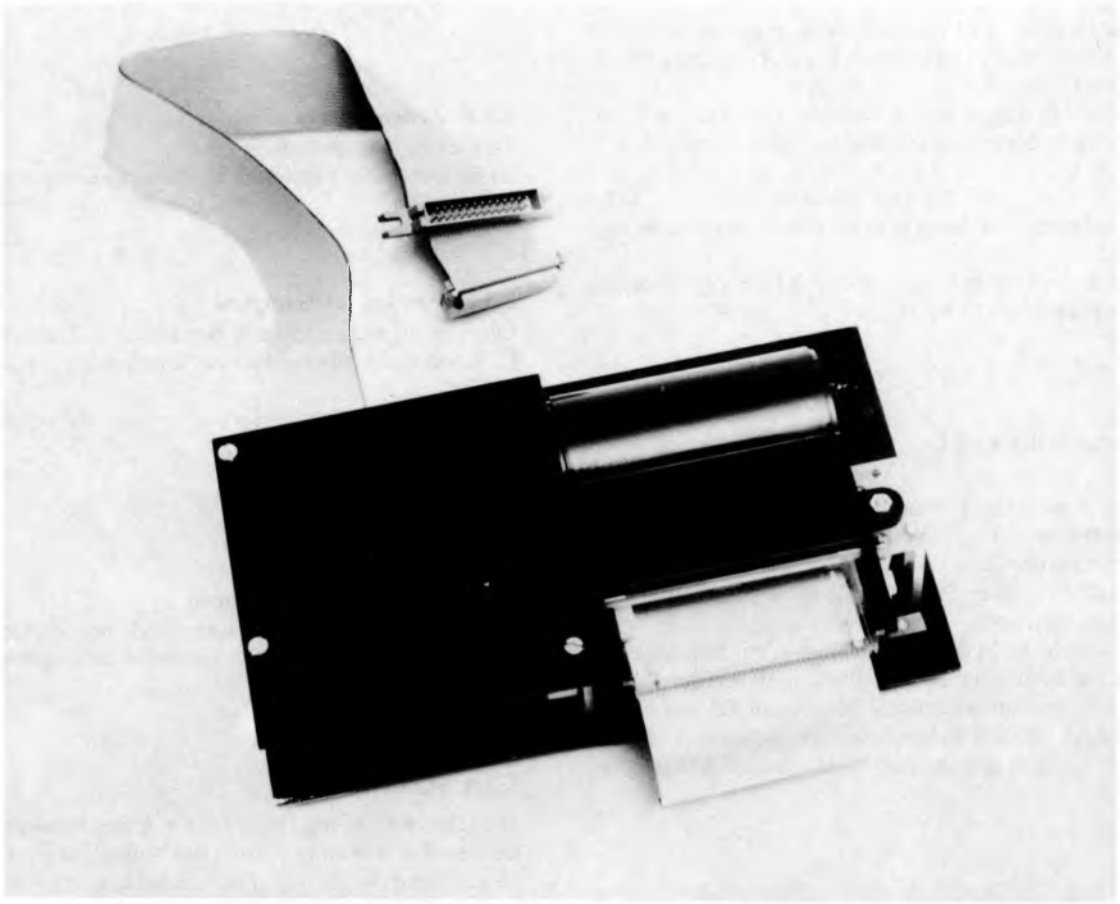


C1 = 4.7µ/16V Tantal
C2 = 0,1µ/20V

KONTRON-KIT/D

Druckerzusatz

KONTRON
ELEKTRONIK GMBH



7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

5. Druckerzusatz Z80-KIT/D

KIT/D ist ein Drucker-Zusatz zur Anfertigung von Hardcopies von Speicherinhalten.

Technische Daten:

Druckwerk:	5×7 Punktematrix
Zeichenzahl:	max. 29
Zeichengröße:	Höhe 3 mm, Breite ca. 2 mm bei 29 Zeichen
Zeilenabstand:	5 mm
Druckgeschwindigkeit:	max. 2 Zeilen/s
Zeichenvorrat:	64 Zeichen nach ASCII
Stromversorgung:	+5 V

5.1 Einleitung

Der Drucker-Zusatz Z80-KIT/D stellt eine preisgünstige Ergänzung des Einfach-Computer- und Lernsystems KONTRON Z80-KIT dar.
Er erlaubt mit der mitgelieferten Software den Ausdruck von Speicherstellen in Hexadezimalform und den Ausdruck von ASCII-Zeichen.
Die Software ist in zwei Unterprogramme (ASCII-, HEX-Ausdruck) aufgeteilt und kann vom Anwender wahlweise aufgerufen werden.
Der Anschluß des Druckers an den Z80-KIT erfolgt über eine parallele Schnittstelle (½ PIO).

5.2 Bestandteile des KIT/D

Der Drucker-Zusatz besteht aus:

- Grundplatte mit Papierhalterklappe, durch die Papier nachgelegt werden kann.
- KONTRON-Drucker 5020 B mit Papierhalter und Ansteuerungselektronik
- 26-pol. Scotchflex-Flachkabelverbinder (mit Zusatzplatine am Drucker befestigt); 26-pol. Pfosten-Verbinder (3M — Nr. 3399) zum Aufstecken auf 26-poligen Pfosten-Steckverbinder. Zusätzlich aufgequetschter Stiftstecker (3M — Nr. 3328) für den Zugang zum nicht benutzten Port B der PIO.

5.3 Technische Beschreibung des KONTRON-Druckwerks

5.3.1 Allgemeines

Es handelt sich um ein seriell arbeitendes Druckwerk. Gedruckt wird mittels eines beweglichen Druckkopfes, der sieben Elektroden trägt. Die Druckrichtung ist dabei je nach Einbaulage von rechts nach links bzw. von links nach rechts. Die Zeichen werden über eine 5×7 Punktematrix durch die Steuer-Logik gebildet.
Das Druckprinzip basiert auf dem punktweisen Abbrennen einer dünnen Metallschicht des Papiers. Die darunterliegende schwarze Schicht ergibt den notwendigen Kontrast.

5.3.2 Zeichengröße

Höhe: ca. 3 mm; festgelegt durch die Anordnung der sieben Elektroden
Breite: Variabel; hängt von der Zahl der in einer Zeile zu druckenden Zeichen ab.
Einstellung bei KIT/D;
Breite: ca. 1,5 mm

5.3.3 Zeichensatz

Der Zeichensatz hängt vom Zeichengenerator der Kontroll-Logik ab; es wird der aus 64 Zeichen bestehende ASCII-Zeichensatz verwendet. Änderungen des Zeichensatzes sind durch Einsatz anderer Zeichengeneratoren (PROM) möglich.

5.3.4 Zeichenzahl

Die Zahl der Zeichen pro Zeile wird mit der Zeichenbreite von der Steuer-Logik und von der dazugehörigen Druckersoftware bestimmt.

Beim KIT/D sind maximal 29 Zeichen pro Zeile erlaubt.
Mit dem Potentiometer auf der Steuer-Logik ist die Zeichenbreite und damit die Breite des gesamten Ausdrucks veränderbar.

5.3.5 Zeilenabstand

Der Zeilenabstand beträgt 5 mm.
Er ist durch die Mechanik des Druckwerkes festgelegt.

5.3.6 Druckgeschwindigkeit

Die Druckgeschwindigkeit beträgt ca. 2 Zeilen pro Sekunde.
Es besteht die Möglichkeit zur Umrüstung auf ca. 1 Zeile pro Sekunde.
Dies kann durch Änderung zweier Lötbrücken erreicht werden.

5.3.7 Umgebungsbedingungen

Das Druckwerk arbeitet ohne jegliche Schmiermittel. Dadurch kann es auch unter extremen Bedingungen arbeiten.

5.3.8 Motor

Das Druckwerk wird von einem Glockenankermotor angetrieben. Die Maximalleistung des Motors ist wesentlich größer als die Betriebsleistung. Dies garantiert eine konstante Laufgeschwindigkeit während des Druckvorgangs, wodurch ein Kopfpositionssignal entfällt und die Zeichenanforderungslogik durch einen frei laufenden Takt gesteuert werden kann.

5.3.9 Metallisiertes Papier

Das metallisierte Papier besteht aus Zellulose, wobei eine Seite mit einer durchgehenden, gleichmäßigen Aluminiumschicht bedampft ist. Zwischen Papier und Aluminium ist eine Zwischenschicht aus schwarzem Kontrastmaterial aufgebracht.
Physikalische und mechanische Eigenschaften:

- Gewicht: 37 Gramm/Quadratmeter (+/− 8%) gemessen nach Tappi 410 0S − 68 Normen
- Dicke: 41 Mikrometer (+/− 10%) gemessen nach Tappi 0S − 68 Normen
- Zugfestigkeit: in Längsrichtung: 6,5 kg/15 mm +/− 2, gemessen nach Tappi T 404 TS Normen
- Reißfestigkeit: längs und quer: 17 Gramm +/− 5, gemessen nach Tappi T 414 TS − 65 Normen

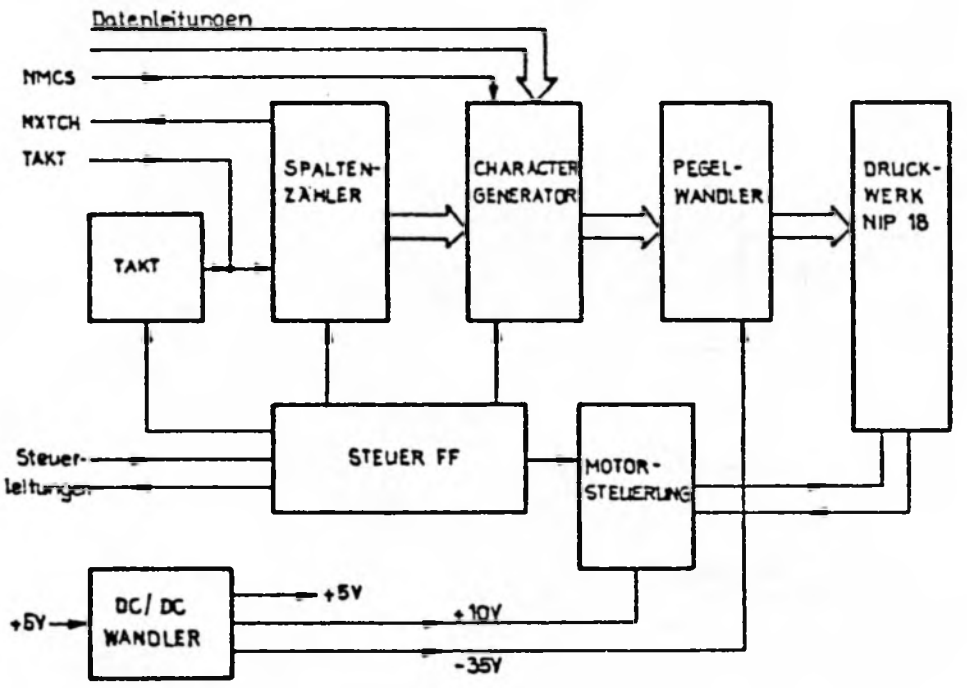
5.3.10 Geräuschentwicklung

Während des Druckvorganges ist der Geräuschpegel des Druckwerks ohne Gehäuse, gemessen in einem echofreien Raum, weniger als 54 Dezibel.
In der Bereitschaftsstellung ist das Druckwerk absolut geräuschlos.

5.4 Elektronischer Aufbau des Druckers

Die Logik des Druckers ist in CMOS-Technologie aufgebaut.
Als Zeichen-Generator wird ein TTL-PROM verwendet.

5.4.1 Blockschaltbild KIT/D



5.4.2 Anschluß des KIT/D

Mittels des aufgequetschten Stiftsteckers (3M — Nr. 3328) des Flachbandkabels kann über das verbleibende PORT B der PIO frei verfügt werden.

Die 3 Steuersignale
- NXTCH (Next Character)
- PB (Printer Busy)
- PRC (Print Command)

laufen ebenfalls über den PORT A (A 6, A 7 und ARDY) derselben PIO-Hälfte.
Die Ansteuerung des Dateneingangs des KIT/D erfolgt über die 6 Datenleitungen A 0 . . . . A 5 des PORT A einer PIO (6 Bit ASCII, Bit-parallel, Zeichen-seriell).

5.4.3 Verwendbare ASCII-Zeichen

Der vom Drucker erzeugbare Zeichensatz ist abhängig vom verwendeten Zeichengenerator.
Grundsätzlich ist eine Darstellung aller 7Bit-ASCII-Zeichen möglich.
Bei Benutzung nur eines PIO-Ports zur Druckeransteuerung ergibt sich für den Drucker die Beschränkung des ASCII-Zeichenvorrats auf sechs Datenbits.
Erzeugt werden können folgende Zeichen:

6bit-ASCII-Zeichensatz

LSD \ MSD		3 1 1	4 0 0	5 0 1
0	0000	0	@	P
1	0001	1	A	Q
2	0010	2	B	R
3	0011	3	C	S
4	0100	4	D	T
5	0101	5	E	U
6	0110	6	F	V
7	0111	7	G	W
8	1000	8	H	X
9	1001	9	I	Y
A	1010	:	J	Z
B	1011	;	K	[
C	1100	<	L	\
D	1101	=	M	]
E	1110	>	N	↑
F	1111	?	O	←

Darüber hinaus war jedoch die Kombination CR (Carriage Return) für das Anfordern einer Neuzeile (ASCII: 0D) wünschenswert. Dieses Problem kann softwaremäßig gelöst werden. Wird 0D (≙ CR) von der Software erkannt, sorgt diese dafür, daß der Rest der aktuellen Druckzeile (der Drucker kennt kein Neuzeile-Kommando!) mit Leerzeichen (im Zeichensatz des Druckers enthalten!) aufgefüllt wird.
Als weiteres Zusatzzeichen ist FF sinnvoll, bei dessen Auftreten ein Abbruch des Ausdruckes erreicht wird.

6. IEC-BUS-Interface

ECB/B (in Vorbereitung)

Die Baugruppe ECB/B dient zur Koppelung ECB-basierender Systeme an Meßsysteme über den IEC bzw. IEEE-Norm-Bus.
Die Realisierung erfolgt mit hochintegrierten spezifischen IEC-Bus-Bausteinen, die für volles Einhalten der IEC-Spezifikation garantieren.

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

KONTRON
ELEKTRONIK GMBH

7. Arithmetik-Prozessor ECB/A (in Vorbereitung)

Die Baugruppe ECB/A ist für rechenintensive Anwendungen und für Probleme gedacht, bei denen arithmetische Operationen in einer Zeit ausgeführt werden müssen, die heute verfügbare Mikrorechner nicht erreichen können. Hiefür wurde ein spezieller, hochintegrierter in MOS-Arithmetik Baustein verwendet, der sowohl Festkomma als auch Gleitkomma-Operationen durchführt.

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN BEIDEN SERIEN Z80A-ECB UND Z80-ECB

ECB/N NETZGERÄTE

KONTRON
ELEKTRONIK GMBH

6. Schaltnetzteile im Einfacheuropaformat ECB/N

6.1 Übersicht

Die Serie ECB/N stellt dem ECB-Anwender ein Spektrum von Netzteilen zur Verfügung, die direkt in den reservierten Steckplatz des Baugruppenträgers ECB/R einsteckbar sind. Sie sind in folgenden Versionen lieferbar:

ECB/N1	+ 5 V	3.5	A
ECB/N2	+ 5 V	3.5	A
	+ 15 V	0.25	A
	— 15 V	0.25	A
ECB/N4	+ 5 V	3.5	A
	+ 12 V	0.3	A
	— 5 V	0.7	A

6.2 Technische Daten:

- Eingangsspannung
  220 Vac ± 10%
- Netzfrequenz
  50/60 Hz
- Ausgangsspannungen
  5 Volt ± 5% einstellbar
  ± 15 Volt ± 4% fest
  + 12 Volt ± 4% fest
  — 5 Volt ± 4% fest
  Die Zusatzspannungen sind kurzschlußfest und besitzen einen thermischen Überlastungsschutz.
- Kurzschlußfestigkeit
  Dauerkurzschlußfest
- Regelung
  Netz: ± 0,1%
  Last: ± 0,4% (Leerlauf-Vollast)
- Restwelligkeit
  40 mV_{SS} max.
- Isolationsspannung
  (primär/sekundär) 1 kV/ac
- Temperatur-Koeffizient
  0,01%/°C
- Schaltfrequenz
  0 — 100 kHz (Lastabhängig)
- Wirkungsgrad
  > 65%
- Power Fail-Signal
  Low aktive, 2 msec vor Abfall der Ausgangsspannung um 50 mV
- Betriebstemperatur
  0 bis + 60°C
- Lagertemperatur
  − 10°C bis + 80°C
- Feuchtigkeit
  bis 90%, nicht kondensierend
- Abmessungen
  160 × 100 × 40 mm
- Gewicht
  600 gr

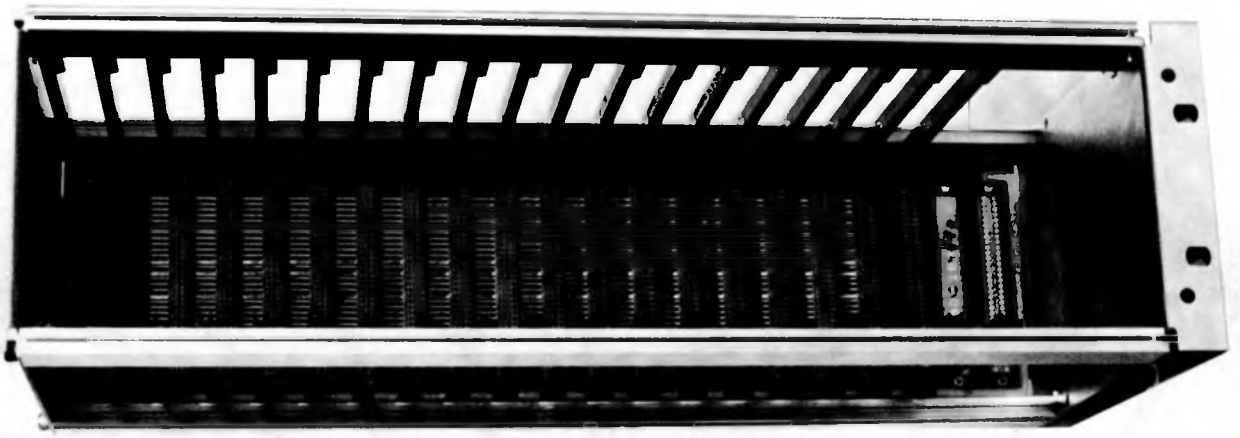
Vorläufige Daten

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN BEIDEN SERIEN Z80A-ECB UND Z80-ECB

ECB/R

KONTRON
ELEKTRONIK GMBH

Baugruppenträger



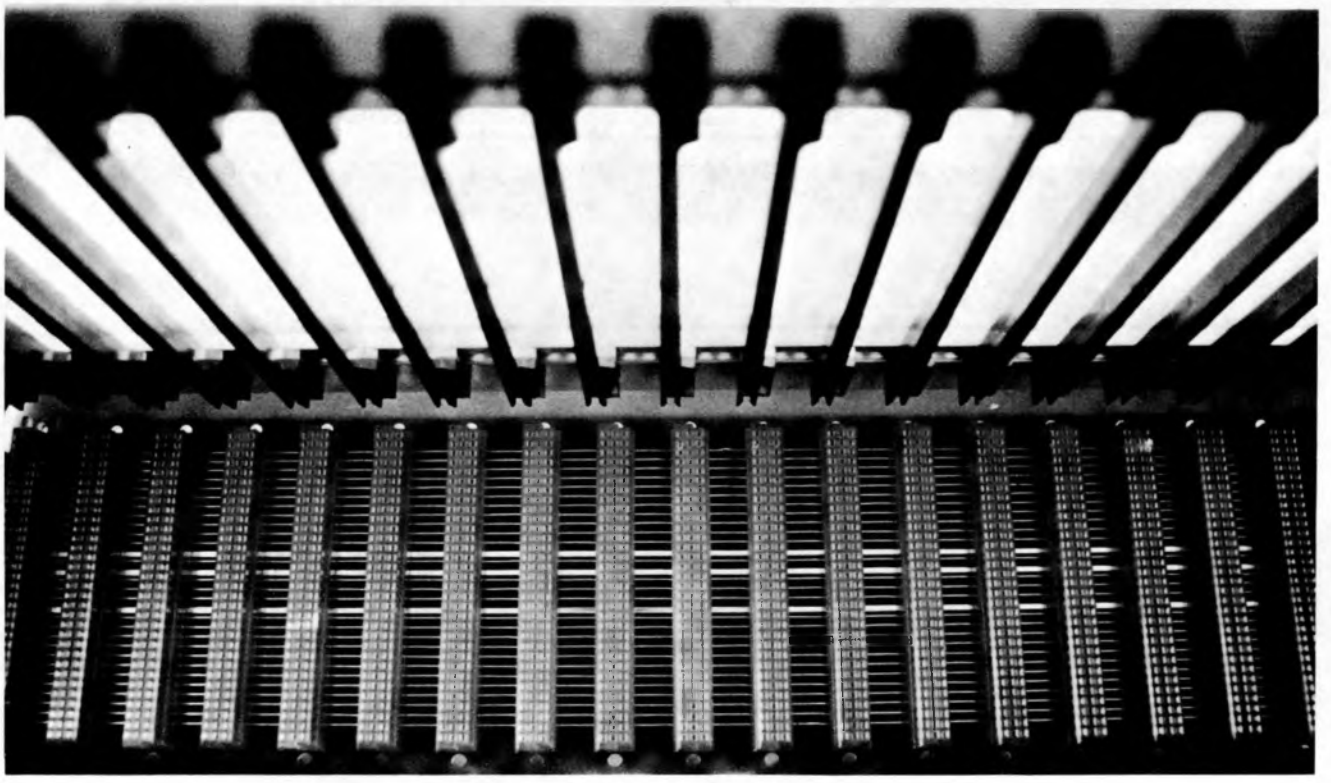
7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

7. Baugruppenträger ECB/R

Der Baugruppenträger ECB/R ist ein besonders sorgfältig ausgeführter 19''-Norm-Einschubrahmen, der besonders für den Betrieb von Computersystemen höherer Taktfrequenzen (z.B 4 MHz) ausgelegt ist.
Die erforderliche Störsicherheit wird durch eine 4-Lagen-Verdrahtung und sorgfältigste Leitungs-Führung und -Bemessung garantiert.
Der Baugruppenträger enthält 18 Norm-Federleisten (64-polig nach DIN 41612, VG 95324), deren Bus-Signale sämtlich 1:1-verdrahtet sind und durch Wire-Wrap-Pins auch in ihrer Verdrahtung modifizierbar sind.
Darüber hinaus ist ein weiterer Steckplatz zur Aufnahme des Europa-Netzteils ECB/N vorgesehen.

Technische Daten:

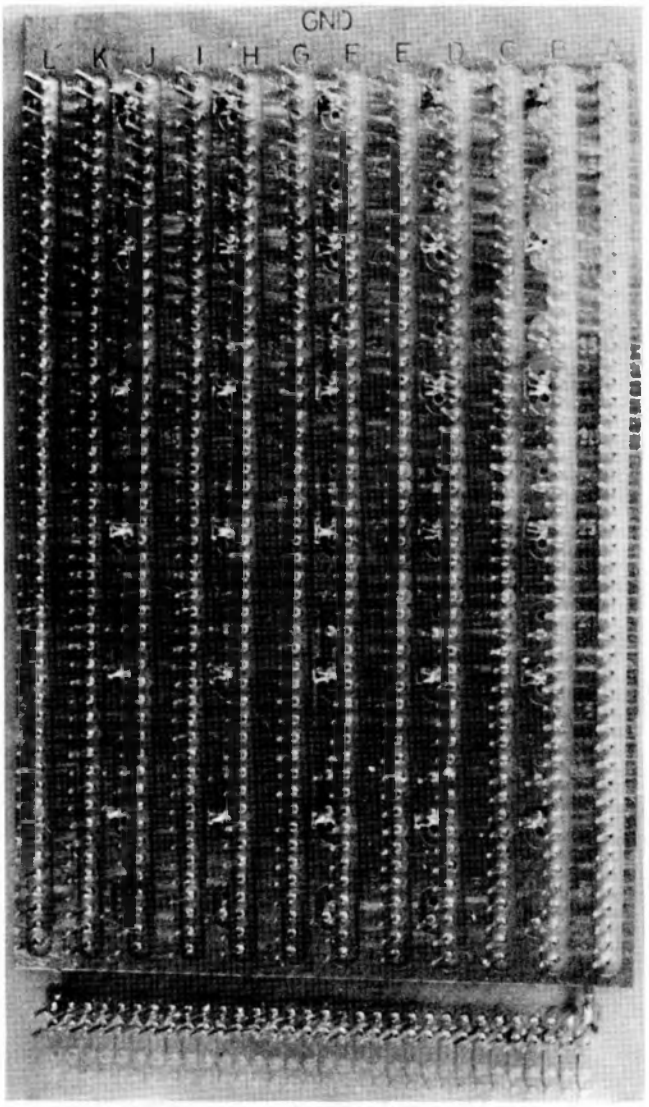
Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Gewicht:	ca. 2100 g
Busseitige Steckverbinder:	64-polige VG-Leisten, Belegung der Reihen a und c nach ECB-Standard, 1:1 durchverdrahtet
Beschriebene Version:	Rev. I



ECB/W

Wire-Wrap-Baugruppe im Einfach-Europa-Format

KONTRON
ELEKTRONIK GMBH



7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

10. Wire-Wrap-Baugruppe ECB/W

Die Baugruppe erlaubt den raschen, kostensparenden Aufbau von anwenderspezifischen, ECB-kompatiblen Z80-Mikrocomputerschaltungen in Wire-Wrap-Technik.
Es können beliebig gemischt 40-, 28-, 20-, 16- und 14-polige Bausteine verdrahtet werden.

Technische Daten:

Spannungsversorgung	+ 5 V, ± 5%
Umgebungstemperatur:	0 . . . 50 °C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 27 mm
Gewicht:	ca. 125 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard

ECB/Y

Verlängerungsplatine im Einfach-Europa-Format

KONTRON
ELEKTRONIK GMBH

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

Die Platine erlaubt das elektromechanische Herausführen der Bussignale aus dem ECB/R für Testzwecke.
Alle 64 Steckeranschlüsse sind 1:1 durchgezogen.

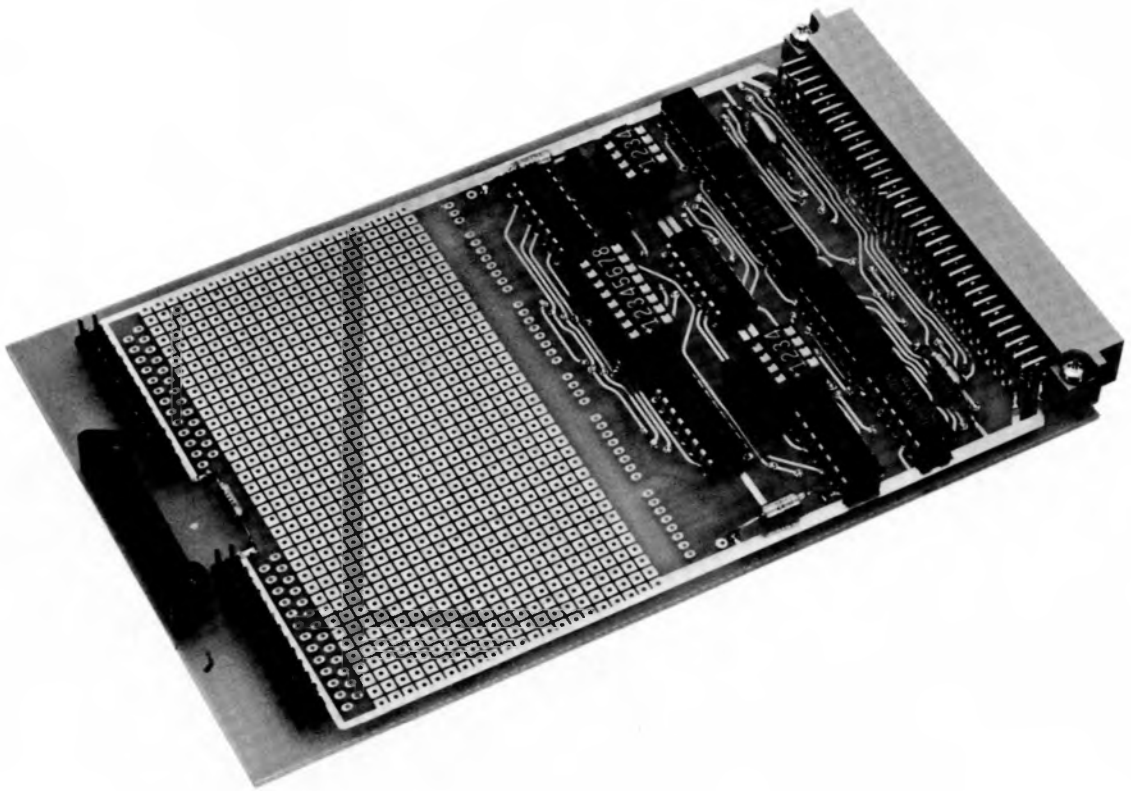
Technische Daten:

Umgebungstemperatur:	0 . . . 50°C
Relative Feuchte:	0 . . . 95% (nicht kondensierend)
Abmessungen:	160 × 100 × 15 mm
Gewicht:	ca. 90 g
Busseitige Steckverbinder:	64-poliger VG-Steckverbinder, Belegung der Reihen a und c nach ECB-Standard
Beschriebene Version:	Rev. I

EML/BF

STOLZ AG

Bus Foundation-Module



7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

12. Bus-Foundation-Module EML/BF

12.1 Übersicht

- enthält gesamte, voll decodierte Bus-Steuerung
- je 2×8 Bit Read- und Write-Register (4 Adressen)
- Interrupt-Möglichkeit mit Daisy-Chain-Technik
- 90 mm×70 mm Platz für Anwenderschaltungen (gebohrt)

12.2 Schaltungsbeschreibung

Die Baugruppe erlaubt die schnelle Entwicklung von Benutzer-Interfaces oder Spezialschaltungen im Zusammenhang mit dem ECB-Sysem. Die gesamte Bus-Steuerung (volle Decodierung) ist auf dem unteren Teil der Platte enthalten. Es stehen je 2×8 Bit Lese- und Schreib-Register zur Verfügung (4 Adressen). Interrupt-Möglichkeit und Daisy-Chain sind implementiert. Am oberen Plattenende sind Sockel für 2 26-polige Flachbandkabel eingesetzt.

Durch Benützer verwendbare Fläche: 90×70 mm gebohrt und quadratisch geätzt im 1/10''-Raster

Z80-EML/TG

STOLZ AG

MICROLOG
Tischgehäuse

7.4. BAUGRUPPEN UND ERGÄNZUNGEN ZU DEN SERIEN Z80A-ECB UND Z80-ECB

13. Tischgehäuse EML/TG

13.1 Eigenschaften

- stabile Konstruktion
- 7 Steckplätze für Einfach-Europa-Karten
- Aluminium farblos eloxiert
- Stromversorgung

13.2 Beschreibung

Das EML/TG Standard Tischgehäuse eignet sich für qualitativ hochstehende, freistehend eingesetzte Geräte (Tisch- oder Standmodelle) im Zusammenhang mit Mikrocomputer-Applikationen. Das Gehäuse besitzt eine hohe mechanische Festigkeit, Konvektionskühlungsschlitze und Platz für 7 Europakarten neben dem Stromversorgungsmodul. Die mitgelieferten Standard Front- und Rückplatten können vom Kunden mit weiteren Bedienungselementen oder Steckern versehen werden.

Abmessungen

Höhe (inkl. Füßchen):	128 mm
Breite:	215 mm
Tiefe:	300 mm
Gewicht:	4,1 kg

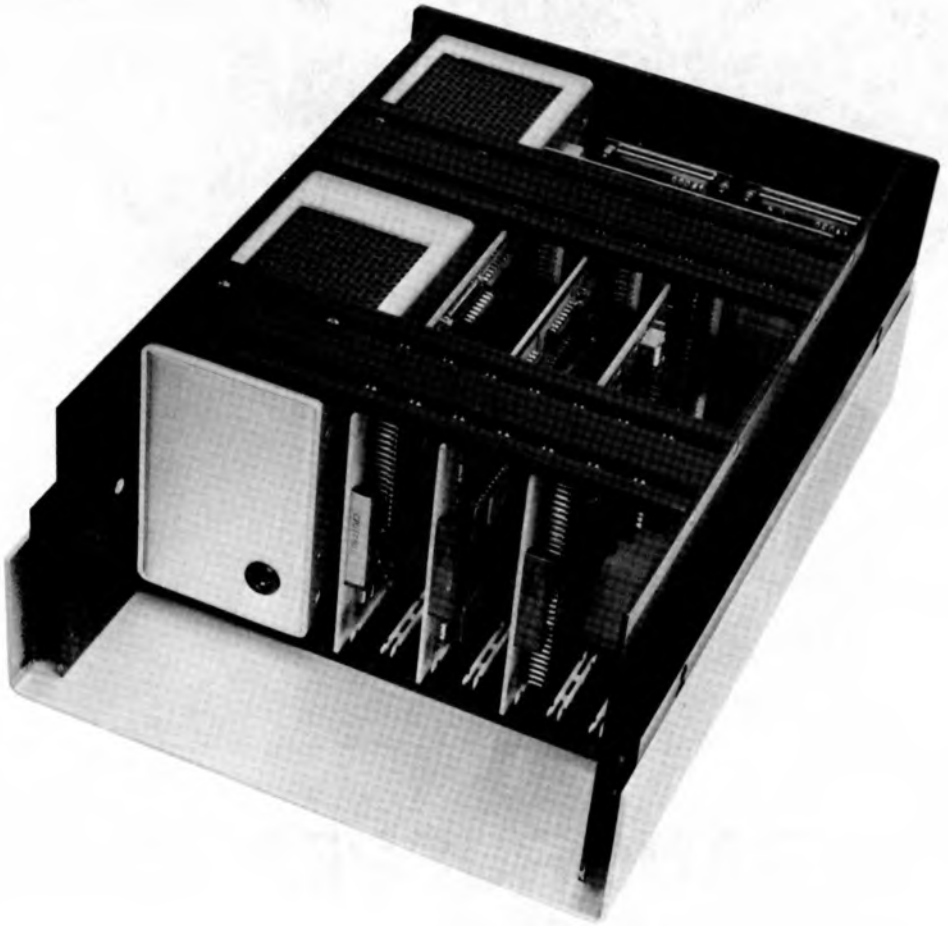
13.3 Rückwandverdrahtung

- Doppelseitig geätzte Backplane für 7 64-polige Normstecker nach DIN 41612.
- Herausgeführte Anschlüsse für die Stromversorgungen
- Interrupt und Daisy-Chain Leitungen nicht durchverdrahtet.

13.4 Stromversorgung

- kompaktes, berührungssicheres Netzteil
- Dauerbelastbarkeit: + 5 Volt: 2400 mA
  + 12 Volt: 400 mA
  —12 Volt: 400 mA
- Leistungsaufnahme: max. 46 VA
- enthält 20 msec Impulstakt für EMS/C-Platine
- alle Ausgänge auf Molex-Stecker geführt
- Netzschalter und Anzeige-LED extern über Front-Molexstecker
- Netzeingang durch Filter gegen Störeinstreuungen geschützt

Zur Beachtung:
Die Nennleistung des Netzteils darf nicht überschritten werden, da keine Thermosicherung eingebaut ist.



7.5. GRUNDSOFTWARE FÜR DIE ECB-SERIEN

Für die Z80-Baugruppen im Einfacheuropaformat (160× 100 mm) sind Monitor-Programme im Umfang von 2 kByte und 1 kByte verfügbar.
Das Programm ECB/2 dient zur Ansteuerung von Floppy-Disk-Systemen mit ECB/F. Die Monitor-Programme stellen beide jeweils ein einfaches Betriebssoftwarepaket dar, das als Firmware (also in einem Festwertspeicher gespeichertes Programm) geliefert wird; das PROM kann direkt in die hierfür auf der ECB-Baugruppe montierte Fassung eingesteckt werden. Der Monitor erlaubt die sofortige Inbetriebnahme des ECB-Computers und erspart dem Anwender die Erstellung gewisser Basis-Routinen, wie Serien-Ein/Ausgabe usw. Darüber hinaus ist mit dem Monitor sogar die Durchführung von Programm-Entwicklungen und -Tests möglich. Die verfügbaren Monitor-Kommandos sind in Punkt 3 beschrieben.

Z80-ECB/1:	1 kByte-Monitorprogramm
Z80-ECB/2:	2 kByte-Monitorprogramm
Z80-MTX:	Echtzeit-Multitask-Betriebssystem

Darüberhinaus sind Standardsoftwarepakete mit arithmetischen Operationen und Wandlungsroutinen und anwendungsorientierte Software verfügbar (siehe hierzu getrennte Veröffentlichungen).

Z80-ECB/1

1 kByte-Betriebsprogramm

KONTRON
ELEKTRONIK GMBH

1. Betriebsprogramm ECB/1

1.1 Einführung

Das Programm ECB/1 übernimmt einfache Grundfunktionen in Anwendersystemen, wie Kaltstartroutine, Steuerung des Serieninterface (z.B. für Testzwecke) und die Abhandlung des Zugriffs auf Speicher und CPU-Register sowie Haltpunkte.

1.2 Hardware-Erfordernisse des Monitors ECB/1

1.2.1 Speicherorganisation

Der 1 kByte Monitor ist in dezimal 1024 (bzw. hexadezimal 400) Festwertspeicherstellen abgelegt. Er benötigt darüber hinaus 35 Byte Schreib/Lesespeicher zuzüglich einem Stack-Bereich im RAM.
Dieser Schreib/Lesespeicher liegt am oberen Ende der vierten Seite, also von Adresse 3CFFH abwärts. (Eine Seite entspricht 4 kByte). Der darauffolgende Stack-Speicherbereich beginnt ab Adresse 3CD8H und wächst in Richtung niederwertiger Speicheradressen. Über den übrigen Speicherbereich kann frei verfügt werden.

1.2.2 Anschluß eines Bedienungs-Terminals

Über die 20 mA-Schnittstelle des ECB/C kann das Monitorprogramm mit jedem beliebigen Terminal verkehren, das 8 bit-ASCII-Code asynchron mit oder ohne Parity-Bit seriell überträgt. Es sind 14 verschiedene Übertragungsraten zugelassen, die auf der ECB/C-Baugruppe über die 4 Schalter einstellbar sind.
Soweit die Fähigkeit des Monitors zur Dateiausgabe bzw. Dateieingabe genutzt werden soll, muß das Terminal über ein Datenspeichermedium verfügen, das softwaremäßig einer Lochstreifen-Leser/Stanzer-Kombination entspricht. Als Start- und Stopsignal muß dieses Kommunikationsmedium die ASCII-Zeichen 11H und 13H erkennen können, oder an geeigneter Stelle manuell zu starten oder zu stoppen sein.

1.3 Beschreibung des Monitor-Programms

1.3.1 Verhältnis zu evtl. Anwenderprogrammen

Anwenderprogramme, die zusammen mit dem Monitor eingesetzt werden, können auf eine Reihe von dessen Unterprogrammen zugreifen.

1.3.2 Formale Regeln

Sämtliche Kommandos können durch ihren ersten Buchstaben abgekürzt oder auch bis zu jeder beliebigen Länge ausgeschrieben werden, da lediglich der erste Buchstabe in einer neuen Eingabe-Zeile als Schlüsselzeichen für ein empfangenes Kommando benutzt wird. Nicht verstandene Kommando-Eingaben werden vom Monitor durch Ausgabe des ASCII-Zeichens für ? quittiert, worauf eine neue Kommandoeingabe erwartet wird. Sämtliche Felder werden durch Leerzeichen begrenzt (= "Delimiters").
Außer den Daten im SET-Kommando können alle Zahlen formatfrei (= "free form") hexadezimal (auch ohne führende Nullen) eingegeben werden. Die als ein Datum eingebbare Zahl ist auf 4 Zeichen begrenzt; werden mehr als 4 Zeichen eingegeben, werden die letzten 4 Zeichen als gültige Zahl interpretiert.

7.5. GRUNDSOFTWARE FÜR DIE ECB-SERIEN

1.3.3 Beschreibung der einzelnen Kommandos

DISPLAY adr n

veranlaßt die Ausgabe des Inhalts der n folgenden Speicherstellen ab (inklusive) der Speicherstelle „adr".

SET adr dat1 dat2 .... datm

speichert die angegebenen Datenwörter (jedes als 2 Hexadezimalzeichen einzugeben!!) ab (inklusive) der Speicheradresse „adr" ab.
Unmittelbar auf das letzte Zeichen von datm muß ein Wagenrücklaufzeichen ("carriage return") eingegeben werden.

PUNCH anfadr endadr

gibt den Speicherinhalt ab (inklusiv) „anfadr" bis (inklusiv) „endadr" auf den Lochstreifenstanzer der Kommandokonsole aus, wobei der Stanzvorgang automatisch durch Ausgabe eines „Tape-on-Zeichens" (12 H ASCII) gestartet und durch Ausgabe von „Tape-off" (14 H ASCII) beendet wird. Nach „Tape-off" wird noch das Stanzen eines Stücks Leerstreifen veranlaßt.

LOAD

Einlesen von Daten von der Lochstreifeneinheit der Kommandokonsole in den Speicher, wobei die Anfangsadresse auf dem Lochstreifen eingestanzt sein muß und der Lochstreifen natürlich vorher (und zwar ein kleines Stück vor Vorkommen des ersten lesbaren Zeichens) in den Leser eingelegt werden muß.
Durch ein "Reader-On"-Signal wird der Leser vom Programm automatisch gestartet und am Ende der eingelesenen Datei wieder abgeschaltet.
Im Fehlerfall (= Check-Summe nicht richtig) wird der Leser ausgeschaltet und eine Nachricht ausgegeben, da anzunehmen ist, daß Daten verloren gegangen sind.

JUMP adr

veranlaßt einen unbedingten Sprung auf Adresse adr und Ausführung des dort befindlichen Programms.

GO

veranlaßt Fortsetzung des Programms nach einem Breakpoint.

BREAK adr

setzt einen Haltepunkt bei Speicheradresse adr, wobei zuvor evtl. vorhandene andere Haltepunkt-Bedingungen außer Kraft gesetzt werden.

REGISTER reg

Mit diesem Kommando können CPU-Registerinhalte ausgegeben und modifiziert werden.
Auf die Angabe des Registernamens hat statt des Wagenrücklaufzeichens ein Leerzeichen (Blank) zu folgen; vom Monitor wird daraufhin der Registerinhalt in der gleichen Zeile ausgegeben, worauf der Benutzer die Ausführung des Kommandos mit der Eingabe eines Wagenrücklaufzeichens abschließen kann.
Soll stattdessen das nächstfolgende Register ausgegeben werden, muß der Benutzer statt des Wagenrücklauf- ein Zeilenvorschub-Zeichen eingeben.
Soll der Inhalt eines eben angezeigten Registers modifiziert werden, ist unmittelbar nach einem weiteren Leerzeichen der gewünschte Registerinhalt einzugeben. Danach beendet wieder ein Wagenrücklauf- oder Zeilenvorschub-Zeichen den Vorgang.

Die Registerinhalte werden von RAM-Speicherstellen aus angezeigt, in denen sie vorher vom Monitor-Programm abgelegt werden.
Die Reihenfolge, in der die Register bei fortlaufender Eingabe von Zeilenvorschubzeichen in REGISTER-Kommandos behandelt werden ist folgende:
A, B, C, D, E, F, H, L, I, A', B', C', D', E', F', H', L', PC, SP, IX, IY.

1.3.4 Haltepunkt-Behandlung (nur beim 1 K-Monitor!)

Die Software-Haltepunkte des Z80-Monitors dienen zur Fehlersuche und -Beseitigung in Anwenderprogrammen. Wird eine Haltepunktbedingung bei der Ausführung des Anwenderprogramms als gültig erkannt, wird diese Ausführung abgebrochen, alle Registerinhalte in den hierfür vom Monitor reservierten Speicherbereich kopiert und eine Meldung ausgegeben, daß und bei welcher Speicherstelle der Haltepunkt erreicht wurde.
Besonders zu beachten ist, daß man den Stackpointer initialisieren muß, wenn man ein Anwenderprogramm ablaufen lassen und durch das Kommando BREAKPOINT behandeln will.

Dies kann geschehen

- durch Initialisierung im Anwenderprogramm selbst oder
- durch direktes Setzen des CPU-Stackpointers durch das Kommando REGISTER

Eine beliebige Anzahl von Haltepunkten kann manuell gesetzt werden, indem als Haltepunktbedingung auf der gewünschten Adresse die Daten 0FFH eingegeben werden.
Damit der Haltepunkt richtig behandelt wird, ist es notwendig, den Stackpointer im User-Programm an eine Stelle im gültigen RAM-Bereich zu setzen, die **nicht** durch die Monitor-Routinen benützt werden. Dies geschieht entweder durch Setzen des SP am Anfang des User-Programms, oder durch Modifizieren des Registerinhalts SP. Günstige Adressen für den ECB 1K-Monitor sind ab 3CD0H abwärts.
Beliebig viele Haltepunkte können manuell durch Angabe der Haltepunktadresse 0 FFH gesetzt werden. Dabei muß die Haltepunktadresse auf das erste Byte einer Anweisung zeigen, und die ursprüngliche Anweisung muß von Hand wieder auf diesen Speicherplatz geschrieben werden, sobald dieser Haltepunkt nicht mehr benötigt wird.
Hintergrund dieses Verfahrens ist die Tatsache, daß der Monitor grundsätzlich anhält, wenn er annimmt, daß er auf eine physikalisch nichtexistierende Speicherstelle (die den Inhalt FFH = RST 38H-Befehl hat) zugreift.

1.4 Bestellbezeichnungen:

Wegen der unterschiedlichen Ein/Ausgabedecodierung weisen die 1 kB-Monitorprogramme für Verwendung mit der Baugruppe ECB/C8 und ECB/C interne Unterschiede auf.
Die verfügbaren Versionen werden durch Anfügen der Ziffer „8" (für ECB/C8) bzw. „0" (für ECB/C) unterschieden; der Zusatz-TV bezeichnet die Eignung zum Betrieb mit TV-Monitor und ASCII-Tastatur als Kommando-Konsole.

Beispiele:

Z80-ECB/18:	1 kB-Monitor für Z80-ECB/C8
Z80-ECB/10:	1 kB-Monitor für Z80-ECB/C
Z80-ECB/18-TV:	1 kB-Monitor für Konfiguration bestehend aus Z80-ECB/C8, Z80-KIT/VZ, TV-Monitor und ASCII-Tastatur.

Z80-ECB/2

Stand-Alone Betriebsprogramm mit Floppy-Disk-Treiber

KONTRON ELEKTRONIK GMBH

2. Z80-ECB/2 ECB 2 K-Monitor

2.1 Übersicht

Der ECB 2k-Monitor enthält als Erweiterung des bekannten ECB 1k-Monitors die Grundfirmware zur Ansteuerung von Floppy-Disk-Laufwerken. Damit ist sowohl das Lesen, als auch das Beschreiben von einzelnen Sektoren der Floppy Disk möglich. Für beide Operationen stehen entsprechende Systemkommandos zur Verfügung.

Leistungsfähigkeit und Benutzerfreundlichkeit der Floppy-Disk-Grundsoftware erlauben mit relativ wenig Aufwand den Aufbau von einfachen Floppy-Disk-Betriebssystemen.

Der ECB-2k-Monitor ist in folgenden Versionen zur Ansteuerung von FD-Laufwerken lieferbar:

a) alternativ bei Verwendung von Z80-ECB/C mit dem Zusatz „-F0S" bzw. bei Verwendung von Z80-ECB/C8 oder Z80A-ECB/C8 mit dem Zusatz „-F8S"

b) alternativ für Einsatz mit hardsektorierter Floppy-Disk in einfacher Schreibdichte (Zusatz „/S"), und Minifloppy Zusatz „/M").

Bestellbeispiel: Z80-ECB/2-F8S/S: Monitor für System mit Z80-ECB/C8 und Normalfloppy, einfache Schreibdichte.

Zur Kennzeichnung hat jedes PROM einen Aufkleber mit den letzten beiden Zeichen der Bestellbezeichnung, also z.B. „0S" oder „8S" für die beiden zu Anfang genannten Versionen.

Zum Lieferumfang gehören:

1. Eine formatierte Diskette mit der „Formatierroutine"
2. Ein Listing des Monitorprogramms (ohne FD-Treiber), bestehend aus dem Listing von fünf verschiedenen relokativen Moduln.
   a.) ECB2K.L
   b.) INIT.ECB.F.L.
   c.) ECBF.RAM.L
   d.) ECB.MON.RAM.L
   e.) ECB.INT.L
3. Ein 2k-Monitor Software Anwender-Handbuch mit Liste von Einsprungpunkten und Adressen von Systemvariablen.

2.2 Hardwareerfordernisse

2.2.1 Computerbaugruppen

Der ECB-2k-Monitor setzt in jeder Version mindestens zwei Baugruppen der Z80-ECB-Serie voraus, nämlich

- die Z80-ECB/C (oder Z80A-ECB/C8 bzw. Z80-ECB/C8)
- die Z80-ECB/F

Um jedoch die Möglichkeiten des Speichermediums Diskette ausnützen zu können, ist in der Regel auch eine Erweiterung des RAM-Bereichs angebracht, z.B. durch eine der folgenden Baugruppen:

- Z80-ECB/E16
- Z80-ECB/D32
- Z80-KIT/P

Der vom Monitor selbst beanspruchte Speicherbereich liegt im RAM-Bereich der ECB/C von XX90H-XXFFH (siehe auch Abschnitte 6 und 7). Hierbei hat XX in Abhängigkeit der verwendeten ECB/C-Version folgenden Wert:

XX = 3C für die Z80-ECB/C
XX = 13 für die Z80-ECB/C8

Diese Symbolik wird im folgenden beibehalten.

Die Adressen XX8E und XX8F enthalten die Sprungadresse für die NMI-Serviceroutine.

Unterhalb der Adresse XX8DH beginnt der Stackbereich der maximal 20H Bytes beansprucht.

2.2.2 Hardwaremodifikationen auf der Z80-ECB/C

Die Z80-ECB/C ist standardmäßig nur zur Aufnahme von ½ kByte bipolaren PROM's ausgerüstet. Um die E-PROM's 2758 einsetzen zu können, sind an den Promsockeln A2, A4 folgende Änderungen notwendig:

1. Adr. A9 (B6,9) → A2,22 und A4,22
2. PIN 18 und 19 der Promsockel statt auf 5 Volt auf Masse legen.
3. Verbindung von PIN 20 der beiden Promsockel auf GND auftrennen und anschließend das Signal von PIN 21 (CS) auf PIN 20 legen.
4. PIN 21 der PROM's jeweils auf +5 Volt legen.

Außerdem muß der Dekoderprom G6 durch den beiliegenden PROM HM 7611 ausgetauscht werden.

7.5. GRUNDSOFTWARE FÜR DIE ECB-SERIEN

2.2.3 Bedienkonsole

Die Kommunikation mit dem „Rechner" erfolgt ausschließlich über die **Serienschnittstelle auf der ECB/F.**
Schnittstellen und Anschlußmöglichkeiten sind im ECB-Anwenderhandbuch beschrieben. (Abschnitt 5.4.3).
Die Schalter S1 bis S4 des 4-fach DIP Switch auf der ECB/F dienen zur Einstellung der Baudrate, die softwaremäßig über einen CTC-Kanal erzeugt wird.
Tabelle 1 enthält die möglichen Baudraten und Schalterstellungen.

S4	S3	S2	S1	Baudrate
0	0	0	0	–
0	0	0	I	110
0	0	I	0	150
0	0	I	I	300
0	I	0	0	600
0	I	0	I	1200
0	I	I	0	2400
0	I	I	I	4800
I	0	0	0	9600
I	0	0	I	19200

Bei nichtzulässigen Schalterstellungen (≥10) läuft der Prozessor in einer Schleife. Nach der Änderung der Schalterstellung ist ein Reset erforderlich.

Es gilt: 0 = EIN
I = AUS

Tabelle 1: Baudrate Einstellung

2.2.4 Floppy Disk Laufwerk

Der 2k-Monitor Z80-ECB/2-F0S bzw. -F8S erlaubt den Anschluß von Standardlaufwerken (Firma Shugart oder ähnliche) für hardsektorierte Disketten mit 77 Spuren à 32 Sektoren bzw. die Versionen /2-F0M und /2-F8M den Anschluß von Mini-Floppy-Disk.

Folgende Verbindungen müssen zwischen ECB/F und Floppy-Disk-Laufwerk(en) vorhanden sein: (Tabelle 2)

LAUFWERK	ECB/F-Anschluß
DISK READ DATA	A12
TRACK 0	A10
WRITE PROTECT	A11
INDEX/SECTOR	A1
WRITE GATE	A9
DISK WRITE DATA	A8
STEP	A7
DIRECTION	A6
DRIVE SELECT 0	A2
(„ 1)	(A3)
(„ 7)	(B9)

Tabelle 2: ECB/F-FD Verbindungen

Zum einwandfreien Betrieb muß der Datenseparator der ECB/F durch den 4-fach DIP-Switch (E4) folgendermaßen programmiert werden:

S1 = OFF
S2 = ON
S3 = ON
S4 = ON

2.3 Beschreibung des Monitorprogramms

2.3.1 Formale Regeln

Sämtliche Kommandos können durch ihren ersten Buchstaben abgekürzt oder auch bis zu jeder beliebigen Länge ausgeschrieben werden, da lediglich der erste Buchstabe in einer neuen Eingabe-Zeile als Schlüsselzeichen für ein empfangenes Kommando benutzt wird.
Nicht verstandene Kommando-Eingaben werden vom Monitor durch Ausgabe des ASCII-Zeichens für ? quittiert, worauf eine neue Kommandoeingabe erwartet wird. Sämtliche Felder werden durch Leerzeichen begrenzt (= „Delimiters").
Außer den Daten im SET-Kommando können alle Zahlen formatfrei (= „free form") hexadezimal (auch ohne führende Nullen) eingegeben werden. Die als ein Datum eingebbare Zahl ist auf 4 Zeichen begrenzt; werden mehr als 4 Zeichen eingegeben, werden die letzten 4 Zeichen als gültige Zahl interpretiert.

2.3.2. Die Kommandos*

DISPLAY adr n
veranlaßt die Ausgabe des Inhalts der n folgenden Speicherstellen ab (inklusive) der Speicherstelle „adr".

SET adr dat1 dat2 . . . datm
speichert die angegebenen Datenwörter (jedes als 2 Hexadezimalzeichen einzugeben!!) ab (inklusive der Speicheradresse „adr" ab.
Unmittelbar auf das letzte Zeichen von datm muß ein Wagenrücklaufzeichen („carriage return") eingegeben werden.

JUMP adr
veranlaßt einen unbedingten Sprung auf Adresse adr und Ausführung des dort befindlichen Programms.

GO
veranlaßt Fortsetzung des Proveranlaßt Fortsetzung des Programms nach einem Breakpoint.

BREAK adr
setzt einen Haltepunkt bei Speicheradresse adr, wobei zuvor evtl. vorhandene andere Haltepunkt-Bedingungen außer Kraft gesetzt werden.
Fehlt die Eingabe adr, so wird ein zuvor gesetzter Break (= „Haltepunkt") gelöscht.

REGISTER reg
Mit diesem Kommando können CPU-Registerinhalte ausgegeben und modifiziert werden. Auf die Angabe des Registernamens hat statt des Wagenrücklaufzeichens ein Leerzeichen (Blank) zu folgen; vom Monitor wird daraufhin der Registerinhalt in der gleichen Zeile ausgegeben, worauf der Benutzer die Ausführung des Kommandos mit der Eingabe eines Wagenrücklaufzeichens abschließen kann.
Soll stattdessen das nächstfolgende Register ausgegeben werden, muß der Benutzer statt des Wagenrücklauf- ein Zeilenvorschub-Zeichen eingeben.
Soll der Inhalt eines eben angezeigten Registers modifiziert werden, ist unmittelbar nach ei-

nem weiteren Leerzeichen der gewünschte Registerinhalt einzugeben. Danach beendet wieder ein Wagenrücklauf- oder Zeilenvorschub-Zeichen den Vorgang.
Die Registerinhalte werden von RAM-Speicherstellen aus angezeigt, in denen sie vorher vom Monitor-Programm abgelegt werden.
Die Reihenfolge, in der die Register bei fortlaufender Eingabe von Zeilenvorschubzeichen in REGISTER-Kommandos behandelt werden ist folgende:
A, B, C, D, E, F, H, L, I,
A', B', C', D', E', F', H', L',
PC, SP, IX, IY.

LOAD — Laden eines Sektors (oder mehrerer Sektoren) von der Diskette. Der entsprechende Vektor wird vom Monitorprogramm ab Speicherstelle XX90H im RAM abgelegt und kann dort leicht durch den Anwender modifiziert werden (siehe Abschnitt 4). Die Voreinstellung des Vektors führt zum Laden eines 500 Byte großen Datenblocks ab Sektor 3 von Spur 00H in einen Speicherbereich ab 8000H. Dort befindet sich bei einer ECB-formatierten Diskette die Formatierroutine.
Nach dem Ladevorgang antwortet der Monitor mit der Frage:
EXECUTE?:
Ein Y führt zum Start des Programms ab Adr. 8000H; ein beliebiges anderes Zeichen zur Rückkehr in den Monitor.

WRITE — Beschreibung eines Sektors (oder mehrerer Sektoren) der Diskette. Der entsprechende Vektor liegt ebenfalls im RAM-Bereich und zwar ab Adresse XX9DH.
Die Voreinstellung des Vektors führt zum Beschreiben von 10 Sektoren ab Sektor 13 von Spur 00H aus einem Speicherbereich ab 8000H.
Auch hier antwortet der Monitor zunächst mit der Frage
EXECUTE?:
Erst durch Eingabe eines Y erfolgt der eigentliche Schreibvorgang.

XECUTE — Führt zur nochmaligen Ausfühfung des zuletzt geladenen Programms. War kein Programm geladen, erfolgt die Rückmeldung
NO PROGRAM LOADED

* Alle Kommandos können bis auf die fett gedruckten Buchstaben abgekürzt werden.

2.4 Floppy Disk Treiber

Der Aufruf der Floppy Disk Treibersoftware erfolgt mit dem Indexregister IY als Parameter-Vektor. Dieser Vektor zeigt auf einen 13 Byte großen Variablenblock, dessen Lage innerhalb des Adreßbereichs der Z80-CPU beliebig ist. Alle für das Treiberprogramm erforderlichen Informationen gehen aus diesem Variablensatz hervor. Im einzelnen sind es folgende Größen:

(IY+0)	unbenutzt
(IY+1)	Operationscode
(IY+2)	Datenübertragungsadresse (LSB)
(IY+3)	Datenübertragungsadresse (MSB)
(IY+4)	Datenlänge (LSB)
(IY+5)	Datenlänge (MSB)
(IY+6)	Rückkehradresse (LSB)
(IY+7)	Rückkehradresse (MSB)
(IY+8)	Rückkehradresse bei Fehlerbedingungen (LSB)
(IY+9)	Rückkehradresse bei Fehlerbedingungen (MSB)
(IY+A)	Beendigungscode
(IY+B)	Sektor-/Laufwerkadresse
(IY+C)	Spuradresse

Es folgt die Beschreibung der jeweils möglichen Parameter und deren Bedeutung.

a) Operationscode (engl. Requestcode)
Sagt dem Treiberprogramm, welche Operation ausgeführt werden soll. Da nur „Schreiben" oder „Lesen" möglich ist, gibt es entsprechend auch nur zwei verschiedene Operationscodes, nämlich:

- WRTBIN = 0EH (Write Binary = Daten zur FD)
- RDBIN = 0AH (Read Binary = Daten von FD).

Hierbei kann durch Bit 0 des Requestcodes festgelegt werden, zu welchem Zeitpunkt nach dem Aufruf des Treiberprogramms der Rücksprung zum Aufrufer erfolgt. Da das Treiberprogramm voll unter Interrupt abläuft, ist der Rücksprung sowohl nach der vollständigen Ausführung einer Operation Bit 0 = 0), als auch bereits nach der ersten Interruptinitialisierung (Bit 0 = I) möglich.
Wird der zweite Fall gewählt (WRTBIN = 0FH, RDBIN = 0BH), so muß der Anwender sicherstellen, daß nach der Rückkehr vom Aufruf der FD-Treiberroutine weder der Interrupt der CPU gesperrt (Befehl: DI), noch das I-Register der CPU verändert wird. Selbstverständlich darf auch die Programmierung des interruptgenerierenden Bausteins Z80-CTC auf der ECB/F nicht modifiziert werden.

b) Datentransferadresse (engl. Data Transfer Address)
Adresse des Pufferspeichers, in dem die zu übertragenden Daten stehen (WRTBIN) oder abgelegt werden (RDBIN).

c) Datenlänge (engl. Data Length)
Anzahl der zu übertragenden Bytes. Da die kleinste adressierbare Einheit auf der Floppy Disk der Sektor mit seinem 128 Byte großen Datenblock ist, können immer nur ganzzahlige Vielfache der Blockgröße gelesen bzw. geschrieben werden.
Ist hier ein Wert ungleich einem ganzzahligen Vielfachen der Blockgröße angegeben, so erfolgt eine entsprechende Aufrundung. Nach Beendigung einer Operation enthalten diese beiden Speicherstellen die Anzahl der tatsächlich übertragenen Bytes (immer Vielfache der Blockgröße).

d) Rückkehradresse bei Beendigung (engl. Completion Return Address)
Programmadresse, zu der der Rücksprung nach Beendigung einer Operation erfolgt. Dieser Mechanismus tritt nur in Kraft, wenn BIT 0 des Requestcodes gesetzt ist.

Z80-MTX

KONTRON
ELEKTRONIK GMBH

Echtzeit-Multitask-Betriebsprogramm

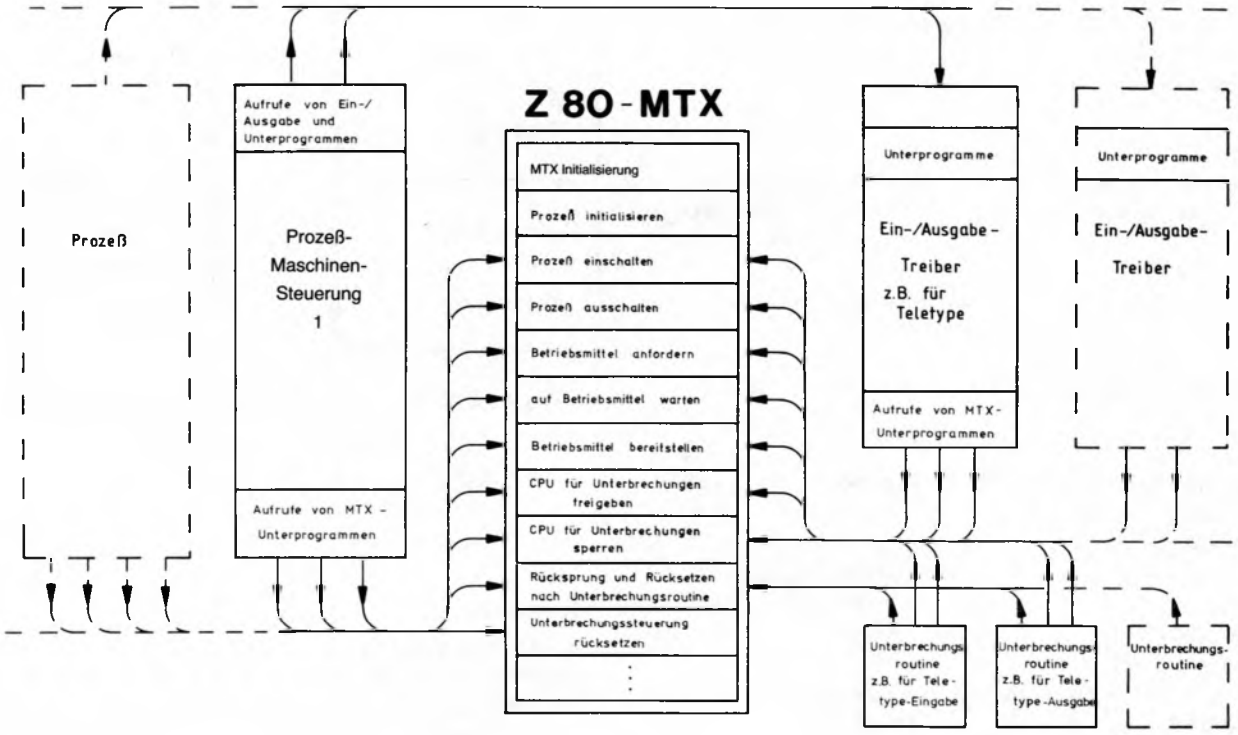


Bild 1

7.5. GRUNDSOFTWARE FÜR DIE ECB-SERIEN

3. Echtzeit-Multitasks-Betriebsprogramm

Für die ständig erweiterte Z80-ECB-Baugruppen-Serie ist jetzt ein Betriebssystem lieferbar, das auf der besonders leistungsfähigen Interrupt-Architektur des Z80-Konzepts basiert und sie ausnutzt.
Z80-MTX ist ein Betriebssystem zur Verwaltung einer großen Anzahl voneinander abhängiger und asynchron verlaufender Vorgänge. Jeder Vorgang (= „Prozeß") kann Ein/Ausgabe-Aktivitäten, Rechnungen und logische Entscheidundungen enthalten.
Das Betriebssystem überwacht den Ablauf der durch Unterbrechungen (Interrupts) gesteuerten Ein- und Ausgaben und weist den Prozessor nach einer Prioritätenliste Rechenzeit zu. Dadurch können z.B. Ein/Ausgabe-bedingte Wartezeiten eines bestimmten Prozesses zur Fortführung anderer Prozesse genutzt werden.
Ebenso koordiniert Z80-MTX logische Verknüpfungen der Prozesse untereinander.

Das Betriebssystem hat folgende hervorstechende Eigenschaften:

- Multitask
  Mehrere Aufgaben, die asynchron anfallen und per Interrupt aufgerufen werden, werden unter Z80-MTX bearbeitet.
- Erweiterbar
  Jede Benutzerroutine kann in einfacher Weise an das Betriebssystem angeschlossen werden.
- Priorisierbar
  Die Rangordnung der Aufgaben wird durch den Benutzer festgelegt.
- Eindeutigkeit
  Sämtliche Benutzer- und Prozeß-Parameter sind in definierbaren Tabellen festgelegt.
- Hohe Geschwindigkeit
  Standardisierte Interrupt-Service-Routinen führen lediglich Eintragung in Prozeßtabellen aus. Die Verarbeitungsgeschwindigkeit kann daher sehr groß sein.
- Standards
  Fertige Ein/Ausgabe-Routinen, wie Terminals-Drücker, Printer-Driver, Floppy-Disk-Driver, sind jetzt oder in Kürze verfügbar.
- Kompakt
  Z80-MTX benötigt durch den leistungsfähigen Befehlssatz der Z80-CPU einen wesentlich geringeren Programm-Speicherbereich als vergleichbare Programme, die auf anderen Prozessoren basieren. Das gesamte Betriebssystem findet in 1 kByte Platz.

Das Betriebssystem ist auf Diskette oder als Firmware (= PROM-resident) lieferbar.

3.1 Allgemeines

Z80-MTX ist ein Multitask-Echtzeit-Betriebssystem, das speziell für Mikroprozessor- und Minicomputer-Anwendungen entwickelt wurde. Multitask-Manager werden sonst im allgemeinen nur für Großrechner und größere Prozeßrechner angeboten. Hier dagegen wurde ein System geschaffen, das auch zur Ablaufsteuerung in kleinen Systemen geeignet ist.

MTX beschränkt sich auf wichtige Funktionen und ist vor allem benutzerfreundlich konzipiert (nur einige wenige Systemaufrufe müssen vom Benutzer beherrscht werden). Sie sind an Hand dieser Beschreibung leicht zu erlernen.
Oft ist der Wunsch nach kurzen Reaktionszeiten ein entscheidender Grund für die Anwendung eines Mikroprozessors. Daher wurde MTX vor allem zeitoptimiert. Nur etwa 0,2 Millisekunden (bei 4 MHz Takt) werden für die längste Systemaktion benötigt, wobei der Durchschnitt wesentlich niedriger liegt. Dennoch ist der Speicherbedarf – ½ KByte für Programm und ¼ KByte für Systemtabellen – äußerst gering.

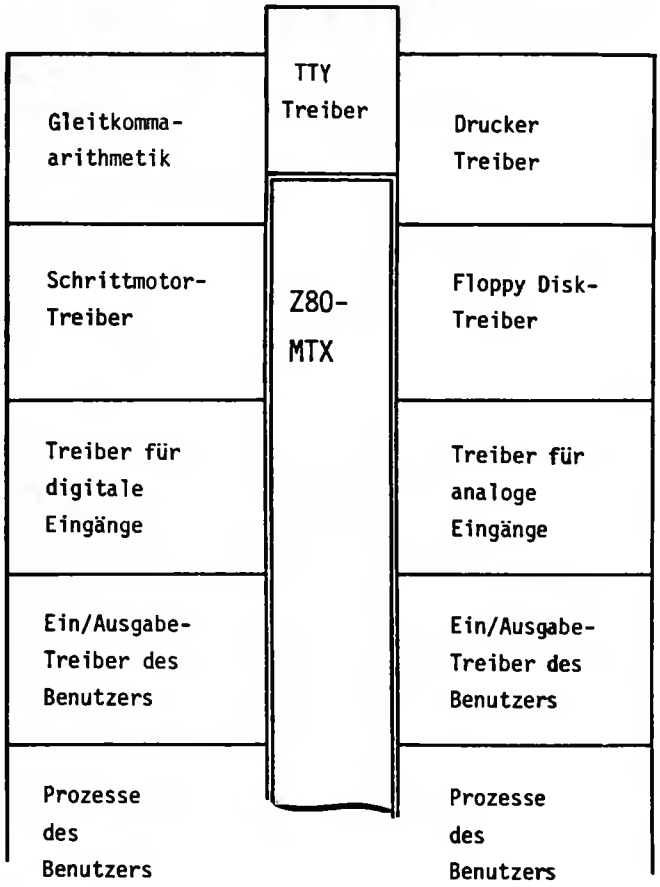


Bild 2

3.2 Aufgabe

Das Multitask-Echtzeit-Betriebssystem Z80-MTX (Multi-Task-Executive) ermöglicht die simultane Abarbeitung mehrerer Teilaufträge unter Berücksichtigung ihrer Priorität und Rechenwilligkeit. Diese Teilaufträge werden im folgenden Prozesse genannt; es kann sich dabei um Benutzer- oder Systemprogramme handeln. Der Systemprogrammierer, d.h. der Benutzer von MTX, muß seine Gesamt-Aufgabe in geeignete Prozesse untergliedern und die Abhängigkeiten der Prozesse untereinander und von Ein/Ausgabe-Aktivitäten analysieren. Entsprechend diesen logischen Abhängigkeiten sind Betriebsmittel vorzusehen. (Der Begriff „Prozeß" wird in Abschnitt 3.4.2. näher erläutert, der Begriff „Betriebsmittel" in 3.4.2.5).
Zur Definition der Prozeß- und Betriebsmittelkonfiguration fügt der Benutzer standardisierte Unterprogramm-Aufrufe für MTX-Routinen im Systemsteuerungsteil ein (siehe auch Bild 1).

Durch MTX werden in optimaler Weise die I/O-Wartezeiten von Prozessen hoher Priorität genutzt, um Prozesse niedriger Priorität zu bearbeiten. MTX regelt außerdem die Kommunikation und Synchronisation der Prozesse untereinander. Es ermöglicht sogar die gleichzeitige Benutzung eines oder mehrerer Programme durch mehrere Benutzer. Durch die Benutzung von MTX ist es also gegenüber herkömmlicher Interrupt-organisierter Programmierung möglich,

1. die Ausführungszeit von Benutzeraufträgen wesentlich zu verkürzen.
2. die Programmierung entschieden zu vereinfachen,
3. verschiedenste Aufgaben gleichzeitig zu erledigen, z.B. mehrere Benutzer auf derselben Anlage gleichzeitig rechnen zu lassen.

In Z80-MTX (REV 2.0) implementierte Funktionen:

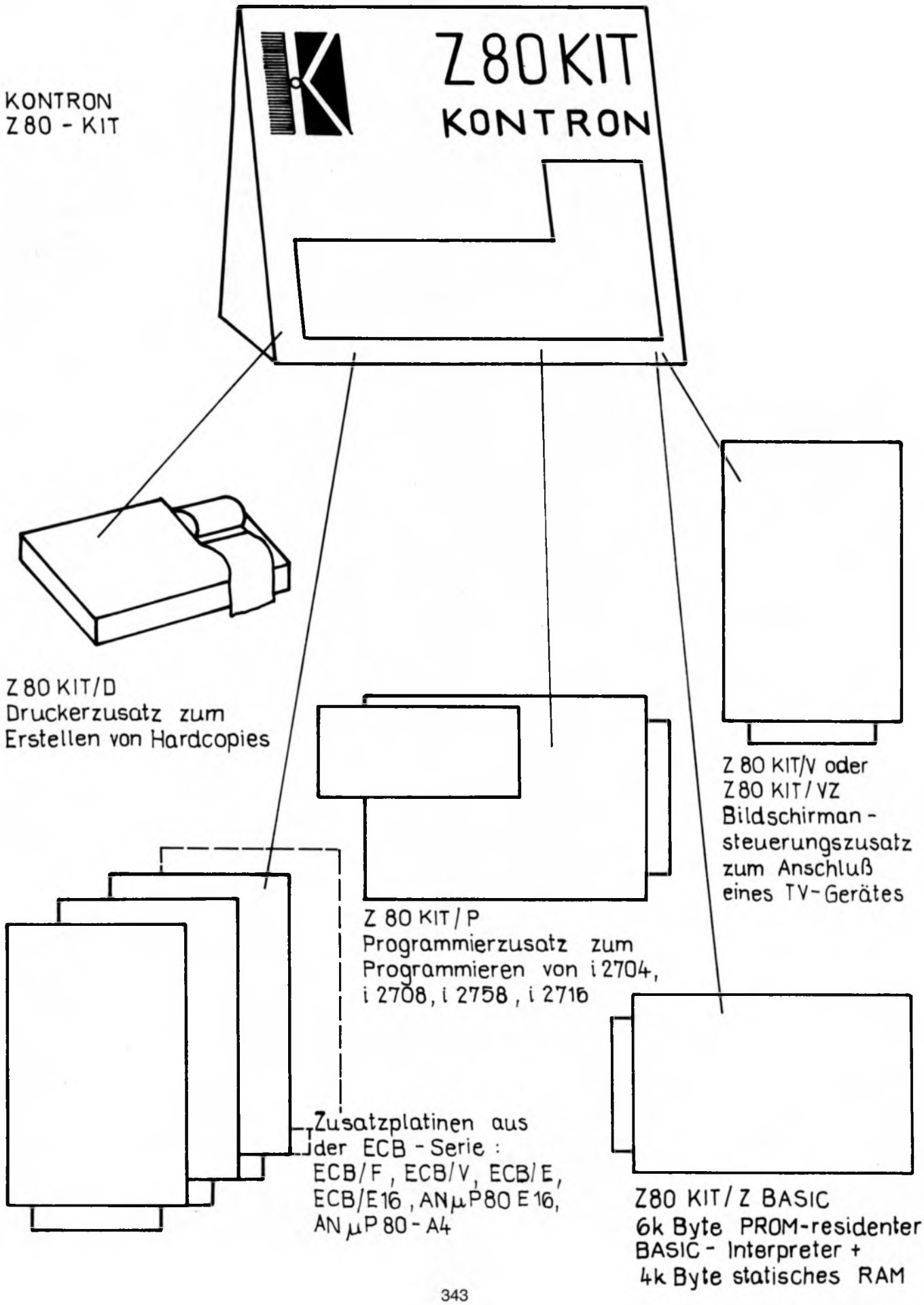
1. CLSYS
   Funktion: Setzt alle Systemparameter zurück. Der Aufruf muß vor der Prozeßinitialisierungen (CALL PROIN) erfolgen.
2. PROIN
   Funktion: Initialisiert den Prozeß N. N ergibt sich aus dem N-ten Aufruf von PROIN. N ist Prozeß-Nummer und Priorität (1 = höchste Priorität). Die Initialisierungsaufrufe müssen unmittelbar aufeinanderfolgen, ihre Reihenfolge legt die Prozeß-Nr. und damit ihre Priorität fest. Prozesse gleicher Priorität sind somit nicht möglich.
3. PROEIN
   Funktion: Der Prozeß N wird eingeschaltet, d.h. rechenwillig gesetzt, jedoch nicht sofort gestartet. MTX startet diesen Prozeß erst dann bei der angegebenen Startadresse, wenn Rechenzeit verfügbar ist und der Prozeß priorisiert ist. Der zweite Registersatz (gestrichene Register) wird an den eingeschalteten Prozeß übergeben.
4. UBRES
   Funktion: Soft Unterbrechung.
   Dieser Sprung ins System ist an zwei Stellen zwingend:
   1. Am Ende der Initialisierung und
   2. hinter CALL PROEIN, wenn einzuschaltender und aufzurufender Prozeß gleich sind.
5. PROAUS
   Funktion: Der Prozeß N wird ausgeschaltet. Er kann erneut und evtl. mit veränderter Startadresse wieder eingeschaltet werden (CALL PROEIN).
   Erfolgt der Aufruf „CALL PROAUS“ in dem ihn betreffenden Prozeß — was die Regel ist — so muß „CALL UBRE“ folgen.
6. PRO?
   Funktion: MTX informiert den Benutzer ob der Prozeß N
   - eingeschaltet oder
   - ausgeschaltet
   - rechenwillig oder
   - im Wartezustand ist.
8. UBRE
   Funktion: Es wird eine Soft-Unterbrechung erzeugt. MTX überprüft, welcher Prozeß fortgesetzt oder gestartet wird.
9. EINT
   Funktion: Der Interrupt wird „ENABLED“ — aber nur für diesen speziellen Prozeß. Ein Aufruf innerhalb einer Unterbrechungsbehandlung (Interrupt-Service-Routine) ist verboten. Die Befehle „EI“ und „DI“ dürfen vom Benutzer nicht gegeben werden.
10. DINT
    Funktion: Der Interrupt wird „DISABLED“, aber nur für diesen speziellen Prozeß. Alles weitere wie „EINT“.
11. ANBM:
    Funktion: Dem System wird mitgeteilt, daß ein Betriebsmittel angefordert wird. Hiernach ist es möglich, auf die Bereitstellung (durch CALL BEBM) des Betriebsmittels zu warten (mit CALL WABM).
    Wird ein Betriebsmittel angefordert, muß auch zu einem späteren Zeitpunkt seine Bereitstellung erfolgen.
12. BEBM
    Funktion: Dem System wird mitgeteilt, daß ein Betriebsmittel bereitgestellt ist.
13. WABM
    Funktion: Der aufrufende Prozeß wird unterbrochen, bis das angeforderte Betriebsmittel bereitgestellt ist. Wurde das Betriebsmittel nicht angefordert (CALL ANBM) oder bereits bereitgestellt (CALL BEBM), wird der Prozeß sofort weitergeführt. Nur ein einziger Prozeß darf auf das gleiche Betriebsmittel zur gleichen Zeit warten.
14. BM?
    Funktion: MTX teilt dem Benutzer mit, ob ein Betriebsmittel:
    - angefordert
    - nicht angefordert
    - bereitgestellt ist.
15. PUREG
    Funktion: Dieser Aufruf muß der erste Befehl in einer Unterbrechungsbehandlung (auch Interrupt-Service-Routine) sein. Das System rettet alle Register (beide Registersätze) und den Status des unterbrochenen Prozesses.
16. RETIN
    Funktion: Dieser Sprung ins System ersetzt alle RETI-Befehle (RETI ist verboten). Er ist der Befehl einer jeden Unterbrechungsbehandlung (auch Interrupt-Service-Routine). In einer Unterbrechungsbehandlung darf der Interrupt-Status auf CPU-Ebene nicht eingeschaltet werden („EI“ verboten).
17. INTRE
    Funktion: Häufig kommt es vor, daß nach dem Einschalten als auch durch Programmier- oder Bedienungsfehler CTC's oder PIO's blokkiert sind, weil sie nach einem Interrupt keinen „RETI“-Befehl für den MTX-Benutzer verboten ist, wurde durch „INTRE“ die Möglichkeit geschaffen, durch wiederholte systemverträgliche Ausführung des „RETI“-Befehls die Interrupt-Hardware von PIO's und CTC's zurückzusetzen. Es bleibt Aufgabe des Benützers, die Kontrollworte für CTC's und PIO's richtig zu laden.
    Dabei ist es sinnvoll, die Interrupt-aktiven Schaltungen auf eine Unterbrechungsbehandlung zeigen zu lassen (über den Interrupt-Vektor), die keine Aktion auslöst.
    „INTRE“ erzeugt Soft-Unterbrechungen. Es gilt also entsprechendes wie bei „UBRE“.

KONTRON
ELEKTRONIK GMBH

8. KONTRON-Z 80-KIT-EINFACHCOMPUTER-UND LERNSYSTEM

8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

HARDWARE ÜBERSICHT



Z80-KIT Einfach-Computer und Lernsystem

Der Z80-KIT ist ein voll funktionsfähiges Einfach-Rechner-System, das auf der Basis des Mikroprozessors Z80 von ZILOG aufgebaut ist. Es besteht aus einer autonomen Rechnereinheit, einer Ein-Ausgabemöglichkeit zur Bedienung des Gerätes, sowie einem Betriebsprogramm, das die Aktivitäten des Z80-KIT ermöglicht.

Der Z80-KIT arbeitet auf Maschinen-Code-Ebene, d.h. die Ein- und Ausgabe von Werten wird in sedezimaler (hexadezimaler) Form durchgeführt.

Eingabemöglichkeiten bestehen in Form der Sedezimaltastatur (0...F) für einzugebende Werte, sowie der Kommandotastatur und deren Hilfe direkte Anweisungen an das System gegeben werden können.

Vom Betriebsprogramm ausführbare Anweisungen sind

- Register auf einen Wert setzen
- Registerinhalt anzeigen
- Speicherzelle auf einen Wert setzen
- Speicherzelleninhalt anzeigen
- Blockeingabe
- Blockausgabe
- Abspeichern von Speicherinhalten auf Cassette
- Laden von auf Cassette gespeicherter Information in den Speicher
- Starten eines Anwenderprogramms
- Einzelbefehlausführung
- Setzen, Anzeigen und Löschen eines Haltepunkts (Breakpoint)

Die Ausgabe von Werten erfolgt auf einer insgesamt sechsstelligen sedezimalen Anzeige.

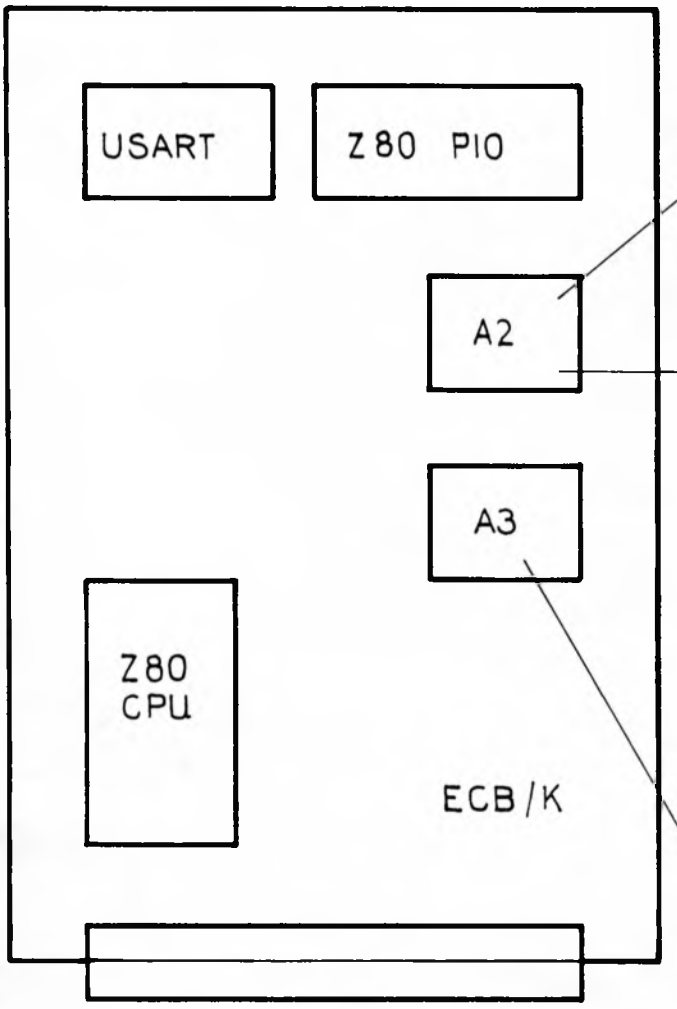
Bei Ein- und Ausgabe ist jeweils das Betriebsprogramm für die Durchführung verantwortlich. Es liest, dekodiert und führt die gegebenen Anweisungen aus. Gleichzeitig bringt es aktuelle Werte zur Anzeige.

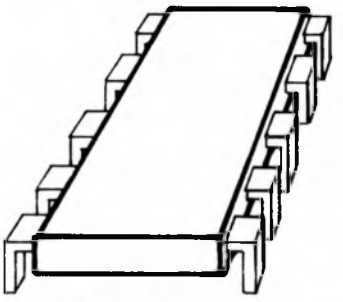
Der Z80-KIT stellt somit ein vollständiges Mikroprozessorsystem dar, das die Möglichkeit bietet, eigene Programme auf hexadezimaler Basis einzugeben, auszutesten und zur Ausführung zu bringen. Dadurch bietet sich ein Einsatz als Entwicklungsgerät an.

Gleichzeitig kann er als Schulungsgerät für Anwender gesehen werden, die sich in die Thematic der Mikroprozessoren einarbeiten wollen. Durch die maschinennahe Arbeitsweise ist es möglich einen grundlegenden Einblick zu erlangen.

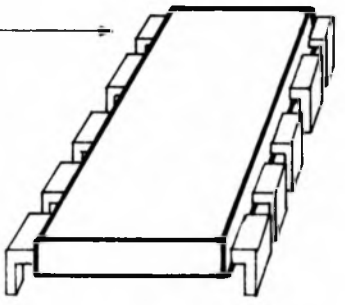
Darüber hinaus kann der Z80-KIT auch zu Anwendungszwecken aller Art herangezogen werden. Von der einfachen Uhrenwerkshaltung über die Steuerung einer Modelleisenbahn bis zur kompletten Hausalarmanlage ist eine Vielzahl von Möglichkeiten denkbar. Unterstützt wird dies durch die beispielhafte Erweiterungsfähigkeit des Z80-KIT und der Vielfalt der angebotenen Erweiterungspakete.

SOFTWARE ÜBERSICHT

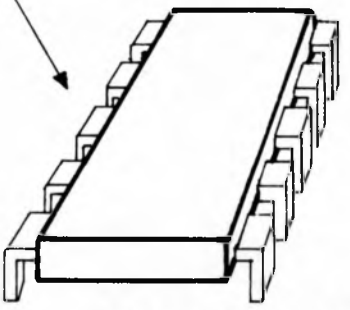




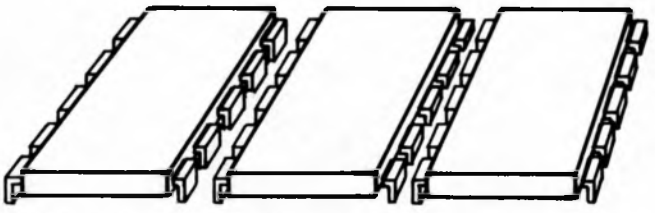
Standardbetriebsprogramm
1k Byte, PROM-resident.
(standardmäßig enthalten)



Z80-KIT /TV
2 k Byte, PROM-resident.
Erweitertes Betriebsprogramm
mit zusätzlicher Ansteuerung
eines TV-Gerätes über
Z80 KIT/V bzw. Z80 KIT/VZ.
(gegen das Standardbetriebs-
programm austauschbar)



Z80 KIT/PDT
2 k Byte, PROM-resident.
Betriebssoftware zu KIT/P und
KIT/D sowie TTY Ansteuer-
routinen.



Z80-KIT / Z BASIC
6 k Byte PROM-residenter
BASIC - Interpreter

KONTRON
ELEKTRONIK GMBH

8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

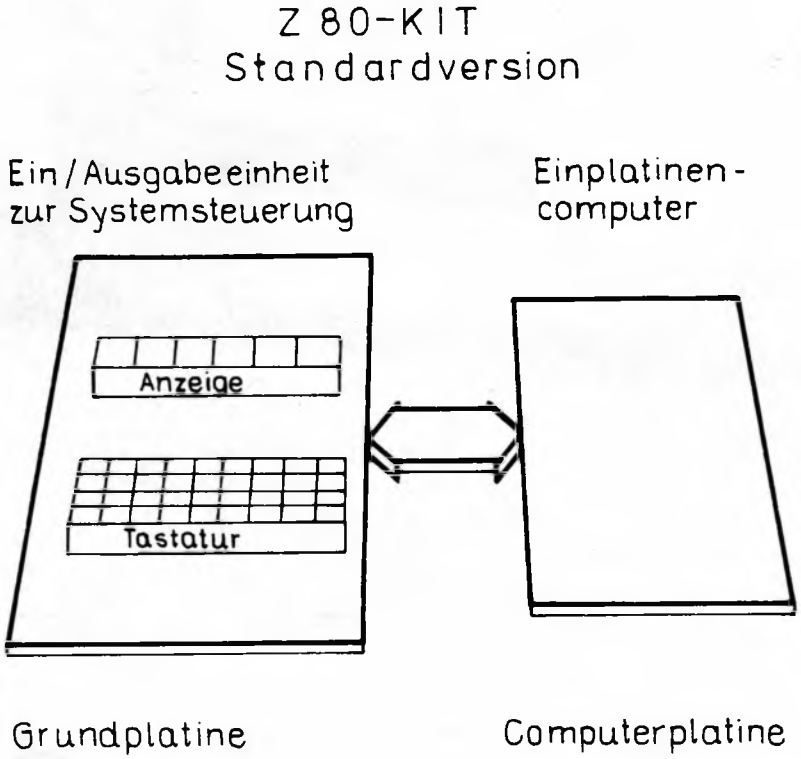
1. Einführung

1.1. Vorbemerkung

Der Z80 KIT ist ein vollständiges Z80-Mikrocomputer-System in Bausatzform, das sich in wenigen Stunden zu einem voll funktionsfähigen System zusammenbauen läßt. Es eignet sich sowohl in hervorragender Weise als Starthilfe bei der Einarbeitung in die Mikrocomputer-Technik als auch als Prototyp für Geräteentwicklungen. Dies umso mehr, da durch umfassende Erweiterungsmöglichkeiten eine Anpassung an den jeweiligen Fortbildungsgrad bzw. an entsprechende Aufgaben jederzeit möglich ist.
Zur Inbetriebnahme zusätzlich notwendig ist lediglich eine 5 V Stromversorgung (±5% Stabilität).
Die Bauanleitung ist vor Beginn des Zusammenbaus genau durchzulesen und dann beim Aufbau des Systems in der vorgegebenen Reihenfolge anzuwenden.

1.2. Grundsätzlicher Aufbau

Mit dem Z80 KIT wurde ein Mikrocomputersystem geschaffen, das in seiner Grundform keinerlei periphere Geräte zu Bedienungszwecken benötigt und dadurch die Verwendung eines funktionsfähigen Mikrocomputers bei minimalen Kosten erlaubt.
Basis dieses Mikrocomputers ist die Grundplatine, auf dem die Bedienungststatur die Anzeigeeinheiten und die dazu notwendige Steuerelektronik untergebracht sind. Ebenfalls auf dem Motherboard wurde ein System-Bus realisiert, der sämtliche Adress-, Daten und Steuersignale über fünf Vielfachsteckverbindungen zur Verfügung stellt. Über einen der genannten Steckplätze erfolgt der Anschluß eines Einplatinencomputers. Erweiterungsmöglichkeiten bezüglich Speicher etc. sind durch die restlichen Bus-Anschlüsse gegeben.



1.3. Erweiterungsmöglichkeiten

Der Z80-KIT läßt sich aufgrund seiner Architektur und seines elektronischen Aufbaus auf insgesamt fünf Platinen erweitern, wobei die vier Zusatzplatinen nach Wunsch gewählt werden können.

Angeboten werden folgende Erweiterungen:

1.3.1. Z80-KIT-Erweiterungspakete

- Z80-KIT/P — Programmier- und Peripherie-Baugruppe zur Programmierung von Festwert-Speicherbausteinen, incl. 4 kByte statische Speichererweiterung (auf 8 kByte ausbaubar) und Ein/Ausgabeerweiterung (um 32 I/O-Leitungen) (Bausatz). (Software auf Cassette).
- Z80-KIT/PZ — wie Z80-KIT/P, jedoch geprüft und zusammengebaut (Software auf Cassette)
- Z80-KIT/D — Drucker-Zusatz zum Z80-KIT zur Anfertigung von Hardcopies von Speicherinhalten. (Software auf Cassette).
- Z80-KIT/V — Bildschirmansteuerungszusatz für Z80-KIT (Anschluß an PIO, ECB/K)
- Z80-KIT/VZ — wie Z80-KIT/V, jedoch geprüft und zusammengebaut, (ECB-Bus kompatible Steckkarte)
- Z80-KIT/PDT — PROM-residente Software für KIT/P und KIT/D sowie zur Ansteuerung der TTY-Schnittstelle der ECB/K
- Z80-KIT/TV — PROM-residenter, erweiterter KIT-Monitor zur zusätzlichen Ansteuerung eines TV-Gerätes (Voraussetzung: KIT/V oder KIT/VZ)
- Z80-KIT/ZBASIC — PROM-residenter Basic-Interpreter incl. 4 kByte statische RAM-Erweiterung.

1.3.2. Baugruppen der ECB-Serie

(Ausnahme: dynamische Speicherkarten)

- Z80-ECB/E — Erweiterungsplatine 1 kB stat. RAM + Sockel für max. 8 kB PROM (2708)
- Z80-ECB/E16 — Erweiterungsplatine 1 kB stat. RAM (auf 4 kB nachrüstbar und Sockel für max. 16 kB Festwertspeicher (i2758, i2716)
- Z80-ECB/I — Ein/Ausgabeplatine mit 4×8 bit-Ports und 4 Zähler/Zeitgeber-Kanälen
- Z80-ECB/V — 4 kByte-CMOS-RAM, nicht flüchtig mit Batterie und Speicherschutzschalter
- Z80-ECB/F — Floppy-Disk-Controller und allgemeine Serien- und Parallel-Ein/Ausgabe
- AN-μP80-E 16 — Platine mit 16 analogen Eingängen (bzw. 8 Gegentakt) und 1 analogen Ausgang
- AN-μP80-A4 — Platine mit 4 analogen Ausgängen und Spannungswandler auf der Platine

1.3.3. Stromversorgungen

Alternativ zum Betrieb mit einer externen Quelle, kann der Z80-KIT auch mit einem, direkt am Bus ansteckbaren Stromversorgungsmodul ausgerüstet werden. Das Modul ist im Europakartenformat aufgebaut und kann in einem, extra dafür vorgesehenen Steckplatz (N) eingesteckt werden. (Steckplatz 1 und Steckplatz N können nur alternativ bestückt werden).

NB.:
Die Stromaufnahme des Z80-KIT beträgt in der Grundausstattung ca. 1A. Bei den Erweiterungen kann ein Bedarf von ca. 0,6 bis 0,8 A pro zusätzlicher Platine zugrunde gelegt werden.

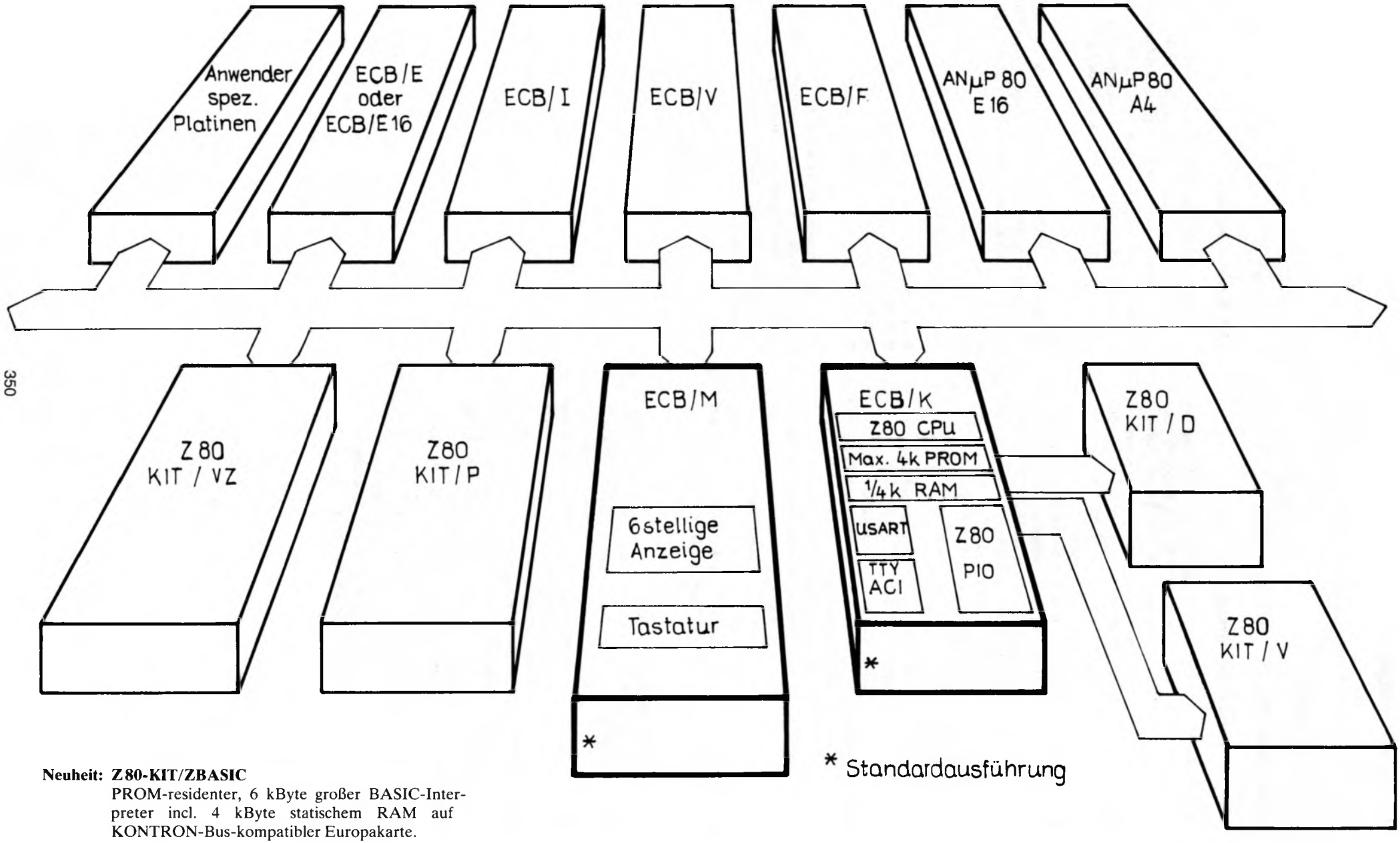
Angeboten werden folgende Stromversorgungskarten:

- KPS 535/1 — 5 Volt/3,5 A
- KPS 535/2 — 5 Volt/3,5 A, ±15 V/0,25 A
- KPS 535/3 — 5 Volt/3,5 A, ±12 V/0,3 A
- KPS 535/4 — 5 Volt/3,5 A, +12 V/0,3 A/−5 V/0,7 A.

1.3.4. Anwenderspezifische Platinen

Selbstverständlich ist auch der Anschluß von Platinen, die vom Anwender aufgebaut werden möglich. Dabei ist lediglich darauf zu achten, daß die elektrischen Grenzwerte der ECB/K nicht überschritten werden.

Erweiterungsübersicht



Neuheit: Z80-KIT/ZBASIC
PROM-residenter, 6 kByte großer BASIC-Interpreter incl. 4 kByte statischem RAM auf KONTRON-Bus-kompatibler Europakarte.

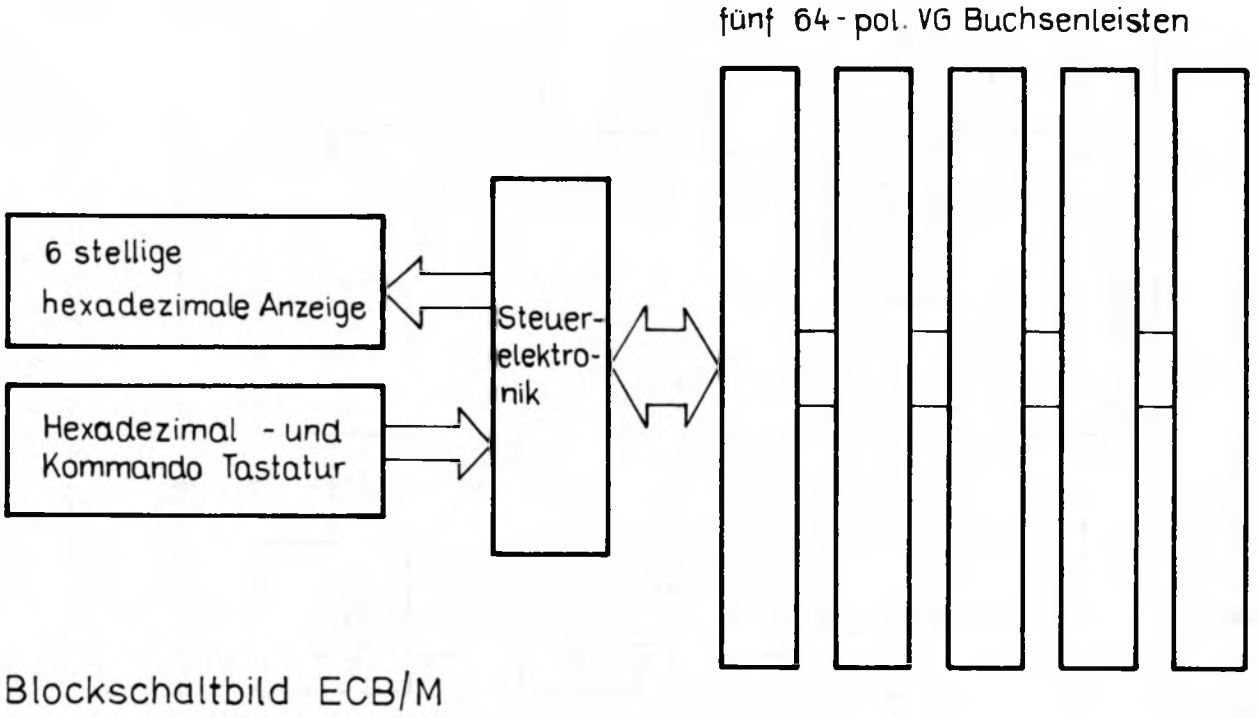
2. Schaltungsbeschreibung des Z80-KIT

2.1. Hardwareausstattung

Die serienmäßige Grundausstattung (Lieferumfang des Z80-KIT) umfaßt folgende Systemkomponenten:

2.1.1. Grundplatine ECB/M

- Eingabetastatur: 26 Tasten zur Bedienung des KIT
- Anzeigeeinheit: 4 hexadezimale Anzeigen für Adressen und 2 hexadezimale Anzeigen für Daten

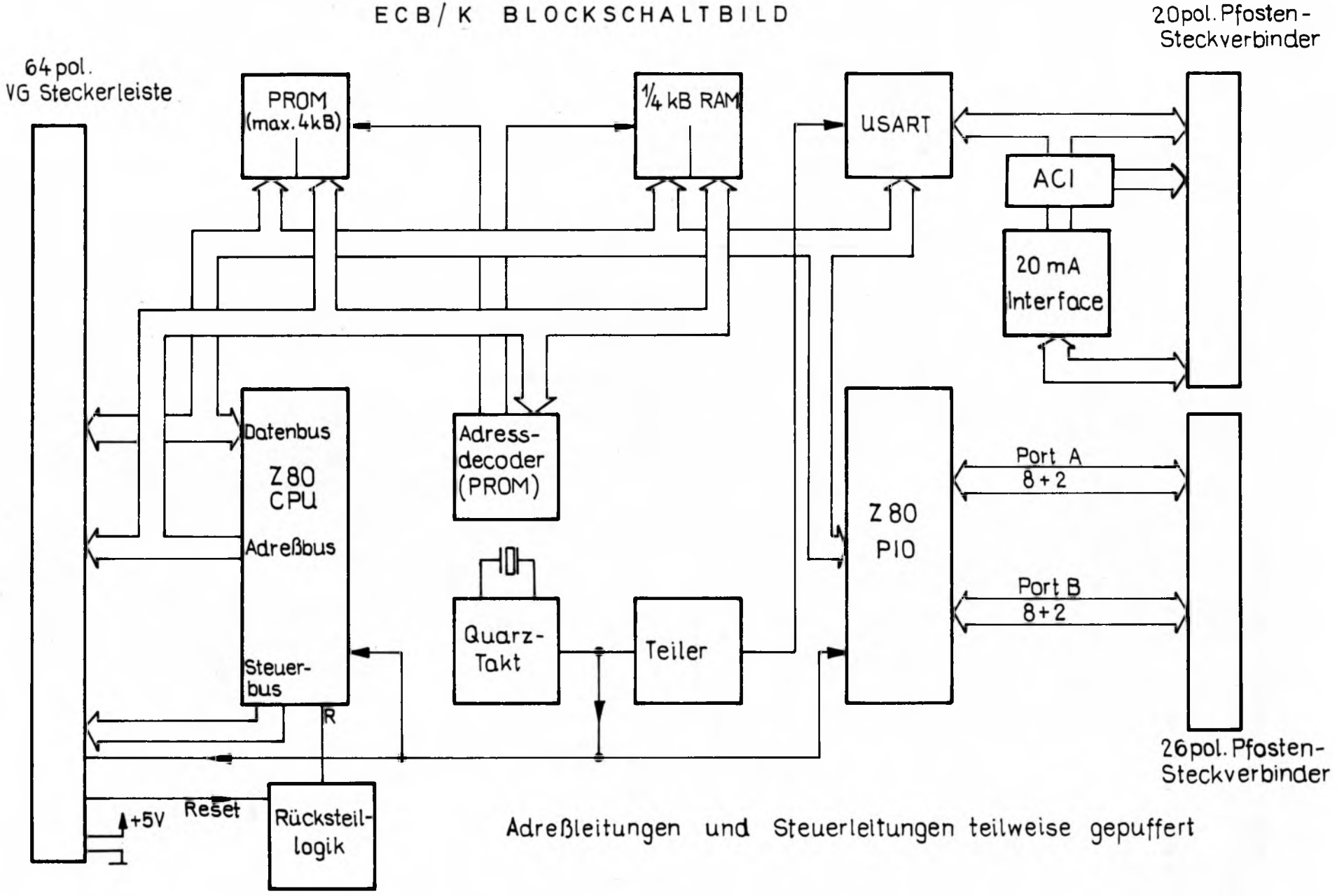


Blockschaltbild ECB/M

2.1.2. Einplatinencomputer ECB/K

- Mikroprozessor: 1 × Z80-CPU/PS
- Taktfrequenz: 2,5 MHz
- Programmspeicher: Max. 4 kByte, 1 kByte Betriebsprogramm serienmäßig mitgeliefert
- Schreib/Lesespeicher: 2 × 2112/A, Statische RAM's (= ¼ kByte)
- Ein/Ausgabeeinheiten: 1 × Z80-PIO 2 × 8 bit Programmierbare Parallelschnittstelle
  1 × 8251 oder 9551 USART
  Programmierbare Serien-Schnittstelle mit Audio-Cassetten-Interface (ACI) und vorbereiteter TTY-Schnittstelle

ECB/K BLOCKSCHALTBILD



Adreßleitungen und Steuerleitungen teilweise gepuffert

2.2. Funktionsbeschreibung

2.2.1. Grundplatine ECB/M

2.2.1.1. Das Tastenfeld

Im Sinne minimaler Hardwarekosten wurde von zwei Voraussetzungen ausgegangen:

- Verwendung billiger Standard-Drucktasten mit einem einzigen Arbeitskontakt
- Vermeidung des Einsatzes eigener Ein/Ausgabe-Bausteine und Mitbenützung der Adreßleitungen zu Ein-/Ausgabezwecken.

Ein Tastenfeld von max. 32 Tasten ist in 4 Reihen und 8 Spalten angeordnet. Die Reihen (= Eingänge) sind durch die, in einem Decoder decodierten Adressen gebildet. Die Spalten (= Ausgänge) sind an die Eingänge eines 3-State-Buffers angeschlossen. Der Zustand der Tasten wird mit dem Signal TRD (Tasten Read) über den Daten-Bus (D0 bis D7) in die CPU übertragen. Dabei ist TRD = (0F ∨ 0E ∨ 0D ∨ 0C) ∧ IORQ ∧ RD.

2.2.1.2. Die Anzeigeeinheiten

Bei der Konzeption der Low-Cost-Anzeige wurde von den gleichen Voraussetzungen ausgegangen. Die Anzeige umfaßt 6 ungepufferte Standard-7-Segment-LED's, die ohne Zwischenschaltung eines Hexadezimal 7-Segment-Decoders direkt per Tabelle aus dem Anwendungsprogramm angesteuert werden.

Dadurch können beliebige numerische, mit gewissen Restriktionen auch hexadezimale und alphanumerische Anzeigen durch Vorgabe einer geeigneten Tabelle im Anwenderprogramm ausgegeben werden.

Die 6 LED's werden kontinuierlich und zyklisch mit den anzuzeigenden Daten über den Datenbus versorgt. Die Anwahl der LED's erfolgt über einen Adreß-Decoder, der durch eine Stromsenke jeweils 1 LED ansteuert. Die Helligkeit der LED's läßt sich über die Zeitkonstante beeinflussen. Das Zeitglied erzeugt mit der Konstante ein WAIT-Signal, das die Dauer der Ausgabe entsprechend dehnt.

2.2.1.3. Interruptprioritätskette

Die mit den Z80-Bausteinen mögliche Prioritätsverteilung per „Daisy-Chain", kann selbstverständlich auch über mehrere Platinen hinweg aufgebaut werden. Dazu ist es notwendig, jeweils den IEO (Interrupt Enable OUT) einer Karte (PIN 16C-STA) mit dem IEI (Interrupt Enable IN) der nachfolgenden Karte (PIN 11C-STA) zu verbinden.

Die fünf Steckplätze des Z80-KIT sind bereits in dieser Weise miteinander verbunden, wobei Steckplatz 5 die höchste Priorität innehat. Soll ein nicht belegter Steckplatz überbrückt werden, so ist dies mit Hilfe einer vorgesehenen Lötbrücke möglich (siehe Schaltplan ECB/M).

2.2.1.4. Mechanischer Umbau der Grundplatine

Die Grundplatine des Z80-KIT ist so aufgebaut, daß der System-Bus vom Tasten- und Anzeige-Block getrennt werden kann. Dadurch ist der Einbau in einen 19''-Einschub möglich, während die Bedienungs- und Anzeigeeinheit in einem getrennten Gehäuse untergebracht werden kann.

Für eine Trennung ist lediglich ein Schnitt in Höhe der unteren Befestigungslöcher der VG Buchsenleisten nötig. Bei Einbau des BUS in einen 19''-Einschubrahmen muß die Platine auf beiden Seiten, entlang der Masse- bzw. +5 V-Fläche abgesägt werden. Die Befestigung im Einschub erfolgt ausschließlich über die VG Buchsenleisten!

Die elektrische Verbindung der geteilten Platine wird am besten über Flachbandkabel hergestellt. Bei Kabellängen über 20 cm empfiehlt es sich zur Abschirmung zwischen jeder Signalleitung eine Masseleitung zu führen.

2.2.2. Einplatinencomputer ECB/K

Die Baugruppe umfaßt folgende Funktionseinheiten:

- Z80-CPU
- Decoder und Puffer für Speicher- und I/O-Ausbau
- Bis zu 4 kByte PROM
- 256 Byte RAM
- 2 Parallelschnittstellen (1 Stück Z80-PIO)
- 1 Serienschnittstelle mit ACI- und TTY-Interface
- Busseitig 64-poliger Stecker nach DIN 41612 (VG 95324)
- I/O-seitig 3M-WWP-Pfostenverbinder.

Aus den Schaltplänen ist die Verknüpfung der einzelnen Funktionseinheiten zu erkennen. Die Busleitungen sind teilweise gepuffert. Die Adreßeingänge der auf der Z80-ECB/K untergebrachten Speicherbausteine werden über diese Puffer, die Datenleitungen vom CPU-Datenbus direkt angesprochen.

Festwert- und Schreib/Lese-Speicherbereich auf der Platine haben durch den Steuerbaustein CIC III vorgegebene Anfangsadressen (siehe Kap. 2.3.2.1. ff).

2.2.2.1. CPU-RESET

Das Rückstellen der Z80-CPU kann über die zwei Standardgatter sowohl durch Aus/Einschalten der Stromversorgung als auch elektronisch oder elektromechanisch über den Eingang RESET (1 Low-Power-Schottky-Eingangslast) erfolgen.

2.2.2.2. Takterzeugung

Aus Stabilitäts- und Kostengründen wird der Systemtakt über einen 9,8304 MHz-Quarz erzeugt, der zusammen mit 1/3 Standard LS-Baustein einen Oszillator bildet. Der für die Z80-CPU nötige Takt Φ von 2,4576 MHz wird über einen D-FlipFlop-Teiler (2 × ½ 74LS74) gewonnen, von dem auch das 2 Φ -Signal abgeleitet und auf den Bus herausgeführt wird.

2.2.2.3. Programmspeicher

Auf der ECB/K sind zwei PROM-Sockel (A2 und A3) für bis zu 4 kByte PROM vorhanden.
Folgende PROM-Typen können verwendet werden:

a)	HARRIS HM 7641	512 × 8 bit organisiert bipolar Bild 13, Kap. 5
b)	HARRIS HM 7681	1 k × 8 bit organisiert bipolar Bild 13, Kap. 5
c)	2708	1 k × 8 bit organisiert E-PROM. (−5 V, +12 V!), Bild 14, Kap. 5
d)	2758	1 k × 8 bit organisiert (single 5 V E-PROM), Bild 15, Kap. 5
e)	2716/6716/2516	2 k × 8 bit organisiert (single 5 V E-PROM) Bild 15, Kap. 5

Eine Einstellung auf einen der genannten Typen erfolgt durch Setzen der Jumper J1 . . . J4 entsprechend den Bildern 13 bis 15 in Kapitel 5 (Zusammenbau). Beachten Sie bitte, daß ein jeweils passender Steuerbaustein CIC III notwendig ist (siehe 2.2.1.). Die Standardausrüstung enthält einen Steuerbaustein CIC III passend zu 2 kByte umfassenden Festwertspeichern.

2.2.2.4. Schreib/Lese-Speicher

Der 256 Byte große Schreib/Lese-Speicher besteht aus 2 statischen 2112/A-RAM-Bausteinen, die 256 × 4-bitweise organisiert sind.

2.2.2.5. Parallel-Ein/Ausgabe

Für den Datenverkehr zwischen Z80-CPU und der „Außenwelt" über parallele Schnittstellen wurde 1 Baustein Z80-PIO auf der Baugruppe Z80-ECB/K untergebracht. Dieser Baustein umfaßt zwei 8bit-Ports, die als Eingabe-Port, Ausgabe Port, bidirektionales Port oder aber in Einzelbit-Schaltung arbeiten können (Einzelheiten siehe Z80-PIO-Produktspezifikation und Z80-PIO-Technical Manual). Beide Ports verfügen über eine interne Logik, die ohne zusätzliche Hardware Quittungs- (="Handshaking")-Betrieb (2 eigene Leitungen pro Port) und priorisierten Vektor-Interrupt ermöglicht. Die die Priorisierung festlegenden Signale IEI (="Interrupt Enable In") und IEO (="Interrupt Enable Out") sind zur Verkettung mit weiteren Ein/Ausgabebausteinen busseitig herausgeführt.

2.2.2.6. Serielle Ein/Ausgabe

Die Serienschnittstelle (1 Duplex-Kanal) ermöglicht den seriellen Datenverkehr zwischen peripherer Elektronik und der Z80-CPU.
Die Serienschnittstelle wurde mit einem Standard-USART-Baustein realisiert, da die Verwendung des Serien-Ein/Ausgabe-Bausteins Z80-SIO, der über 2 Duplex-Kanäle zur Arbeit mit SDLC- und ähnlichen Prozeduren und Logik zur Floppy-Disk-Steuerung verfügt, an dieser Stelle redundant wäre.
Der USART ist übrigens selbst auch interruptfähig, hat aber nicht die Möglichkeit zur automatischen Interrupt-Vektor-Behandlung, die alle Zilog Z80-Ein/Ausgabe-Bausteine aufweisen. Soll die Serienschnittstelle in Vektor-Interrupt-Betriebsart gefahren werden, müßte deshalb eine zusätzliche Logik zur Interrupt-Vektor-Erzeugung aufgebaut werden.
Angeschlossen an diese Serienschnittstelle sind ein Audio Cassetten Interface und ein TTY-Interface. Der Einfachheit halber liegen beide Schaltungen parallel; es ist aber jeweils nur eine von beiden benutzbar!

Das Audio Cassetteninterface

Es besteht aus einer einfachen Modulation/Demodulation. Die Ausgangsschaltung wandelt ein high-Signal in eine Frequenz um, ein low Signal hat keine Wirkung, so daß sich eine Folge von Frequenz/keine Frequenz ergibt. Der Ausgang wird direkt mit dem NF Eingang eines Cassettenrecorders verbunden.
Bei der Eingangsschaltung wird das vom Band gelieferte Signal demoduliert und damit wieder in logische low bzw. high Pegel umgesetzt.

Der Anschluß dieser Schaltung erfolgt an den Lautsprecheranschluß des Cassettengerätes. Beachten Sie dabei den Ausgangswiderstand des Gerätes, der in der Größenordnung 10 Ω liegen muß! Sollte dies nicht der Fall sein, muß R29 entsprechend angepaßt werden. Auf diese Weise ist auch der Anschluß an eine Ohrhörerbuchse möglich.

Das TTY Interface

Beachten Sie bitte, daß es sich hier nur um eine **vorbereitete** TTY Schnittstelle handelt, da im Monitor des KIT keine softwaremäßige Steuerung enthalten ist. Ein entsprechendes Programm kann selbstverständlich jederzeit durch Verwendung von Programmspeichererweiterungen zusätzlich eingebaut werden (siehe Z80-KIT/PDT).

Die Eingangsschaltung der Stromschleifenschnittstelle (TTY-Schnittstelle) besteht aus einem Optokoppler mit nachfolgendem Impedanzwandler und TTL-Puffer. Die Leuchtdiode des Optokopplers kann in zwei verschiedene Arten vom angeschlossenen Terminal angesteuert werden:

- ☐ Liefert das Terminal den Strom für die Stromschleifenschnittstelle, genügt es, die Leuchtdiode in diesen Stromkreis zu schalten.

- ☐ Stellt das Terminal nur einen Kontaktschluß zur Verfügung, wie es üblicherweise bei einem Fernschreiber (z. B. Teletype ASR 33) der Fall ist, ist es notwendig, die Anode der Leuchtdiode über einen Widerstand (150 Ohm) auf +5 Volt zu legen und den Kontakt zwischen Leuchtdioden-Kathode und Masse zu legen.

Der Fototransistor auf der Ausgangsseite des Optokopplers ist als Emitterfolger geschaltet, der einen weiteren Transistor schaltet. Die Basis des Fototransistors ist hochohmig mit dem Emitter verbunden, so daß der Fototransistor beim Ein- und Ausschalten an seinen Anschlüssen keine große Spannungsdifferenzen auszugleichen hat. Dadurch wird der Einfluß interner Transistorkapazitäten ausgeglichen und eine stabile Datenübertragung selbst bei hohen Baudraten gewährleistet.

Der dem Treibertransistor folgende TTL-Puffer 74LS04 ist zusätzlich auf den Pfostensteckverbinder der Serienschnittstelle zur Realisierung einer Spannungsschnittstelle entsprechend RS 232 herausgeführt, wobei allerdings die Normpegel nicht im vollen Umfang eingehalten werden. Dies hat jedoch für Datenübertragungen auf Strecken < 20 m praktisch keine Bedeutung.

3. Beschreibung des Z80-KIT Betriebsprogramms

3.1. Allgemeines

Für den Z80-KIT werden Betriebsprogramme im Umfang von 1 kByte und 2 kByte angeboten. Die 2 kByte Version (siehe KIT/TV) verfügt im Gegensatz zur 1k Version über eine zusätzliche Ansteuerungsmöglichkeit für ein TV-Gerät. Beide Betriebsprogramme werden als Firmware (also in einem Festwertspeicher) geliefert. Das PROM kann direkt in den dafür vorgesehenen Sockel A2 auf der ECB/K gesteckt werden.

Die Adreßraumaufteilung nach 2.3.2.1. weist dem Sockel A2 in der ausgelieferten Version einen Umfang von 2 kByte zu. Bei Verwendung des 1 k Betriebsprogramms bleibt der Adreßraum 0400H bis 07FFH (= 1 kByte) unbelegt.

Zusätzlich zum Betriebsprogramm kann auf dem Sockel A3 ein zusätzlicher Festwertspeicher von 2 kByte Kapazität untergebracht werden. (Siehe hierzu Teil V, KIT/PDT).

3.2. Aufbau des Betriebsprogramms

3.2.1. Unterprogrammstruktur

Der Z80-KIT-Monitor ist weitestgehend in Unterprogrammtechnik geschrieben. Das heißt, daß neben einem Rahmenprogramm mehrere in sich abgeschlossene Routinen existieren, die jeweils vom Rahmenprogramm her aufgerufen werden. Da die Routinen mit dem RETURN-Befehl (RET) abgeschlossen sind, ist ein Aufruf nicht nur vom Rahmenprogramm des Betriebsprogramms, sondern auch vom Anwenderprogramm her möglich. Dies bedeutet neben Platzersparnissen im Speicher eine wesentliche Erleichterung bei der Erstellung von Programmen, da die zur Verfügung stehenden Routinen nicht selbst programmiert werden müssen.

Das 2 kByte Betriebsprogramm unterscheidet sich von der 1 k Version im Wesentlichen durch erweiterte Routinen, wobei die Erweiterung jeweils zur Ansteuerung des TV-Gerätes dient.

3.2.2. Ein/Ausgabe von Werten

Werte die am KIT zur Anzeige kommen, stehen immer in den jeweils zugehörigen LED-Puffern, d.h. in Speicherzellen, auf die die Anzeigeroutine zyklisch zugreift.

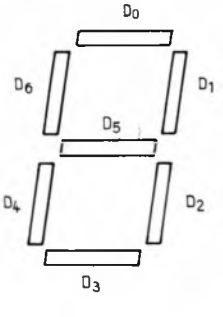
Soll ein Wert, der in einer bestimmten Speicherzelle steht angezeigt werden, so wird eine Routine aufgerufen, die den gewünschten Wert aus der Speicherzelle in die entsprechenden LED-Puffer transferiert.

Umgekehrt existiert eine zweite Routine, die einen, evtl. durch zuvoriges Eintippen in den LED-Puffern stehenden Wert in den Speicher überführt.

Sämtliche Anzeige- und Eingabe-Kommandos verlaufen nach diesem Prinzip.

3.2.3. Ausgabe nicht hexadezimaler Werte

Die Anzeigeroutine selbst verfügt über die Möglichkeit, die in den LED-Puffern stehenden Werte mit oder ohne vorherige Hex-zu-7-Segment-Konversion anzuzeigen (siehe 2.2.1.2.). Dadurch können auch nichthexadezimale Zeichen dargestellt werden. Die Zuordnung zwischen Segment und Daten-bit für nicht hexadezimale Zeichen ist nachfolgend dargestellt.



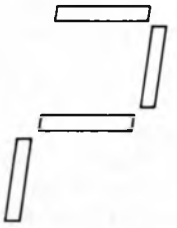
Bei Aufruf der Anzeigeroutine muß das HL Registerpaar die Adresse einer Speicherzelle (hier ‚MARKE' genannt) enthalten, deren Bit 0 entsprechend dem Wunsch, mit oder ohne Umrechnung, gesetzt sein muß.

Bit 0 = 1 → Anzeige mit
Hex-zu-7 Segment Konversion

Bit 0 = 0 → Anzeige ohne Konversion

Beispiel für die Anzeige eines nichthexadezimalen Zeichens:

Darstellung des Zeichens:



(Fragezeichen stilisiert)

Auf „1" gesetzt sind die Bits D_0 D_1 D_5 und D_4. Dies ergibt binär:

0 0 1 1 0 0 1 1

oder hexadezimal

3 3

Schreibt man 3 3 in einen der LED PUFFER (Adressen siehe Kap. 3.5.) und ruft die Anzeigeroutine **ohne** Konversion auf, erscheint das obige Zeichen auf der entsprechenden Anzeigeeinheit.

3.2.4. Verwendung von Interrupts

Die Monitorprogramme verwenden für eigene Belange (STEP-Routine) den NMI (Non Maskable Interrupt). Dadurch ist der Einsatz des NMI durch den User ausgeschlossen.

Ebenfalls nicht benutzbar für Userzwecke sind MODE 0 und MODE 1 des maskable Interrupts, da sämtliche Restart-Adressen bereits vom Monitorprogramm belegt sind. Eine Interruptbehandlung durch das Userprogramm muß deshalb ausschließlich im MODE 2 erfolgen.

3.2.5. Besonderheit der Einschrittverarbeitung

Bedingt durch die Hardware der Grundplatine des Z80-KIT, treten bei der Abarbeitung von Befehlen mit Doppelanweisungsteil (Zwei M1 Zyklen) im STEP-Mode sog. „Zwischenschritte" auf. Dabei springt der Befehlszähler nicht wie erwartet auf die Adresse des nächsten Befehls, sondern auf 0068. Bei nochmaligem Drücken der Taste STEP erhält man den erwarteten Programmzählerstand. Diese „Zwischenschritte" haben keinerlei Auswirkungen auf das Userprogramm!

3.3. Mitbenutzbare Routinen des Betriebsprogramms

Im Anschluß finden Sie eine Zusammenstellung der Routinen des 1k Betriebsprogramms. Eine Zusammenstellung des 2k Betriebsprogramms finden Sie in Teil VI, KIT/TV. Neben einer Beschreibung der einzelnen Aktionen werden die jeweils von der Routine benutzten Register und die Einsprungadresse angegeben. Der Aufruf einer Routine erfolgt dann nach folgendem Schema:

Anwenderprogramm
evtl. Retten der Register
Aufruf Monitorroutine
evtl. Rückholen der Register
Anwenderprogramm Fortsetzung

Am Schluß dieser Aufstellung wird dieses Schema an einem konkreten Beispiel erläutert.

3.3.1. Zusammenstellung

OBTAST
Prüft, ob eine Taste des KIT Tastenfeldes gedrückt wurde. Bei der Rückkehr sind zwei Fälle möglich:
a) eine Taste wurde gedrückt
   ZERO FLAG zurückgesetzt
b) kein Tastendruck
   ZERO FLAG gesetzt

Keine Übergabeparameter
Benützte Register: A B C D E F H L IX
Einsprungpunkt: 0251

WETAST
Prüft um welche Taste es sich handelt. Bei der Rückkehr sind zwei Fälle möglich:
a) es war immer noch die gleiche Taste gedrückt (während eines Tastendruckes mehrere Abfragen, da Prozessor sehr schnell). Das A Register enthält den Wert FF
b) neue Taste erkannt. Das A Register enthält den Tastencode der entsprechenden Taste (siehe Schaltplan!)

Keine Übergabeparameter
Benützte Register: A B C D E F H L IX
Einsprungpunkt: 0287

LEDS1
Bringt die, in den zur DATA Anzeige gehörenden LED Puffern stehenden Werte bei DATA zur Anzeige.
Adresse der ‚Marke' beim Aufruf im HL Register-Paar.
a) bit 0 von Marke gleich 0 → Anzeige ohne Konversion
b) bit 0 von Marke gleich 1 → Anzeige mit Konversion
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0309

LEDS2
Bringt die, in den zur Adreß-Anzeige gehörenden LED Puffern stehenden Werte bei Adreß zur Anzeige
Adresse der ‚Marke' beim Aufruf im HL Register-Paar
a) bit 0 von Marke gleich 0 → Anzeige ohne Konversion
b) bit 0 von Marke gleich 1 → Anzeige mit Konversion
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0317

ZEITKO
Zeitschleife von ca. 0,5 Sec. Dauer.
Keine Übergabeparameter
Benützte Register: A B C F
Einsprungpunkt: 03CA

LOAD
Liest auf Kassette abgespeicherte Programme oder Daten ein.
Startadresse des Zielspeichers beim Aufruf im Speicher. 3C44/45 (siehe Kommando LOAD)
Benützte Register: A F H L
Einsprungpunkt: 0090

STORE
Speichert im RAM stehende Programme oder Daten auf Cassette ab.
Startadresse und Endadresse des Quellspeichers beim Aufruf im Speicher 3C40/41 und 3C42/43
(siehe Kommando STORE)
Benützte Register: A B C D E F H L
Einsprungpunkt: 00BC

ABSPE1
Entnimmt den beiden zur DATA Anzeige gehörenden LED Puffern die dort gespeicherten (und angezeigten) Halbworte (0, 1 . . . F), bildet ein Gesamtwort (00, 01 . . . FF) und speichert es in die gewünschte Speicherzelle ab
Adresse der gewünschten Speicherzelle beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 0361

ABSPE2
Entnimmt den vier zur Adreß-Anzeige gehörenden LED Puffern die dort gespeicherten (und angezeigten) Halbworte, bildet zwei Gesamtworte (high und low Byte) und speichert sie in die gewünschte Speicherzelle (low Byte) bzw. die nächstfolgende (high Byte) ab
Niedrigere Adresse der beiden gewünschten Speicherzellen beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 0369

RESPE1
Entnimmt dem gewünschten Speicher den Wert, teilt dieses Gesamtwort auf in zwei Halbworte und belegt damit die beiden zur DATA Anzeige gehörenden LED Puffer
Adresse der gewünschten Speicherzelle beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 0386

RESPE2
Entnimmt dem gewünschten und dem nachfolgendem Speicher den Wert, teilt die beiden Gesamtworte (low und high Byte) jeweils auf in zwei Halbworte und belegt damit die vier zur Adreß Anzeige gehörenden LED Puffer
Niedrigere Adresse der beiden gewünschten Speicherzellen beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungpunkt: 038E

FEHLER
Bringt die ERROR Lampe zum Leuchten
Keine Übergabeparameter
Benützte Register: A F
Einsprungpunkt: 02FF

FEHLEX
Löscht die ERROR Anzeige
Keine Übergabeparameter
Benützte Register: A F
Einsprungpunkt: 0303

ANZABF
Zusammengesetzte Routine aus LEDS1, LEDS2, OBTAST und WETAST; zeigt die in den LED Puffern gespeicherten Werte an und führt eine Tastenabfrage durch. Rücksprung bei Erkennen einer neuen Taste.
Adresse der ‚Marke' im HL Registerpaar
Bit 0 siehe LEDS1 bzw. LEDS2
a) Bit 1 von Marke gleich 1 → DATA Anzeige
b) Bit 1 von Marke gleich 0 → keine DATA Anzeige
c) Bit 2 von Marke gleich 1 → ADDRESS Anzeige
d) Bit 2 von Marke gleich 0 → keine ADDRESS Anzeige
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0346

TAREG1
Verschiebt die Inhalte der zur DATA Anzeige gehörenden LED Puffer um eine Stelle in Richtung höherwertiger Stelle. Die dadurch freiwerdende unterste Stelle (DATA 1) wird mit einem neuen Wert belegt
Neu einzulesender Wert beim Aufruf im A-Register
Benützte Register: A B C D E F H L
Einsprungpunkt: 02CC

TAREG2
Verschiebt die, in den zur Addreß Anzeige gehörenden LED Puffern stehenden Werte in Richtung höherwertiger Stellen. Die freiwerdende unterste Stelle (Addreß1) wird mit einem neuen Wert belegt
Neu einzulesender Wert beim Aufruf im A-Register
Benützte Register: A B C D E F H L
Einsprungpunkt: 02E0

4. Bedeutung der Tasten

Vorbemerkung:
In diesem Abschnitt wird eine Übersicht über die Funktion der einzelnen Tastenelemente gegeben. Bitte versuchen Sie an dieser Stelle noch nicht, Ihr System in Betrieb zu nehmen; an späterer Stelle wird Inbetriebnahme, Programm-Eingabe- und -Ausführung noch eingehend behandelt.

Kommandotastatur

RESET	Die Rücksetzungs (RESET) Taste gibt Ihnen die Möglichkeit, das System zu jeder Zeit in einen definierten Anfangszustand zu bringen. Sie ist nach jedem Einschalten der Speisespannung zu bestätigen.
EX	Abschlußkommando (EXECUTE) veranlaßt das Mikrocomputersystem zur Übernahme und Abspeicherung des zuvor eingetippten, auf der Anzeige sichtbaren Wertes.
LOAD	Ladeanweisung, dient zum Laden eines Programms, das auf Cassette gespeichert ist, in einen bestimmten Speicherbereich
STORE	Speicheranweisung, dient zum Abspeichern eines bestimmten Speicherbereiches auf Band.
DIS	Anzeigeanweisung (DISPLAY) dient zur Ausgabe eines Speicher- oder Registerinhalts
SET	Schreibanweisung dient zur Eingabe eines Wertes in eine Speicherzelle bzw. in ein Register
MEM	Speicheranweisung (MEMORY) wird benötigt, wenn eine Speicheroperation durchgeführt werden soll (z.B. SET MEM . . .gibt an, daß eine Speicherzelle belegt werden soll)
START	Startkommando dient zum Starten eines Anwenderprogramms
STEP	Einzelschrittkommando bewirkt Ausführung eines Programmschritts (=CPU-Befehl)
BR	Haltepunktanweisung (BREAKPOINT) mit dieser Taste kann ein Haltepunkt auf eine gewünschte Adresse gelegt werden
IN	Eingabekommando (INPUT) versetzt den Mikrocomputer in den Eingabemodus für die Eingabe eines Anwenderprogramms
IDM	Ausgabekommando (INCREMENT AND DISPLAY MEMORY) dient zur zusammenhängenden Ausgabe größerer Speicherbereiche
STORN	Aussprunganweisung (STORNO INPUT) dient zum Verlassen des INPUT-Modus

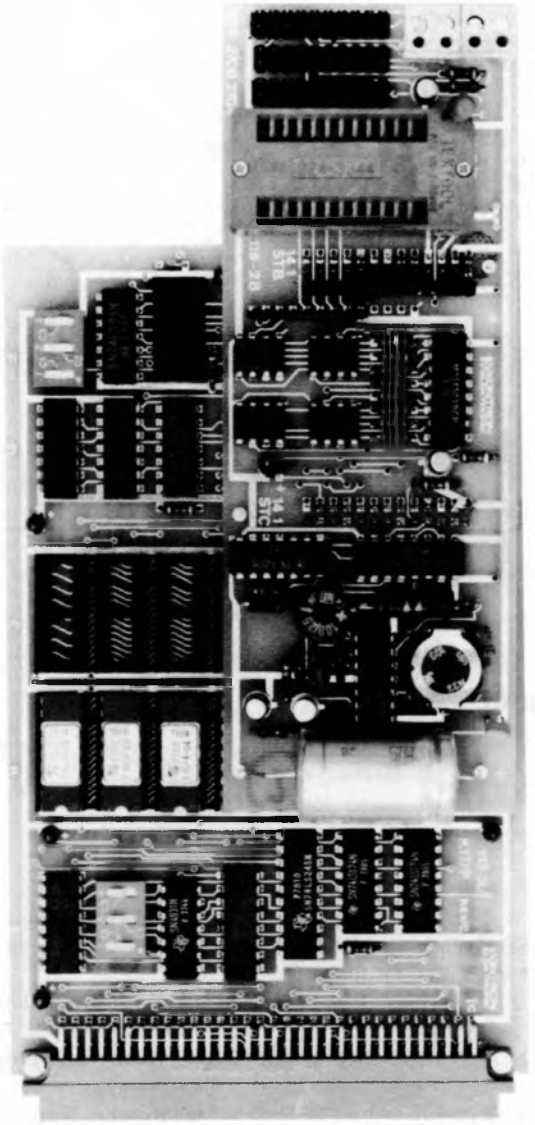
Hexadezimaltastatur

0/′	Hexadezimalziffer 0 / Kennung der Zweitregister (z.B. H′ L′ etc.)
1	Hexadezimalziffer 1
2	Hexadezimalziffer 2
3/I	Hexadezimalziffer 3 / I Register
4/P	Hexadezimalziffer 4 / Befehlszähler (Programm Counter = PC Register)
5/S	Hexadezimalziffer 5 / Stack Pointer (SP Register)
6/Y	Hexadezimalziffer 6 / IY Register
7/X	Hexadezimalziffer 7 / IX Register
8/H	Hexadezimalziffer 8 / H Register
9/L	Hexadezimalziffer 9 / L Register
A	Hexadezimalziffer A / A Register
B	Hexadezimalziffer B / B Register
C	Hexadezimalziffer C / C Register
D	Hexadezimalziffer D / D Register
E	Hexadezimalziffer E / E Register
F	Hexadezimalziffer F / F Register

KONTRON KIT®

PROM-Programmier- und Speichererweiterung KONTRON-KIT/P

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

1. Einleitung

Durch die beispielhafte Erweiterungsfähigkeit des Z80-KIT kann der Anwender je nach Wunsch oder Notwendigkeit einen entsprechenden Ausbau vornehmen, z.B. durch zusätzliche Speicher-, Ein/Ausgabe oder A/D bzw. D/A-Wandler-Karten. Durch die Kompatibilität des KIT's mit dem ECB-Bus kann diese Erweiterung einfach durch Zustecken von Baugruppen der ECB-Serie erfolgen.
Durch das in der Grundausstattung enthaltene Cassetten-Interface ist es darüber hinaus möglich, Programme auch in nichtflüchtiger Form abzuspeichern. Zur nichtflüchtigen Abspeicherung von Programmen in residenter Form benützt man programmierbare Nur-Lese-Bausteine (PROM's). Geräte zur Programmierung solcher PROM's sind allerdings meist relativ teuer (ca. 4000,– DM).
Die Firma KONTRON hat daher einen kostengünstigen Programmierzusatz KIT/P für den Z80-KIT entwickelt, der technisch und vor allen Dingen preislich äußerst attraktiv ist.

2. Überblick über die Möglichkeiten des KIT/P

Der KIT/P PROM-Programmer ist für das Programmieren von UV-löschbaren PROM's ausgelegt worden. Diese PROM's können beliebig oft gelöscht und wieder programmiert werden.

Folgende Typen sind mit dem KIT/P programmierbar:

Der KIT/P PROM-Programmer ist für das Programmieren von UV-löschbaren PROM's ausgelegt worden. Diese PROM's können beliebig oft gelöscht und wieder programmiert werden. Folgende Typen sind mit dem KIT/P programmierbar:

EPROM	i2704	½k×8	PROM's mit + 5V, – 5V + 12V
EPROM	i2708	1k×8	Versorgungsspannung
EPROM	i2758	1k×8	PROM's mit einer + 5V
EPROM	i2716	2k×8	Versorgungsspannung
EPROM	Z6716	2k×8	

Durch den modularen Aufbau der Hardware (Programmiermodul) und der Software (Unterprogramm-Aufruf-Technik) ist es möglich, den KIT/P auch auf andere Typen zu erweitern.

---

3. Schaltungsbeschreibung des KIT/P

3.1. Hardwareausstattung des KIT/P

Die Hardware des KIT/P besteht aus folgenden zwei Komponenten:

3.1.1. Speicher-I/O-Erweiterung

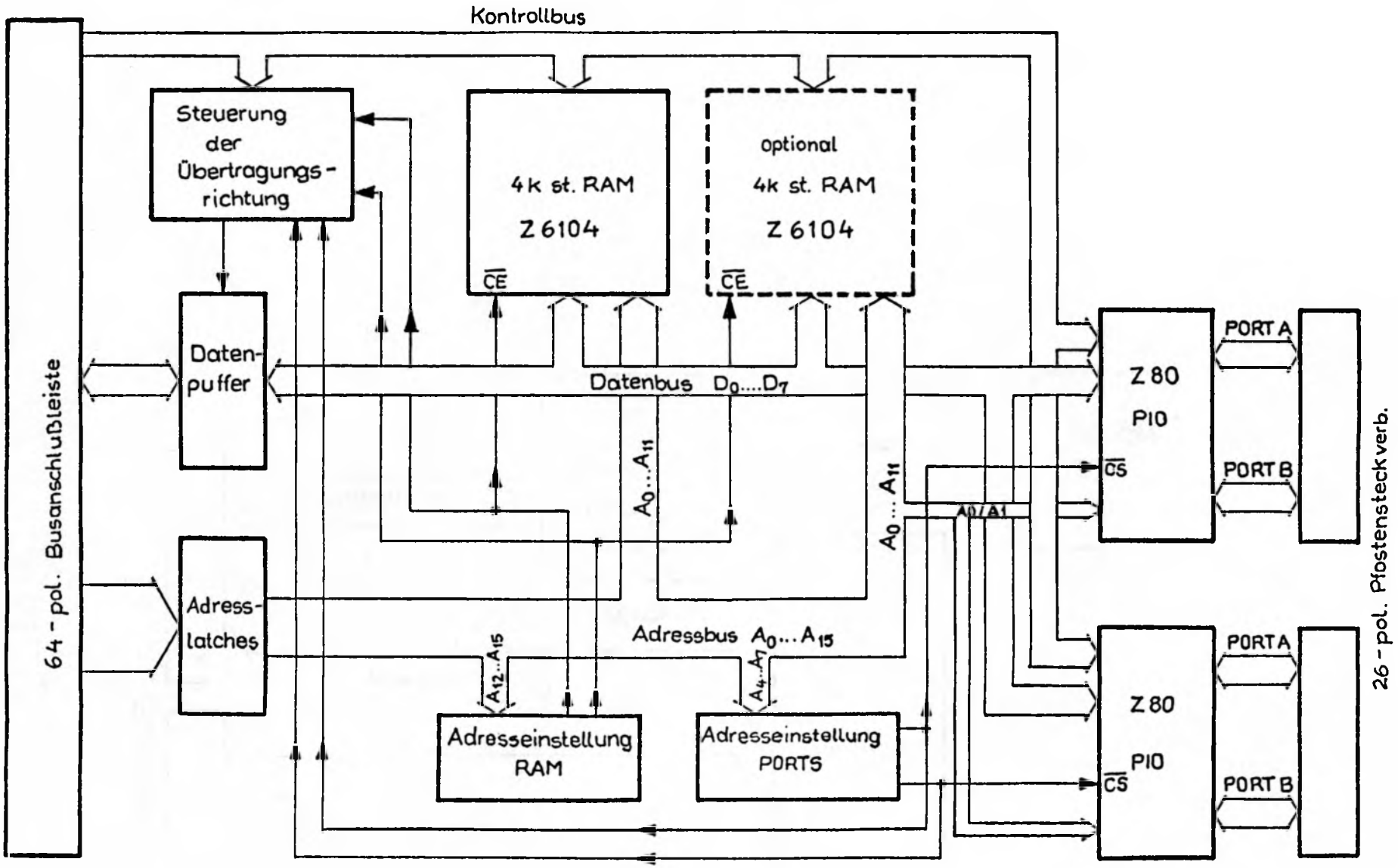
Europakarte steckbar auf einen der fünf Steckplätze des KIT,
- ☐ 2×Z80-PIO ≙ 4 parallele 8bit I/O Ports mit Quittungsleitungen
- ☐ 4 kByte statisches RAM (austauschbar auf 8 kB)
- ☐ Einstellung der Speicheranfangsadresse über Kippschalter in Schritten von 8 k
- ☐ Einstellung der Portadressen über Kippschalter
- ☐ sämtliche Signale mit Busanschlußwert von einer TTL-LS-Last
- ☐ alle Adressen zwischengespeichert
- ☐ Daten über bidirektionalen Schmitt-Trigger-Tri-State-Buffer 74LS245 geführt
- ☐ Ausgangsseitig (= PIO Ein/Ausgänge) zwei 26-polige Stiftleisten

3.1.2. PROM-Programmier-Modul

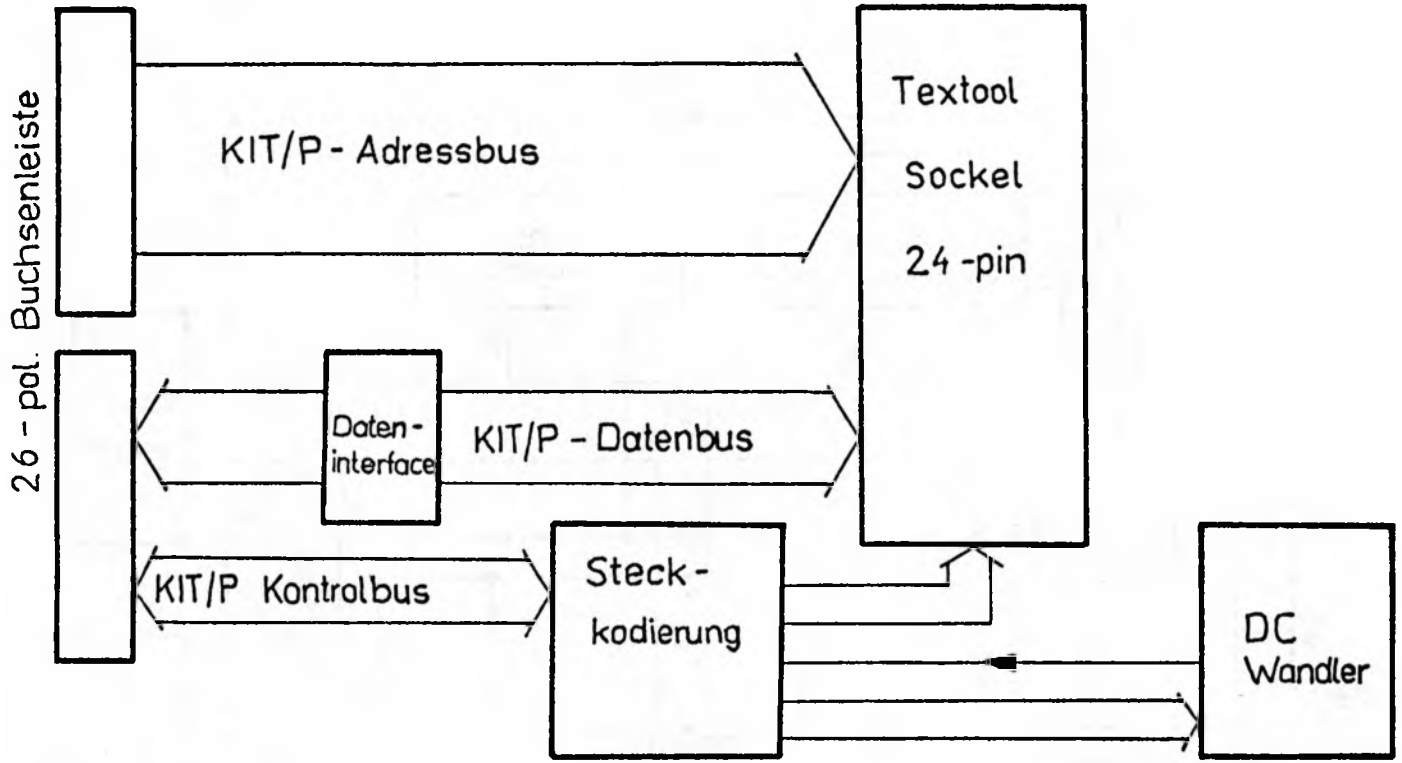
Programmiermodul, ca. 17 × 5 cm, als „Huckepackplatine" aufsteckbar
- ☐ 24-poliger Textool Sockel zur Aufnahme der PROM's
- ☐ Kodierstecker zur Auswahl des Promtyps.*
- ☐ Erzeugung der Programmierspannung von 26 V mittels DC-Wandler.
  Dadurch bei Verwendung von „5 V-EPROM's" keine zusätzliche Stromversorgung notwendig!
- ☐ Verbindung zu Speicher-I/O-Erweiterung über zwei Buchsenleisten, passend zu den Stiftleisten.
- ☐ Eindeutigkeit der Aufsteckposition durch Kodierstifte auf der Speicher-I/O-Erweiterung gewährleistet.
- ☐ Zuführung der Versorgungsspannungen von außen.
- ☐ Möglichkeit zur Abnahme der +5 V vom KIT direkt**

* Es gibt drei gekennzeichnete Möglichkeiten, den Kodierstecker aufzubringen.
2704 gilt für i2704 EPROM's
2708 gilt für i2708 EPROM's
2716 gilt für i2758, i2716 und Z6716 EPROM's

** Diese Möglichkeit empfiehlt sich, wenn nur PROM's mit einer einzigen +5 V Versorgung verwendet werden sollen. In diesem Fall kann auf dem PROM-Programmierer-Modul ein Jumper J eingelötet werden.



Blockschaltbild Z80-KIT/P
Teil 1 (ECB/S)



Blockschaltbild Z80-KIT/P
Teil 2 (ECB/P)

4. Beschreibung des KIT/P-Programms

4.1. Programmumfang

- ☐ folgende vier Betriebsarten (Modes) sind möglich:
  - Programmieren (Quelle = RAM)
  - Duplizieren (Quelle = PROM)
  - Listen
  - Verifizieren (Vergleichen)
- ☐ Fehlermeldung bei nicht aufgestecktem Kodierstecker.
- ☐ Anfangsadresse des RAM beim Programmieren und Verifizieren frei wählbar.
- ☐ Beim Programmieren von i2758, i2716 oder Z6716 zusätzlich Anfangsadresse des PROM's und Blocklänge (= Anzahl der zu programmierenden Bytes) wählbar.
- ☐ Fehlermeldung bei Überschreitung der PROM-Größe bei i2716 und Z6716
- ☐ Fehlermeldung bei nicht vollständig gelöschtem PROM
- ☐ Überprogrammieren möglich
- ☐ Fehlermeldung bei Unterschieden zw. PROM-Inhalt und RAM-Inhalt nach dem Programmiervorgang.
- ☐ Anfangsadresse des PROM's beim Listen beliebig einstellbar.
- ☐ Voreinstellung von RAM- und PROM-Startadresse mit 0000 und der Blocklänge entsprechend dem gewählten Typ beim Einsprung in die Betriebsarten Programmieren und Duplizieren
- ☐ Lieferung des KIT/P-Programmes auf Cassette
- ☐ Optionale Lieferung des KIT/P Programms im PROM

4.2. Allgemeine Beschreibung

Das KIT/P-Programm beginnt mit der Frage nach der gewünschten Betriebsart (Mode), wobei die Möglichkeiten

a) Programmieren – Daten im Speicher
b) Duplizieren – Daten in einem PROM
c) Listen – Lesen eines PROM-Inhaltes
d) Verifizieren – Vergleich PROM/Speicher

zur Verfügung stehen.

Vor der Wahl der gewünschten Betriebsart muß der Kodierstecker für die Typenwahl gesteckt werden, da andernfalls eine Fehlermeldung erfolgt.

Zu den einzelnen Betriebsarten können folgende Parameter angegeben werden:

zu a) – Anfangsadresse des RAM-Bereiches in dem die Daten stehen
bei i2716, i2758, Z6716, **zusätzlich:**
– Anzahl der Bytes
– Anfangsadresse im PROM

zu c) – Anfangsadresse im PROM

zu d) Anfangsadresse im RAM und Blocklänge

Das Programmieren der angegebenen EPROM-Typen ist unterschiedlich. Während die Typen i2704 und i2708 immer als Ganzes, d.h. alle Zellen zusammenhängend programmiert werden müssen, ist es bei den Typen i2758, i2716 und Z6716 möglich, auch einzelne Teilbereiche anzusprechen. Im Extremfall heißt das, das Programmieren eines einzelnen Bytes. Auf Grund dessen erfolgt für diese Typen bei der Betriebsart Programmieren zusätzlich die Abfrage der PROM Startadresse und der Anzahl der zu programmierenden Bytes (= Blocklänge).

Vor dem Beginn der Programmierroutine wird der Inhalt des zu programmierenden PROM's geprüft. Ist es nicht gelöscht (d.h. weist nur ein Wort nicht den Wert FF auf), so erscheint eine entsprechende Fehlermeldung. Ein Überprogrammieren ist jedoch möglich.

Die eigentlichen „Brenn Routinen" erfolgen nach den Angaben der Hersteller. Für die Typen i2758, i2716 und Z6716 wurde zusätzlich ein sog. QUICK EXIT eingebaut. Dabei wird **vor** dem Brennvorgang jeder Zelleninhalt mit dem einzuprogrammierenden Wert verglichen. Stimmt er überein, geht das Programm sofort zur nächsten Zelle über.

Im gegenteiligen Fall wird die entsprechende Zelle gegebenenfalls bis zu maximal 5 mal „gebrannt". Dadurch ergeben sich variable Programmierzeiten.

Die typischen Zeiten liegen bei gelöschten EPROM's im Bereich von Minuten.

Während des Programmiervorganges erlöschen alle Anzeigen.

Anschließend an die Brennroutine wird der PROM-Inhalt mit dem RAM-Inhalt verglichen. Bei einer Diskrepanz erfolgt eine Unterbrechung sowie eine Fehlermeldung mit der Möglichkeit, beide Werte (jeweils Adresse und Daten) anzuzeigen. Erst nach einem entsprechenden Kommando wird die Verifizierroutine fortgesetzt.

Ist die Verifizierroutine nach 0 oder n Fehlern abgeschlossen, kehrt das Programm an eine Stelle zurück, die ein sofortiges nochmaliges Programmieren mit den gleichen Daten und Parametern zuläßt. Es kann also jeweils sofort ein weiteres PROM mit den gleichen Werten programmiert werden, bzw. ein sofortiges Nachprogrammieren des gleichen PROM's ist möglich.

4.3. Dialog Programm, Benutzer

Das Programm bringt nach seinem Start zur Festlegung der Parameter Systemmeldungen bzw. -fragen in kodierter Form zur Anzeige und erwartet eine entsprechende Tasten-Eingabe. Treten im Verlauf der Vorbereitung Fehler in Bezug auf Promgröße oder Promtyp auf, so erscheint ebenfalls in kodierter Form eine Fehlermeldung. Desgleichen bei nicht gelöschtem PROM oder bei einem Programmierfehler, der bei der abschließenden Prüfung entdeckt wird. Auch hier erwartet das System eine entsprechende Eingabe.

NB: In allen Teilen des Programmes führen nur dafür vorgesehene Tasten zu einer Aktion. Alle anderen Tasten werden ignoriert und haben keinerlei Wirkung.

Grundsätzlich gilt:

EX Taste: Abschluß des aktuellen und Einleitung des nächsten Abschnittes
0 Taste: Rücksprung in den vorhergegangenen Abschnitt
A Taste: Anzeige eines im aktuellen Abschnitt gültigen Wertes.

Wurde durch Drücken der Taste A ein Wert zur Anzeige gebracht, so **kann** er durch Eintippen einer anderen Zahl verändert werden. Die Zahlen werden dabei sequentiell von rechts nach links in die Anzeige geschoben. Erst wenn der gewünschte Wert angezeigt wird, kann mittels EX Taste dieser Wert abgespeichert und der nächste Abschnitt eingeleitet werden.

4.4. Zusammenstellung der Systemmeldungen und Eingabemöglichkeiten

4.4.1. Auswahl der Betriebsart

Anzeige Anzeige	Aktivität des Benutzers	Erläuterung
Co?		Kommando? Frage nach der gewünschten Betriebsart (Mode)
	Taste 0	Rücksprung ins Monitorprogramm
	Taste 1	Sprung zum Programmieren
	Taste 2	Sprung zum Duplizieren
	Taste 3	Sprung zum Listen
	Taste 4	Sprung zum Verifizieren (Check)

4.4.2. Programmteil Programmieren

Anzeige	Aktivität des Benutzers	Erläuterung
– P 1		Gib Startadresse des RAM-Bereiches!
	Taste 0	Rücksprung zu „Co?“
	Taste EX	Sprung zu ‚P2‘ (i2758, i2716 und Z6716) bzw. ‚P4‘? (i27040 (RAM Startadresse: Voreinstellung = 0000)
	Taste A	Sprung zu „Anzeige RAM-Startadresse“
– Anzeige RAM-Startadresse	Taste 0...F	Einlesen ins Anzeigen register
	Taste EX	Abspeichern RAM Startadresse und Sprung zu ‚P2‘ (i2758, i2716 und Z6716) bzw. ‚P4‘ (i2704/08).
– P 2		Gib Blocklänge (nur i2758, i2716 und Z6716)!
	Taste 0	Rücksprung zu ‚P1‘
	Taste EX	Sprung zu ‚P3‘ (Blocklänge: Voreinstellung = 0800)
	Taste A	Sprung zu ‚Anzeige Blocklänge‘
– Anzeige Blocklänge	Taste 0...F	Einlesen ins Anzeigenregister
	Taste EX	Sprung zu ‚P3‘
– P 3		Gibt PROM Startadresse!
	Taste 0	Rücksprung zu ‚P2‘
	Taste EX	Sprung zu ‚P4?‘ (PROM Startadresse: Voreinstellung = 0000)
	Taste A	Sprung zu ‚Anzeige PROM-Startadresse‘
– Anzeige PROM-Startadresse	Taste 0...F	Einlesen ins Anzeigenregister
	Taste EX	Sprung zu ‚P4?‘
– P 4?		Fertig zum Programmieren?
	Taste 0	Rücksprung zu ‚P3‘ (2716 etc.) bzw. ‚P1‘ (2704/08)
	Taste EX	Sprung zur Programmierroutine. Anschließend Rückkehr zu ‚P4?‘

4.4.3 Programmteil Duplizieren

Anzeige	Aktivität des Benutzers	Erläuterung
– D 1		Original PROM einsetzen!
	Taste 0	Rücksprung zu ‚Co?'
	Taste EX	Einlesen PROM-Daten, RAM-Startadresse = 6800H und Sprung zu ‚D2'
– D 2		PROM's wechseln!
	Taste 0	Rücksprung zu ‚D1'
	Taste EX	Sprung zu ‚D3?'
– D 3?		Fertig zum programmieren?
	Taste 0	Rücksprung zu ‚D2'
	Taste EX	Sprung zur Programmierroutine. Anschließend Rückkehr zu ‚D3?'

4.4.4 Listen

Anzeige	Aktivität des Benutzers	Erläuterung
– L 1		Gib PROM Startadresse!
	Taste 0	Rücksprung zu ‚Co?'
	Taste EX	Sprung zur Anzeigenroutine (PROM Startadresse: Voreinstellung = 0000)
	Taste A	Sprung zu ‚Anzeige PROM Startadresse'
– Anzeige PROM Startadresse		
	Taste 0...F	Einlesen ins Anzeigenregister
	Taste EX	Sprung zur Anzeigenroutine
– Anzeigenroutine		Zeigt die aktuelle PROM-Adresse und Daten an
	Taste 0	Rücksprung zu ‚L1'
	Taste IDM	Adresse inkrementieren neue Adresse und Daten anzeigen

4.4.5 Verifizieren (Checken)

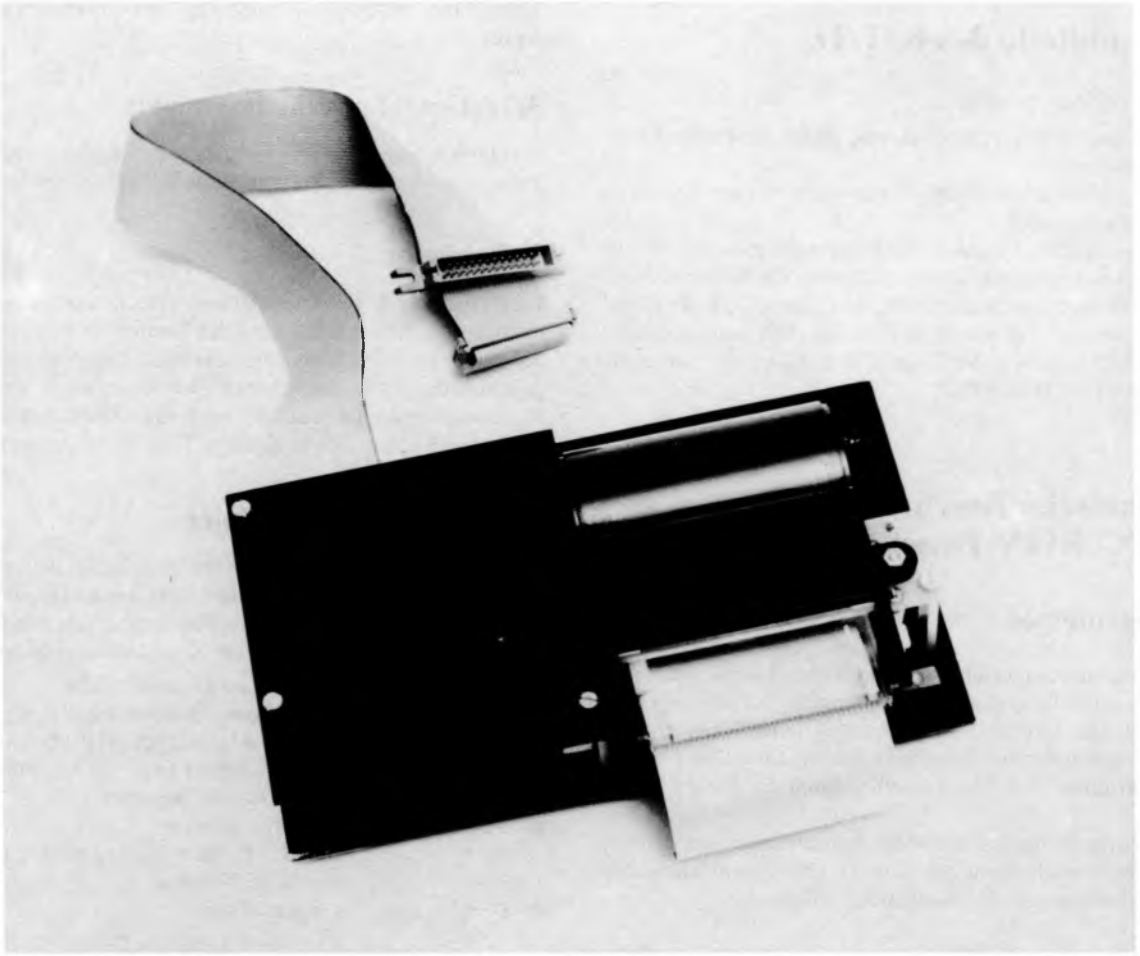
Anzeige	Aktivität des Benutzers	Erläuterung
– C 1		Gib RAM Startadresse!
	Taste 0	Rücksprung zu ‚Co?'
	Taste EX	Sprung zu ‚C2'
	Taste A	Sprung zu ‚Anzeige RAM Startadresse'
– Anzeige RAM-Startadresse		
	Taste 0...F	Einlesen in Anzeigenregister
	Taste EX	Sprung zu ‚C2'
– C 2		Gib Blocklänge!
	Taste 0	Rücksprung zu ‚C1'
	Taste EX	Sprung zu ‚C3?'
	Taste A	Sprung zu ‚Anzeige Blocklänge'
– Anzeige Blocklänge		
	0...F	Einlesen in Anzeigenregister
	Taste EX	Abspeichern Blocklänge und Sprung zu ‚C3?'
– C 3?		Fertig zum Verifizieren?
	Taste 0	Rücksprung zu ‚C1'
	Taste EX	Sprung zur Verifizierroutine, anschließend zurück zu ‚C3?'

4.4.6 Fehlermeldungen

Anzeige	Aktivität des Benutzers	Erläuterung
– F 1		PROM-Typ undefiniert
	Taste 0	Rücksprung zu ‚Co?'
– F 2		Größenüberschreitung
	Taste 0	Rücksprung zu ‚P3'
– F 3 ?		PROM nicht leer, trotzdem programmieren?
	Taste 0	Rücksprung zu ‚Co?'
	Taste EX	Sprung zur Programmierroutine; anschließend nach ‚P4?' oder ‚D3?' zurück
– F 4		Programmierfehler
	Taste 0	Rücksprung zu ‚P4?' bzw. ‚D3?'
	Taste EX	Weiter in Verifizierroutine
	Taste A	Sprung zu ‚Anzeige PROM'
– Anzeige PROM		Anzeige PROM Adresse und Daten
	Taste 0	Rücksprung zu ‚P4?' bzw. ‚D3?'
	Taste EX	Weiter in Verifizierroutine
	Taste A	Sprung zu ‚Anzeige RAM'
– Anzeige RAM		Anzeige RAM Adresse und Daten
	Taste 0	Rücksprung zu ‚P4?' oder ‚D3?'
	Taste EX	Weiter in Verifizierroutine
	Taste A	Sprung zu ‚Anzeige PROM'

KONTRON-KIT/D

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

1. Einleitung

Der Drucker-Zusatz Z80-KIT/D stellt eine preisgünstige Ergänzung des Einfach-Computer- und Lernsystems KONTRON Z80-KIT dar.
Er erlaubt mit der mitgelieferten Software den Ausdruck von Speicherstellen in Hexadezimalform und den Ausdruck von ASCII-Zeichen.
Die Software ist in zwei Unterprogramme (ASCII-, HEX-Ausdruck) aufgeteilt und kann vom Anwender wahlweise aufgerufen werden.
Der Anschluß des Druckers an den Z80-KIT erfolgt über die parallele Schnittstelle (PIO) der ECB/K.

2. Bestandteile des KIT/D

Der Drucker-Zusatz besteht aus

- Grundplatte mit Papierhalterklappe, durch die Papier nachgelegt werden kann.
- KONTRON-Drucker 5020 B mit Papierhalter und Ansteuerungselektronik
- 26-pol. Scotchflex-Flachkabelverbinder (mit Zusatzplatine am Drucker befestigt); 26-pol. Pfosten-Verbinder (3M – Nr. 3399) zum Aufsecken auf den 26-poligen Pfosten-Steckverbinder STB der ECB/K. Zusätzlich aufgequetschter Stiftstecker (3M – Nr. 3328) für den Zugang zum nicht benutzten Port B der PIO.

3. Technische Beschreibung des KONTRON-Druckwerks

3.1. Allgemeines

Es handelt sich um ein seriell arbeitendes Druckwerk. Gedruckt wird mittels eines beweglichen Druckkopfes, der sieben Elektroden trägt. Die Druckrichtung ist dabei je nach Einbaulage von rechts nach links bzw. von links nach rechts. Die Zeichen werden über eine 5 × 7 Punktematrix durch die Steuer-Logik gebildet.
Das Druckprinzip basiert auf dem punktweisen Abbrennen einer dünnen Metallschicht des Papiers. Die darunterliegende schwarze Schicht ergibt den notwendigen Kontrast.
(Siehe 3.9.).

3.2. Zeichengröße

Höhe: ca. 3 mm; festgelegt durch die Anordnung der sieben Elektroden
Breite: Variabel; hängt von der Zahl der in einer Zeile zu druckenden Zeichen ab.
Einstellung bei KIT/D;
Breite: ca. 1,5 mm

3.3. Zeichensatz

Der Zeichensatz hängt vom Zeichengenerator der Kontroll-Logik ab; es wird der aus 64 Zeichen bestehende ASCII-Zeichensatz verwendet. Änderungen des Zeichensatzes sind durch Einsatz anderer Zeichengeneratoren (PROM) möglich.

3.4. Zeichenzahl

Die Zahl der Zeichen pro Zeile wird mit der Zeichenbreite von der Steuer-Logik und von der dazugehörigen Druckersoftware bestimmt.
Beim KIT/D sind maximal 29 Zeichen pro Zeile erlaubt.
Mit dem Potentiometer auf der Steuer-Logik (siehe Lageplan der Lötbrücken; Anhang) ist die Zeichenbreite und damit auch die „Breite des gesamten Ausdrucks veränderbar.

3.5. Zeilenabstand

Der Zeilenabstand beträgt 5 mm.
Er ist durch die Mechanik des Druckwerkes festgelegt.

3.6. Druckgeschwindigkeit

Die Druckgeschwindigkeit beträgt ca. 2 Zeilen pro Sekunde.
Es besteht die Möglichkeit zur Umrüstung auf ca. 1 Zeile pro Sekunde. (Siehe 7.).
Dies kann durch Änderung von zwei Lötbrücken erreicht werden.

3.7. Umgebungsbedingungen

Das Druckwerk arbeitet ohne jegliche Schmiermittel. Dadurch kann es auch unter extremen Bedingungen arbeiten.

3.8. Motor

Das Druckwerk wird von einem Glockenankermotor angetrieben. Die Maximalleistung des Motors ist wesentlich größer als die Betriebsleistung. Dies garantiert eine konstante Laufgeschwindigkeit während des Druckvorgangs, wodurch ein Kopfpositionssignal entfällt und die Zeichenanforderungslogik durch einen frei laufenden Takt gesteuert werden kann.

3.9. Metallisiertes Papier

Das metallisierte Papier besteht aus Zellulose, wobei eine Seite mit einer durchgehenden, gleichmäßigen Aluminiumschicht bedampft ist. Zwischen Papier und Aluminium ist eine Zwischenschicht aus schwarzem Kontrastmaterial aufgebracht.
Physikalische und mechanische Eigenschaften:

- Gewicht: 37 Gramm/Quadratmeter (+/– 8%) gemessen nach Tappi 410 0S – 68 Normen
- Dicke: 41 Mikrometer (+/– 10%) gemessen nach Tappi 0S – 68 Normen
- Zugfestigkeit: in Längsrichtung: 6,5 kg/15 mm +/– 2, gemessen nach Tappi T 404 TS Normen
- Reißfestigkeit: längs und quer: 17 Gramm +/– 5, gemessen nach Tappi T 414 TS – 65 Normen

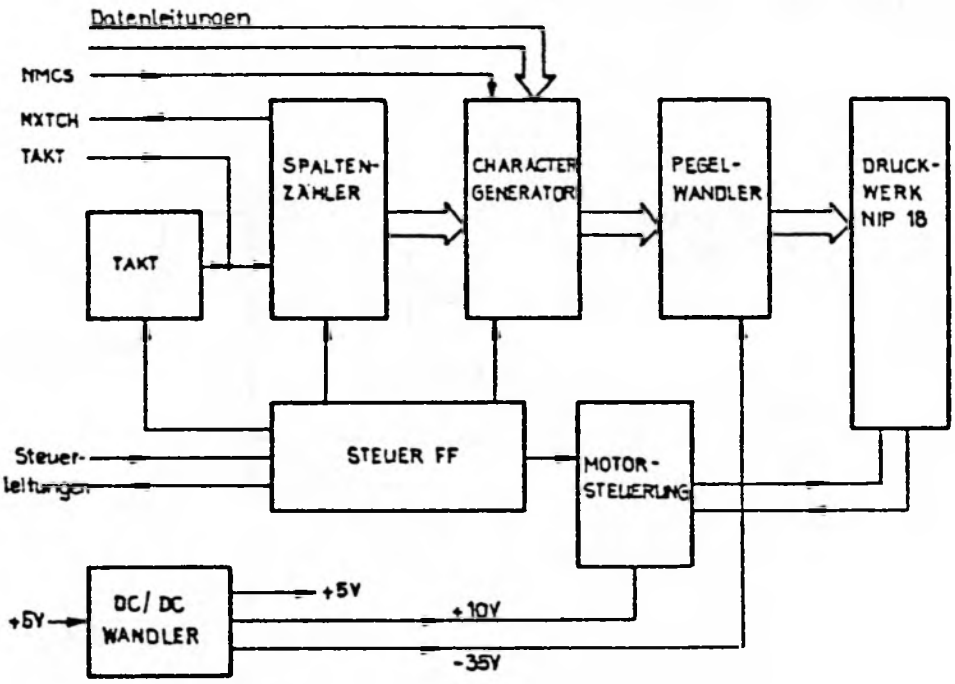
3.10. Geräuschentwicklung

Während des Druckvorganges ist der Geräuschpegel des Druckwerks ohne Gehäuse, gemessen in einem echofreien Raum, weniger als 54 Dezibel.
In der Bereitschaftsstellung ist das Druckwerk absolut geräuschlos.

4. Elektronischer Aufbau des Druckers

Die Logik des Druckers ist in CMOS-Technologie aufgebaut.
Als Zeichen-Generator wird ein TTL-PROM verwendet.

4.1. Blockschaltbild KIT/D



4.2. Anschluß des KIT/D

Der Anschluß des KIT/D an den Z80-KIT erfolgt über PORT A der parallelen Schnittstelle (PIO) der ECB/K.
Die Ansteuerung des Dateneingangs des KIT/D erfolgt über die 6 Datenleitungen A 0 . . . . A 5 des PORT A der PIO auf der ECB/K. (6 Bit ASCII, Bit-parallel, Zeichen-seriell).

Die 3 Steuersignale

- NXTCH (Next Character)
- PB (Printer Busy)
- PRC (Print Command)

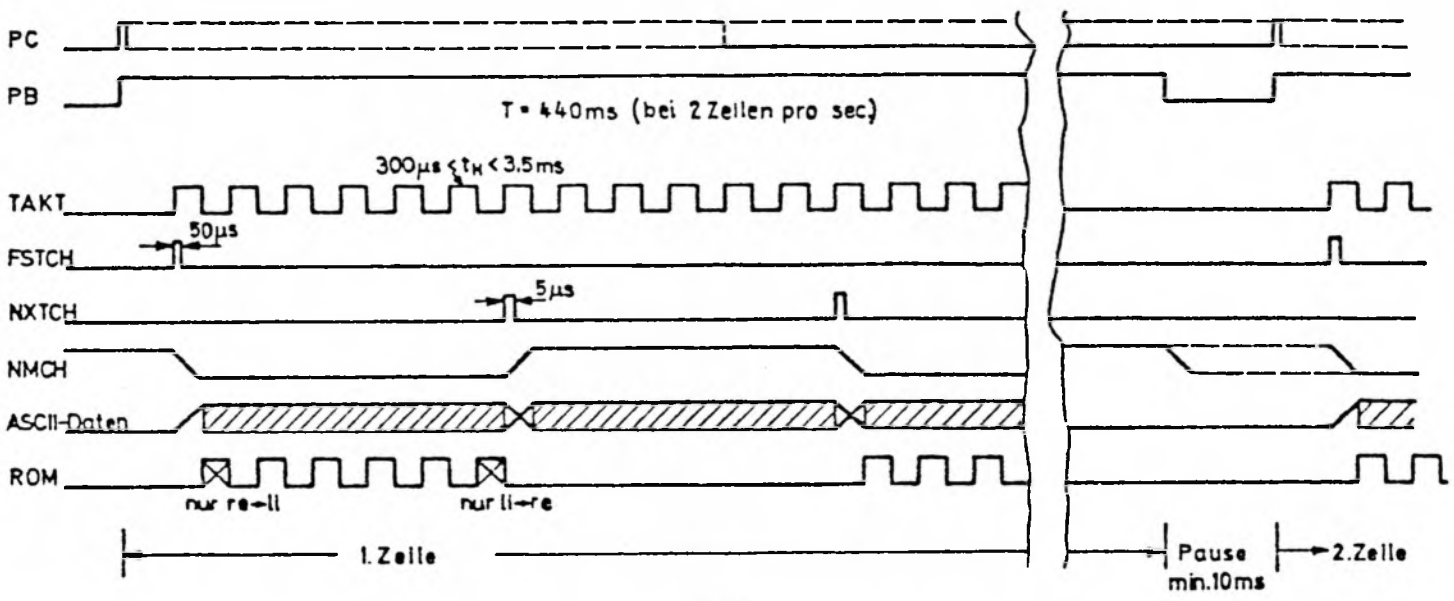
laufen ebenfalls über den PORT A (A 6, A 7 und ARDY) der ECB/K-PIO.
Mittels des aufgequetschten Stiftsteckers (3M – Nr. 3328) des Flachbandkabels zwischen ECB/K und KIT/D kann über das verbleibende PORT B der PIO auf der ECB/K frei verfügt werden.

4.3. Verbindungen zwischen ECB/K und KIT/D

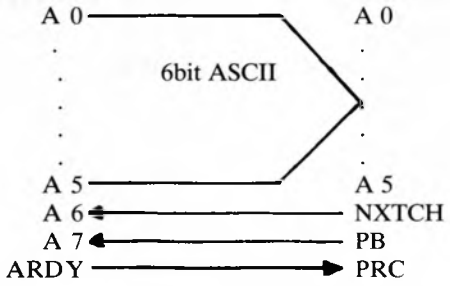
ECB/K STB Stift Nr.	Signal		KIT/D 20-pol. Steckverbinder Stift Nr.	Signal	
13	+5V		1	+5V	
26	+5V		2	+5V	
4	A 5		3	A 5	
5	A 4		4	A 4	
6	A 3		6	A 3	Daten
7	A 2	Port A	8	A 2	
8	A 1		10	A 1	
9	A 0		12	A 0	
3	A 6		13	NXTCH	Steuer-
2	A 7		16	PB	leitungen
12	ARDY		9	PRC	
1	GND		19	GND	
4	GND		20	GND	

4.4 Impulsdiagramm

ASCII-Daten: schraffierter Bereich kennzeichnet Gültigkeit. 500 μs nach FSTCH bzw. NXTCH gültig (bei 2 Zeilen/s, 32 Zeichen/Zeile)
NMCH muß H werden mit NXTCH, das dem letzten zu druckenden Zeichen folgt,
muß L werden mit FSTCH, kann L werden mit PB (bzw. PB)



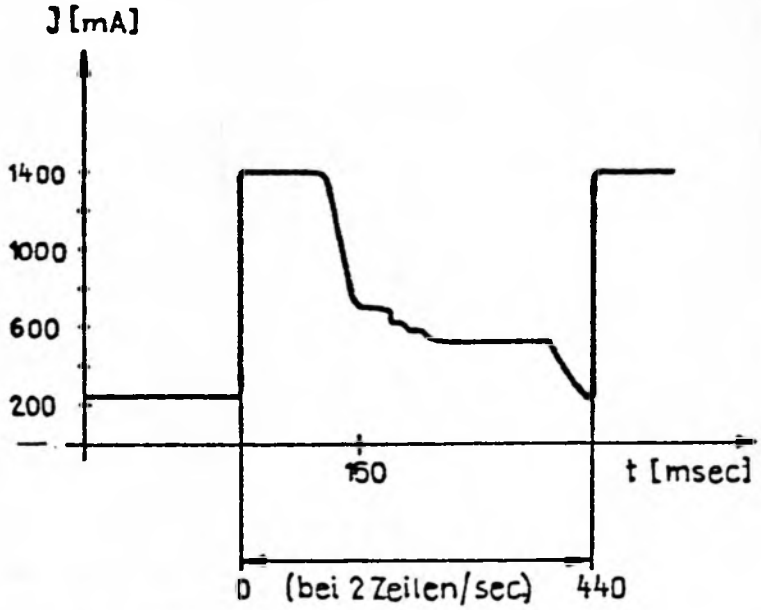
ECB/K /D



Die gezeigten Verbindungen werden durch das mitgelieferte Flachband hergestellt.

4.5. Stromaufnahme

Die Stromaufnahme im Ruhezustand beträgt ca. 235 MA.



Stromaufnahme für $U_{Betrieb} = 5{,}0$ Volt

5. Beschreibung des KIT/D Programms

5.1. Allgemeine Beschreibung

Das KIT/D-Programm umfaßt folgende zwei Möglichkeiten:
a) ASCII-Ausdruck eines Speichers
b) Hexadezimaler Speicherausdruck in formatierter Form
Beide Programmteile können wie Routinen des Betriebsprogramms aufgerufen werden (siehe 5.3.).

ACHTUNG:
Das KIT/D-Programm benötigt zur Zwischenspeicherung der auszudruckenden Werte einen 32 Bytes umfassenden Speicherbereich.
Dieser Bufferbereich liegt bei der Adresse 3CA2 beginnend. Es wird demnach der Speicher im Bereich

3CA2 bis 3CC1

vom KIT/D-Programm belegt.

ACHTUNG:
Das KIT/D-Programm arbeitet mit Interrupts im Interrupt-Mode-2 (IM2).

Auf diese Tatsache ist besonders zu achten, wenn die Drucker-Routinen von einem Anwender-Programm aufgerufen werden, das ebenfalls mit Interrupt arbeitet.

Das Retten des I-Registers der CPU muß vor dem Aufruf im Anwenderprogramm realisiert werden.

Das Laden der I-Register von CPU und PIO (Int.-Vektor) für die Druckerroutinen erfolgt automatisch im Programmteil PIOIN, dem Initialisierungsprogramm, das von den beiden Routinen DASCII und DHEX aufgerufen wird.

Die Interrupttabelle, die die Adresse der Interruptserviceroutine enthält, liegt auf den Adressen

3CA0 und 3CA1

5.1.1. Unterprogramm DASCII

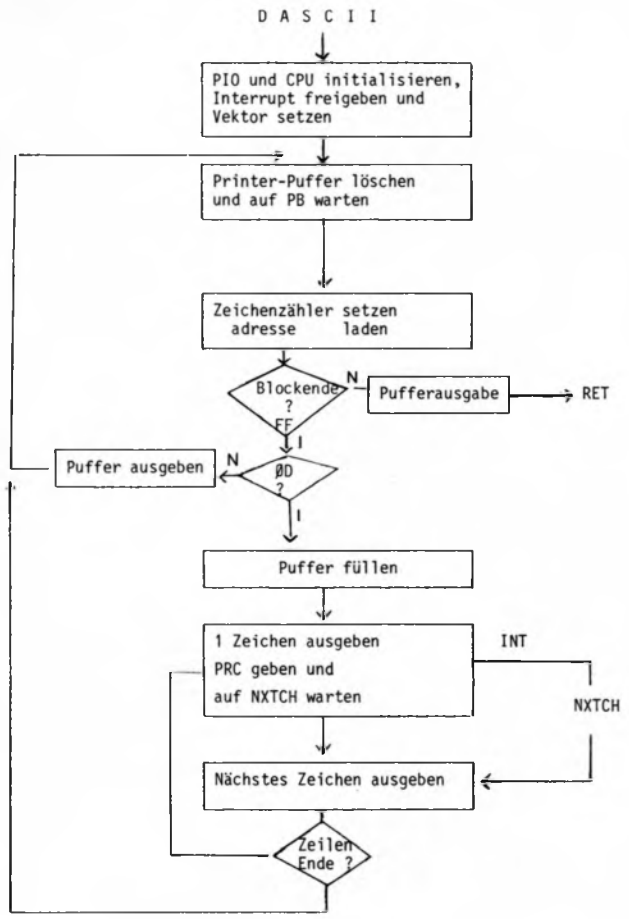
Ein mit ASCII-äquivalenten Hexadezimal-Zeichen gefüllter Speicherbereich beliebiger Länge kann mit dem Unterprogramm „DASCII" auf dem KIT/D ausgegeben werden. Dabei werden n Zeilen zu je max. 29 Zeichen gedruckt.
Bei Aufruf der Routine muß die Anfangsadresse des Quellspeichers angegeben werden.
Die Beendigung des ASCII-Andruckes erfolgt, wenn im Speicher das Zeichen

FF ≙ Blockende

gefunden wird. Dieses Zeichen wird nicht mehr ausgedruckt.
Wird das Zeichen

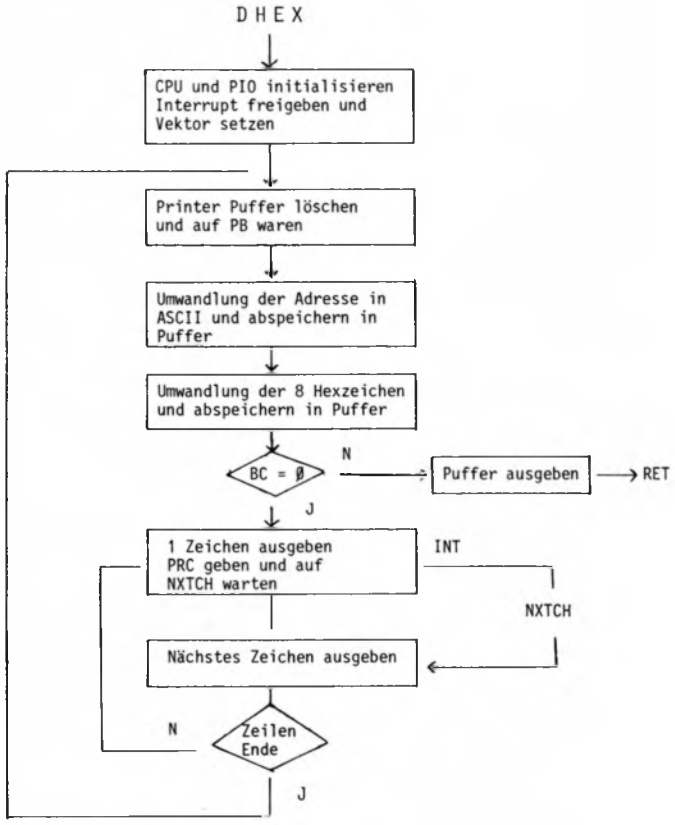
0D ≙ NEUE ZEILE!

erkannt, so wird anschließend eine neue Zeile begonnen. Der automatische Beginn einer Neuzeile nach 29 Zeichen erfolgt davon unabhängig.



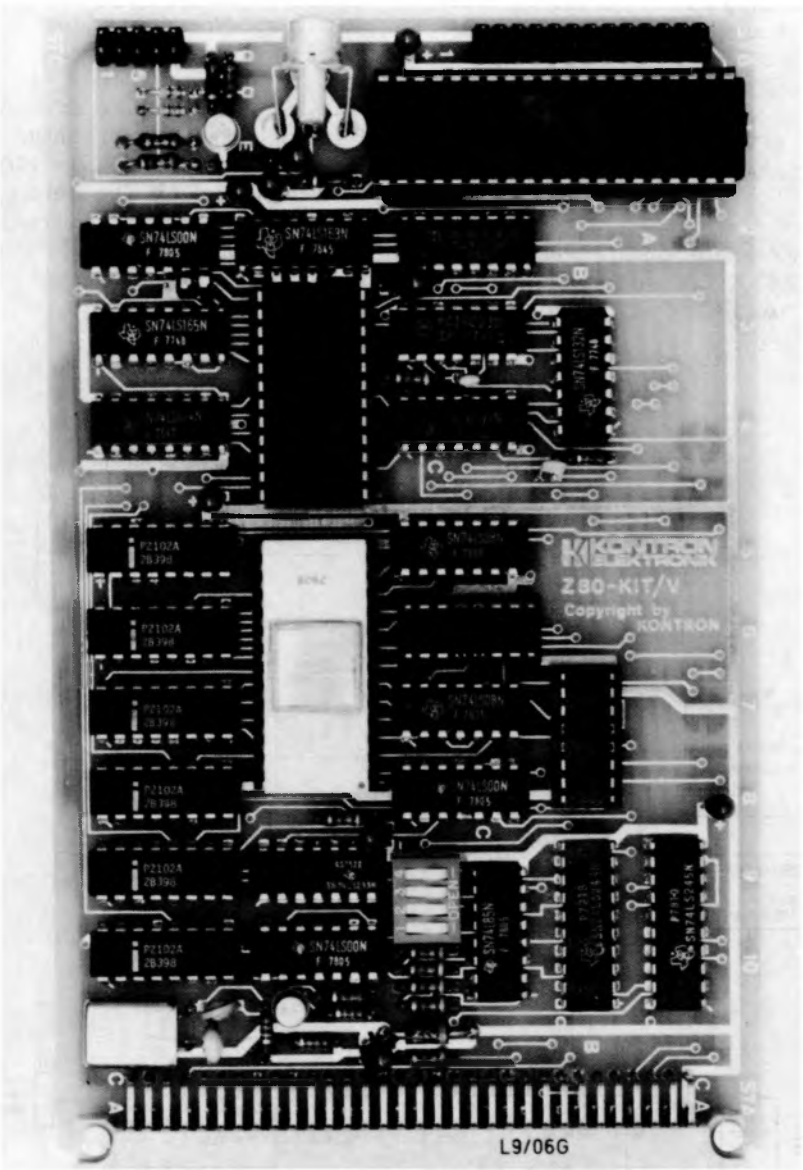
5.1.2. Unterprogramm DHEX

Der Ausdruck von Hexadezimal-Zeichen dient zur Ausgabe von Programmen und Speicherstellen und erfolgt in formatierter Form.
Dabei wird in folgendem Format ausgegeben:
Als erstes wird die 16-Bit Adresse des ersten Speicherplatzes ausgegeben.
Danach folgen zwei Leerzeichen und darauf der Ausdruck von max. 8 Speicherstellen pro Zeile, jeweils durch Leerzeichen getrennt.
Bei jeder neuen Zeile wird wieder mit dem Ausdruck der aktuellen Adresse begonnen.
Bei Aufruf der Routine müssen die Anfangsadresse des Quellspeichers und die Anzahl der zu druckenden Bytes angegeben werden.
Die Beendigung des Ausdruckes erfolgt automatisch nach dem letzten Zeichen.



KONTRON-KIT/V
KONTRON-KIT/VZ

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

1. Gegenüberstellung von Z80-KIT/V und Z80-KIT/VZ

Bei den Baugruppen Z80-KIT/V und Z80-KIT/VZ handelt es sich in beiden Fällen um Baugruppen zur Bildschirmansteuerung.

Unterschiede bestehen in folgenden 3 Punkten:

- Z80-KIT/VZ wird zusammengebaut und getestet geliefert. und kann direkt auf jeden ECB- oder KIT-Bus aufgesteckt werden. Z80-KIT/V ist ein Bausatz, der als besonders preisgünstige Erweiterung zum Z80-KIT zu verstehen ist.

- Während die Baugruppe Z80-KIT/V über die parallele Schnittstelle des, im Z80-KIT befindlichen Einplatinencomputers (ECB/K bzw. ECB/C) angesteuert wird, besitzt die Baugruppe Z80-KIT/VZ eine zusätzliche Parallelschnittstelle.

- Die Einheit Z80-KIT/VZ besitzt einen eigenen Busanschluß, die Baugruppe Z80-KIT/V wird an die parallele Schnittstelle der ECB/K bzw. ECB/C angeschlossen.

2. Schaltungsbeschreibung

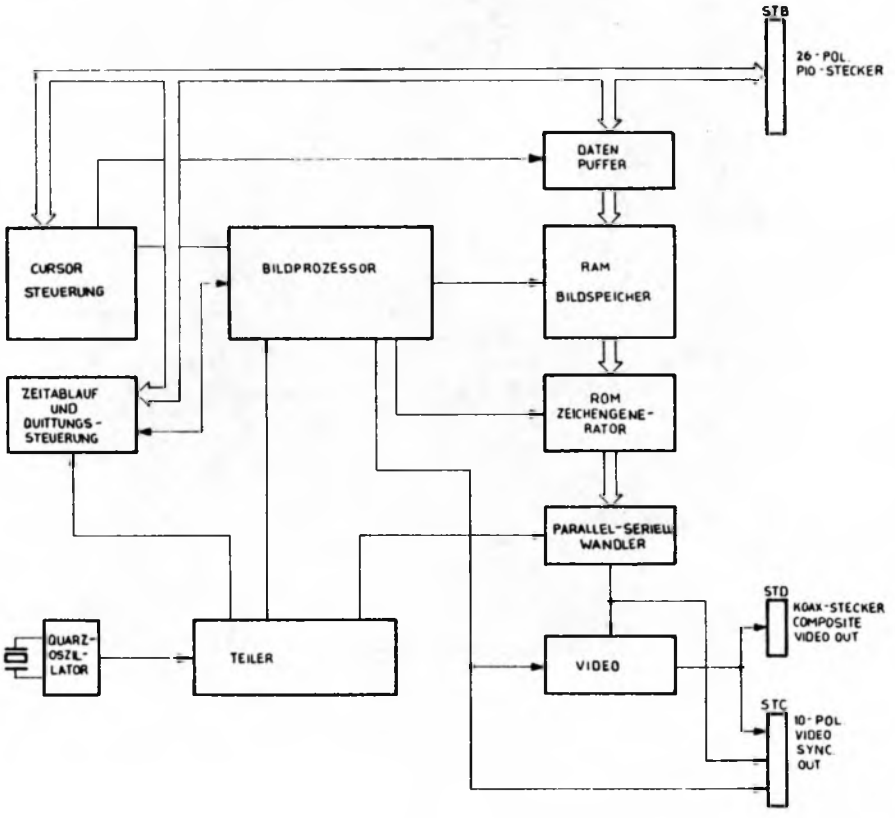
2.1. Schaltungsbeschreibung Z80-KIT/V

Die Baugruppe umfaßt folgende Funktionseinheiten:

Bildprozessor (Video-Controller)
1 K RAM-Bildspeicher
Zeichengenerator
Parallel-Serienwandler
Video-Mischer und Ausgangsverstärker
Anschlußseitig 26-Pol Pio-Stecker, 10 pol. Videostecker für getrennte Video- und Synchronisationssignale, Koaxstecker für Composite-Video-Out Signale.

Aus dem Blockschaltbild ist die Verknüpfung der einzelnen Funktionseinheiten zu erkennen.
Sämtliche I/O Leitungen des PIO's sind u.a. an den PIO-Stecker STB geführt. Die Pinbelegung ist die gleiche wie bei den ECB/C und ECB/K-Baugruppen.
Der Anschluß des Z80-KIT/V erfolgt über eine 26-polige Flachkabelstechverbindung mit dem I/O Port der ECB/C bzw. ECB/K Baugruppe.

Blockschaltbild Z80-KIT/V



2.2. Schaltungsbeschreibung Z80-KIT/VZ

Die Baugruppe umfaßt folgende Funktionseinheiten:

Zwei Parallelschnittstellen (1 Stück Z80-PIO)
Decoder
Bus-Pufferung
Bildprozessor (Video-Controller)
1 K RAM-Bildspeicher
Zeichengenerator
Parallel-Serienwandler
Video-Mischer und Ausgangsverstärker
Anschlußseitig 26-pol. PIO-Stecker, 10-pol. Videostecker für getrennte Video- und Synchronisationssignale, Koaxstecker für Composite-Video-Out Signale.
Busseitig 64-pol. Stecker nach DIN 41612 (VG 95324)

Aus dem Blockschaltbild ist die Verknüpfung der einzelnen Funktionseinheiten zu erkennen.
Die Busleitungen sind gepuffert.

Die Adresse des Z80-PIO kann über DIL-Schalter S1 festgelegt werden.
Es stehen 16 Adressen zur Verfügung. Die Schalterstellung ist invertiert zu betrachten.

ADRESSE	SCHALTER			
	S1/4	S1/3	S1/2	S1/1
F X H	AUS	AUS	AUS	AUS
E X H	AUS	AUS	AUS	EIN
USW.				
I X H	EIN	EIN	EIN	AUS
0 X H	EIN	EIN	EIN	EIN

Durch direkte Auswahl der I/O-Bausteine auf der ECB/C oder ECB/K Karte müssen die Adreßbits 2 und 3 immer auf LOG 1 sein. Bei einer Schalterstellung

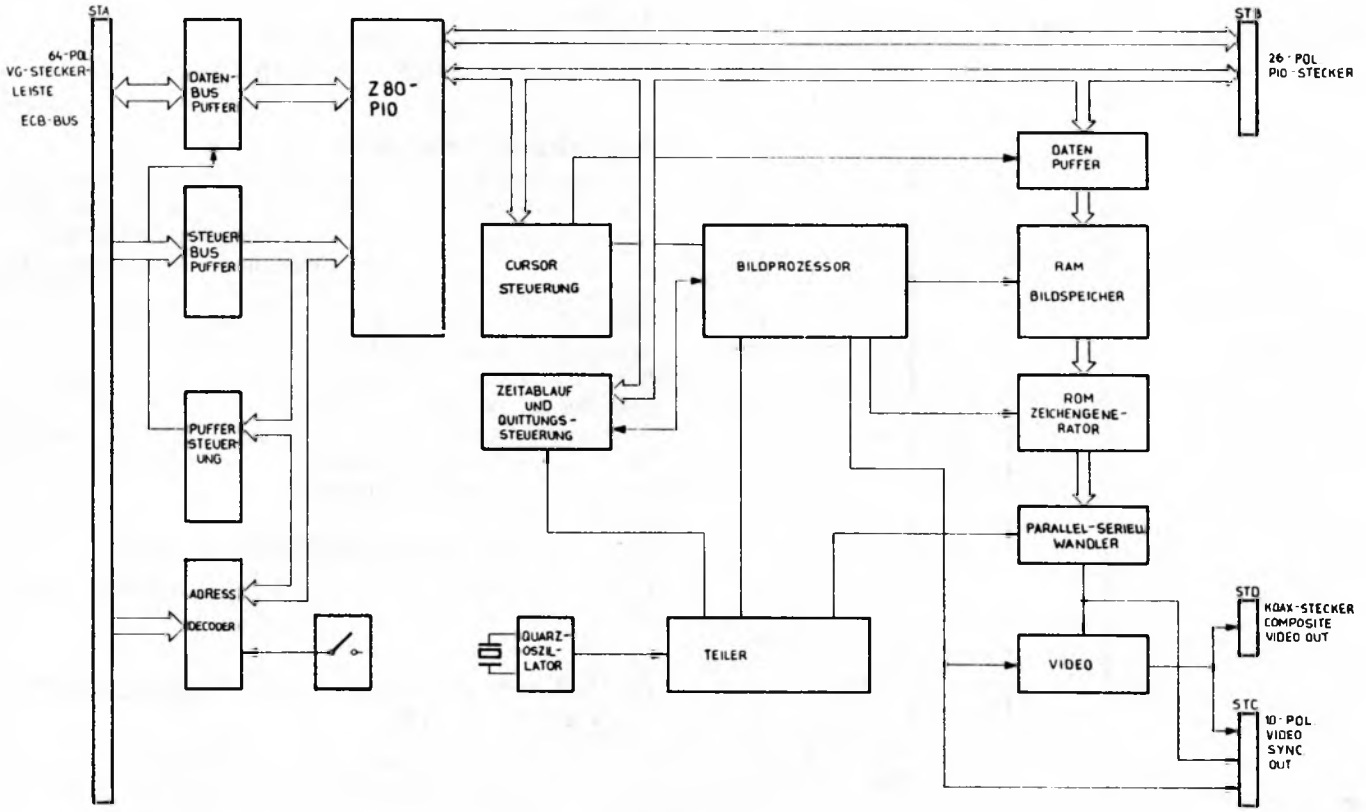
Zum Beispiel:	S1/4	S1/3	S1/2	S1/1
	AUS	EIN	AUS	EIN

Am Schalter S1 muß der PIO wie folgt adressiert werden.
I/O Adresse (hex)

ACH = PIO PORT A DATA
ADH = PIO PORT B DATA
AEH = PIO PORT A CONTROL
AFH = PIO PORT B CONTROL

Sämtliche I/O-Leitungen des PIO's sind an den PIO-Stecker STB geführt. Die Pinbelegung ist die gleiche wie bei den ECB/C und ECB/K-Baugruppen.

Blockschaltbild Z80-KIT/VZ



3. Ansteuerung der Z80-KIT/V bzw. Z80-KIT/VZ Baugruppe

Maximal werden von der Bildschirmansteuerung 120 Zeichen pro Sekunde verarbeitet (entspricht 8,3 Ms/Zeichen). Das Löschen des Bildschirmes benötigt ca. 132 Ms.
Eine entsprechende Zeitablaufsteuerung erfolgt automatisch über die Handshakeleitungen BRDY und BSTB des Z80-PIO in der Betriebsart Output.
(Mode 0, siehe Z80-PIO-Manual)

Arbeitsweise:
Bei Ausgabe eines Zeichens geht die BRDY-Leitung auf LOG1.
Nach Abarbeitung (8,3 Ms, bzw. 132 Ms) durch das Videoboard wird BSTB aktiv und löst einen Interrupt aus. Bei entsprechender softwaremäßiger Bearbeitung kann somit die maximale Datenausgabegeschwindigkeit der Z80-KIT/V bzw. Z80-KIT/VZ Baugruppe genutzt werden.

4. Darstellungsweise

Die Zeichendarstellung erfolgt in 16 Zeilen zu 64 Zeichen. Der Zeichenumfang beträgt 64 Zeichen (ASCII-UPPER CASE), die Darstellung erfolgt in einer 5 × 7 Punktmatrix.
Die augenblickliche Schreibposition wird durch den Cursor angezeigt. Durch Umstecken des Jumpers J1 kann zwischen Darstellung schwarze Zeichen auf weißem Grund oder Invers gewählt werden.

4.1. Zeichensatz ASCII

LSB	MSB 0 1	2	3	4	5
0			0	C	P
1		!	1	A	Q
2		"	2	B	R
3		≠	3	C	S
4		$	4	D	T
5		%	5	E	U
6		&	6	F	V
7	ASCII-	/	7	G	W
8	Control-	(	8	H	X
9	zeichen	)	9	I	Y
A	nicht	*	:	J	Z
B	darstellbar	+	;	K	[
C		,	<	L	/
D		–	=	M	J
E		.	>	N	↑
F		/	?	O	–

4.2. Bildschirm- und Cursorsteuerung

ASCII	KEY	HEX	Bedeutung
BS	CNTRL H	08H	Cursor links
HT	CNTRL I	09H	Cursor rechts
LF	CNTRL J	0AG	Cursor nach unten
VT	CNTRL K	0BH	Cursor nach oben
FF	CNTRL L	0CH	Bild Löschen und Cursor zum Bildanfang
CR	CNTRL M	0DH	Zeilenende löschen und Cursor zum Zeilenanfang
SUB	CONTR Z	1AH	Löschen der laufenden Zeile
ESC	SHIFT+CONTR K	1BH	Zeilenvorschub (Bild 1 Zeile nach O. rollen)
FS	SHIFT+CONTR L	1CH	Cursor zum Bildanfang
GS	SHIFT+CONTR M	1DH	Cursor zum Zeilenanfang

Bermerkung:
Mit Erreichen der untersten Bildzeile erfolgt automatisch Übergabe in den ROLL-MODE.

5. Video-Schnittstelle

Die Z80-KIT/V bzw. Z80-KIT/VZ-Baugruppe ist für den Anschluß eines Schwarz-weiß-Monitors oder eines handelsüblichen SW-Fernsehers mit Video-Eingang ausgelegt. Dadurch erreicht man wesentlich höhere Bildqualität und Störsicherheit gegenüber der Verwendung des Antenneneingangs des Fernsehers.
Bei Verwendung eines Fernsehgerätes ohne Video-Eingang ist ein einfacher Eingriff im Gerät notwendig (siehe 5.3.)

5.1. Monitoranschluß

Der Anschluß eines SW-Monitors ist wahlweise über das Composite-Video-Signal (STC oder STD) oder über Getrennte Video- und Composite-Sync.-Signale (STC) möglich.
Die Impedanz des Ausgangstreibers für das Videosignal (T1) beträgt 75 Ohm +/– 5%.
Koaxleitung wird empfohlen, die Verwendung ist aber bei Leitungslängen unter 1,80 Meter nicht zwingend.
Das Kabel soll am Leitungsende mit 75 Ohm reflektionsfrei abgeschlossen werden. Die Widerstände R7 und R16 bestimmen die Amplitude des Composite-Video-Out-Signals. Eine Anpassung an den Sichtschirm kann durch ändern dieser Widerstände vorgenommen werden.

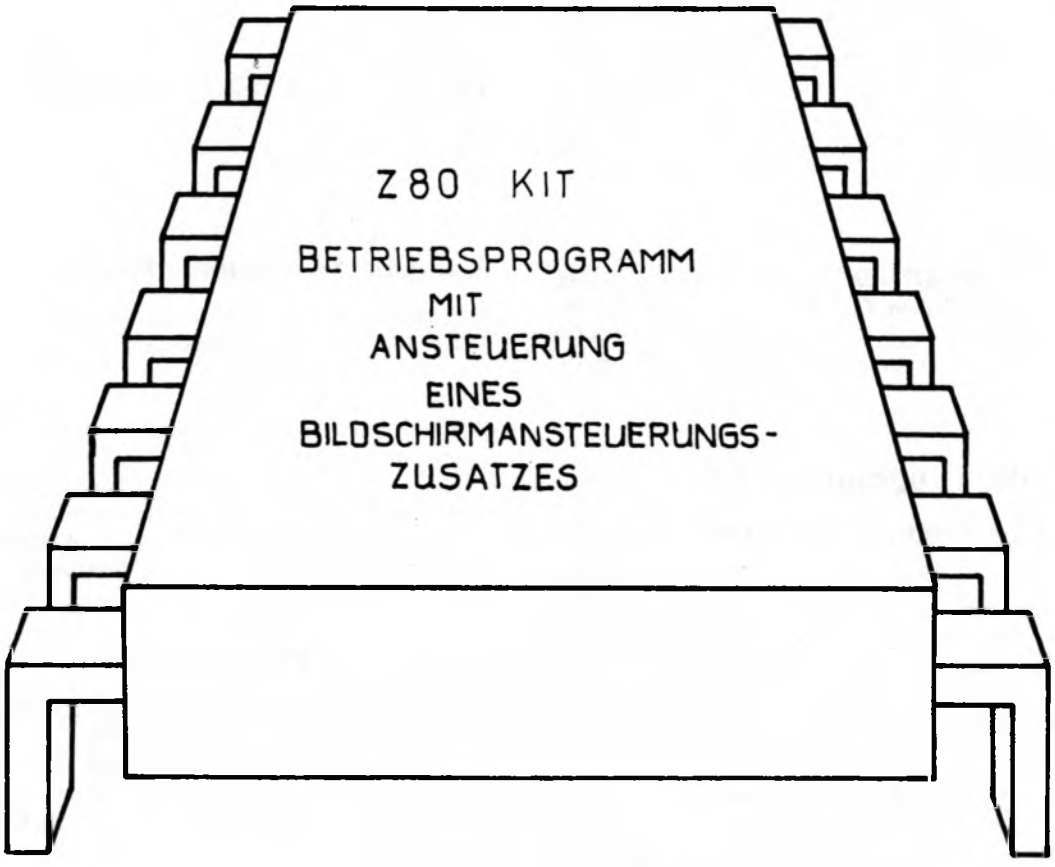
5.2. SW-Fernseher mit Videoanschluß

Der Anschluß ist über STD (Composite-Video-Out) vorgesehen. Hierzu gilt sinngemäß das unter 5.1. Gesagte.

6. Betrieb mit dem Betriebsprogramm Z80-KIT/TV

Beide Baugruppen (Z80-KIT/V und Z80-KIT/VZ) eignen sich im Zusammenhang mit dem TV Betriebsprogramm Z80-KIT/TV zum Aufbau einer Bildschirmanzeige für das Z80-KIT. Für weitere Informationen siehe Teil VI Z80-KIT/TV.

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

1. Einleitung

Beim Z80-KIT/PDT handelt es sich um die PROM-residente Software zu den Erweiterungspaketen KIT/P und KIT/D. Zusätzlich sind Treiberroutinen zur Ansteuerung der TTY-Schnittstelle auf der ECB/K sowie Standardarithmetikroutinen enthalten.
Durch die PROM-residente Version der Programme entfällt die eventuelle Notwendigkeit des Einlesevorgangs von der Cassette. Gleichzeitig kann der sonst vom entsprechenden Programm belegte Speicherbereich anderweitig benützt werden.

2. Einsatz des Z80-KIT/PDT

Der Z80-KIT/PDT wird in einem 2k Byte Festwertspeicher (= „Firmware") geliefert.
Der Festwertspeicher kann ohne Hardwareänderung auf Steckplatz A3 der ECB/K untergebracht werden (Adreßbereich 0800H bis 0FFFH). Ein Einsatz auf einer anderen, entsprechend ausgerüsteten Platine (z.B. ECB/E16) ist möglich, sofern der Adreßbereich 0800H bis 0FFFH dort eingestellt wird.

NB.:
Das Betreiben des Z80-KIT/PDT ist nur im Zusammenhang mit den Betriebsprogrammen Z80-KIT/M2 (1 k Byte) oder Z80-KIT/TV (2 k Byte) möglich, da das Programm Z80-KIT/PDT auf Routinen dieser Betriebsprogramme zurückgreift.

3. Aufruf der Programme

3.1. Aufruf des Promprogrammers

Der Aufruf erfolgt wie bei der Cassettenversion. Dabei wird gemäß dem Starten eines Anwenderprogramms der Programmzähler auf die Startadresse gesetzt und anschließend die Taste START gedrückt. (Siehe auch KIT/P, Teil II)

Einsprungadresse: 0800

3.2. Aufruf der Drucker-Routinen

Der Aufruf erfolgt wie bei der Cassetten-Version. Dabei werden die beiden Programmteile über CALL-Befehle in ein Anwenderprogramm eingebunden.
(Siehe auch KIT/D, Teil III)

ASCII druckt den Inhalt eines Speicherbereiches in Form von ASCII-Zeichen aus.
Startadresse des Quellspeichers beim Aufruf im DE Registerpaar.
Benützte Register: A B C D E F H L I (IM2!)
Benützter Speicher: 3CA0 . . . . 3CC3
Einsprungspunkt: 0E7F

HEX druckt den Inhalt eines Speicherbereichs in hexadezimaler Form aus. Angabe der Adresse am Zeilenanfang.
Startadresse des Quellspeichers beim Aufruf im DE Registerpaar, Anzahl der Bytes im BC Registerpaar.
Benützte Register: A B C D E F H L I (IM2!)
Benützter Speicher: 3CA0 . . . . 3CC3
Einsprungpunkt: 0EE9

3.3. Aufruf der TTY Routinen

Der Aufruf erfolgt über CALL-Befehle aus dem Anwenderprogramm.

USART INITIALISIERUNG
Bringt den auf der ECB/K sitzenden seriellen Schnittstellenbaustein in einen definierten Zustand.
Keine Übergabeparameter
Einsprungpunkt: 0F6B

TTYIN
Wartet auf einen Tastendruck von der angeschlossenen ASCII-Tastatur ASCII-Zeichen der entsprechenden Taste beim Verlasssen der Routine im A-Register.
Benützte Register: A, F
Einsprungpunkt: 0F81

TTYOUT
Gibt einen Charakter an das angeschlossene Device aus. Auszugebendes ASCII-Zeichen beim Einsprung in die Routine im A-Register.
Benützte Register: A, F
Einsprungpunkt: 0F8C

3.4. Aufruf der Standardarithmetik

Es sind folgende drei Routinen verfügbar:
16 × 16-BIT Integer Multiplikation
16/ 8-BIT Integer Division
32/ 16-BIT Integer Division
Der Aufruf erfolgt über CALL-Befehle aus einem Anwenderprogramm.

N.B.: Die Ergebnisse der drei Routinen stellen binäre Werte dar. Sind für eine Darstellung dezimale Werte im BCD-Format notwendig, ist eine entsprechende Umrechnung vorzunehmen!

3.4.1. 16 × 16-BIT Integer Multiplikation

Beim Einsprung in die Routine Multiplikand im DE-Registerpaar, Multiplkator im HL-Registerpaar.
Beim Verlassen der Routine Ergebnis in den Registerpaaren HL (höherwertiger Teil) und DE (niederwertiger Teil)
Benützte Register: A, B, C, D, E, H, L, D', E', H', L'
Einsprungpunkt: 0F97

3.4.2. 16/8-BIT Integer Division

Beim Einsprung in die Routine Dividend im HL-Registerpaar, Divisor im A-Register.
Beim Verlassen der Routine Ergebnis im HL-Registerpaar, Rest im DE-Registerpaar.
Benützte Register: A, B, C, D, E, H, L, B'
Einsprungpunkt: 0FBE

3.4.3. 32/16-BIT Integer Division

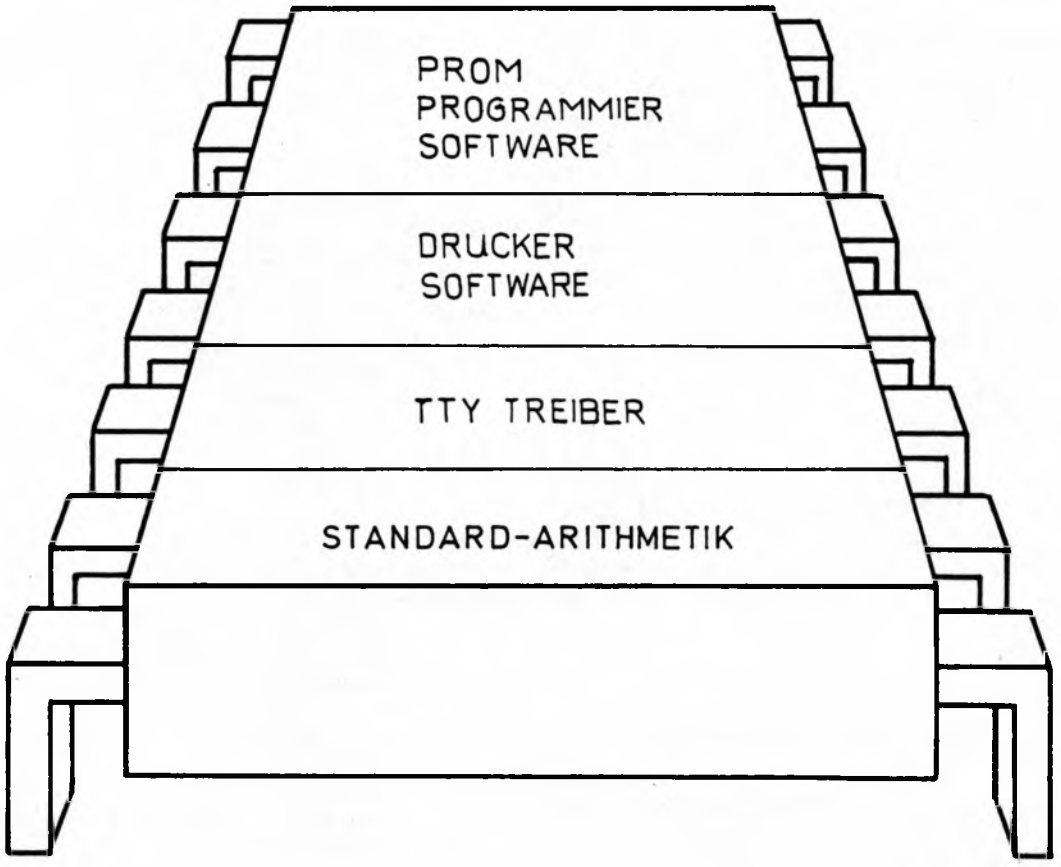
Beim Einsprung in die Routine Dividend in den Registerpaaren HL (höherwertiger Teil) und DE (niederwertiger Teil).
Beim Verlassen der Routine Ergebnis ebenfalls in den Registerpaaren HL (höherwertiger Teil) und DE (niederwertiger Teil).

Benützte Register: A, B, C, D, E, H, L, B', C', H', L'
Einsprungpunkt: 0FDD

KONTRON®

KONTRON-KIT/TV

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

Z80-KIT/TV

1. Einleitung

Z80-KIT/TV ist ein für Bildschirmbetrieb erweitertes Z80-KIT-Betriebsprogramm. Neben den bereits im Z80-KIT Manual beschriebenen Möglichkeiten, verfügt das Z80-KIT/TV Betriebsprogramm über eine zusätzliche Ansteuerung eines TV-Gerätes.
Bedingung für diese zusätzliche Anzeige ist eine der Baugruppen Z80-KIT/V (Bausatz) oder Z80-KIT/VZ. Ohne entsprechende Video-Hardware kann das TV-Betriebsprogramm wie das 1k-Betriebsprogramm benutzt werden.

2. Beschreibung des Betriebsprogramms Z80-KIT/TV

2.1. Allgemeine Beschreibung

Das Z80-KIT/TV Betriebsprogramm wird wie das 1k Betriebsprogramm als Firmware in einem Festwertspeicher geliefert. Es umfaßt 2 kByte und kann ohne jede Hardware-Änderung auf Steckplatz A2 der ECB/K untergebracht werden. Es umfaßt sämtliche Möglichkeiten des 1k-Programms (siehe hierzu Teil I).
Hinzu kommt eine Darstellung der KIT-Kommandos auf dem Bildschirm. Es werden jeweils das Kommando sowie die entsprechenden Daten bzw. Adressen dargestellt. Zusätzlich ist die Ausgabe von Speicherbereichen und Registerblöcken möglich. Da es sich dabei um zusätzliche Kommandos handelt, die mit Hilfe der KIT-Anzeige nicht dargestellt werden können, erfolgt in diesem Fall am KIT-Display keine Reaktion.
Die Darstellung auf dem Bildschirm erfolgt gemäß der Hardware (siehe Z80-KIT/V oder Z80-KIT/VZ) in 16 Zeilen zu 64 Zeichen. Selbstverständlich besteht auch beim Z80-KIT/TV die Möglichkeit, Routinen des Betriebsprogramms vom Anwenderprogramm aus aufzurufen. Eine Aufstellung finden Sie in Kap. 3.2.

2.2. Adreßeinstellung für Z80-KIT/V bzw. Z80-KIT/VZ

Das Betriebsprogramm Z80-KIT/TV ist zur Ansteuerung eines Bildschirms mit Hilfe der Einheiten Z80-KIT/V bzw. Z80-KIT/VZ geeignet. Die beiden Baugruppen unterscheiden sich lediglich durch die Adresse, des zur Ansteuerung verwendeten PIO. Standardmäßig ist das Betriebsprogramm Z80-KIT/TV zur Zusammenarbeit mit dem Z80-KIT/VZ bzw. Z80-KIT/V ausgelegt.

	Adresse für KIT/V Z80-KIT/TV1	Adresse für KIT/VZ Z80-KIT/TV2
Control	0B	FF
Data	09 (= PIO ECB/K)	FD (= PIO KIT/VZ)
Im Betr. prog. auf Adr.		
0497	0B	FF
049B	0B	FF
049F	0B	FF
04B1	09	FD
04D5	09	FD

2.3. Interruptverhalten

Das Betriebsprogramm Z80-KIT/TV arbeitet bezüglich der Bildschirmdarstellung mit Interrupt im MODE 2.
Auf diese Tatsache ist besonders zu achten, wenn die Bildschirm-Ansteuerroutinen des Betriebsprogramms von einem Anwenderprogramm aufgerufen werden, das ebenfalls mit Interrupt arbeitet.
Die vom Betriebsprogramm verwendete Interrupt-Service-Routine kann bei Aufruf der Bildschirm-Ansteuerroutinen durch das Anwenderprogramm mitverwendet werden. Der entsprechende Interruptvektor lautet

0718 H (I Reg. CPU 07)
(I Reg. PIO 18)

und ist bereits durch die Systeminitialisierung in den angegebenen Registern gespeichert. Die Verwendung anderer, vom Anwender erstellter Interrupt-Service-Routinen ist selbstverständlich ebenfalls möglich.

Achtung:
Bei Rücksprüngen vom Anwender ins Betriebsprogramm muß das I Register der CPU auf 07, das I Register der PIO auf 18 gesetzt sein!

2.4. Mitbenutzbare Routinen

Zusammenstellung

OBTAST
prüft, ob eine Taste des KIT Tastenfeldes gedrückt wurde. Bei der Rückkehr sind zwei Fälle möglich:
a) eine Taste wurde gedrückt
ZERO FLAG zurückgesetzt
b) kein Tastendruck
ZERO FLAG gesetzt
keine Übergabeparameter
Benützte Register: A B C D E F H L IX
Einsprungpunkt: 0251

WETAST
prüft, um welche Taste es sich handelt. Bei der Rückkehr sind zwei Fälle möglich:
a) es war immer noch die gleiche Taste gedrückt (während eines Tastendruckes mehrere Abfragen, da Prozessor sehr schnell!) Das A-Register enthält den Wert FF.
b) neue Taste erkannt
Das A-Register enthält den Tastencode der entsprechenden Taste
Keine Übergabeparameter
Benützte Register: A B C D E F H L IX
Einsprungpunkt: 0287

LEDS1
bringt die, in den zur Data Anzeige gehörenden LED Puffern stehenden Werte bei Data zur Anzeige.
Adresse der ‚Marke' beim Aufruf im HL Register-Paar.
a) Bit 0 von Marke gleich 0 → Anzeige ohne Konversion
b) 0 von Marke gleich 1 → Anzeige mit Konversion
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0309

LEDS2
bringt die, in den zur Address-Anzeige gehörenden LED Puffern stehenden Werte bei Address zur Anzeige.
Adresse der ‚Marke' beim Aufruf im HL Register-Paar
a) Bit 0 von Marke gleich 0 → Anzeige ohne Konversion
b) Bit 0 von Marke gleich 1 → Anzeige mit Konversion
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0317

ZEITKO
Zeitschleife von ca. 0,5 sec. Dauer
Keine Übergabeparameter
Benützte Register: A B C F
Einsprungpunkt: 0489

LOAD
Liest auf Cassette abgespeicherte Programme oder Daten ein
Startadresse des Zielspeichers beim Aufruf im Speicher 3C44/45 (siehe Kommando Load)
Benützte Register: A F H L
Einsprungpunkt: 00BE

STORE
speichert im RAM stehende Programme oder Daten auf Cassette ab
Startadresse und Endadresse des Quellspeichers beim Aufruf im Speicher 3C40/41 und 3C42/43 (siehe Kommando STORE)
Benützte Register: A B C D E F H L
Einsprungpunkt: 00FB

ABSPE1
entnimmt den beiden zur Data Anzeige gehörenden LED Puffern die dort gespeicherten (und angezeigten) Halbworte (0,1 . . . F), bildet ein Gesamtwort (00,01 . . . FF) und speichert es in die gewünschte Speicherzelle ab
Adresse der gewünschten Speicherzelle beim Aufruf im IY-Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 041E

ABSPE2
entnimmt den vier zur Address-Anzeige gehörenden LED Puffern die dort gespeicherten (und angezeigten) Halbworte, bildet zwei Gesamtworte (high und low Byte) und speichert sie in die gewünschte Speicherzelle (low Byte) bzw. die nächstfolgende (high Byte) ab
Niedrigere Adresse der beiden gewünschten Speicherzellen beim Aufruf im IY-Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 0426

RESPE1
entnimmt dem gewünschten Speicher den Wert, teilt dieses Gesamtwort auf in zwei Halbworte und belegt damit die beiden zur Data Anzeige gehörenden LED Puffer
Adresse der gewünschten Speicherzelle beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungadresse: 0443

RESPE2
entnimmt dem gewünschten und dem nachfolgenden Speicher den Wert, teilt die beiden Gesamtworte (low und high Byte) jeweils auf in zwei Halbworte und belegt damit die vier zur Address-Anzeige gehörenden LED Puffer
Niedrigere Adresse der beiden gewünschten Speicherzellen beim Aufruf im IY Register
Benützte Register: A B F H L IX IY
Einsprungpunkt: 044B

TAREG1
verschiebt die Inhalte der zur Data Anzeige gehörenden LED Puffer um eine Stelle in Richtung höherwertiger Stelle. Die dadurch freiwerdende unterste Stelle (Data 1) wird mit einem neuen Wert belegt
Neu einzulesender Wert beim Aufruf im A-Register
Benützte Register: A B C D E F H L
Einsprungpunkt: 036A

TAREG2
verschiebt die, in den zur Address-Anzeige gehörenden LED Puffern stehenden Werte in Richtung höherwertiger Stellen. Die freiwerdende unterste Stelle (Address1) wird mit einem neuen Wert belegt
Neu einzulesender Wert beim Aufruf im A-Register
Benützte Register: A B C D E F H L
Einsprungpunkt: 037E

ANZABF
zusammen gesetzte Routine aus LEDS1, LEDS2, OBTAST und WETAST, zeigt die in den LED Puffern gespeicherten Werte an und führt eine Tastenabfrage durch. Rücksprung bei Erkennen einer neuen Taste.
Adresse der ‚Marke' im HL Registerpaar
Bit 0 siehe LEDS1 bzw. LEDS2
a) Bit 1 von Marke gleich 1 → Data Anzeige
b) Bit 1 von Marke gleich 0 → keine Data Anzeige
c) Bit 2 von Marke gleich 1 → Address Anzeige
d) Bit 2 von Marke gleich 0 → keine Address Anzeige
Benützte Register: A B C D E F H L IX IY
Einsprungpunkt: 0400

FEHLER
bringt die ERROR Lampe zum Leuchten
Keine Übergabeparameter
Benützte Register: A F
Einsprungpunkt: 03B7

FEHLEX
löscht die ERROR Anzeige
Keine Übergabeparameter
Benützte Register: A F
Einsprungpunkt: 03BB

Folgende Unterprogramme sind nur in Verbindung mit einer Platine ECB-V oder ECB-VZ sinnvoll.

Bitte beachten Sie unbedingt, daß vor Aufruf dieser Unterprogramme das User-I-Register auf 07 gesetzt ist (siehe 2.3.).

SCREEN
Bildschirm wird gelöscht und Cursor geht nach links oben
Keine Übergabeparameter
Benützte Register: A B C F
Einsprungpunkt: 04AE

TVOUT
Das in Register A enthaltene ASCII-Zeichen wird auf dem Bildschirm ausgegeben bzw. wenn das in Register A enthaltene ASCII-Zeichen ein Cursor-Befehl ist, so wird dieser ausgeführt.
Keine Übergabeparameter
Keine Registerbeeinflussung
Einsprungpunkt: 04CB

PUTA
Die beiden im Register A enthaltenen Hexadezimalhalbbytes werden als 2 ASCII-Zeichen auf dem Bildschirm dargestellt (0 . . . F)
Keine Übergabeparameter
Keine Registerbeeinflussung
Einsprungpunkt: 04EB

PUTAB
Wie PUTA, jedoch folgt nach den beiden Zeichen ein BLANK.
Einsprungpunkt: 04FC

LINE FEED
Cursor geht zum Zeilenanfang der nächsten Zeile
Keine Übergabeparameter
Keine Registerbeeinflussung
Einsprungpunkt: 0507

PUTHL
Die im Registerpaar HL enthaltenen 4 Hexadezimalhalbbytes werden als 4 ASCII-Zeichen auf den Bildschirm ausgegeben (0 . . . F). Nach dem 4. Halbbyte folgt ein BLANK.
Keine Übergabeparameter
Keine Registerbeeinflussung
Einsprungpunkt: 051E

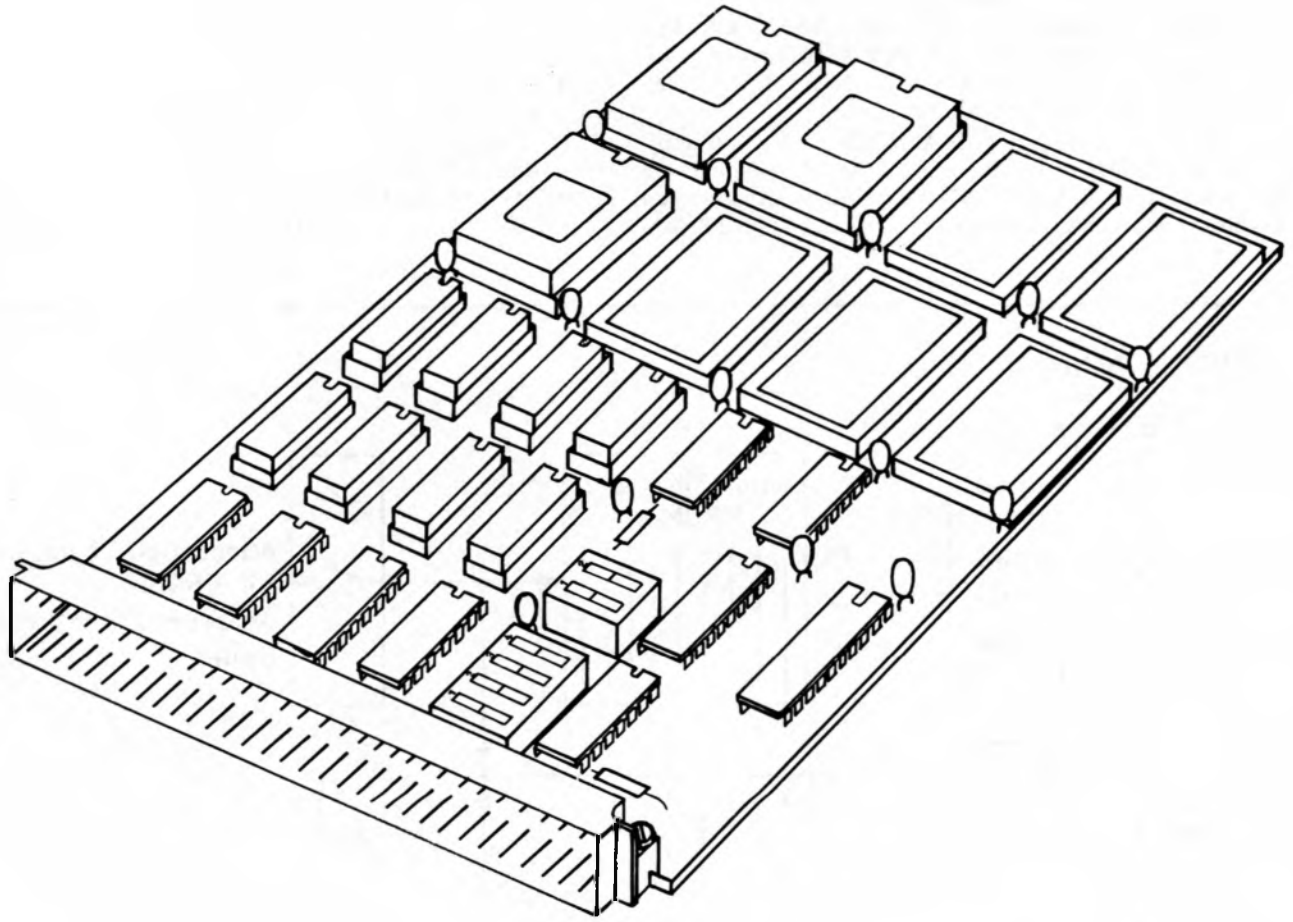
OUTSTR
Mit diesem Unterprogramm ist es möglich, lange Textblocks auszugeben. Die Anfangsadresse des abgespeicherten Textblockes muß im Registerpaar HL stehen. Der Text muß mit dem Zeichen * enden. (Dieses Zeichen = 2A). Das Endezeichen * wird nicht ausgegeben.
Benützte Register: HL (= Textzeiger)
Einsprungpunkt: 0510

ROUT1
Die Register A B C D E F H L IX IY PC SP werden incl. Überschrift auf den Bildschirm ausgegeben.
Benützte Register: A F H L
Keine Registerbeeinflussung
Einsprungpunkt: 0537

ROUTX
Wie ROUT;, jedoch ohne Überschrift
Einsprungpunkt: 0546

ROUT2
Wie ROUT1, jedoch werden statt der Register A . . . F die Zweitregister A′ . . . F′ ausgegeben.
Einsprungpunkt: 0579

KONTRON
ELEKTRONIK GMBH



8. Z80-KIT-EINFACH-COMPUTER UND LERNSYSTEM

1. Übersicht

KIT/ZBASIC ist neben den bereits bestehenden Bausteinen eine weitere Möglichkeit zum Ausbau des bekannten Billig-Computer-Systems Z80-KIT.
Die Erweiterung besteht aus einer Steckkarte im Einfach-Europaformat, auf der der BASIC-Interpreter ZBASIC in PROM-residenter Form untergebracht ist. Für den Aufbau von Benutzerprogrammen steht zusätzlich statisches RAM zur Verfügung.
ZBASIC stellt einen äußerst leistungsfähigen BASIC-Interpreter dar, der speziell über ein umfangreiches Mathematikpaket verfügt. Als Beispiele sei die Möglichkeit zur Berechnung des Hyperbelsinus oder der Arcustangensfunktion genannt.
ZBASIC ist für den Betrieb mit einer ASCII-Tastatur und einer Bildschirmanzeige vorbereitet.
Die Ausgaben erfolgen über eine parallele Schnittstelle. Das verwendete Format paßt zu den beiden Bildschirm-Ansteuerungs-Einheiten Z80-KIT/V bzw. Z80-KIT/VZ, die beide ohne Veränderung eingesetzt werden können.
Die Eingaben werden von ZBASIC über eine interruptgesteuerte Parallelschnittstelle erwartet. Bei Verwendung von Z80-KIT/V bzw. Z80-KIT/VZ ist ein direkter Anschluß an den noch verbleibenden Port der entsprechenden PIO möglich.

KIT/ZBASIC kann durch Einstecken in einen freien Steckplatz des Z80-KIT-Busses und anschließendes Starten sofort in Betrieb genommen werden.
Einzige Voraussetzung hierfür ist das Vorhandensein einer ASCII-Tastatur und einer Bildschirmanzeige.

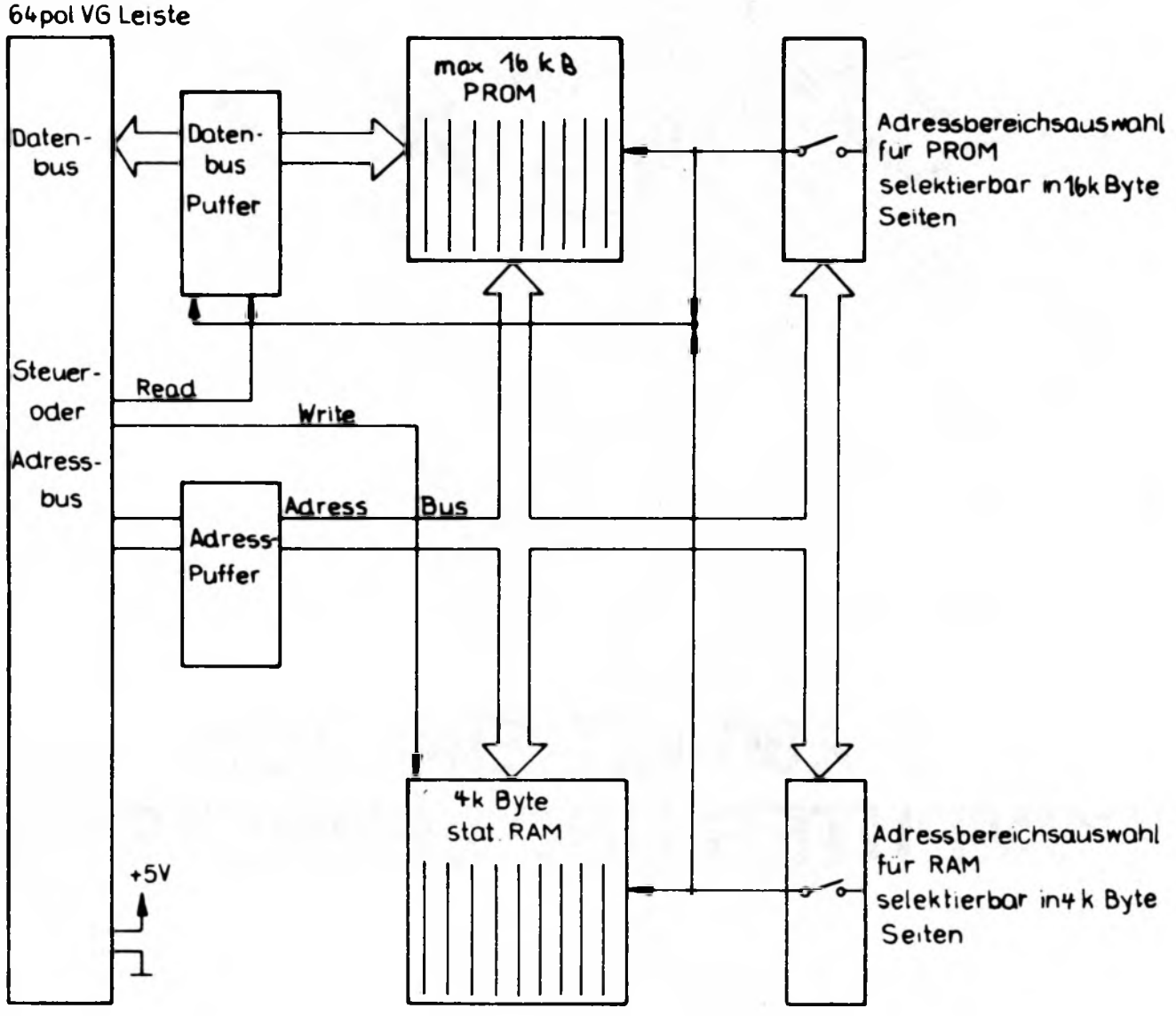
2. Schaltungsbeschreibung

2.1. Hardwareausstattung

KIT/ZBASIC enthält folgende Hardwarekomponenten:

- 6 KByte ROM (ZBASIC in insg. drei EPROMs)
- Platz für den Aufbau von weiteren max. 10 KByte ROM
- 4 KByte statisches RAM
- Einstellung der Speicheranfangsadressen von ROM und RAM in Schritten zu 16 bzw. 4 KByte
- volle Daten- und Adreßbuspufferung; Busanschlußwert = 1 TTL-LS Last.
- BUS-Anschluß über 64-poligen Stecker nach DIN 41612 (VG 95324).

2.2. Blockschaltbild



Zusammenstellung

1. Verwendbare Zeichen

☐ Einfache Variable
ein Buchstabe von A bis Z oder
ein Buchstabe mit einer nachfolgenden Ziffer

z. B. D, F5, E2, A, Z, I9, W0, etc.

☐ Zahlen aus dem Bereich

$\pm 1{,}7\ 10^{38}$ bis $\pm 2\ 10^{38}$

mit sieben signifikanten Dezimalstellen
Eingabe in beliebigem Format, mit oder
ohne Dezimal-Exponent.

2. Implementierte Mathematik

☐ Addition
☐ Subtraktion
☐ Multiplikation
☐ Division
☐ Exponenzierung
☐ Exponenzierung zur Basis e
☐ Logarithmus zur Basis e
☐ Quadratwurzel
☐ SINUS
☐ COSINUS
☐ TANGENS
☐ Arcustangens
☐ Hyperbelsinus
☐ Absolutbetrag
☐ Integer
☐ Signum
☐ Zufallszahl

Weitere Einzelheiten über KIT-ZBASIC finden Sie in dem 190 Seiten starken „KONTRON-Z80-KIT-Anwenderhandbuch".

3. ZBASIC-Beschreibung

BASIC ist eine leicht zu erlernende Programmiersprache und hat eine weite Verbreitung gefunden. Sie wurde für den den interaktiven Betrieb mit dem Computer entwickelt, bei dem der Programmierer im direkten Dialog mit dem Computer steht, der ihn schon bei der Eingabe auf Syntaxfehler aufmerksam macht. Das eingegebene Programm kann sofort gestartet werden, da es nicht in Maschinensprache übersetzt wird, sondern nur Anweisungen an den Computer gibt, der es „interpretiert". Entsprechen die Ergebnisse des Programmes nicht den Vorstellungen, können beliebig Zeilen (Statements) ersetzt, gelöscht oder eingeschoben werden und danach ein neuer Versuch gestartet werden. Diese Eigenschaften verkürzen die benötigte Zeit für die Programmentwicklung ganz wesentlich.

Das im folgenden beschriebene ZBASIC-Interpreter-Programm wurde für den Z80-Mikroprozessor geschrieben. Es ist ebenso komfortabel in der Bedienung wie die für die größeren Computerbrüder geschriebenen Interpreter und steht in der Genauigkeit nicht, und in der Geschwindigkeit nur wenig seinen „großen Brüdern" nach.

Hauptbestandteile der BASIC-Programme sind im Allgemeinen numerische Berechnungen. Deshalb stellt ZBASIC eine Gleitkommaarithmetik, einen Algorithmus zur Lösung algebraischer Ausdrücke und die wichtigsten transzendentalen Funktionen zur Verfügung.

Die verwendete Syntax ist allgemein unter dem Namen Darthmouth-Basic bekannt (nach dem Namen des Ortes an dem sie entwickelt wurde). Erweitert wurde die Sprache durch Eingabe, Ausgabe- und Unterbrechungs-Befehle, die es gestatten, das vorliegende Programm zur Steuerung in Echtzeit zu benutzen („Prozess-Basic"). Damit eröffnet sich die Möglichkeit, das BASIC-Programm auch in Geräten oder Anlagen zu verwenden. Besonders dann, wenn im großen Umfang numerische Daten zu verarbeiten sind.

Werden in ZBASIC-Programmen Fehler festgestellt, können Unterbrechungspunkte gesetzt werden. Im „Direkt-Mode" lassen sich dann sehr einfach Speicherstellen lesen und evtl. verändern.

Ist das Programm ausgetestet, kann es in PROMs oder EPROMs abgespeichert werden und auch in einem anderen Gerät abgearbeitet werden. Die vom Interpreter aufgebaute Programmdatei ist adress-unabhängig! Auch mehrere BASIC-Programmteile können also unabhängig voneinander geprüft und anschließend zusammengesetzt werden, solange sich die Zeilennummern nicht überschneiden.

Soll das fertige BASIC-Programm nur noch ausgeführt werden, so sind die Editing-Funktionen, die Syntaxprüfung und der LIST-Teil des Interpreters überflüssig. Es ist nur noch ein relativ kurzer „RUNTIME" Programmteil der BASIC-Datei selbst notwendig.

Die im Interpreter verwendete Gleitkommaarithmetik stellt bei der gewählten Auflösung ein Optimum bezüglich der entwickelten Verarbeitungsgeschwindigkeit dar.

Die Vorteile der sofortigen Übersetzung des BASIC-Programms in eine neue Datei, der „quasi Compilierung", wie sie der ZBASIC-Interpreter durchführt, sollen noch einmal zusammengefaßt werden:

1. der benötigte Speicherplatz ist gering,
2. der Ablauf ist schnell und
3. das Programm ist mit einem kleinen RUNTIME-Modul lauffähig.

PSI ψ 80

Z80A-Mikrocomputer-Anwendersystem-Familie

KONTRON
ELEKTRONIK GMBH

9. Z80A-MIKROCOMPUTER-SYSTEME

PSI Ψ 80

Produktübersicht

Die Computersysteme der Serie PSI Ψ 80 sind betriebsbereite, besonders preisgünstige Anlagen für industrielle und private Anwendungen. Sie beinhalten einen Computer auf Basis der ZILOG Z80A-CPU mit 4 MHz Taktfrequenz (Befehlsausführungsdauer min. 1 µsec.) mit 32 kByte Speicher, Bildschirm, graphikfähigem Bildprozessor, Ein/Ausgabe-Schnittstellen und Tastatur.

Das System ist in einem kompakten ansprechenden Gehäuse untergebracht, das sowohl auf dem Arbeitstisch verwendet, als auch in 19''-Racks eingebaut werden kann. Die Tastatur ist getrennt.

Die Software-Ausstattung umfaßt neben Betriebssystemen, Fehlersuchpaketen, Editier- und Assemblierprogrammen auch höhere Sprachen wie z. B. BASIC, FORTRAN und PASCAL.

Hardware-Beschreibung

Basis der PSI Ψ 80 -Computerserie ist eine hochintegrierte Computerbaugruppe, die eine vollständige Zentraleinheit mit weitreichenden Ausbaumöglichkeiten umfaßt.

Auf dieser Platine sind in der maximalen Ausbaustufe möglich:

Zentralprozessor, Programm- und Datenspeicher (max. 8 kByte PROM, 64 kByte RAM), Bildschirmprozessor mit 16 kByte Wiederholspeicher, Tastaturanschluß, Controller für 2 Minifloppy-Laufwerke (600 kByte), Parallel- und Serial-Ein/Ausgabe, Steuerung für direkten Speicherzugriff (DMA), Zähler/Zeitgeber, Statusschalter und Businterface.

Zusätzliche Ausbaumöglichkeiten bietet ein optionaler Einschubrahmen für anwendungsorientierte Funktionaleinheiten. Als verfügbare Busstandards für diese Erweiterung wurden der ECB-Bus und der S 100-Bus gewählt, wodurch sowohl den industriellen, als auch den Hobby-Anwendern Rechnung getragen wird.

Beide Kartentypen sind gemischt einsetzbar. Als Erweiterungen kommen unter anderem in Frage:

- ☐ Subprozessor-Baugruppen zur Realisierung leistungsfähiger Multiprozessor-Systeme
- ☐ Arithmetikprozessor-Baugruppen für rechenintensive Anwendungen (z. B. Z80A-ECB/V)
- ☐ Bussteuerungs-Schnittstellen (z. B. IEC/IEEE-Controller Z80A-ECB/B)
- ☐ Nichtflüchtige Speicher für Stromausfallgesicherte Systeme (z. B. Z80A-ECB/V)
- ☐ Funktionssteuerungs-Baugruppen (z. B. Infrarot-Fernsteuerung, Joystick-Steuerung)
- ☐ Standard Ein/Ausgabe-Erweiterungen aller Art

Die PSI Ψ 80 -Computerserie umfaßt derzeit vier Ausbaustufen:

PSI Ψ 80-H	Computer mit ZILOG's Z80A-CPU 4 MHz-Takt, eingebautem Bildschirm mit 25 Zeilen à 80 Zeichen, getrennte alphanumerische Tastatur mit Groß- und Kleinschreibung und deutschen Umlauten, Bildprozessor, 32 kByte Schreib/Lesespeicher, Audio-Cassetten-Interface, Zähler/Zeitgeber, 2 kByte PROM-residentes Betriebssystem , Texteditor, BASIC-Interpreter, Z80-Assembler, Fehlersuchprogrammpaket	**H**eimcomputer und industrielle Systeme ohne Massenspeicher
PSI Ψ 80-S	Computer mit ZILOG's Z80A-CPU 4 MHz-Takt, eingebautem Bildschirm mit 25 Zeilen à 80 Zeichen, getrennte alphanumerische Tastatur mit Groß- und Kleinschreibung und deutschen Umlauten, Bildprozessor, 48 kByte Schreib/Lesespeicher, Audio-Cassetten-Interface, Zähler/Zeitgeber, 2 kByte PROM-residentes Betriebssystem, Texteditor, BASIC-Interpreter, Z80-Assembler, Fehlersuchprogrammpaket, 1 Floppy-Disk-Laufwerk, vorbereitet auf 80 kByte Speicherausbau	**S**tandard-Computersystem mit 1 Mini-FD-Laufwerk
PSI Ψ 80-P	Computer mit ZILOG's Z80A-CPU 4 MHz-Takt, eingebautem Bildschirm mit 25 Zeilen à 80 Zeichen, getrennte alphanumerische Tastatur mit Groß- und Kleinschreibung und deutschen Umlauten, Bildprozessor, 48 kByte Schreib/Lesespeicher, Audio-Cassetten-Interface, Zähler/Zeitgeber, 2 kByte PROM-residentes Betriebssystem, insgesamt zwei Vollduplex-Serien-Ein/Ausgabe-Kanäle mit Datenübertragungsprotokollen und RS 232-Pegel, 4 für den Anwender verfügbare Ein/Ausgabe-Schnittstellen, Composite-Video-Interface, vorbereitet auf 80 kByte Speicherausbau und DMA, Einschubrahmen für Erweiterungs-Baugruppen	**P**rozeßsteuerungs-orientiertes System zum Einsatz in der Meß/Steuer/Regeltechnik
PSI Ψ 80-M	Computer mit ZILOG's Z80A-CPU 4 MHz-Takt, eingebautem Bildschirm mit 25 Zeilen à 80 Zeichen, getrennte alphanumerische Tastatur mit Groß- und Kleinschreibung und deutschen Umlauten, Bildprozessor, 80 kByte Schreib/Lesespeicher, Audio-Cassetten-Interface, Zähler/Zeitgeber, 2 kByte PROM-residentes Betriebssystem, Texteditor, BASIC-Interpreter, Z80-Assembler, Fehlersuchprogrammpaket, insgesamt zwei Vollduplex-Serien-Ein/Ausgabe-Kanäle mit Datenübertragungsprotokollen und RS 232-Pegel, für den Anwender verfügbare Ein/Ausgabeschnittstellen, Composite-Video-Interface, 80 kByte Schreib/Lesespeicher, 2 Floppy-Disk-Laufwerke, vorbereitet für DMA.	**M**inirechner für Anwendungen mit 2 Mini-Floppy-Disk-Laufwerken

Diese Grundkonfigurationen lassen sich durch standardmäßige Erweiterungszusätze in Hardware und Software ergänzen.
Die Aufrüstung kann in den meisten Fällen vom Anwender selbst durchgeführt werden; lediglich bei der Version PSI Ψ 80 H muß der Umbau beim Hersteller erfolgen.
Aufschluß über solche Aufrüstungssätze, die eine Eingriff in die Hardware des Grundsystems bedeuten, gibt die nachstehende Tabelle.

Erweiterungssatz	Lieferumfang	Aufrüstung von PSI Ψ 80-H	Aufrüstung von PSI Ψ 80-P	Aufrüstung von PSI Ψ 80-S
PSI Ψ 80-FD1	**F**loppy-**D**isk-Erweiterung für PSI Ψ 80 -H und PSI Ψ 80 -P, bestehend aus: – 1 FD-Laufwerk – 1 FD-Prozessor und Zusatzlogik – Betriebssystem KOS – Diskresidenter EDITOR, ASSEMBLER, Dateiverwaltungspaket, Testprogramme – Elektromechanisches Zubehör	beim Hersteller	kundenseitig oder beim Hersteller	Standard
PSI Ψ 80-FD2	Zweites **FD**-Laufwerk, bestehend aus: – 1 FD-Laufwerk – Elektromechanisches Zubehör		kundenseitig oder beim Hersteller	kundenseitig oder beim Hersteller
PSI Ψ 80-RM6	Einschub-**R**ahmen zur Erweiterung des Systems, um Baugruppen im Europaformat und S-100 (für PSI Ψ 80 -H und -S), bestehend aus: – Einschubrahmen – Buspufferung – Elektromechanisches Zubehör	beim Hersteller	Standard	kundenseitig oder beim Hersteller
PSI Ψ 80-S16	16 kByte dynamische Speichererweiterung (max. Ausbaumöglichkeit auf der Zentralbaugruppe ist 80 kByte).	beim Hersteller	kundenseitig oder beim Hersteller	kundenseitig oder beim Hersteller
PSI Ψ 80-PER	Erweiterung um Peripherieansteuerungen, bestehend aus: – Zweiter Serien-Ein/Ausgabekanal – RS 232-C-Schnittstelle mit ±12 V Pegel – 4 8-Bit Ein/Ausgabe-Schnittstellen – Composite Video-Schnittstelle	beim Hersteller	Standard	Standard
PSI Ψ 80-DMA	Ausbau auf direktem Speicherzugriff (**D**irect **M**emory **A**ccess)		kundenseitig oder beim Hersteller	kundenseitig oder beim Hersteller
PSI Ψ 80-S100	Zwischenstück zur Verwendung von S100-Karten (1 Stück PSI Ψ 80 pro einzusteckender S100-Karte erforderlich).	–	für Einschubrahmen	–

10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

Das Zilog Z80-MCZ Mikrocomputersystem ist ein Allzweckcomputer mit hoher Leistungsfähigkeit zu einem niedrigen Preis. Er eignet sich insbesondere zur Realisierung von Mikrocomputeranwendersystemen, die Floppy Disks als Massenspeicher verwenden, ohne jeglichen Hardware-Entwicklungsaufwand und mit einem stark reduzierten Software-Entwicklungsaufwand. Durch die hohe Integration der Elektronik und die ausgereifte Technik der verwendeten Floppy Disk-Laufwerke wurde eine hohe Zuverlässigkeit des Systems bei extrem niedrigem Wartungsaufwand erreicht. Kernstück des MCZ-Systems ist der Z80-CPU Mikroprozessor, der maschinencodekompatibel zum 8080A System ist und darüberhinaus auch über eine Vielzahl besonders effizienter Befehle verfügt (insges. 158), wodurch Entwicklungsaufwand und Speicherkosten gegenüber bisher lieferbaren Systemen in hohem Maße reduziert werden (typisch um 50%). Das MCZ beinhaltet in seiner Standardkonfiguration neben der Zentraleinheit 16 kByte Schreib/Lese-Speicher, 3 kByte Festwertspeicher, Parallel- und Serienschnittstellen, 2 Floppy Disk Laufwerke. Die Speicherkapazität des vom Z80-CPU adressierbaren Schreib/Lese-Speichers ist bis zu 64 kByte ausbaubar.

☐ Systemkonzept

Das Zilog Mikrocomputersystem Z80-MCZ ist ein weiterer Schritt in Richtung der Verwendung vorhandener Hardware- und Software-Standards auf dem Gerätesektor und Datenverarbeitungssektor an Stelle von jedesmal neu zu entwickelnden Speziallösungen. Das Z80-MCZ ist so konzipiert, daß es direkt als OEM-Einheit in einem Mikrocomputer-Anwendersystem zu verwenden ist. Der Entwicklungsaufwand beschränkt sich auf das Design der Interfaceschaltungen und der Erstellung und des Austestens eines Anwenderprogramms, wobei auch dabei selbstverständlich auf von Zilog zur Verfügung gestellte Anwenderprogrammteile (Anwenderbibliothek) zurückgegriffen werden kann. Darüberhinaus ist auf dem MCZ die gesamte von Zilog gelieferte Betriebssoftware ablauffähig, wie der absolute Assembler, der relokative Assembler (RIO), Basic Interpreter, PL/Z und FORTRAN Cobol-Compiler usw.

☐ Systemeigenschaften

Zentraleinheit: Z80-CPU (158 Befehle, 2,5 MHz)

- Speicherkapazität: standardmäßig mitgeliefert werden 60 kByte dyn. Speicher und 3 kByte Festwertspeicher (Betriebsprogramm).
- Massenspeicher (Floppy Disks oder Hard Disks)
- Serienschnittstelle (RS 232 und Stromschleife 20 mA) zum Anschluß eines Bedienterminals oder eines Magnetkassettengerätes ist standardmäßig eingebaut. Die Datenübertragungsrate reicht von 10 bis 38600 baud.
- Dem Anwender stehen zwei 8bit Parallel-Ein/Ausgabe-Kanäle incl. zwei Handshaking-Leitungen zur Verfügung.
- Steckplätze zur Erweiterung des Systems um Speicher-, Ausgabe-, oder sonstiger (auch kundenspezifischer) Steckkarten.
- Mechanischer Aufbau: Massives Metallgehäuse.
- Alternativ in Reck-Einbauversion lieferbar.
- Mitgelieferte Standardbetriebssoftware:
  - ☐ PROM-residente Ein/Ausgabesteuerung, Fehlersuchprogramme und Kaltstartprogramm (bootstrap)
  - ☐ Texteditor
  - ☐ Plattenbetriebssystem
  - ☐ RIO relokativer Makro-Assembler und Linker
- Dokumentation

☐ Eigenschaften der Zentraleinheit

Kernstück des Z80 MCZ ist der Z80-CPU-Mikroprozessor, der modernste Technologie, höchste Leistungsfähigkeit und volle Kompatibilität zu den Standardcomputersystemen miteinander verbindet. Hervorstechendste Eigenschaft ist seine fortschrittliche Architektur, die eine Reduktion der zur Lösung eines bestimmten Problems nötigen Befehle um typisch 50% ermöglicht. Hierdurch werden Programmentwicklungs- und Testkosten eingespart, was besonders bei in kleinen Stückzahlen gefertigten Geräten ausschlaggebend ist; gleichzeitig wird die Anzahl der im System benötigten Programmspeicherbausteine reduziert (wichtig für Systeme in großen Stückzahlen) und die Verarbeitungsgeschwindigkeit in hohem Maß gesteigert. Erzielt wird diese Wirkung durch die folgenden Eigenschaften der Z80-CPU:

- **zwei** vollständige Registersätze zur schnellen Behandlung von Interrupt und Unterprogrammen.
- ein Registersatz enthält einen 8bit Akkumulator, ein Flagregister und sechs allgemeine 8bit Register, die auch als 16 bit Speicheradreßregister verwendet werden können.
- Zusätzlich zwei 16 bit Indexregister und ein 16 bit Stapelspeicherzeiger (stackpointer) zur unbegrenzten Verschachtelung von Unterprogrammen, Realisierung von Zwischenspeicherbereichen und Vektorinterruptbehandlung.
- Echte Speicher — indirekte Interruptbearbeitungstechnik möglich (auf Interruptmode 2); dadurch ist eine in der Mikrocomputertechnik bisher unerreichte Flexibilität in der Interruptbehandlung möglich, wie sie bisher nur bei Prozeßrechnern bekannt war.
- In der CPU eingebauter Refresh Controller zum direkten Anschluß von dynamischen Speichern an die Z80-CPU ohne zusätzliche Hard- oder Software. Zu den genannten architektonischen Eigenschaften kommen folgende Softwarebesonderheiten:
  - ☐ Befehle zur Behandlung von 4 bit, 8 bit und 16 bit Datenwörter
  - ☐ Befehle zum Register rotieren **und** schieben
- Als einziger Prozessor ist die Z80-CPU in der Lage sowohl Einzelbits als auch ganze Dateien mit einem einzigen Befehl zu bearbeiten (Blocktransfers mit einer Blockgröße bis zu 64 kByte bei einer Übertragungsrate von 8,4 μsec/Byte, Blocksuchbefehl, Block-Ein/Ausgaben mit einer Ein/Ausgaberate von 125 kByte/sec, Setzen, Testen oder Rücksetzen irgend eines einzelnen Bits in einem der CPU Register oder einer Speicherstelle).

☐ Beschreibung

Das Z80-MCZ ist eine nach Anlegen der Versorgungsspannung (220 V, 50 Hz) sofort arbeitsfähige Anlage mit Massenspeicher. Der Dialog mit dem Computersystem findet über eine Serienschnittstelle (RS 232 oder 20 mA Stromschleife) statt, wodurch der Anschluß eines jeden Standardbildschirms oder teletypeähnlichen Fernschreibers möglich ist. Die Datenübertragungsfrequenz ist im Bereich von 10 bis 38600 baud einstellbar. Zusätzlich zu diesem Anschluß der Bedienkonsole sind an der Rückwand des Systems weitere Stecker zum Anschluß von Prozeß- oder sonstiger Peripherie vorgesehen. Die freien Kartensteckplätze können durch Wirewrap-Technik vom Anwender entsprechend der von ihm verwendeten Karten beliebig verdrahtet werden.

☐ Z80-MCZ-Betriebssoftware

Zu dem Standardlieferumfang des Z80-MCZ gehört das Betriebssystem RIO, das folgende Programme umfaßt:

- Bootstrap- und Monitor-Firmware
- Platte-Betriebssystem, Ein/Ausgabe-Treiber-Routinen
- Editorprogramm
- relokativer Makroassembler und Linker
- Fehlersuchprogramm

Das Z80-MCZ beinhaltet einen Festwertspeicherbereich (nicht flüchtig) von 3 kByte, in dem die Bootstrap-Routine, das Bedienkonsolentreiberprogramm, das Floppy Disk Driver-Programm und Fehlersuchprogramme untergebracht sind. Über dieses Programmpaket greift der Anwender auf weitere umfangreiche Betriebsprogramme zu, die auf Floppy Disk gespeichert sind: Das ZDOS-Betriebssystem, das MCZ-Executiv-Programm, den hochleistungsfähigen Texteditor und einen Makro-Assembler. Durch Anwendung dieser Technik wird einerseits teurer Schreib/Lesespeicher gespart, andererseits die Speicherfähigkeit des Systems gegenüber herkömmlich aufgebauten Systemen drastisch erweitert. Optional für das MCZ verfügbar sind weitere Betriebs- und Anwenderprogramme.

In die übrigen Steckplätze des MCZ lassen sich die von Zilog optional angebotenen Karten für Speicher und Ein/Ausgabeerweiterungen oder aber beliebige kundenspezifische Baugruppen einstecken. Der Kontakt zur anzuschließenden Peripherie erfolgt in all diesen Fällen über die auf der Rückseite des Systems vorgesehenen Ein/Ausgabestecker. Vom System können bis zu acht Floppy-Disk-Laufwerke ohne externe Decodierung adressiert werden. Der Computerkarte stehen vier Sockel für Festwertspeicher zur Verfügung, von denen drei durch die standardmäßig mitgelieferte Betriebsfirmware belegt sind. Der Benutzer kann entweder die verbleibende Festwertspeichererfassung für seine spezifischen Zwecke verwenden oder aber auch die eingesteckten Festwertspeicher herausnehmen, wodurch er dann alle vier Sockel zu seiner freien Verfügung hat. Auch parallele Ein/Ausgabe ist bereits auf dem Rechnerbaustein des MCZ aufgebaut; auch stehen vier 16 Pin Sockel zum Einsetzen anwendungsspezifischer Treiberbausteine nach dem Parallel-Ein-Ausgabe-Baustein zur Verfügung. Um den Anwender in den Möglichkeiten des Systemdesigns nicht einzuschränken, wurden diese Fassungen nicht mit den Anschlüssen des PIO's verdrahtet; auf diese Weise kann die Anschlußbelegung vom Anwender durch Legen von Drahtbrücken festgelegt werden. Optional ist das Z80-MCZ jedoch auch in für einen Schnelldrucker fertig verdrahteter Form lieferbar. Für Echtzeitaufgaben wurde ein Z80-CTC-Baustein eingesetzt, der die Festlegung der Übertragungsrate der Bedienkonsole und die Floppy-Disk-Synchronisation vornimmt. Zwei seiner Kanäle stehen dem Benutzer zur Realisierung von Echtzeituhr oder ähnlichen Anwendungen zur freien Verfügung. Der in dem System enthaltene Ein/Ausgabe-Port-Adreßdecoder adressiert einen Adreßbereich von 32 zusammenhängenden Ein/Ausgabeadressen. Einige davon werden für interne Zwecke verwendet; es besteht volle Erweiterbarkeit auf den von der Z80-CPU adressierbaren Ein/Ausgabebereich. Durch Verlegung von Drahtbrücken kann man auf der Zentralkarte Speicher und Ein/Ausgabebereiche nach Wunsch festlegen. Im übrigen ist die Erweiterbarkeit selbstverständlich auch durch volle Pufferung sämtlicher Systembussignale gesichert. Ein Rücksetzschalter ist an der Vorderseite des Gehäuses angebracht, der Netzschalter befindet sich auf der Rückseite. Zum Standardlieferumfang des Systems gehört, wie eingangs bereits aufgelistet, das Betriebssystem RIO mit Dateiverwaltung, Editor, relokativem Makro-Assembler, Binder/Lader (= „Linker") und Fehlersuchprogrammen.

☐ Ausbau des Z80-MCZ

Die Pufferung der Zentralbaugruppe im MCZ und die sieben freien Kartensteckplätze erlauben einfach durch Einstecken von Standard- oder anwendungsspezifischer Baugruppen eine Erweiterung der Fähigkeiten des MCZ.

Eine Erweiterung ist um folgende Baugruppen möglich:

- Z80-RMB: Schreib/Lesespeicher-Erweiterung mit 16...48 kByte Kapazität, aufgebaut mit 16 kByte dynamischen RAM-Speicherbaustein. Zusätzlich auf dieser Baugruppe untergebracht sind acht Fassungen für Festwertspeicherbausteine.
- Z80-IOB: Parallel-Ein/Ausgabe-Erweiterung zur Erweiterung des MCZ-Systems um acht Ein/Ausgabeports á 8 bit incl. Steuerleitungen.
- Z80-SIB: serielle Ein/Ausgabe Erweiterung des Z80-MCZ um vier serielle Ein/Ausgabe-Duplexschnittstellen;
- Z80-VDB: ermöglicht den direkten Anschluß des Z80-MCZ an einen beliebigen Fernsehbildschirm (Heim- oder Industriemonitor). Mit dem Z80-VDB ist sowohl alphanumerisches als auch graphisches Arbeiten möglich. In der alphanumerischen Betriebsweise sind ca. 2 kByte, in der graphischen Betriebsweise 20 kByte Bildwiederholspeicher nötig. Über einen Z80-PIO-Baustein auf der Baugruppe ist der Anschluß einer Standard-ASCII-Tastatur möglich.
- Z80-PMB: Festwertspeicher-Erweiterungsbaugruppe mit Parallel-Ein/Ausgabe-Erweiterung. Auf der Baugruppe untergebracht sind 16 Fassungen für Festwertspeicher-Bausteine 1 Parallel-Ein/Ausgabe-Baustein (Z80-PIO) und ein Zähler/Zeitgeber-Baustein (Z80-CTC).
- Z80-PPB: PROM Programmierbaugruppe. Diese Baugruppe wird mit der zugehörigen Betriebssoftware geliefert und erlaubt das Programmieren von bipolaren und MOS-Festwertspeicherbausteinen (PROM's und EPROM's). Folgende Festwertspeichertypen lassen sich elektrisch direkt ohne Eingriff in der Schaltung des Z80-PPB programmieren: 2704, 2708, Harris 7620, 7621, 7640 und 7641. Die Version PPB-16 programmiert i2716 EPROM's.
- Z80-SCC: Im MCZ-System ist ein neun-steckplätziger Einbaurahmen untergebracht, der auch als Einzelteil zu beziehen ist.
- Z80-WWB: Wirewrapplatine zum Aufbau kundenspezifischer Schaltung.
- Z80-EXB: Verlängerungsplatine; mit Hilfe dieses Zusatzes können Baugruppen so in den Z80-MCZ-Bus eingesteckt werden, daß sie von außen her zu Testzwecken zugänglich sind.

□ Anwendung

Die Prozeßrechner bzw. Minicomputer der ZILOG-Serie MCZ sind für Einsatz in allen Anwendungen kommerzieller, technisch-wissenschaftlicher und meß/steuer/regelungstechnischer Art geeignet, in denen externe Massenspeicher mit wahlfreiem Zugriff wünschenswert oder erforderlich sind. Sie werden in einer Form geliefert, die auf einfachste Weise eine Anpassung an jedes beliebige Anwenderproblem gestatten; hierfür sind freie Steckplätze vorgesehen, die je nach Einsatzzweck mit Steckkarten aus dem umfangreichen und voll zum MCZ kompatiblen Spektrum der MCB-Kartenserie bestückt werden können.

Die zum Teil im Preis inbegriffene, z.T. optional lieferbare Betriebs- und Programmentwicklungssoftware eröffnet insbesondere Systemhäusern und OEM-Kunden die interessante Möglichkeit der Erstellung und Ausentwicklung ihres Endprodukts unter Original-Bedingungen, da die Gesamtanlage selbst nach Abschluß der Entwicklungsarbeiten gleichzeitig das vom Endkunden einzusetzende Anwendersystem darstellt.

Dadurch ist keinerlei Hardware-Investment für die Entwicklung des Anwendersystems nötig.

Wesentlich ist für den Anwender bei der hohen Innovationsrate in der μC-Technik die Zukunftssicherheit. Die Systeme der MCZ-Serie erlauben nicht nur die Realisierung von Z80-Systemen, sondern auch die Entwicklung von Z8- und Z8000-Programmen:
Damit wird der Schritt zur Vierten Mikrorechnergeneration leicht gemacht.

Dabei liefern wir Ihnen nicht nur das Computersystem, sondern auch die erforderliche Peripherie inclusive der nötigen Hard- und Software-Anpassung.

Als Denkanstoß sind folgende (z.T. bereits realisierte) Anwendungsbereiche interessant:

- Analysentechnik mit graphischer Ausgabe
- Meßdatenerfassungssystem in Labors und Kliniken
- Verwaltung von Patientenkarteien in Arztpraxen
- Patientenüberwachung in Intensivstationen
- Steuerung und Regelung von Maschinen und industrieller Prozesse
- Verwaltung von Büchereien
- Zeichnungserstellung und Bearbeitung in Architekturbüros
- Textverarbeitung in Sprachen mit schwierig zu handhabendem Alphabet (z.B. Chinesisch, Arabisch)
- Steuerung von Filmstudios
- Bilderzeugung, -modifikation und -überwachung in Fernsehstudios
- Lehrcomputer mit alphanumerischer und graphischer Ein/Ausgabemöglichkeit; programmierter Unterricht
- Kommerzieller Universal-Rechner zu minimalen Kosten
- Datenkonzentrator/intelligentes Terminal
- Mehrprozessoranlage in einem Gehäuse
- Koordinator für Meßsysteme
- Wartungssystem für Industrie, Werkstatt und Heim
- Alarmsysteme
- Steuereinheit für automatische Testsysteme

Das Z80-MCZ ist eine nach Anlegen der Versorgungsspannung (220 V, 50 Hz) sofort arbeitsfähige Anlage mit Floppy-Disk oder Harddisk-Laufwerken. Der Dialog mit dem Computersystem findet über eine Serienschnittstelle (RS 232 oder 20 mA Stromschleife) statt, wodurch der Anschluß eines jeden Standardbildschirms oder teletypeähnlichen Fernschreibers möglich ist. Die Datenübertragungsfrequenz ist im Bereich von 10 bis 38600 baud einstellbar. Zusätzlich zu diesem Anschluß der Bedienkonsole sind an der Rückwand des Systems weitere Plätze für Stecker zum Anschluß von Prozeß- oder sonstiger Peripherie vorgesehen. Die freien Kartensteckplätze können durch Wirewrap-Technik vom Anwender entsprechend der von ihm verwendeten Karten beliebig verdrahtet werden.

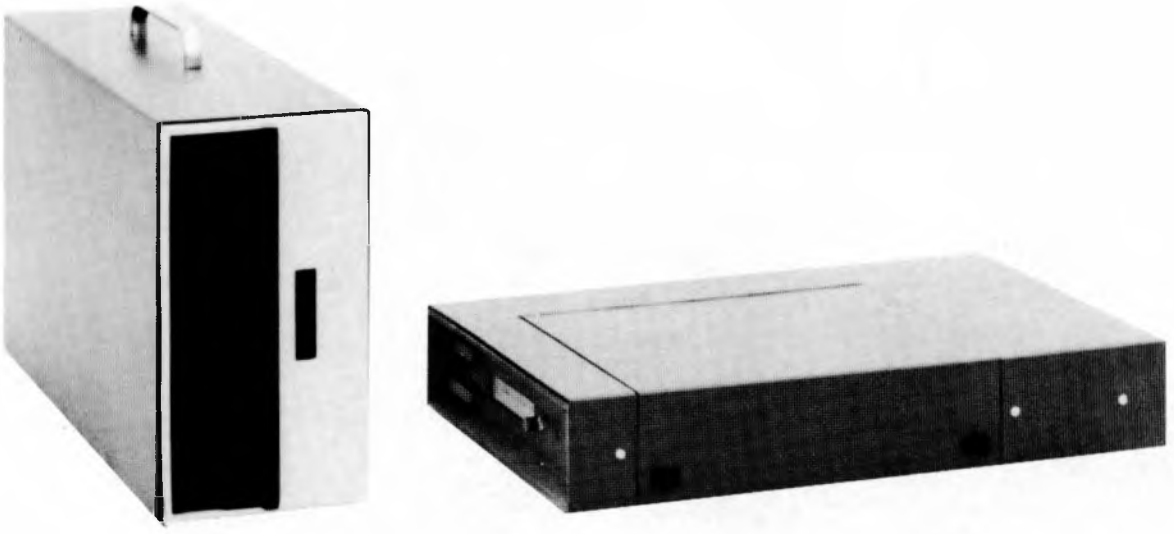
Im MCZ sind eine Reihe Steckplätze verfügbar, von denen minimal zwei von der Minimalsystemkonfiguration (Computerkarte und Floppy Disk-Steuerungskarte) besetzt werden. In die übrigen Steckplätze lassen sich die von Zilog optional angebotenen Karten für Speicher und Ein/Ausgabeerweiterungen oder aber beliebige kundenspezifische Baugruppen einstecken. Der Kontakt zur anzuschließenden Peripherie erfolgt in all diesen Fällen über die auf der Rückseite des Systems vorgesehenen Ein/Ausgabestecker. Vom System können bis zu acht Floppy-Disk-Laufwerke ohne externe Decodierung adressiert werden. Der Computerkarte stehen vier Sockel für Festwertspeicher zur Verfügung, von denen drei durch die standardmäßig mitgelieferte Betriebsfirmware belegt sind. Der Benutzer kann entweder die verbleibende Festwertspeichererfassung für seine spezifischen Zwecke verwenden oder aber auch die eingesteckten Festwertspeicher herausnehmen, wodurch er dann alle vier Sockel zu seiner freien Verfügung hat. Auch parallele Ein/Ausgabe ist bereits auf dem Rechnerbaustein des MCZ aufgebaut; auch stehen vier 16 Pin Sockel zum Einsetzen anwendungsspezifischer Treiberbausteine nach dem Parallel-Ein-Ausgabe-Baustein zur Verfügung. Um den Anwender in den Möglichkeiten des Systemdesigns nicht einzuschränken, wurden diese Fassungen nicht mit den Anschlüssen des PIO's verdrahtet; auf diese Weise kann die Anschlußbelegung vom Anwender durch Legen von Drahtbrücken festgelegt werden. Optional ist das Z80-MCZ jedoch auch in für einen Schnelldrucker fertig verdrahteter Form lieferbar. Für Echtzeitaufgaben wurde ein Z80-CTC-Baustein eingesetzt, der die Festlegung der Übertragungsrate der Bedienkonsole und die Floppy-Disk-Synchronisation vornimmt. Zwei seiner Kanäle stehen dem Benutzer zur Realisierung von Echtzeituhr oder ähnlichen Anwendungen zur freien Verfügung. Der in dem System enthaltene Ein/Ausgabe-Port-Adreßdecoder adressiert einen Adreßbereich von 32 zusammenhängenden Ein/Ausgabeadressen. Einige davon werden für interne Zwecke verwendet; es besteht volle Erweiterbarkeit auf den von der Z80-CPU adressierbaren Ein/Ausgabebereich. Durch Verlegung von Drahtbrücken kann man auf der Zentralkarte Speicher und Ein/Ausgabebereiche nach Wunsch festlegen. Im übrigen ist die Erweiterbarkeit selbstverständlich auch durch volle Pufferung sämtlicher Systembussignale gesichert. Ein Rücksetzschalter ist an der Vorderseite des Gehäuses angebracht, der Netzschalter befindet sich auf der Rückseite. Zum Standardlieferumfang des Systems gehört, wie eingangs bereits aufgelistet, ein vollständiges Betriebssystem mit Dateiverwaltung, Editor, Makro-Assembler und Fehlersuchprogrammen.

Die Computer der MCZ-Serie sind in einer Vielfalt verschiedener Gehäuse und Konfigurationen lieferbar, die entsprechend ihrer Ausbaufähigkeit und den mitgelieferten Massenspeichern in einem breiten Preisspektrum lieferbar sind:

MCZ 1-05/W64

Zilog

Betriebsbereites Anwender-Computer- und Entwicklungs-System mit 1 Floppy-Disk



Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert (also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!

10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

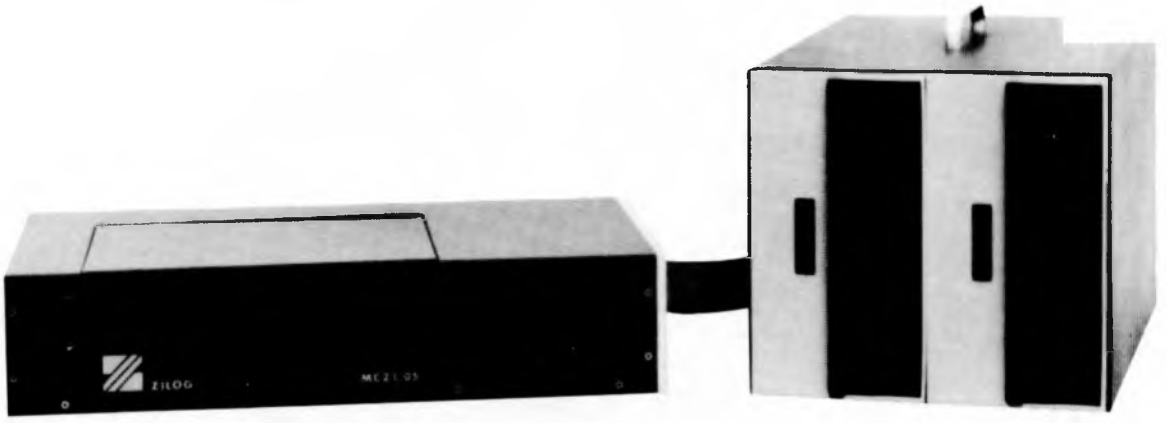
Z80-MCZ 1/05-W

- Speicherausbau 64 kByte
- 1 Floppy-Disk-Laufwerk in getrenntem Gehäuse,
- 2 freie Steckplätze
- Tischmodell

MCZ 1-05/R64

Zilog

Betriebsbereites Anwender-Computer- und Entwicklungs-System mit 2 Floppy-Disks



10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

Z80-MCZ 1/05-R

- Speicherausbau 64 kByte
- 2 Floppy-Disk-Laufwerke in getrenntem Gehäuse,
- zwei freie Steckplätze,
- Tischmodell

MCZ 1-20/R64

Zilog

Betriebsbereites Anwender-Computer- und Entwicklungs-System mit 2 Floppy-Disks

10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

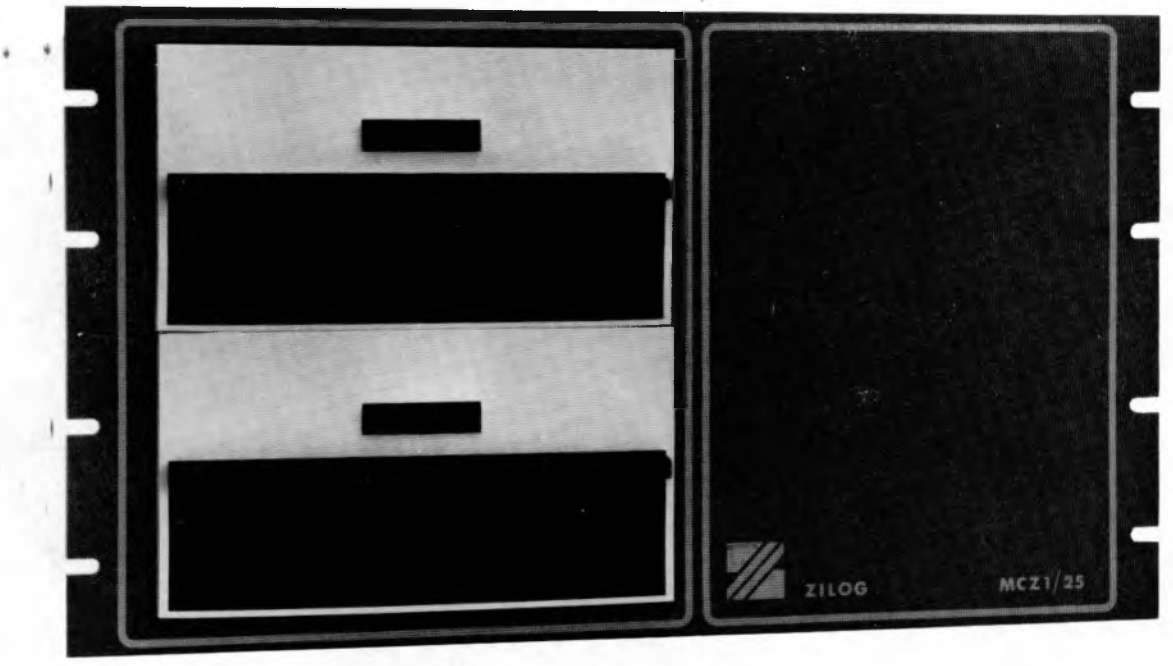
Z80-MCZ 1/20:

- Speicherausbau 64 kByte
- 2 Floppy-Disk-Laufwerke eingebaut,
- 7 freie Steckplätze
- Tischmodell

MCZ 1-25/R64

Zilog

Betriebsbereites Anwender-Computer- und Entwicklungs-System mit 2 Floppy-Disks



10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

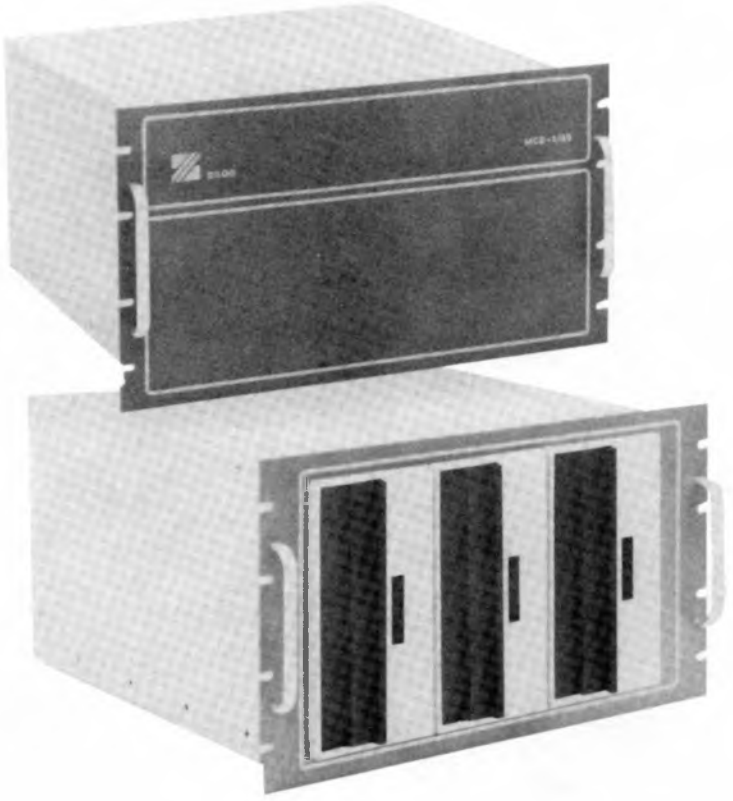
Z80-MCZ 1/25:

- Speicherausbau 64 kByte
- 2 Floppy-Disk-Laufwerke eingebaut
- 7 freie Steckplätze
- robuste 19-Zoll-Rack-Ausführung für industrielle und kommerzielle Einbau-Anwendungen

MCZ 1-30/R64

Zilog

Betriebsbereites Anwender-Computer- und Entwicklungs-System mit 3 Floppy-Disks



10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

Z80-MCZ 1/30:

- Speicherausbau 64 kByte
- in 19''-Zentralrechner-Gehäuse
- 16 freie Steckplätze
- 3 Floppy-Disk-Laufwerke in getrenntem 19''-Rack-Gehäuse
- insgesamt sind 8 Floppy-Disk-Laufwerke anschließbar.
- Ausführung für industrielle und kommerzielle Einbau-Anwendungen
- Auf Mehrfachcomputer ausbaubar

MCZ 1-35/R64

Zilog

Betriebsbereites Anwendercomputer- und Entwicklungssystem mit 12 MByte-Festplatte

10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

Als sensationelles Beispiel für ZILOG's Mikrocomputer-Leistung stellen wir Ihnen hier ein System vor, das weit in den Mini- und Prozeßrechnermarkt vorstößt:

ZILOG's Festplatten-Basierendes Multicomputersystem Z80-MCZ 1/35
mit 12 MByte Speicherkapazität und zwei Z80-Prozessoren, DMA und einem Arbeitsspeicher von 80 kByte zu einem absolut konkurrenzlosen Preis.

Da das System, wie alle Anlagen der MCZ-Serie, voll von Höheren Sprachen wie COBOL, FORTRAN IV, BASIC und PLZ unterstützt wird, ist es in praktisch allen Anwendungen zu verwenden, in denen bisher aufwendige Minicomputer eingesetzt werden mußten.

Hier die technischen Daten dieser Anlage:

- Speicherausbau 64 kByte
  in 19''-Zentralrechnergehäuse
  Einbaurack nicht im Lieferumfang)
- 13 freie Steckplätze
- Ein Hard-Disk-Controller ist eingebaut, der eine 12 MByte Festplatte (6 MByte austauschbar, 6 MByte fest) steuert.
- Floppy-Disk-Controller für max. 8 FD-Laufwerke
- Editorprogramm
- Makroassembler (relokativ) mit Binder/Lader
- Fehlersuchprogramm.

Betriebssoftware für MCZ-1 Systeme

10. ZILOG-Z80 MIKROCOMPUTER-SYSTEME

MCZ-Betriebssoftware

(siehe auch Abschnitt 11)

Zu dem Standardlieferumfang des Z80-MCZ gehört ein Standardbetriebssystem, das folgende Programme umfaßt:

- Bootstrap- und Monitor-Firmware
- Platte-Betriebssystem, Ein/Ausgabe-Treiber-Routinen (RIO)
- Editorprogramm
- Makroassembler (relokativ)
- Fehlersuchprogramm.

Das Z80-MCZ beinhaltet einen Festwertspeicherbereich (nicht flüchtig) von 3 kByte, in dem die Bootstrap-Routine, das Bedienkonsolentreiberprogramm, das Floppy Disk Driver-Programm und Fehlersuchprogramme untergebracht sind. Über dieses Programmpaket greift der Anwender auf weitere umfangreiche Betriebsprogramme zu, die auf Floppy Disk gespeichert sind: Das ZDOS-Betriebssystem, das MCZ-Executiv-Programm, den hochleistungsfähigen Texteditor und einen Makro-Assembler. Durch Anwendung dieser Floppy-Disk-Technik wird einerseits teurer Schreib/Lesespeicher gespart, andererseits die Speicherfähigkeit des Systems gegenüber herkömmlich aufgebauten Systemen drastisch erweitert (600000 Byte). Dieses Betriebssystem (RIO) hat einen relokativen Makroassembler zur weiteren Einsparung von Programmentwicklungs- und Testzeit durch getrenntes Übersetzen und Testen von einzelnen Programmteilen.

RIO umfaßt daneben einen Binder/Lader, erweiterte Editier-, Dateiverwaltungs- und Testmöglichkeiten und außer indexsequentieller Dateiadressierung auch direkten Zugriff.

Seine transparente Ein/Ausgabe-Architektur ermöglicht die individuelle Generierung und Erweiterung des Betriebssystems nach Anwendererfordernissen.

Über Kommando-Dateien, die vom Rechner ebenso wie andere Datengesamtheiten akzeptiert und verarbeitet werden, läßt der Rechner durch RIO **automatisierte Datenverarbeitung im Stapelbetrieb** (z.B. Abarbeitung großer Datenmengen in der Nacht) zu und **Bedienung des fertig implementierten Systems durch ungeschultes Personal.**

RIO ist übrigens das von allen Höheren Sprachen geforderte Grund-Betriebssystem.

Optional für das MCZ verfügbar ist folgende Software:

☐ MCZ/PLZ

PLZ ist eine neue, hochleistungsfähige, speziell für Mikrocomputerentwicklung ausgelegte Sprache, die Elemente von PL/1, PASCAL und ALGOL-68 miteinander vereint.

Der Lieferumfang umfaßt einen PLZ-Compiler, der maschinenunabhängiges Programmieren erlaubt, einen PLZ-„Assembler", der für zeitkritische Anwendungsteile das Mischen von PLZ- und ASSEMBLER-Anweisungen des jeweiligen Prozessors ermöglicht und einen PLZ-Interpreter zum interaktiven Austesten der Programme. Durch diese Architektur kann in optimaler Weise programmiert werden, da die Programmteile in der jeweils günstigsten Weise codiert und dann einfach zusammengebunden werden können.

PLZ ist bzw. wird für alle ZILOG-Prozessoren verfügbar. Speicherbedarf: 48 kB. Betriebssystem RIO erforderlich.

☐ MCZ/COBOL

Residenter COBOL-Compiler für Z80-MCZ; entspricht ANSI-STANDARD 1974 LEVEL 1; zusätzlich einige Anweisungen aus LEVEL 2.

Außerdem sind Besonderheiten für Mikrocomputer berücksichtigt, wie etwa spezielle Ein/Ausgabe-Befehle (ACCEPT, DISPLAY) und Testhilfen.

Der Compiler läuft unter dem Betriebssystem Z80-RIO ab. Mindestspeicherbedarf ist 48 kByte.

☐ MCZ/BASIC-Interpreter

BASIC-Interpreter mit wahlweise 13 BCD-Zeichen Genauigkeit oder 32 Bit-Gleitkommaarithmetik.

Beinhaltet hochleitungsfähige Arithmetik, Dateiverarbeitung und die Möglichkeit des Einbindens von ASSEMBLER-Routinen.

Speicherbedarf ist ca. 23 kByte, minimal erforderliche Systemspeicher 48 kB; Betriebssystem RIO ist Voraussetzung.

☐ FORTRAN

Befehlsvorrat umfaßt die Anweisungen des ANSI-66x 3.9 Standards mit Ausnahme des Datentyps COMPLEX.

Der minimal erforderliche Schreib/Lesespeicher ist 48 kByte, Betriebssystem RIO ist Voraussetzung.

☐ MCZ/DEV-Z8-Programmpaket

zur Entwicklung und zum Austesten von Programmen für ZILOG's Einchipcomputer Z8.

Das Paket umfaßt einen strukturierten Assembler (PLZ/ASM-Z8), einen Cross-Simulator und erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.

☐ MCZ/ASM-Z8000-Programmpaket

zur Entwicklung von Programmen für ZILOG's Z8000, den ersten 16 bit-Mikrocomputer der Vierten Generation.

Das Programm erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.

ZILOG-
ZDS-1-Serie

Zilog

**Mikrocomputer-
Entwicklungs-Systeme**

**Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert
(also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!**

11. ZILOG-
MIKROCOMPUTER-
ENTWICKLUNGS-SYSTEME

6. ZILOG-Entwicklungssysteme der ZDS-1-Serie

Mit dem Z-80-Mikrocomputer-Entwicklungssystem steht ein komplettes, betriebsbereites Computersystem zur Verfügung, das speziell auf die Entwicklung von Zilog-Anwendersystemen zugeschnitten ist.

Damit ist dem Entwickler ein Mittel in die Hand gegeben, Mikrocomputer-Systeme auf die rationellste und kostengünstigste Weise zu realisieren. Wie wichtig und rentabel an dieser Stelle der Einsatz ausgefeilter und modernster Technik ist, wird klar, wenn man sich vergegenwärtigt, daß von den Entwicklungskosten eines Mikrocomputergesteuerten Gerätes im Durchschnitt 50...90% auf die Software-Entwicklung und Tests entfallen; die Investition amortisiert sich dadurch in kürzester Zeit.

Das Z-80-Mikrocomputer-Entwicklungssystem ist so flexibel aufgebaut, daß es auch für alle zukünftigen Zilog-Mikroprozessoren und Mikrocomputer geeignet ist.

Wichtigster Bestandteil ist neben seinem Zentralgerät die Doppel-Floppy-Disk-Einheit, die sekundenschnellen Zugriff auf sämtliche System- und im Test befindlichen Anwender-Programme gestattet, außerdem komfortable Dateiverwaltung und -Änderung.

Zum System gehört daneben ein Echtzeit-Testadapter ("Users Hardware Interface") zur direkten Kopplung zwischen dem Entwicklungscomputer und dem zu testenden Anwendersystem, wodurch in kürzester Zeit sowohl Hardware- als auch Software-Tests durchgeführt werden können.

Selbstverständlich wird zu allen diesen Einheiten die zugehörige System-Software auf Diskette mitgeliefert.

Die Systeme dieser Serie integrieren Funktionen in ihrer Standardausrüstung, die sonst nur als Sonderzusätze angeboten werden:

Als **ECHTZEIT-TESTADAPTER** stellt das Entwicklungssystem die direkte Verbindung zum Prototypsystem her und verhält sich dabei wie ein vom Anwender gesteuerter Z80- oder Z80A-Mikroprozessor, der es ermöglicht:

- Alle Register und RAM Speicherbereiche zu setzen und rückzusetzen
- Interruptstatus zu beeinflussen
- Befehle einzeln oder abschnittweise auszuführen
- I/O Port-Daten zu setzen oder anzuzeigen
- den Programmablauf an beliebiger Stelle zu starten und zu unterbrechen.

Als **LOGIC ANALYZER** überwacht und speichert das Entwicklungssystem CPU-Bus Aktivitäten und schließt einen Programmablauf definiert ab, wenn ein vom Anwender vorher bestimmtes Ereignis eintritt.

Beispiele:

- Abspeichern von Informationen, die während irgendeines, vom Anwender vorgegebenen Vorganges auf dem Adreß-, Daten- und Steuerbus auftreten.
- Abspeichern von Vorgängen auf den Bussen, die vom Anwender bestimmt werden, z.B. bei einem Lese- oder Schreibbefehl
- Abbruch des Programmes bei Eintritt eines vom Anwender vorher bestimmten Ereignisses
- Anzeige der letzten bis zu 256 Bus-Ereignisse.

Als **ROM/RAM SIMULATOR** werden dem von Anwender zu entwickelndem System Speicherbereiche des Entwicklungssystems zur Verfügung gestellt (=Speicherabbildung).

- Mit der Speicherabbildung kann der Benutzer seinem Prototypsystem die physikalischen Eigenschaften des Entwicklungssystems zuweisen.
  Eine Speicherabbildung wird dadurch erreicht, daß der Anwender selbst bestimmt, welche Speicherbereiche des Prototyps benützt und welche vom Entwicklungssystem entliehen werden sollen.
- Mit dem Entwicklungssystem ZDS-1/40 können sogar Speicherbereiche als schreibgeschützt und nicht vorhanden erklärt werden, so daß jeder Zugriff auf einen solchen Bereich zu einem Programmabbruch führt.

Als **PROM/EPROM PROGRAMMIEREINRICHTUNG** dient das Entwicklungssystem dazu, PROM's zu schreiben, zu lesen, zu prüfen und zu duplizieren. Dies ist eine Zusatzeinrichtung, die durch die Baugruppen ZDS/PPB und ZDS/PPB16 realisiert wird.

Erst durch das äußerst vielseitige Betriebssystem RIO ist Mikrocomputerentwicklung in der Effizienz möglich, die die ZILOG-ZDS-Serie bietet. RIO dient dazu, Software für Z80-Mikroprozessor in optimaler Weise zu erstellen und zu übersetzen. Darüber hinaus steht unter RIO ein umfangreiches Softwarepaket zur Verfügung, das eine einwandfreie und rasche Fehlersuche in Z80-Systemen gewährleistet; die standardmäßig mitgelieferten Echtzeittestadapter und die zugehörige Testsoftware garantieren Austesten von Hard- und Software in minimal möglicher Zeit.

Das Betriebssystem RIO mit seiner Verwendbarkeit relokativer Modulen und seinem transparenten I/O-Management wurde speziell dazu geschaffen, den Entwicklungsvorgang wesentlich zu erleichtern. Außerdem wird durch die Verwendung verschiedener Systemeigenschaften eine Anpassung an die individuellen Anforderungen jedes einzelnen Anwenders erreicht.

Wichtigstes Bestandteil: Die Betriebs- und Testsoftware

■ **OS Executive**
- ☐ Kommandos zur System-Speicherkonfiguration
- ☐ Kommandos können von der Konsole oder über ein Steuerprogramm vorgegeben werden.
- ☐ Dem System können beliebige anwenderindividuelle Kommandos hinzugefügt werden.
- ☐ Kommandofolgen können in Dateien abgespeichert und ausgeführt werden. Dadurch läßt sich Stapelverarbeitung (=„Batchbetrieb", z.B. in „Dritter Schicht") zur automatisierten Datenverarbeitung bei langwierigen Operationen durchführen; außerdem ist die Abwicklung komplexer Vorgänge durch einige wenige Tastenbetätigungen auszulösen, wozu in Routinefällen beispielsweise auch angelernte Hilfskräfte eingesetzt werden können.

■ **Assembler**
- ☐ wahlweise relokativer oder absoluter Objektcode
- ☐ externe Bezugsmerkmale
- ☐ Globale Merkmalbestimmungen
- ☐ Makro-fähig
- ☐ Symboltabelle
- ☐ Mit INCLUDE-Anweisungen werden zusätzliche Dateien bei der Assemblierung in das Quellprogramm mit aufgenommen.

■ **Binder/Lader**
- ☐ Zuordnung absoluter Adressen für Programmodule
- ☐ Auflösung externer Bezugsmerkmale
- ☐ Erlaubt Überlagerungen oder Speicherzwischenräume
- ☐ Erstellt eine Speicherübersicht und globale Adreßliste

■ **Text Editor**
- ☐ Seitenorientierter Arbeitsbereich für jede beliebige Dateigröße
- ☐ Automatische Dateiduplizierung
- ☐ Zugriff auf weitere Dateien auch während des Editierens
- ☐ Zeichenweise Suche und Änderung von Zeileninhalten einer Datei

■ **PROM Monitor**
- ☐ Steuerroutinen für die Systemkonsole und Floppy Disk
- ☐ Urlader zur Erleichterung der Systeminitialisierung
- ☐ Fehlersuchpaket auf Maschinenebene (HEX)

Anwendung des ZDS-1-Entwicklungssystems
Das Höchstleistungs-Entwicklungswerkzeug zum Höchstleistungs-Mikroprozessor

Es spart Ihnen Geld
an der Stelle, wo es wichtig ist — durch Reduktion der Software-Kosten, die bekanntlich 50—90% der Entwicklungskosten eines Anwendersystems ausmachen und zwar durch

- Doppel-Floppy-Disk, das Wartezeiten zum Programm- und Datei-Laden auf ein Minimum verkürzt.
- System-Software, die Editor-, Makroassembler-, Datei-Verwaltungs- und Fehlersuchprogrammpakete umfaßt. Der Computer reduziert die zum Schreiben, Übersetzen und Testen eines Programms nötige Zeit auf das technisch mögliche Minimum. Und das in Echtzeit — sodaß Sie nicht riskieren, daß die eigentliche Fehlersuche erst dann anfängt, wenn die Fähigkeiten Ihres Entwicklungssystems ausgeschöpft sind.
- BASIC-Interpreter, FORTRAN-Compiler und die PLZ-Übersetzer-Familie erlauben Ihnen sogar das Verwenden bekannter, höherer Programmiersprachen!
- Echtzeit-Testadapter: Sie ersetzen einfach die CPU Ihres Anwendersystems durch den Testadapter-Stecker und lassen Ihr Anwenderprogramm im Arbeitsspeicher ablaufen. Welche Speicher und Ein/Ausgaben (die des Entwicklungs- oder die des Anwendersystems) Sie verwenden, können Sie frei festlegen; dadurch kreisen Sie Fehler unter Echtzeitbedingungen in der kürzest möglichen Zeit ein!
- Echtzeit-Testeinheit: Bei Ausführung des Anwenderprogramms läuft dauernd ein zusätzlicher 256×32 bit großer Speicher mit, der sämtliche Aktivitäten auf Adreß-, Daten- und Steuerbus registriert.
- Jedes fehlerhafte Verhalten Ihres Anwendersystems (z. B. Setzen eines bestimmten Bits bei einem bestimmten Port) können Sie nun dem Entwicklungscomputer in Form einer Haltepunkt-Bedingung vorgeben. Er hält an dieser Programmstelle an und zeigt Ihnen über die letzten 255 Busaktivitäten, wie der Fehler zustande kam!
  Sie ändern nun einfach per Kommando von der Bedienungskonsole die fehlerhaften Inhalte von Registern, Speicherstellen oder I/O-Ports und setzen die Ausführung des Programms fort — ohne Programmkorrektur, Neuübersetzung oder gar Neuprogrammieren eines PROM's!

Und warum beziehen Sie Ihr Z80-Entwicklungssystem von uns?

- Wir können Ihnen garantieren, daß Sie von uns die im Hause Zilog entwickelte Original-Systemsoftware bekommen — auf dem neuesten Stand in sämtlichen Weiterentwicklungs-Stufen.
- Nur wir stellen für Sie einen Partner dar, der gleichzeitig Z80-Anbieter und Z80-Anwender ist, denn unsere hauseigene Geräteentwicklung arbeitet seit September 76 mit Zilog 80.
  Wir kennen als Anwender Ihre Probleme und haben das Know-How, den Service und die Beratungsmöglichkeiten, die Sie brauchen!
  Wir bieten Ihnen ein **komplettes** Entwicklungssystem **mit** zugehöriger Peripherie, wie Datensichtgeräte, Fernschreiber, Zeilen- oder Matrixdrucker, PROM-Programmiergeräte . . . wie immer Sie Ihr System ausstatten wollen, an das Entwicklungssystem angepaßt und aus einer Hand!
- Und schließlich: Wir bieten Ihnen die Schlüssel-Dokumentation in Deutsch — damit Sie Ihre Muttersprache auch bei Anwendung von Z80 nicht verlernen.
- Am besten, Sie rechnen sich selbst aus, wie schnell sich Ihr Zilog-Entwicklungssystem amortisiert, oder Sie rufen uns an, wir helfen Ihnen gerne dabei!

ZILOG ZDS/1 – 40

Zilog

4 MHz-Mikrocomputer-Entwicklungs-System

Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert (also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!

11. ZILOG-MIKROCOMPUTER-ENTWICKLUNGS-SYSTEME

Entwicklungssystem ZDS-1/40-4MHz

Mit ZDS-1/40 steht ein leistungsfähiges System zur Entwicklung von Anwendersystemen auf Basis von ZILOG Mikrocomputern (Z80A, Z80, Z8, Z8000) zur Verfügung. Durch den Einsatz zweier Mikroprozessoren (einer Z80-CPU im Entwicklungssystem und einer Z80A-CPU im Testadapter selbst) ist Echtzeittest bei Z80-Systemen bis zu Taktfrequenzen von 4 MHz ermöglicht.

Hier eine Übersicht über die technischen Daten:

- Externe Z80A-CPU für den direkten Anschluß des Prototyps
- Exakter Echtzeittest für Systeme bis 4 MHz
- Speicherabbildung und Schutz in Blöcken bis zu 1024 Bytes
- Refresh des Anwenderspeichers bei Abbruch eines Echtzeittests
- Überprüfung des Taktes des Anwendersystems
- Speicheradreßumsetzung
- Meldung von Speicherzugriffen auf schreibgeschützte oder nicht verfügbare Blöcke
- Leistungsfähige Software zur Fehlersuche

Bestandteile

☐ Z80A-CPU	Z80-Mikroprozessor mit 4 MHz Taktfrequenz im Emulator
☐ Hauptspeicher	standardmäßig 64 kB
☐ Zwei Floppy-Disk-Laufwerke	Speicherkapazität von je 300 kB
☐ Betriebssystem RIO	relokativer Makro-Assembler, Binder/Lader, Text-Editor und Dateiverwaltung
☐ Echtzeit-Speicher-Modul	Überwachung von Adreß-, Daten- und Steuerleitungen während ausgewählter Operationen, wie z.B. Lesen, Schreiben von Speicherzellen, oder Lesen, Schreiben einer Ein/Ausgabe-Schnittstelle
☐ Haltepunkt-Modul	Überwachung und Kontrolle bestimmter Zustände auf Adreß-, Daten- und Steuerbus zum Abbrechen des Programmablaufs oder Ausgabe eines Synchronisationsimpulses für einen Oszillographen.
☐ Speicherabbildung	Zuordnung von physikalischen Eigenschaften für den einzusetzenden Speicherbereich. Dabei kann der Anwender Speicherbereiche des Entwicklungssystems, seines eigenen Systems (Prototyp) Kombination von beiden in 1024 Bytes-Blöcken mischen. Zusätzlich kann jeder Block als schreibgeschützt oder nicht verfügbar definiert werden, was die Fehlersuche und die Entwicklung des Prototyps wesentlich erleichtert.
☐ Adreßumsetzer	Steht zusätzlich mit der Speicherabbildung zur Verfügung. Ein Block (1024 Bytes) kann damit im Anwendersystem an einer anderen Adresse als im Entwicklungssystem stehen.
☐ Überprüfung des Anwendertaktes	wird durchgeführt, um sicherzustellen, daß die Taktfrequenz des Anwendersystems nicht unter 4,9 kHz liegt.
☐ Refresh	wird immer dann gestartet, wenn ein Echtzeittest abgebrochen wird, um den Verlust von Daten bei dynamischen RAM's zu vermeiden.
☐ I/O Ports	Der Anwender kann auf alle I/O Ports zugreifen, wenn er sich im „Users-Mode" befindet und der Takt des Anwendersystems Verwendung findet.
☐ Betriebssystem RIO	relokativer Makro-Assembler, Text-Editor, Binder/Lader und ZDOS Plattendateiverwaltung.

Technische Daten

CPU	Z80-CPU im Zentralgerät und Z80A-CPU in der Echtzeittesteinheit
Speicher	3 kB ROM/1 kB stat. RAM reserviert für Betriebssystem; 60 kB RAM für allgemeine Zwecke
Systemtakt	2,5 MHz Quarzgesteuert
Stromversorgung	230 V, 50 Hz
Umgebungstemperat.	Betriebstemperatur 0 . . . 50°C
Gehäuse	2 getrennte Gehäuse, davon eines für die Floppy-Disk-Laufwerke und die Stromversorgung, das andere beinhaltet den Computer
Abmessung	je Gehäuse Länge 480 × Breite 381 × Höhe 229 mm
Gewicht	ca. 16 kg

ZDS-1/40-4MHz Kommandos zur Fehlersuche

Mit dem ZDS-1/40 Vorrat an Kommandos zur Fehlersuche steht dem Anwender eine leistungsfähige plattenorientierte Software zur Verfügung, die das Z80A-CPU Echtzeittest-Modul optimal unterstützt.
Die Kommandos dienen nicht nur der Überwachung und Analyse von Programmabläufen unter Testbedingungen, sondern auch dazu, die physikalischen Eigenschaften des genutzten Speicherbereichs zu beschreiben. Ist die Speichereinteilung einmal festgelegt, kann sie auf Platte gesichert und bei Bedarf wieder geladen werden. Dies erspart es dem Anwender, die Speicherkonfiguration vor jedem Echtzeittest neu festzulegen.

Hierzu stehen folgende Kommandos zur Verfügung:

☐ MAP	Generiert Eintragungen über die Zusammensetzung des vom Anwender verwendeten Speicheradreßraumes. Damit kann der Benutzer Speicherbereiche von 1024 Byte (=Block) entweder dem Anwendersystemspeicher oder dem Entwicklungssystem zuordnen. Darüber hinaus können jedem Block Eigenschaften wie schreibgeschützt oder nicht verfügbar zugewiesen werden.
☐ DMAP (Display MAP)	Darstellung einzelner Einträge, einem Bereich von Einträgen oder der gesamten Einträge und Editierung der Einträge.
☐ WRITE	Bestimmung einer Unterbrechung, wenn auf einen schreibgeschützten Bereich geschrieben werden soll.
☐ BREAK	legt fest, wann ein Echtzeitlauf eines Programms abgebrochen wird. Im variablen Teil des Kommandos wird definiert, bei welchen Ereignissen eine Unterbrechung eintreten soll: ● Adressenvergleich ● Datenvergleich oder Vergleich mit Maske ● Speicher Schreib/Lesebefehl ● I/0 Port Schreib/Lesebefehl
☐ DISPLAY MEMORY	Inhalt eines oder mehrerer Speicherbereiche werden angezeigt. Werden einzelne Bytes angezeigt, kann der Anwender den Inhalt jedes einzelnen Bytes ändern.

☐ FILL — Ein vorgegebener Datensatz wird in allen Bereichen abgespeichert, die durch das Kommando festgelegt werden.

☐ COMPARE — Vergleich von Speicherblöcken

☐ DISPLAY — Zugriff auf Speicherbereiche, die als Block oder einer nach dem anderen angezeigt werden, um zu prüfen und zu ändern.

☐ FILL — Speicherung eines frei wählbaren Datenbytes in Bereichen von Speicheradressen.

☐ INTERRUPT STATUS — Dient der Anzeige und Änderung der Schalterzustände von Interruptmöglichkeiten.

☐ GET — Lädt eine Datei von der Platte in den Speicher und ihre Eingangsadresse in den Befehlszähler.

☐ GO — Startet das Benutzerprogramm. Die Adresse an der das Programm beginnen soll kann festgelegt werden. Wird das Programm vorzeitig durch eine Unterbrechung abgeschlossen, setzt GO das System wieder in Echtzeittestmodus zurück und startet erneut das Programm.

☐ HISTORY — Abspeicherung der letzten n (1-256) Ereignisse, die im Echtzeitspeicher Modul abgespeichert sind; darstellbare Informationen:
- Adreßbus (16 Bits)
- Datenbus (8 Bits)
- Steuerbus (7 Bits)

☐ INTERRUPT — Darstellung des Interrupt Mode der Echtzeittesteinheit (Mode 0, 1, 2, freige-

☐ INTERRUPT MODE — Darstellung des Interrupt Mode der Echtzeittesteinheit (Mode 0, 1, 2, freigegeben oder gesperrt).

☐ NEXT — 1 bis n Befehle werden schrittweise ausgeführt. Nachdem die ausgewählte Anzahl von Befehlen abgearbeitet ist, werden alle CPU-Register angezeigt.

☐ OS — Verläßt das Fehlersuchprogrammpaket, lädt das Betriebssystem und geht auf Anfangsstellung

☐ PORT — Ein Zeichen kann von einem beliebig auszuwählendem Port eingegeben wer-

☐ PORT — Ein Zeichen kann von einem beliebig auszuwählendem Port eingegeben und angezeigt werden. Anschließend kann ein Zeichen vom Benutzer eingegeben und auf das ausgewählte Port ausgegeben werden.

☐ PROM — übergibt die Programmsteuerung an die in den PROM's des Entwicklungssystems residente Software zur Fehlersuche.

☐ PULSE — Hat gleiche Wirkung wie BREAK identisch; das Programm wird jedoch nicht unterbrochen, sondern es wird stattdessen ein Synchronisationsimpuls für eine externe Testeinrichtung, z. B. einen Oszillographen über eine hierfür vorgesehene BNC-Buchse ausgegeben.

☐ QUIT — Übergibt die Steuerung dem OS

☐ REGISTER — Abfrage und Änderung von CPU-Registern

☐ SAVE — Im RAM befindliche Programme und Subroutinen werden auf Platte gespeichert. Diese Dateien können mit einem GET wieder geladen werden. Zur Adressierung auf der Platte wird ein Dateiname vergeben.

☐ SET — Sequentielle Speicherung von Daten im Speicher ab einer vom Benutzer festgelegten Adresse.

☐ STATUS — Zeigt den momentanen Zustand des Echtzeittests, wie z. B. interrupt Mode, Taktversorgung, sowie Angaben über Programmunterbrechung und -überwachung.

☐ TRACE — Der Benutzer bestimmt damit, welche Bus-Aktivitäten im Echtzeitspeicher Modul abgespeichert werden. Dabei ist irgendeine oder eine Kombination folgender Operationen frei bestimmbar:
- Speicher Lesebefehl
- Speicher Schreibbefehl
- Port Lesebefehl
- Port Schreibbefehl

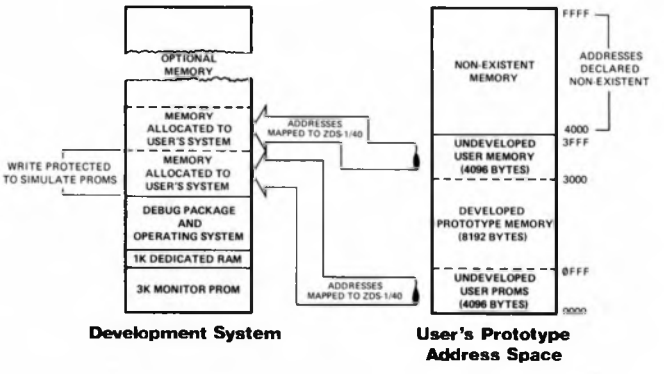
Speicherabbildung und Adreßumsetzung

Mit der Speicherabbildung und dem Adreßumsetzer kann der während eines Echtzeittests benutzte Speicherbereich konfiguriert werden. Hierbei wird der Speicher in Blöcke von 1024 Bytes eingeteilt. Softwaremäßig läßt sich nun festlegen, ob diese Blöcke sich physikalisch, oder aber in der Anwenderhardware befinden sollen.

Dadurch ist das schrittweise Austesten des Anwenderspeichers in sehr einfacher Weise möglich. Ein weiteres starkes Hilfsmittel der ZDS-Entwicklungssysteme ist ihre Fähigkeit, solche Blöcke als nicht verfügbar und schreibgeschützt zu deklarieren. Dadurch ist eine naturgetreue Abbildung eines realen Anwendersystems realisiert, das bekanntlich im allgemeinen aus PROM, RAM und einem nichtbenutzten Speicherbereich besteht.

Auf diese Weise lassen sich korrekte Aussagen über das Verhalten einer Mikrocomputer-Hard/Software-Kombination treffen, noch bevor die tatsächliche Anwenderhardware in Betrieb genommen oder erstellt wurde. Obige Illustration verdeutlicht diese Vorgänge.

Nichtentwickelten PROM-Bereichen ist z. B. Speicher aus dem Entwicklungssystem zugewiesen, so daß das System des Anwenders zuerst absolut fehlerfrei ausgetestet wird, bevor die entsprechenden PROM's gebrannt werden. Der Bereich im Entwicklungssystem ist dabei schreibgeschützt, um das PROM-Verhalten zu simulieren. Wird versucht, diesen Bereich zu beschreiben, wird das Programm sofort unterbrochen, wenn dies mittels BREAK-Kommando veranlaßt wurde. Die Adressen 4000 - FFFF sind in obigem Beispiel als nicht vorhanden deklariert, so daß jeder versuchte Zugriff ebenfalls eine Programmunterbrechung hervorrufen würde.



ZILOG-ZDS/1-25

Zilog

2.5 MHz-Mikrocomputer-Entwicklungs-System

Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert (also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!

11. ZILOG-MIKROCOMPUTER-ENTWICKLUNGS-SYSTEME

Entwicklungssystem ZDS-1/25 (2,5 MHz)

Das Entwicklungssystem ZDS-1/25 mit der Z80-CPU gehört zu den preiswertesten Mikrocomputerentwicklungssystemen, die heute angeboten werden. Es stellt einen eigenständigen Mikrocomputer mit zwei Floppy Disks, verschiedener Peripherie und Schnittstellen dar, das den Anwender bei der Erstellung von Soft- und Hardware für Z80-Systeme in komfortabler Weise unterstützt. Seine zwei Betriebsarten, der Anwender-Mode und Monitor-Mode, tragen hierzu entscheidend bei.

Im Monitor-Mode setzt der Benutzer das Betriebssystem RIO ein, um seine Software zu entwickeln, zu editieren und zu modifizieren.

Im Anwender-Mode sind Speicher und Peripherie ganz oder teilweise dem Anwendersystem-Prototyp zugeordnet, mit Ausnahme der I/O-Port's E0H bis EFH. Damit kann der Anwender sein Programm unter Echtzeitumgebung ablaufen lassen.

Bestandteile

- ☐ 60 kB RAM
- ☐ 3 kB PROM Steuerprogramm, 1 kB reserviert für Monitor Zwischenspeicher
- ☐ Programmierbares Hardware Unterbrechungsmodul, um bei einer gewünschten Adresse oder Aktivität einen Abbruch herbeizuführen.
- ☐ Programmierbares Echtzeitspeicher-Modul um CPU-Aktivitäten zu speichern (Adreß-, Daten- und Steuerbus).
- ☐ Echtzeittesteinheit mit Verbindungskabel (ca. 1 m) zum Anschluß des Anwendersystems
- ☐ Speicherabbildung, für 256 Byte-Blöcke
- ☐ 2 Floppy-Disk-Laufwerke, Gesamtkapazität 600 kBytes
- ☐ Betriebssystem RIO mit Text-Editor, Assembler, Binder und ZDOS II Plattenverwaltung

Technische Daten

CPU:	Standard Z80-CPU
Systemtakt:	2,5 MHz quarzgesteuert
Speicher:	3 kB ROM/1 kB stat. RAM reserviert für Betriebssystem, 60 kB für allgemeine Zwecke.
Stromversorgung:	230 V, 50 Hz
Umgebungstemp.	Betriebstemperatur 0 . . . 50°C
Gehäuse:	1 Gehäuse mit 2 Floppy-Disk-Laufwerken, Stromversorgung und sonstige Systembestandteile.
Abmessung:	Länge 480 × Breite 381 × Höhe 229 mm
Gewicht:	ca. 16 kg

ZDS-1/25 Kommandos zur Fehlersuche

Der mit dem System ZDS-1/25 vorhandene Sprachumfang zur Fehlersuche gewährleistet eine rasche Steuerung, Analyse und Fehlersuche von Programmen, die entweder im Entwicklungssystem, im Prototyp oder in einer Kombination beider ablaufen.

Die Kommandos zur Fehlersuche sind PROM-resident und benötigen 1 kB RAM für die Zwischenspeicherung.

Nachfolgend eine Liste der zur Fehlersuche vorhandenen Kommandos und deren Funktionen:

☐ BREAK — Veranlaßt einen automatischen Hardware-Abbruch im Echtzeit-Fehlersuch-Modul. Der Abbruch kann ausgelöst werden durch

- Speicher Schreib/Lesebefehl
- I/O Port Schreib/Lesebefehl
- Adressenvergleich
- Datenvergleich oder Vergleich mit Maske
- Betätigen der MONITOR-Taste

In jedem Fall wird bei einem Abbruch der Status der Anwender-CPU abgespeichert, so daß die Programmausführung später wieder fortgesetzt werden kann.

☐ HISTORY — Darstellung der letzten n (1—256) Ereignisse, die im Echtzeitspeicher-Modul abespeichert sind; darstellbare Zeichen:

- Adreßbus (16 Bits)
- Datenbus (8 Bits)
- Steuerbus (7 Bits)

☐ JUMP — Übergibt die Steuerung an eine Startadresse, das System selbst bleibt jedoch im Monitor-Mode.

☐ MOVE — Blocktransfer beliebiger Größe von einem beliebigen Bereich in einen anderen.

☐ NEXT — 1 bis n Befehle werden schrittweise ausgeführt. Nachdem die ausgewählte Anzahl von Befehlen abgearbeitet ist, werden alle CPU-Register angezeigt.

☐ PORT — Prüfung und/oder Änderung von I/O-Port-Daten

☐ PULSE — Ist dem BREAK identisch, das Programm wird jedoch nicht unterbrochen, sondern es wird stattdessen ein Synchronisations-Impuls für eine externe Testeinrichtung gegeben (z.B. einen Oszillographen).

☐ QUIT — Übergibt die Steuerung dem OS

☐ REGISTER — Abfrage und Änderung von CPU-Registern.

☐ SAVE — Im RAM abgelegte Programme und Subroutinen werden auf Platte gespeichert. Diese Dateien können mit GET wieder geladen werden. Zur Adressierung auf der Platte wird ein Dateiname vergeben.

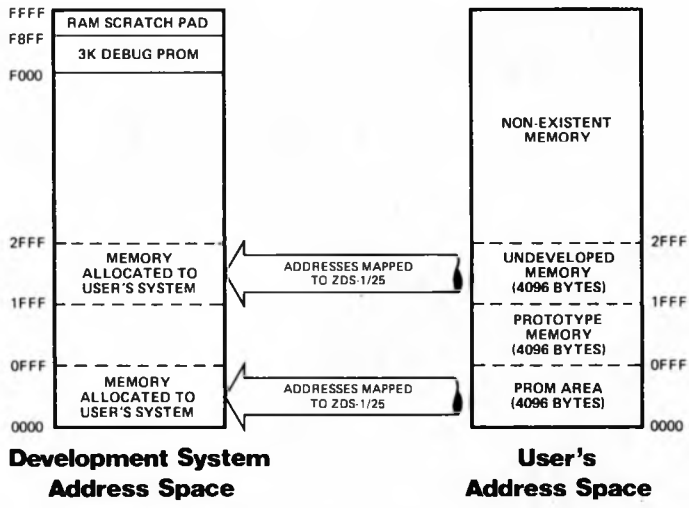
☐ SET — Sequentielle Speicherung von Daten im Speicher ab einer vom Benutzer festgelegten Adresse.

☐ TRACE — Der Benutzer bestimmt damit, welche Bus-Aktivitäten im Echtzeitspeicher Modul abgespeichert werden. Dabei ist irgendeine oder eine Kombination folgender Operationen frei wählbar:

- Speicher Lesebefehl
- Speicher Schreibbefehl
- Port Lesebefehl
- Port Schreibbefehl

Speicherabbildung

Mit dem Zusatz Speicherabbildung können Speicherblöcke entweder im ZDS-1/25 oder im Mustersystem angesprochen werden. Dieser Zusatz ist hauptsächlich dann sinnvoll, wenn Blöcke des Mustersystems aus ROM, PROM oder EPROM zusammengesetzt sind. Bei Verwendung der Speicherabbildung können Blöcke aus ROM, PROM oder EPROM quasi in den ZDS-1/25 RAM-Speicher verlegt werden, um die Fehlersuche in der Benutzer-Firmware wesentlich zu erleichtern, während der übrige Teil adressierbaren Speichers ruhig im Benutzersystem bleiben kann. Genausogut kann der Benutzer Speicherblöcke in das ZDS-1/25 laden, die bislang im Prototyp noch nicht entwickelt waren.



Obiges Bild zeigt eine typische Speicherabbildung, wie sie bei der Entwicklung von Prototypen auftritt. Die Abbildung erfolgt unter Verwendung einer festen Blocklänge von 256 Bytes. Ebenfalls zu beachten ist, daß bei der Speicherabbildung die Adreßbereiche im Benutzer- und Prototypsystem gleich sind.

ZDS-1-Serie

Zilog

Zusätze zu ZILOG's Mikrocomputer-Entwicklungs-System

Peripheriegeräte und Erweiterungszusätze

Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert (also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!

11. ZILOG-MIKROCOMPUTER-ENTWICKLUNGS-SYSTEME

Wir bieten eine Vielzahl an Hardwareerweiterungen, die die optimale Anpassung an die individuellen Anforderungen jeder einzelnen Mikroprozessorentwicklung gewährleisten. Eine Standard-Bus-Schnittstelle bietet die Möglichkeit, modular aufgebaute, höchstintegrierte Baugruppen zu verwenden, die ein Maximum an Verfügbarkeit und Durchsatz garantieren. Darüber hinaus steht eine leistungsfähige Software zur Verfügung, die einen unverzüglichen Einsatz der Schnittstellen bietet.

Zeilendrucker HOUSTON INSTRUMENTS, HI-8210/M
- ☐ Druckgeschwindigkeit 2400 Zeilen/min.
- ☐ 80 Zeichen/Zeile (auch Version mit 132 Zeichen/Zeile lieferbar)

Schnelldrucker OKI, DP-100/M
- ☐ Druckgeschwindigkeit 125 Zeilen/min.
- ☐ 132 Zeichen/Zeile
- ☐ Groß-/Kleinschreibung (96 ASCII-Zeichenvorrat)
- ☐ 9×7 Punktematrix

Schnelldrucker OKIDATA, CP-110/M
- ☐ Druckgeschwindigkeit 65 Zeilen/min
- ☐ 80 Zeichen/Zeile
- ☐ 5×7 Punktematrix
- ☐ 64 ASCII-Zeichenvorrat

Nadeldrucker INTEGRALSYSTEMS, IP-125/M
- ☐ Druckgeschwindigkeit 35 Zeilen/min.
- ☐ 80 Zeichen/Zeile
- ☐ Groß-/Kleinschreibung (96 ASCII-Zeichenvorrat)
- ☐ 7×7 Punktmatrix
- ☐ Stackelradwalze und Form Feed (für IP-225/M)

Nadeldrucker IP-225/GM
- ☐ wie IP-125/M, jedoch graphik-fähig, insbesondere als Hardcopyzusatz zusammen mit ZILOG Graphik-Video-Baugruppe Z80-VDB zu empfehlen.

Universalconsole COMPUTER DEVICES MINITERM, CD-1202/M
- ☐ Druckgeschwindigkeit bis zu 30 Zeichen/sec.
- ☐ Universalkonsole, geräuscharm, mit Serienschnittstelle (V24) und Tastatur

Bildschirmkonsole KONTRON 2100/M
- ☐ Serienschnittstelle (V24 oder 20 mA)
- ☐ Groß/Kleinschreibung
- ☐ zusätzliche Dezimaltastatur
- ☐ Übertragungsgeschwindigkeit einstellbar bis 9600 baud

Für die im vorangegangenen aufgeführten Peripheriegeräte stehen gesonderte Datenblätter zur Verfügung.

ZDS/CIB, Zusatzbaugruppe mit Centronics-Schnittstelle
- ☐ Zwei parallele Schnittstellen für den Anschluß eines PROM-Programmiergerätes PROLOG 90 und eines Druckers (2 St. Z80-PIO).
- ☐ Softwareunterstützung der Schnittstellen

ZDS/PPP, Zusatzbaugruppe mit Interface für OKIDATA-Drucker
- ☐ Zwei parallele Schnittstellen für Anschluß von Druckern und sonstigen Parallel-Ein/Ausgabe-Geräten
- ☐ Optional (ZDS/PPP-SE) eine Vollduplex-Serienschnittstelle
- ☐ Softwareunterstützung der Schnittstelle

ZDS/PPB Zusatzbaugruppe mit Programmiereinrichtung
- ☐ PROM/EPROM Programmierbaugruppe für folgende Bauelemente:
  EPROM 2704 und 2708
  BIPOLAR 7610, 7611, 7620, 7621, 7640 und 7641
- ☐ Softwareunterstützung durch entsprechende Hilfsprogramme

ZDS/PPB16 Zusatzbaugruppe mit Programmiereinrichtung
- ☐ wie ZDS/PPB, jedoch für folgende Bauelemente:
  EPROM i2758 und i2716
  BIPOLAR 82S27, 82S181, 82S131

ZILOG-
ZDS/1-Serie

Betriebssoftware
für Mikrocomputer-
Entwicklungs-Systeme

☐ **RIO**
☐ **PLZ**
☐ **FORTRAN**
☐ **BASIC**

Wir liefern Ihnen alle Systeme auf Wunsch fertig konfiguriert (also hard- und softwaremäßig angepaßt) mit Computer-Peripherie!

11. ZILOG-MIKROCOMPUTER-ENTWICKLUNGS-SYSTEME

Höhere Programmiersprachen

Über das Betriebssystem RIO hinaus bietet ZILOG ein breites Spektrum höherer Programmiersprachen.

BASIC/ZDS

BASIC ist eine leicht erlernbare, höhere Programmiersprache. Die gesamte Programmerstellung erfolgt in direktem Dialog mit dem ZDS, wodurch Editiervorgänge und Rückassemblierungen entfallen. Mit dem ZILOG-BASIC-Interpreter stehen eine Reihe sehr leistungsfähiger Kommandos und Anweisungen für Programmeingabe, -ausgabe, -start und Fehlersuche zur Verfügung.
BASIC erlaubt die Manipulation verschiedener Datentypen und Ausdrücke z.B. INTEGER (ganzzahlige)-Zahlen, Gleitkommazahlen und Zeichen-Ketten-Ausdrücke.
BASIC läuft unter dem RIO-Betriebssystem; damit steht die gesamte Dateiverwaltung von RIO auch für BASIC zur Verfügung, z.B. sequentieller und wahlfreier Zugriff. Ferner können Unterprogramme, die in Assemblersprache oder PLZ geschrieben sind, aufgerufen werden.

FORTRAN/ZDS

Mit dem Z80-FORTRAN-Compiler können ZDS-Anwender FORTRAN-Programme auf ihr System übernehmen und darüberhinaus die Vorteile nutzen, die durch die Vielzahl bereits existierender Programme vorhanden sind. FORTRAN entspricht voll der ANSI Norm X3.9-1966, (mit Ausnahme des Datentyps „Complex"); s. auch Punkt 5.
Z80-FORTRAN wird auf Floppy-Disk, komplett mit Bibliothekssoftware, Dokumentation und Benutzerhandbuch geliefert.

PLZ/ZDS

PLZ ist eine Familie von Programmiersprachen, die insbesondere zur softwarekostensparenden und effizienten Anwendung von Mikrocomputern entwickelt wurde. Durch die Verwendung einer ganzen Sprachfamilie werden die Nachteile bisher üblicher Programmiersprachen für die Mikrorechnertechnik eliminiert.
In PLZ werden maschinenunabhängige Programmteile wie Arithmetik und komplexe Entscheidungen über PLZ/SYS erstellt und übersetzt, während PLZ/ASM durch Mischen von Höheren Struktur- und Assembler-Anweisungen maschinenbezogene Programmierung zeitkritischer Probleme wie Interrupt und Ein/Ausgabe ermöglicht.
Diese getrennte Behandlung von Teilproblemen des Anwender-Programms führt zu einer äußerst effizienten Programmentwicklung.
PLZ ist durch seine klar durchkonstruierte Syntax und seine Übersichtliche Notation sehr leicht erlernbar.

DEV/Z8-Programmpaket
zur Entwicklung und zum Austesten von Programmen für ZILOG's Einchip-Minicomputer Z8.
Das Paket umfaßt einen strukturierten Assembler (PLZ/ASM-Z8), einen Cross-Simulator und erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.

ASM-Z8000-Programmpaket
zur Entwicklung von Programmen für ZILOG's Z8000, den ersten 16 bit-Mikrocomputer der Vierten Generation. Das Programm erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.

- ☐ Betriebssysteme
- ☐ BASIC-Interpreter
- ☐ PLZ-Familie
- ☐ FORTRAN-IV-Compiler
- ☐ COBOL-Compiler
- ☐ Z8-Entwicklungspaket (PLZ-Assembler und Simulator)
- ☐ Z8000-Assembler

12. ZILOG-SYSTEM-BETRIEBSSOFTWARE

Zu den im Vorhergehenden beschriebenen Mikrocomputersystemen sind die folgenden Standardprogrammpakete lieferbar

a) Zu den Computerbaugruppenfamilien MCB und MCB/E sind die Betriebsprogramme MO1 und MO3 lieferbar. All diese Programme enthalten die Steuerroutinen für die serielle Kommandoschnittstelle, die Bootstrap-Routinen, Programme zur Verzweigung zu einer beliebigen Anwenderroutine und die Möglichkeiten zum Lesen und Einschreiben in Speicherstellen und ganze Speicherbereiche. Sie können zusätzlich CPU-Registerinhalte lesen und ändern, außerdem Haltepunkte (= „Breakpoints") setzen und auswerten. MO3 umfaßt über den Leistungsumfang des MO1 hinaus Ansteuerungsprogramme für hardsektorierte Floppy-Disks.

b) Zu der Computerbaugruppenfamilie ECB existieren die gleichen Betriebsprogramme wie für die MCB-Serie (Bestellbezeichnungs-Beispiel: Z80-ECB/1 ist das leistungsmäßig dem MO1 entsprechende Programm für die ECB-Serie).

c) Im Lieferumfang des Z80-KIT ist bereits ein Betriebsprogramm inbegriffen, das die Entwicklung und das Hantieren von Anwenderprogrammen auf Hexadezimalebene (auch auf Audio-Cassettenrecordern!) erlaubt.
Die Software für die Programmierung von PROM-Bausteinen ist optional.

d) Das Betriebssystem für den OEM-Computer Z80-MCZ umfaßt (im Preis inbegriffen) standardmäßig:
- 3kB-Monitor (= MO3) inclusive Testsoftware
- Dateiverwaltungs-Programmpaket (RIO)
- Editor
- Makro-Assembler (relokativ)

e) Das Betriebssystem für die Z80-Entwicklungssysteme umfaßt (im Preis inbegriffen) standardmäßig:
- 4 kByte-Betriebsprogramm mit Kaltstart-, Serienschnittstellen- und Test-Software
- Dateiverwaltungs-Programmpaket (RIO)
- Editor
- Makroassembler (relokativ).

Hinzuzufügen ist, daß diese Programme auch die Testmöglichkeiten mit dem Hardware-Testadapter und dem Echtzeitspeichermodul bestimmen.

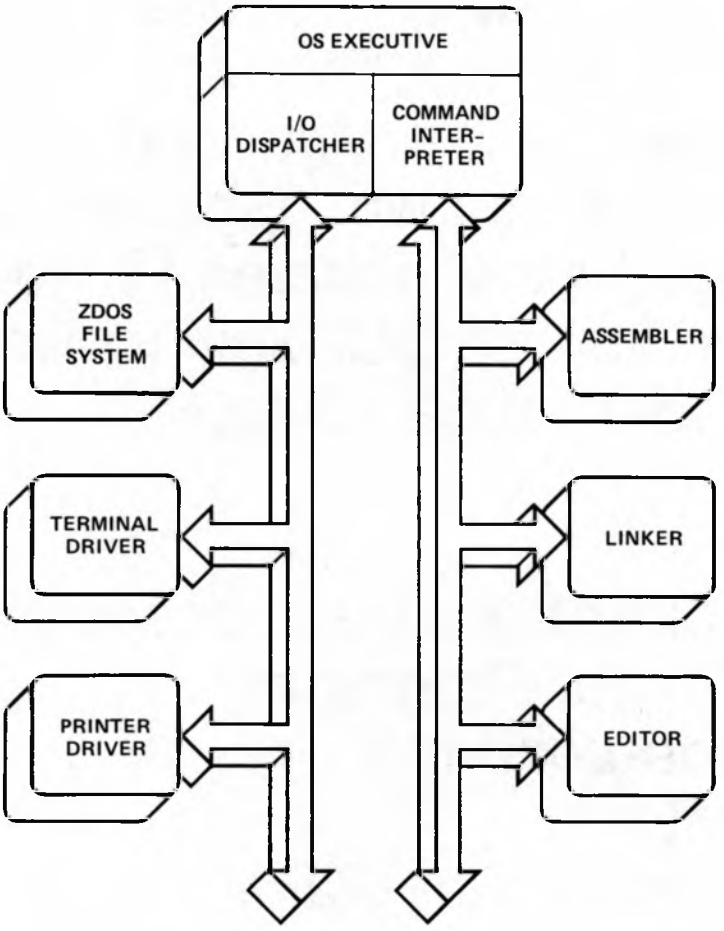
Das System Z80 umfaßt zur Zeit folgende Übersetzungsprogramme.

- Z80-ZCON: Übersetzungsprogramm für 8080- oder 8008-Primärprogramme in den Z80-Source-Code.
- Z80-ASM: Relokatives Makro-Assemblerprogramm für Z80-ZDS oder Z80-MCZ, (s. dort).
- Z80-BASIC: BASIC-Interpreter für Z80-ZDS oder Z80-MCZ (Speicherbedarf: 48 kByte) Er umfaßt die Möglichkeit des Einbindens von Assemblerroutinen und erfordert RIO als Betriebssystem
- Z80-PLZ: Compiler zur Übersetzung von PLZ-Programmen in Z80-Maschinensprache. Die Sprache PLZ ist eine völlig neue, erstmals der Struktur eines Mikrocomputers tatsächlich angepaßte, höhere Programmiersprache, die extrem hohe Effizienz (typ. 95%) in Speicherbedarf und Verarbeitungsgeschwindigkeit erzielt. Sie vereint in sich Elemente der bekannten Programmiersprachen PL/I, PASCAL und ALGOL 68, ist jedoch keine in irgendeiner Form abgeänderte Version von PL/M oder einem seiner Derivate.
- Z80-COBOL: Compiler zur Übersetzung von in COBOL geschriebenen Programmen auf dem MCZ
- Z80-FORTRAN: Compiler zur Übersetzung von in FORTRAN geschriebenen Programmen auf dem MCZ oder ZDS.
- DEV/Z8-Programmpaket
zur Entwicklung und zum Austesten von Programmen für ZILOG's Einchip-Minicomputer Z8.
Das Paket umfaßt einen strukturierten Assembler (PLZ/ASM-Z8), einen Cross-Simulator und erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.
- ASM-Z8000-Programmpaket
zur Entwicklung von Programmen für ZILOG's Z8000, den ersten 16 bit-Mikrocomputer der Vierten Generation. Das Programm erfordert zum Ablauf das Betriebssystem RIO und 60 kByte Speicherausbau.

Z80-RIO

Zilog

Betriebssystem mit relokativem Assembler und I/O-Management



12. ZILOG-SYSTEM-BETRIEBSSOFTWARE

Z80-RIO ist ein Universal-Betriebssystem für die ZILOG-Mikrocomputersysteme (wie ZILOG-ZDS und Z80-MCZ).

Es stellt Entwicklungs- und Arbeitshilfen für Mikrorechner in einer Form zur Verfügung, wie sie bisher nur bei Mini- und Mega-Computern bekannt war. Gleichzeitig bildet es die Basis für alle höheren Übersetzer-Programme von ZILOG. Sein minimaler Speicherbedarf ist 32 kByte.

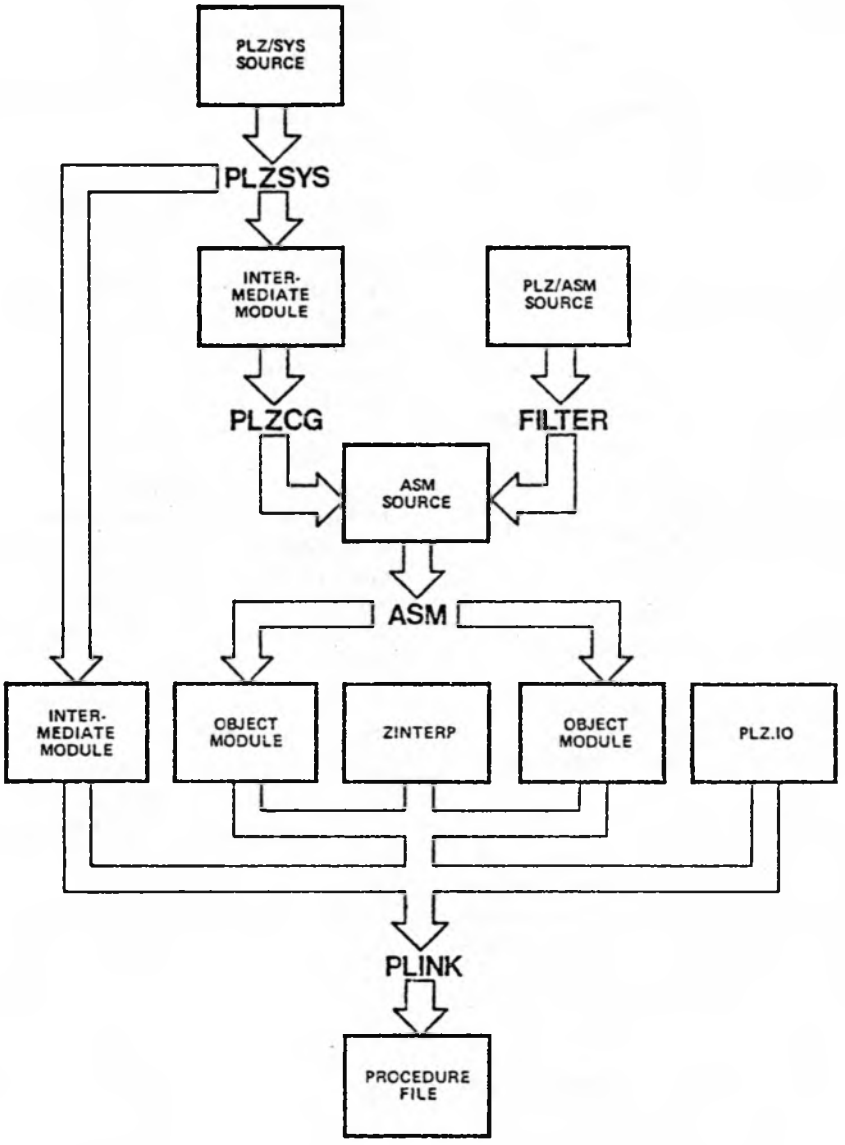
Es umfaßt:

- Einen relokativen Makroassembler
- Binder/Lader-Programm („Linkage-Loader“)
- Text-Editor
- Datei-Verwaltungs-Paket
- Treiber für Systemkonsole
- Fehlersuchprogrammpaket
- Völlige Transparenz und freie Konfigurierbarkeit des Betriebssystems
- Möglichkeit zur BATCH-Verarbeitung
- Computerbedienung ohne Fachpersonal

Z80-PLZ

Zilog

Sprachfamilie für effiziente Mikrocomputer-Programmentwicklung



12. ZILOG SYSTEM-SOFTWARE

Z80-PLZ

PLZ ist eine Familie von Programmiersprachen, die insbesondere zur softwarekostensparenden und effizienten Anwendung von Mikrocomputern entwickelt wurde. Durch die Verwendung einer vollständigen Sprachfamilie werden die Nachteile bisher üblicher Mikrocomputer-Programmiersprachen vermieden.
In PLZ werden maschinenunabhängige Programmteile wie Arithmetik und komplexe Entscheidungen über PLZ/SYS erstellt und übersetzt, während PLZ/ASM durch Mischen von Höheren Struktur- und Assembler-Anweisungen maschinenbezogene Programmierung zeitkritischer Probleme wie Interrupt und Ein/Ausgaben ermöglicht.
Diese getrennte Behandlung von Teilproblemen des Anwender-Programms führt zu einer äußerst effizienten Programmentwicklung.
PLZ ist durch seine klar durchkonstruierte Syntax und seine übersichtliche Notation sehr leicht erlernbar.

Beschreibung des PLZ/SYS-Compilers

- ☐ PLZ/SYS ist eine höhere, prozedurorientierte Sprache, besonders geeignet für die Entwicklung von
  - ● Betriebssystemen
  - ● Betriebsnahen Programmen und
  - ● Anwendersoftware
- ☐ Die Sprachelemente erlauben eine effiziente Implementierung der Anwendersoftware unabhängig vom verwendeten Mikroprozessor.
- ☐ Das übersichtliche Konzept zielt auf eine schnelle und einfache Programmübersetzung ab und ermöglicht hierdurch eine weitere Produktivitätssteigerung in der Programmierung
- ☐ Der effiziente Befehlscode bietet neben der Einsparung von Speicherplatz eine erhebliche Verringerung der Programmlaufzeit.
- ☐ Eine dynamische Datenzuordnung erlaubt sowohl reentrante als auch rekursive Prozeduren.
- ☐ Die Sprache führt zur Optimierung der Software-Erstellung und -Pflege.

Beschreibung des PLZ/ASM-Übersetzers

- ☐ PLZ/ASM erlaubt die Kombination der Vorteile einer höheren Programmiersprache mit dem Zugriff auf die Prozessorarchitektur: Alle Möglichkeiten der Assemblersprache einschließlich Register-, Speicher- und Ein/Ausgabeoperationen stehen zur Verfügung.
- ☐ Mit PLZ/ASM werden zeitkritische und Interruptprozeduren implementiert und später mit den getrennt übersetzten PLZ/SYS-Moduln zusammengebunden.
- ☐ Datendeklarationen wie RECORDS und ARRAYS, Abfragen wie IF.. THEN.. ELSE.. FI.. oder Schleifenprogrammierung über D0..0D bilden so eine Erweiterung des Assemblers in der Struktur als auch in der Programmierung auf höherer Ebene.
- ☐ Erleichterung der Assembler-Programmierung durch die Verwendung von high-level-Anweisungen.

PLZ ist ein Teil aus dem Programmpaket für Zilog Computersysteme und wird auf Floppy-Disk geliefert.
PLZ arbeitet in Verbindung mit dem RIO-Betriebssystem und benötigt einen minimalen Programmspeicher von 48 K bytes.
PLZ-Moduln lassen sich mit von anderen Übersetzern wie BASIC, FORTRAN oder COBOL generierten Moduln zusammenbinden.

PLZ/SYS-Sprachbeschreibung

PLZ/SYS ist eine prozedurorientierte Sprache. Die Sprachelemente bilden ein Derivat der bekannten Sprachen PASCAL, ALGOL und PL/I. Die Programmierung von Algorithmen erfolgt einfach in einer höher strukturierten Sprache, ohne die Architektur des Prozessors berücksichtigen zu müssen.
PLZ/SYS übernimmt dabei die Verwaltung der Computerfunktionalbestandteile wie Register und Speicher und ermöglicht auf diese Weise dem Anwender, sich auf die Bearbeitung seines Anwenderproblems konzentrieren zu können, ohne dabei die Hardwaregegebenheiten des Computers berücksichtigen zu müssen.
Der Anwendung des Verfahrens der Modularen Programmierung bietet sich durch den eindeutigen, übersichtlichen Datenfluß zwischen den Modulen geradezu an und begünstigt die Programmentwicklung durch den Vorteil der getrennten Übersetzung einzelner Module, die sowohl aus Datenerklärungen als auch aus Programmbefehlen gebildet werden.
Entsprechend der Aufgabenstruktur können vom Anwender eigene Datentypen definiert werden. Zu den standardmäßig implementierten Datentypen gehören u.a. Zeichenketten (ARRAYS), Datensätze (RECORDS) und Adressenfelder (POINTERS).
Sorgfältige Datentypprüfung seitens des Compilers verhindert den unbeabsichtigten Übergang von einem zum anderen Datentyp und eine damit evtl. verbundene ungewollte Veränderung von Daten.
Für Fälle, die einen solchen Übergang gezielt erfordern, sind eigene Anweisungen implementiert.
PLZ-Programme sind gut lesbar und tragen so wesentlich zur Dokumentation bei.
Die klare Interpunktion erlaubt, den Programmtext praktisch wie eine natürliche Sprache zu lesen.
PLZ/SYS-Programme werden direkt auf ZILOG-Mikrocomputersystemen kompiliert.
In PLZ/SYS sind bewußt keine Ein/Ausgabe-Anweisungen implementiert, da diese durch die meist problemorientierte, unterschiedliche Hardware in Mikrocomputern sehr wenig wirksam sind. Stattdessen wird mit dem Compiler ein ganzer Satz Ein/Ausgabe-Moduln standardmäßig mit zur Verfügung gestellt.

PLZ/SYS Elemente

Datenfelder und Dateien

- ☐ LITERALE — dezimale wie hexadezimale Zahlen und mit Hochkommata abgegrenzte Zeichenketten können wie Festwerte behandelt werden.
- ☐ KONSTANTEN — erlauben die Verwendung symbolischer Namen für feste Werte.
- ☐ VARIABLEN — ihr Wert ändert sich entsprechend der Programmausführung. VARIABLEN können initialisiert und durch den Zusatz „GLOBAL" über das gesamte Programm hin benutzt werden. Durch Kennzeichnung einer Variablen mit „INTERNAL" definiert man sie **innerhalb eines Moduls** und durch die Kennzeichnung „LOCAL" **innerhalb einer Prozedur.**

  Folgende Variablen stehen zur Verfügung:
  - ● BYTE — 8-bit ohne Vorzeichen
  - ● SHORT-INTEGER — 8-bit mit Vorzeichen
  - ● WORD — 16-bit ohne Vorzeichen
  - ● INTEGER — 16-bit mit Vorzeichen
  - ● POINTER (Zeiger) — maschinenabhängiger Wert, der die Adresse einer anderen Variablen angibt.
- ☐ MATRIZEN — eine Zusammenstellung von einem oder mehreren Werten gleichen Typs. Innerhalb einer Matrix („ARRAY") können die einzelnen Elemente durch einen variablen Index adressiert werden (z.B. A [I]).

Mehrdimensionale Matrizen und Matrizen mit Sätzen (= „RECORDS") als Elemente sind zulässig.

☐ SÄTZE — eine Zusammenstellung einer oder mehrerer Werte bzw. Daten verschiedenen Typs die über ein Namenkonstrukt (wie z.B. PATIENT.GEBURTG) angesprochen werden kann.
Ein Satz („RECORD") kann selbst aus Sätzen und Matrizen bestehen. Über Zeigervariable können beliebig komplexe Datensatzketten aufgebaut werden. Innerhalb eines Satzes können Felder als Zeigervariablen definiert sein, die wiederum die Adresse weiterer RECORDS angeben.

☐ TYP — erlaubt dem Anwender zusätzliche Datentypen bestimmter Eigenschaft selbst zu definieren, Teilemengen verwendeter Variablen zu kategorisieren und den Zugriff auf sie gezielt zu kanalisieren.

☐ AUSDRÜCKE — folgende Ausdrücke (= „EXPRESSIONS") für vergleichende, logische und arithmetische Operationen stehen zur Verfügung:

- Addition, Subtraktion, Multiplikation, Division und Modulus
- Inkrement, Dekrement, Bildung von Absolut-Werten und Negation
- Logisches Komplement, Und, Oder, Exclusiv- oder
- Gleich-, Größer- und Kleiner-Anweisung (=, >=, >, <=, <, <>)
- ASIZE gibt die Anzahl der Elemente in einer Matrix an
  kennzeichnet den Adreßwert einer gegebenen Variablen.

Ausführbare Anweisungen in PLZ/SYS

☐ ASSIGNMENT — erlaubt die Zuweisung (= „ASSIGNMENT") eines Ergebnisses aus einem Ausdruck zu einer Variablen.

☐ IF..THEN..ELSE..FI — Steuerung der Programmausführung abhängig von Bedingungen oder Ausdrücken.

☐ SELECT-STATEMENT — erlaubt die Ausführung von einer oder mehreren verschiedenen Anweisungen (statements) abhängig vom Wert des „Selectorausdrucks".
Eine Voreinstellung ist optional.

☐ DO..OD — ermöglicht den Aufbau von Schleifen, die Anweisungen innerhalb der Schleife wiederholt zur Ausführung bringen.

☐ EXIT — veranlaßt das Verlassen einer Programmschleife; mehrere Ausgänge innerhalb einer Schleife sind zulässig.

☐ REPEAT — veranlaßt die Wiederholung einer Schleife vom Beginn an. Mehrfach-Repeat-Anweisungen sind zulässig.

☐ RETURN — schließt eine bestehende Prozedur ab und übergibt die Kontrolle wieder dem aufrufenden Programm.

☐ PROCEDURE — enthalten ausführbare Anweisungen. Parameter können an die Prozedur und von der Prozedur übergeben werden. Innerhalb einer Prozedur können Variablen als LOCAL deklariert werden, die dynamisch zugewiesen werden. Dadurch wird Rekursivität und Reentranz der Prozeduren gewährleistet.
Prozeduren, die nur 1 Wert als Ergebnis haben, können in Ausdrücken eingesetzt werden. Andernfalls müssen sie ihrerseits in einer Prozedur aufgerufen werden, die Mehrfach-Zuweisungen erlaubt.
Prozeduren können als GLOBAL (im ganzen Programm zugänglich) oder INTERNAL (nur im betreffenden Modul zugänglich) erklärt werden.

PLZ/ASM-Sprachbeschreibung

PLZ/ASM ist eine strukturierte Assemblersprache mit zusätzlichen Anweisungen aus der Höheren Programmiersprache PLZ/SYS.
PLZ/ASM bietet sämtliche Möglichkeiten der Assemblersprache, die auf der Prozessorebene Zugriff auf Register, Speicher und Ein/Ausgabekanäle ermöglicht. Hinzu kommen die höheren Sprachanweisungen für die Steuerung des Programmablaufs und Datendeklarationen, die eine Programmierung auf dieser Ebene weniger zeitaufwendig und transparenter gestalten.
Die Datendeklarationen bilden eine Untermenge der PLZ/SYS-Deklarationen einschließlich RECORDS und ARRAYS.
Die Organisationsanweisungen entsprechen denen in PLZ/SYS.
Register und Bedingungsbits werden durch PLZ/ASM-Anweisungen nicht beeinflußt. Auf dieser Weise ist sichergestellt, daß die gesamte Prozessorsteuerung über das Anwenderprogramm erfolgen kann.
Die Anweisungen sind nicht formatgebunden und ermöglichen damit einen individuellen Aufbau hinsichtlich Dokumentation und Lesbarkeit.

PLZ/ASM Sprache-Elemente

Datentypen

☐ LITERALE — binäre, oktale und hexadezimale Zahlen und durch Hochkommata abgegrenzte Zeichenketten können als Festwerte benutzt werden.

☐ KONSTANTEN — erlauben die Verwendung symbolischer Namen für feste Werte.

☐ VARIABLEN — ihr Wert ändert sich entsprechend der Programmausführung. VARIABLEB können initialisiert und durch den Zusatz „GLOBAL" über das gesamte Programm hin benutzt werden. Durch Kennzeichnung einer Variablen mit „INTERNAL" definiert man sie **innerhalb eines Moduls** und durch die Kennzeichnung „LOCAL" **innerhalb einer Prozedur.**

Folgende Variablen stehen zur Verfügung.

- BYTE — 8-bit ohne Vorzeichen
- WORD — 16-bit ohne Vorzeichen

folgende Speicherstrukturen können erklärt werden:

- ARRAY — eine Zusammenstellung von mehreren Werten gleichen Typs. Anfangswertzuweisung ist möglich.
- RECORD — eine Zusammenstellung von Daten verschiedenen Typs, einschließlich BYTE, WORD, ARRAY und andere RECORDS.

PLZ/ASM Ausführbare Anweisungen

Zusätzlich zu dem ASSEMBLER-Befehlsvorrat bietet PLZ/ASM folgende Anweisungen der höheren Programmiersprache PLZ/SYS:

□ IF..THEN ELSE..FI	Bedingte Ausführung von Programmteilen. Verschachtelungen sind realisierbar.
□ SELECT	erlaubt die Ausführung von einem oder einer Gruppe von Befehlen abhängig von dem Inhalt eines Registers. Standardwertzuweisung ist möglich.
□ DO..OD	ermöglicht den Aufbau von Schleifen
□ EXIT	Anweisung zum Verlassen einer Programmschleife.
□ REPEAT	veranlaßt die Wiederholung einer Schleife von Anfang an.
□ PROCEDURE	enthalten sowohl ausführbare Anweisungen als auch Deklarationen von Variablen des Typs LOCAL. Prozeduren können sowohl GLOBAL (im ganzen Programm zugänglich) oder INTERNAL (nur im betreffenden Modul zugänglich) erklärt werden.

PLZ-Programmimplementierung

Die PLZ-Sprachenfamilie wurde zunächst für die Mikroprozessoren Z80-CPU und Z80A-CPU realisiert. Die PLZ-Soft ware steht auf Floppy-Disketten zur Verfügung und läuft unter dem Betriebssystem RIO auf ZILOG-Mikrocomputersystemen ab.

— PLZ-SYS-Compiler —

Der PLZSYS-Compiler erstellt aus einem Quellenprogramm oder Modul einen Zwischencode und ein Programmlisting mit Fehlermeldungen. Der Zwischencode kann dann entweder mittels eines „Code Generators" in den Z80-Objektcode transformiert werden oder direkt mit dem ZINTERP-Interpreter ausgeführt werden.

— PLZCG-Code Generator —

Der Code Generator übersetzt den Zwischencode aus dem PLZSYS-Compiler in Z80-Assemblerbefehle und bildet nach dem Assemblieren das Maschinenprogramm. Das kompilierte Programm ermöglicht wesentlich kürzere Programmlaufzeiten gegenüber dem Interpreter.

— ZINTERP-Interpreter —

Der Interpreter führt den Zwischencode direkt aus und bietet auf dieser Ebene wirkungsvolle Testmöglichkeiten.

— PLINK-Linker —

Binder Lader PLINK ermöglicht neben den Objektmoduln auch das Binden von Moduln mit dem Zwischencode aus dem PLZSYS-Compiler.

— FILTER-PLZ/ASM Translator —

Der Translator bildet aus einem PLZ/ASM-Quellenmodul unter Einbeziehung der höheren Sprachelemente aus PLZ/SYS einschließlich der Datendeklarationen ein entsprechendes Assemblerprogramm. Über den Z80-Assembler wird dann das Objektprogramm erzeugt.

— PLZ.IO-Ein/Ausgabe-Programme —

Die mitgelieferten Ein/Ausgabe-Programme erlauben die Implementierung einer ablauforientierten EIN/AUSGABE-Schnittstelle im PLZ-System. Die Datenkommunikation zwischen Dateien oder peripheren Geräten kann daher einfach über symbolische Zuweisungen erfolgen.

Z80®-BASIC

Zilog

BASIC-Interpreter für Systeme der Serien MCZ und ZDS

Z80-BASIC-INTERPRETER

Zilog BASIC ist eine leicht erlernbare, höhere Programmiersprache für MCZ- und ZDS-Anwender.
Die gesamte Programmerstellung erfolgt in direktem Dialog mit dem ZILOG-Computer, wodurch zeitraubende Editiervorgänge und Rückassemblierungen entfallen. Mit dem Zilog BASIC-Interpreter stehen eine Reihe sehr leistungsfähiger Kommandos und Anweisungen für Programmeingabe, -ausgabe, -start und Fehlersuche zur Verfügung.
BASIC erlaubt die Manipulation verschiedener Datentypen und Ausdrücke z.B. INTEGER (ganzzahlige)-Zahlen, Gleitkommazahlen und Zeichen-Ketten-Ausdrücke.
BASIC läuft unter dem RIO-Betriebssystem; damit steht die gesamte Dateiverwaltung von RIO auch für BASIC zur Verfügung, z.B. sequentieller und wahlfreier Zugriff. Ferner können Unterprogramme, die in Assemblersprache oder PLZ geschrieben sind, aufgerufen werden.

12. ZILOG SYSTEM-SOFTWARE

Eigenschaften des Z80-BASIC-Interpreters

Programmeingabe und Editierung

- Syntax-Überprüfung bereits bei der Befehlseingabe
- genaue Kontrolle über das Format der Programm-Ausgabe
- Löschen und korrigieren von einzelnen Programmzeilen oder Programmteilen
- Neunumerierung von Zeilengruppen oder des gesamten Programms ist möglich

Spracherweiterungen

- Z80-BASIC entspricht schon dem vorgeschlagenen ANSI-Standard mit wenigen geringfügigen Abweichungen
- IF-THEN, ELSE mit DO-DOEND bietet Möglichkeit zur strukturierten Programmierung
- Mehrzeilige Funktions-Definitionen
- Volle Zeichenketten-Manipulationsfähigkeit
- Zeichenketten-Felder
- Überprüfung von Eingabeoperationen während des Programmablaufes
- Verkettung von Programmen und Weitergabe von Variablenwerten

Ein/Ausgabe

- Dateiverwaltung mit RIO
- Wahlfreier Dateizugriff
- Binär- und ASCII-Dateien

Mathematische Funktionen

- Binär- oder BCD-Mathematik
- INTEGER-Zahlendarstellung zur Erhöhung der Geschwindigkeit und Reduktion des Speicherbedarfs
- Trigonometrische, logarithmische und exponentielle Funktionen
- Umwandlung von Zeichen- in Zahlen- und Zahlen- in Zeichenvariablen

Fehlersuche

- Unterbrechungsmöglichkeit des Programms zur Kontrolle und Änderung von Variablen
- Einzelschritt-Steuerung
- sofort durchführbare Befehle

48k Speicherbedarf

BASIC-Anweisungen

CALL prozedurname
Aufruf eines Assembler- oder PLZ-Unterprogramms (oder -prozedur) mit der Möglichkeit Variable und Ausdrücke zu übernehmen

CHAIN programmname
Beendet das laufende Programm und lädt das neue, dessen Name im CHAIN-Befehl angegeben ist. Variable können mit dem COM-Befehl übergeben werden.

CLOSE [#dateinummer [,#dateinummer [,...]]]
Schließt alle angegebenen Dateien ab und löst die durch die durch FILE-Anweisung hergestellten Zuordnungen (Dateinahme — Dateinummer) wieder auf.

COM variable, variable, ...
Zuweisung der angegebenen Variablen zu mehr als einem Programm

DATA konstante, konstante, ...
stellt Daten zum Lesen mit dem READ-Befehl zur Verfügung.

DEF funktionsname = funktion
Eröffnet eine Funktions-Definition

DIM variable [integer], variable [integer], ... oder
DIM variable [integer, integer], ...
Reserviert Speicherplatz für Felder und setzt Obergrenzen für die Anzahl der Feldelemente. Bereits existierende Felder können nochmals neu dimensioniert werden.

DO...DOEND Nur in Verbindung mit IF...THEN oder ELSE, umschließt eine Befehlsgruppe, die nur bei einer gültigen IF oder ELSE-Bedingung ausgeführt werden.

ELSE programmzeile/befehl/DO
Nur in Verbindung mit IF...THEN. Erlaubt die Ausführung eines Befehls, wenn die IF-Bedingung „falsch" ist.

END Beendet die Durchführung des laufenden Programms; kann weggelassen werden, falls die letzte Zeile des Programms automatisch als Programmende interpretiert wird.

ERASE dateiname [,variable]
Löscht die angegebene Datei

FILE dateinummer; dateiname [,variable]
Dem Dateinamen wird eine Dateinummer zugewiesen und die genannte Datei wird eröffnet. Eine vorher mit derselben Nummer eröffnete Datei wird auch automatisch abgeschlossen.

FNEND Beendet eine mehrzeilige Funktionsdefinition

FOR variable = funktion TO funktion
[STEP funktion]
.
.
.

NEXT variable
Erlaubt die Wiederholung einer Gruppe von Befehlen zwischen FOR und NEXT. Die Wiederholungsrate wird durch Anfangs- und Endwert der FOR-Variablen bestimmt (mit wählbarer Schrittgröße)

GOTO programmzeile
Sprung zur angegebenen Programmzeile

GOSUB programmzeile
Bewirkt die Ausführung eines Unterprogramms ab der angegebenen Programmzeile. Nach einem RETURN-Befehl im Unterprogramm wird der Programmablauf in der nach GOSUB folgenden Programmzeile wieder fortgesetzt.

IF funktion
.
.
.

THEN programmzeile/befehl/DO
Weist der IF-Bedingung einen Wert zu und spezifiziert die durchzuführende Operation bei erfüllter Bedingung. War die Bedingung ein mathematischer Ausdruck, wird sie beim Ergebnis ungleich null, als „wahr" betrachtet, als „falsch" beim Ergebnis null. Die Operation kann ein Sprung zu einer Programmzeile, ein einzelner Befehl oder eine DO...DOEND-Befehlsgruppe sein.

INPUT variable, variable,...
Erlaubt die Eingabe von einer oder mehreren Variablen von der Konsole. Die Aufforderung erfolgt mit „?"

INPUT dateinummer; variable, variable,...
Wie INPUT, nur als Eingabe von einer Datei.

LET variable = funktion
Ermöglicht die Zuweisung eines oder mehrerer Werte zu einer Variablen oder zu Feldelementen.

LINPUT variable
Erlaubt die Eingabe einer alphanumerischen Zeichenfolge von der Konsole, die einer einzigen Variablen zugeordnet wird.

LINPUT dateinummer; variable
Wie LINPUT, nur Eingabe von einer Datei.

NEXT	siehe FOR...NEXT Beendet eine Schleife, die mit FOR eröffnet wurde. Die NEXT-Variable muß der FOR-Variablen entsprechen.
ON . . . GOSUB	funktion programmzeile, programmzeile,... GOSUB-Befehl mit Aufruf eines Unterprogramms aus einer Liste. Die Verzweigung erfolgt in Abhängigkeit vom ganzzahligen Wert eines mathematischen Ausdruckes.
ON . . . GOTO	funktion programmzeile, programmzeile, ... GOTO-Befehl mit Sprung zu einer Programmzeile aus einer Liste. Die Verzweigung erfolgt wie bei ON...GOSUB.
ON . . . RESTORE	variable programmzeile, programmzeile,... Erlaubt die Wiederverwendung einer „markierten" Datenliste. Die Auswahl erfolgt wie bei ON...GOSUB.
PRINT	[variablen] Ausgabe der nachfolgenden Liste von Variablen und mathematischen Ausdrücken.
PRINT	#dateinummer [; variablen] Ausgabe wie PRINT, jedoch ASCII-Datei.
PRINT USING	format wie PRINT, aber in vorgegebenem Format.
RANDOMIZE	Auswahl einer zufälligen, aber nach jedem Programmstart normalerweise wiederkehrenden Startzahl für die RND-Funktion.
READ	variable, variable, ... Lesen von Werten aus einer oder mehreren Detaillisten.
READ	#dateinummer; variable, variable, ... wie READ, jedoch Lesen einer Binär-Datei.
REM	zeichen Zum Einfügen einer Kommentarzeile.
RESTORE	[programmzeile] Wiederverwendung sämtlicher Datenlisten oder einer durch die Zeilennummer markierten Dateiliste.
RESTORE	#dateinummer [,marke] Wiederverwendung einer gesamten Datei oder ab einer markierten Stelle.
RETURN	Rücksprung aus einem Unterprogramm (siehe GOSUB)
SPACE	dateinummer, richtung Bewegt den Zeiger in einer Datei vorwärts oder rückwärts.
STOP	Unterbricht die Ausführung des laufenden Programms.
SYSTEM	systemfunktion Steuert Systemfunktionen (z. B. Zeilenlänge, Fehlermeldungen, usw.)
TRAP	Erlaubt oder verbietet die Überprüfung der Bedingungen ESC, KEYS, ERR, EOF und EXT.
TRUNCATE	dateinummer setzt die EOF-Marke einer Datei
WRITE	dateinummer [,marke]; variablen Schreiben einer Binär-Datei.

Z80-BASIC-Kommandos

APPEND	programmname Verknüpft ein bestimmtes Programm (ASCII-Form) mit dem im Arbeitsspeicher befindlichen.
ASAVE	programmname Kopiert das im Arbeitsspeicher stehende Programm auf Diskette.
CAT	RIO-Kommando für die Ausgabe des „Inhaltsverzeichnisses" der Diskette.
CLEAR	Löscht den gesamten Speicherbereich, der **nicht** direkt vom Programm belegt wird.
CONTINUE	Fortsetzung des Programmablaufes nach ESC oder STOP.
DELETE	programmzeile(n) Löschen einer oder mehrerer Programmzeilen.
GET	programmname Laden eines neuen Programms.
LIST	[programmzeile(n)] Ausgabe des Programms oder der angegebenen Programmzeilen auf die Konsole.
NEW	Löscht den gesamten Arbeitsspeicher des momentanen Anwenderprogramms.
QUIT	Rückkehr ins RIO-Betriebssystem.
RENUMBER	[neuordnungsangaben] Neuordnung der Programmzeilennumerierung beginnend mit Zeilennummer 10 in Zehnerschritten, falls nicht anders angegeben.
RUN	[programmzeile] Ausführung des im Arbeitsspeicher stehenden Programms.
RSAV	programmname Kopiert das im Arbeitsspeicher stehende Programm in eine bestehende Datei.
SAV	programmname Kopiert das im Arbeitsspeicher stehende (übersetzte) Programm auf Diskette.
SIZE	Status-Ausgabe von: freier Speicherplatz, Programmgröße, Größe des Variablenspeichers, Anzahl der geschützten 512-byte Speicherblöcke.
STEP	Ausführung des Programms in Einzelschritten (Einzelbefehlen = einzelnen Zeilen), ausgenommen mehrzeilige Funktionsdefinitionen.
XEQ	programmname Laden und sofortige Durchführung des Programms.

Z80-BASIC — vorprogrammierte Funktionen

ABS(x)	Betrag von x
ASC(s)	ASCII-Code des ersten Zeichens einer Zeichenkette.
ATN(x)	Arcustangens von x (x in rad).
CHR$(x)	Umwandlung einer Zahl zwischen 0 und 255 in entsprechendes ASCII-Zeichen.
COS(x)	Cosinus von x (x in rad).
EOF(x)	Information über EOF-Marke in Datei x (x ÷ Zahl zwischen 1 und 15): vorhanden ergibt 1 nicht vorhanden ergibt 0
ERR(x)	Liefert die Nummer des mit TRAP ERR zuletzt entdeckten Fehlers.
ESC	Liefert die Information über das Drücken der ESC-Taste in Verbindung mit dem TRAP EOF Befehl.
EXP(x)	e^x
INT(x)	Ergibt den größten ganzzahligen Wert kleiner oder gleich x.

KEYS$	Liefert die Zeichen, die während der Programmausführung über die Konsole eingegeben wurden.
LEFT$(s,n)	Liefert die n linksbündigen Zeichen der Zeichenfolge s. Left (A$,n) entspricht A$[1,n]
LEN(s)	Anzahl der Zeichen in der Zeichenkette s.
LOG(x)	Natürlicher Logarithmus von x (x größer 0).
POS(s1, s2)	Position des ersten Zeichens der Zeichenkette s2 innerhalb der Zeichenkette s1.
RIGHT$(s,n)	Liefert die n rechtsbündigen Zeichen der Zeichenfolge s. RIGHT (A$,n) entspricht A$[n]
RND	Zufallszahl zwischen 0 und 1 (kleiner 1).
SEG$(s, n, m)	Liefert die Zeichenfolge zwischen dem n-ten undm-ten Zeichen der Zeichenfolge s. SEGS (A$, n, m) entspricht A$[n,m].
SGN(x)	Vorzeichentest ergibt: 1 für x größer 0, 0 für x gleich 0, −1 für x kleiner 0
SIN(x)	Sinus von x (x in rad).
STR$(x)	Interpretation von x als Zeichenkette.
SQR(x)	Quadratwurzel von x (x größer gleich 0).
TAB(x)	Sprung zur Druckposition x (modulo Zeichenlänge).
TRP	Ausgabe der Nummer der zuletzt durchgeführten Programmzeile vor dem Auftreten einer TRAP-Bedingung.
VAL(s)	Konvertierung und Interpretation einer Zeichenkette s als Zahl.

Z80-FORTRAN

Zilog

FORTRAN-IV-COMPILER für MCZ- und ZDS-Syssteme

Mit dem Z80-FORTRAN-COMPILER können Anwender der Zilog-Mikrocomputersysteme MCZ und ZDS FORTRAN-Programme implementieren. Die vielen existierenden FORTRAN-Anwendungsprogramme werden zugänglich.

Z80-FORTRAN ist mit auf Minicomputern laufenden Compilern vergleichbar. Es umfaßt das gesamte ANSI-Standard FORTRAN X 3.9–1966 mit Ausnahme des Datentyps COMPLEX.

Z80-FORTRAN wird auf Floppy-Diskette oder Plattenkassette mit Bibliotheks-Software-Dokumentation und Anwendungshinweisen geliefert.

Relokativer Code
Z80-FORTRAN ist insofern einzigartig, als daß verschiebbare Objektmodule erzeugt werden, die mit Z80-Assembler-Moduln verbunden werden können. Nur die für den Ablauf des FORTRAN-Programms notwendigen Systemroutinen müssen vorab geladen werden.

12. ZILOG SYSTEM-SOFTWARE

Bibliotheksunterstützung

Unterprogramme können in eine System-Bibliothek eingebracht und zu einem allgemein verwendeten Satz von Routinen werden. Bei modularem Programmaufbau muß nur der von einer Änderung betroffene Modul erneut compiliert werden.

Standardmäßig sind folgende Routinen in der Bibliothek enthalten:

ABS	LABS	DABS	AINT
INT	IDINT	AMOD	MOD
AMAXO	AMAX1	MAX0	MAX1
DMAX1	AMINO	AMIN1	MINO
MIN1	DMIN1	FLOAT	IFIX
SIGN	ISIGN	DSIGN	DIM
IDIM	SNGL	DBLE	EXP
DEXP	ALOG	DLOG	ALOG10
DLOG10	SIN	DSIN	COS
DCOS	TANH	SQRT	DSQRT
ATAN	DATAN	ATAN2	DATAN2
DMOD	PEEK	POKE	INP
OUT			

Außerdem Routinen für 32- und 64-bit Gleitpunktrechnungen usw.

Erweiterungen

Der Z80-Fortran Compiler bietet eine Reihe von Erweiterungen gegenüber dem ANSI-Standard:

1. LOGICAL-Variable der ganzzahligen Größen im Bereich von −128 bis +127.
2. LOGICAL DO-Schleifen für dichteren Code und schnellere Ausführung bei Zeitschleifen.

Z80-COBOL

COBOL-Compiler für MCZ-Systeme

ZILOG's Z80-COBOL entspricht der Norm ANSI X3.34 COBOL (1974; FIPS-designation Low Intermediate. Dadurch ist es möglich, vorhandene COBOL-Programme auf Mikrocomputersystemen zu übersetzen und Z80-Mikro-prozessor-Maschinencode zu erzeugen, ebenso natürlich kommerzielle Programme in dieser weitverbreiteten Sprache zu erstellen.

Die Leistungsfähigkeit des ZILOG Z80-COBOL-Compilers entspricht der vieler Minirechner bzw. übertrifft diese sogar.
Der Compiler läuft auf ZILOG-Floppy-Disk-basierenden Computern der Serie Z80-MCZ ab.
Voraussetzung ist ein minimaler Schreib/Lesespeicherausbau von 48 kByte und das Betriebssystem ZILOG RIO.

12. ZILOG SYSTEM-SOFTWARE

Eigenschaften

☐ 1974 ANSI-COBOL mittlerem Implementierungsniveaus.
- Sequentieller Datei-Zugriff
- Indizierter Datei-Zugriff
- Wahlfreier Datei-Zugriff
- Segmentierung
- Bibliothek
- Datenaustausch zwischen Programmen
- 18-stellige dezimale und binäre Datentypen

☐ Erweiterte Konsolen-Ein/Ausgabefunktion
- ACCEPT/DISPLAY-Anweisung
- Cursor-Positionierung
- Bildschirmlöschen
- Eingabe-Editierfunktion
- Eingabe-Quittungsfunktion

☐ Interaktive Fehlerbeseitigung
- keine besondere Kompilierung
- Einzelschrittausführung des Programms auf Anweisungsebene

Adreßhalt bei vorgegebener Marke

☐ Prüfroutinen
- „Reserve Word Check" gegenüber Anweisungen Höherer COBOL-Implementierungs-Levels.
- Undermark Inline
- Leicht verständliche Konsol-Meldungen

☐ Cross-References
- References für alle Symbole
- References für alle Deklarationen
- References für alle möglichen Ziele

☐ Anwenderfreundlicher Ablauf des Compilers
- 1-Pass-Compiler
- Kein Bindelauf erforderlich
- Schnelles Laden

Einzelheiten des implementierten Befehlssatzes

☐ **Nukleus**

Erweiterungen über Level 1 hinaus:
- Kommata und Semikolon als Separatoren zulässig (Level 2)
- Zeichen < = > dürfen in Bedingungen verwendet werden (Level 2)
- Qualifikation ist zulässig; daher muß nicht das lediglich einmalige Vorkommen von Daten-, Bedingungs- und Paragraph-Namen gewährleistet werden (Level 2)
- Datennamen müssen nicht mit einem Buchstaben beginnen (Level 2)
- Ein Wort oder ein numerisches Literal darf in einer Fortsetzungszeile fortgeführt werden (Level 2)
- Alle Format-Konstanten inclusive ZEROS, ZEROES, SPACES, HIGH-VALUES, QUOTES (Level 2)
- Level-Nummern 1....49 können einstellig sein (Level 2)
- Level-Nummer 88 ist zur Definition von Bedingungsnamen reserviert, die Bedingungs-Variablen zugeordnet sind (Level 2)
- Datentyp-Beschreibung beinhaltet USAGE bei COMPUTATIONAL-1 oder COMP-1 zur Beschreibung von vorzeichenbehafteten 1-Wort-Binärdaten in Zweierkomplement-Darstellung
- Datentyp-Beschreibung erlaubt verschachtelte REDEFINES (Level 2)
- Vergleich nichtnumerischer Operanden ungleichen Formats ist erlaubt (Level 2)
- Bedingungsausdrücke lassen Bedingungs-Name, Bedingung und komplexe Bedingung über die logischen Operatoren AND, OR und NOT zu (Level 2)
- Die ACCEPT-Anweisung gestattet die Zusätze FROM DATE, FROM DAY und FROM TIME (Level 2)
- ACCEPT gestattet multiple Operanden
- Bei ACCEPT und DISPLAY ist Spezifikation von Bildschirmkonsolen-Steuerfunktionen möglich.
- Die ALTER-Anweisung gestattet eine Prozedurnamen-Folge (Level 2)
- Bei der IF-Anweisung ist Verschachtelung zugelassen; Anweisungen im Rahmen einer IF-Bedingung können imperative oder Bedingungs-Anweisungen sein (Level 2)
- PERFORM beinhaltet Format 3 (PERFORM....UNTIL) und Format 4 (PERFORM....VARYING) aus Level 2.

Ausnahmen von Level 1
- In Datentyp-Beschreibung kann die SIGN-Klausel nicht LEADING für das Vorzeichen spezifizieren. Weglassen der SEPARATE-Phrase hat keine Wirkung; alle Vorzeichen sind eigene führende Zeichen.

☐ **Tabellenbearbeitung** (entspricht Level 1)

☐ **Sequentielle Ein/Ausgabe**

Erweiterungen über Level 1 hinaus
- Die Dateibehandlungsklausel SELECT erlaubt die Angabe des externen Dateinamens als ein Literal oder eine Datenmenge.
- Die Dateibeschreibungsklausel BLOCK CONTAINS beinhaltet die optionale Bereichssyntax von Level 2
- Die CLOSE-Anweisung erlaubt die Zusätze WITH LOCK und WITH NO REWIND, außerdem Dateinamen-Folgen (Level 2)
- Die OPEN-Anweisung erlaubt die Zusätze WITH LOCK und WITH NO REWIND, außerdem Dateinamen-, INPUT-, OUTPUT-, I-O- und EXTEND-Folgen (Level 2)
- Die USE-Anweisung erlaubt den EXTEND-Zusatz und Dateinamen-Folgen (Level 2)
- Die WRITE-Anweisung erlaubt die Benutzung eines Identifiers in BEFORE/AFTER ADVANCING... LINES

Ausnahmen von Level 1
- Bei der RERUN-Klausel ist für I-O-CONTROL keine Formatangabe zulässig.
- AM Anfang einer Dateibeschreibung darf kein CODE-SET stehen.

☐ **Relative Ein/Ausgabe**

Erweiterungen über Level 1 hinaus
- Die Dateibehandlungsklausel SELECT erlaubt die Angabe des externen Dateinamens als ein Literal oder eine Datenmenge.
- Die Dateibeschreibungsklausel BLOCK CONTAINS beinhaltet die optionale Bereichssyntax von Level 2
- Die DYNAMIC Zugriffsweise ist implementiert (Level 2)
- Bei der READ-Anweisung ist die NEXT-Option zugelassen (Level 2)
- START-Anweisung des Level 2 ist implementiert
- Die USE-Anweisung erlaubt Dateinamen-Folgen (Level 2)

Ausnahmen gegenüber Level 1
- Bei der RERUN-Klausel ist für I-O CONTROL keine Formatangabe zulässig.

☐ **Indizierte Ein/Ausgabe**

Erweiterungen über Level 1 hinaus
- Die Dateibehandlungsklausel SELECT erlaubt die Angabe des externen Dateinamens als ein Literal oder eine Datenmenge.
- Die Dateibeschreibungsklausel BLOCK CONTAINS beinhaltet die optionale Bereichssyntax von Level 2
- Die DYNAMIK Zugriffsweise ist implementiert (Level 2)

- Bei der READ-Anweisung ist die NEXT-Option zugelassen (Level 2)
- START-Anweisung des Level 2 ist implementiert
- Die USE-Anweisung erlaubt Dateinamen-Folgen (Level 2)

Ausnahmen gegenüber Level 1

- Bei der RERUN-Klausel ist für I-O CONTROL keine Formatangabe zulässig.

☐ **Segmentierung** entspricht Level 1

☐ **Bibliotheksprogramme**

Ausnahmen gegenüber Level 1

- Die COPY-Anweisung darf nur an Stellen auftreten, an denen auch ein Trennungszeichen vorkommen darf.
- Eine COPY-Folge muß die letzte Beginnadresse in Feld B eines Quell-Satzes sein.

☐ **Fehlersuchhilfen**

Erweiterungen über Level 1 hinaus

- Fehlersuchbetriebsart kann bei Aufruf des Compilers festgelegt werden.
- automatische, interaktive Fehlerbeseitigung

Ausnahmen gegenüber Level 1

- USE DEBUGGING ist nicht implementiert

☐ **Datenaustausch zwischen Programmen** ist implementiert.

COBOL Befehlsvorrat

IDENTIFICATION DIVISION

PROGRAM–ID

ENVIRONMENT DIVISION

CONFIGURATION SECTION

SOURCE–COMPUTER

OBJECT–COMPUTER (. . . MEMORY . . . SEQUENCE)

INPUT-OUTPUT SECTION

FILE–CONTROL

I/O–CONTROL
SAME AREA FOR FILENAME-1,–

SPECIAL NAMES (. . . ON 15, OFF 15 . . . ,
CURRENCY IS . . . , DECIMAL-POINT IS COMMA)

INPUT-OUTPUT SECTION

FILE-CONTROL (SELECT . . . ASSIGN . . .
ORGANIZATION . . . SEQUENTIAL . . .
RELATIVE . . . INDEXED . . . ACCESS . . .
SEQUENTIAL . . . RANDOM . . . DYNAMIC . . .
RECORD KEY IS . . . RELATIVE KEY IS . . .
STATUS IS)

I/O CONTROL (SAME . . . AREA)

DATA DIVISION

FILE SECTION

FD (BLOCK CONTAINS . . . RECORD CONTAINS . . .
LABEL RECORDS ARE . . . VALUE OF LABEL . . .
DATA RECORDS ARE)

88 CONDITION NAME; (VALUE IS . . . THROUGH)

WORKING STORAGE SECTION

LINKAGE SECTION

PROCEDURE DIVISION (USING . . .)

ACCEPT . . . FROM
ADD . . . TO . . . ROUNDED . . . SIZE ERROR . . .
GIVING
ALTER . . . TO
CALL . . . USING
CLOSE . . . NO REWIND . . . LOCK
DELETE . . . INVALID
DISPLAY . . . (UNIT . . . LINE . . . BEEP . . .
POSITION . . . SIZE . . . ERASE . . .)
DIVIDE . . . INTO/BY . . . ROUNDED . . . SIZE
ERROR . . . GIVING
EXIT (PROGRAM)
GO TO (DEPENDING ON)
IF . . . NEXT SENTENCE . . . ELSE . . . NEXT
SENTENCE
INSPECT . . . TALLYING . . . FOR
. . . REPLACING . . . BY . . .
. . . ALL/LEADING/FIRST CHARACTERS . . .
BEFORE/AFTER INITIAL
MOVE . . . TO
MULTIPLY . . . BY . . . (GIVING . . . ROUNDED . . . ON
SIZE ERROR)
OPEN (INPUT . . . OUTPUT . . . I-O . . . EXTEND)
PERFORM (THRU . . . TIMES . . . UNTIL . . . VARYING
READ . . . NEXT . . . (INTO . . .
AT END . . . INVALID KEY)
REWRITE . . . (FROM . . . INVALID KEY)
SET . . . TO . . . (UP BY . . . DOWN BY . . .)
START . . . (KEY . . . EQUAL . . . GREATER . . .
NOT LESS . . . INVALID)
STOP . . . (RUN . . .)
SUBTRACT . . . FROM . . . (ROUNDED . . . SIZE
ERROR . . . GIVING)
USE AFTER . . . (EXCEPTION. . . ERROR . . .) PRO-
CEDURE . . . (INPUT . . . OUTPUT . . . I-O . . . EX-
TEND) . . .
WRITE . . . (FROM . . . BEFORE/AFTER ADVANCING
. . . PAGE . . . LINES . . . INVALID KEY . . .)
LIBRARY . . . COPY

KONTRON
ELEKTRONIK GMBH

- Standard-Arithmetik-Paket (SAP)
- Gleitkomma-Paket (FPA)
- Höhere Funktionen in Gleitkommaformat (EXP)
- Schrittmotorsteuerung (SM)
- Programmierter Unterricht (PU)
- Adressen-Verwaltungspaket (ML)
- Bücherei-Verwaltung (BIB)
- Textverarbeitung (FORMAT)
- Regeltechnische Programme (PID)

13. Z80-Anwendersoftware

Standard-Anwender-Software

Selbstverständlich ist auch ein Spektrum von Standardprogrammen für das System Z80 verfügbar, das dauernd erweitert wird.

Insbesondere ist durch unsere jahrelange Entwicklungsaktivität mit dem System Z80 und verwandten Prozessoren ein Fundus von Standard-Anwender-Software vorhanden, der auf Anfrage dem Benutzer verfügbar gemacht werden kann. Teil dieser Aktivität ist beispielsweise das viel verwendete Gleitkommaarithmetik-Paket (Z80-FPA) das Erweiterungspaket mit höheren mathematischen Funktion (Z80-EAP) und das Standard-Festkomma-Paket (Z80-SAP).

Das Multi-Task-Betriebssystem MTX erlaubt die Koordinierung und Verwaltung mehrerer konkurrierender, zeitkritischer Probleme und Interrupt und ist insbesondere für die Verwendung mit Baugruppen der ECB-Serie geeignet. Ebenso ist für diese Baugruppenfamilie ein Satz verschiedener Monitorprogramme verfügbar, die von einfachen Steuerfunktionen (ECB/1) bis zur Floppy-Disk-Treiber-Software (ECB/2) reichen.

Für spezielle Anwendungen ist eine Bibliothek von anwendungsorientierten Programmen lieferbar. Hierzu gehören Schrittmotorsteuerungsprogramme (Z80-SM), Programme zur Durchführung von automatisiertem Unterricht (Z80-PU), ein Kundendatei-Verwaltungspaket (Z80-ML), ein Programmsystem zur Verwaltung von Büchereien (Z80-BIB) und ähnlich organisierten Organisationen und vieles andere mehr.

Z80-SAP Standard-Arithmetik-Programmpaket

```
DAS STANDARD ARITHMETIK PAKET (Z80-SAP) ENTHAELT
FOLGENDE UNTERPROGRAMME

ARI-01: 08 X 16 BIT INTEGER MULTIPLIKATION (BINAER)
ARI-02: 16 X 16 BIT INTEGER MULTIPLIKATION (BINAER)
ARI-03: 16 / 08 BIT INTEGER DIVISION   (BINAER)
ARI-04: 32 / 16 BIT INTEGER DIVISION   (BINAER)

CON-01: 16 BIT BINAER/BCD - CONVERSION
CON-02: 32 BIT BINAER/BCD - CONVERSION
CON-06: 05 DIG BCD/BINAER - CONVERSION

TRI-01: COSINUS/SINUS X  -  08 BIT FORMAT
TRI-02: COSINUS/SINUS X  -  16 BIT FORMAT

CON-03: 16 BIT HEX/ASCII - CONVERSION
CON-05: 04 CHR ASCII/HEX - CONVERSION


SUBROUTINE:         H16BCD
                    16 BIT BINAER/BCD - CONVERSION
NUMBER OF BYTES:    23
EXECUTION TIME:     470.0 MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L
MEMORY USED:        -

ON ENTRANCE:        BINAER DATEN IN HL
ON EXIT:            PACKED BCD DATEN IN A,HL


-------------------------------------------------------
SUBROUTINE:         H32BCD
                    32 BIT BINAER/BCD - CONVERSION
NUMBER OF BYTES:    44
EXECUTION TIME:     1 707.6 MICROSEKUNDEN

REGISTERS USED:     A,B,D,E,H,L,B',D',E',H',L'
MEMORY USED:        -

ON ENTRANCE:        BINAER DATEN IN HL,DE
ON EXIT:            PACKED BCD DATEN IN A,HL,DE


-------------------------------------------------------
SUBROUTINE:         H04ASC
                    HEX/ASCII - CONVERSION (4 CHARACTERS)
NUMBER OF BYTES:    31
EXECUTION TIME:     143.8 - 146.8 MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,H,L
MEMORY USED:        4 LOCATIONS FROM (DE) UNTIL (DE + 3)

ON ENTRANCE:        HEX DATEN IN HL
                    DE POINTS TO WORKSPACE-BUFFER (WSB)
                    LENGTH = 4
ON EXIT:            DATEN IN WORKSPACE-BBUFFER
                    DE POINTS TO MSB
                    DE + 3    TO LSB


-------------------------------------------------------
SUBROUTINE:         A04HEX
                    ASCII (4 CHR.) / HEX CONVERSION
NUMBER OF BYTES:    44
EXECUTION TIME:     184.4 - 232.4 MICROSEKUNDEN

REGISTERS USED:     A,B,C,H,L,IX
MEMORY USED:        4 LOCATIONS FROM (IX) TO (IX+3)

ON ENTRANCE:        4 ASCII CHR. IN WORKSPACE-BUFFER (WSB)
                    (IX + 0) --> MSB
                    (IX + 3) --> LSB
ON EXIT:            HEX NUMBER (16 BIT) IN HL
                    CARRY BIT IS SET IF A NON HEX-CHR.
                    SHOULD BE CONVERTED
                    OTHERWISE IT IS RESET


-------------------------------------------------------

SUBROUTINE:         B05HEX
                    5 DIGIT BCD/BINAER CONVERSION
NUMBER OF BYTES:    48
EXECUTION TIME:     Z MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L
MEMORY USED:        5 LOCATIONS FROM (HL-2) UNTIL (HL+2)

ON ENTRANCE:        BCD DATEN IN WORKSPACE BUFFER (WSB)
                    LENGTH = 5 (HL-2) --> (HL+2)
                    (HL)    --> LSB IN WSB
                    (HL+2) --> MSB IN WSB
ON EXIT:            MSB IN (HL) AND LSB IN (HL-1)


SUBROUTINE:         COS108
                    COSINUS X OR SINUS X IN 8 BIT FORMAT
NUMBER OF BYTES:    51
EXECUTION TIME:     X MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L
MEMORY USED:        -

ON ENTRANCE:        X IN DEGREES IN A WITH:
                    A = X * 255/90
ON EXIT:            RESULT IN A WITH:
                    A =  SIN X * 255  OR
                      = -COS X * 255


-------------------------------------------------------
SUBROUTINE:         COS116
                    COSINUS X OR SINUS X IN 16 BIT FORMAT
NUMBER OF BYTES:    126
EXECUTION TIME:     X MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L,IX,IY
MEMORY USED:        -

ON ENTRANCE:        X IN DEGREES IN BC WITH:
                    BC = X * 2 EXP08
ON EXIT:            COSINUS X IN DE = COS X * 2 EXP15
                    SINUS X   IN HL = SIN X * 2 EXP15


-------------------------------------------------------


SUBROUTINE:         MHEX24
                    8 X 16 BIT INTEGER MULTIPLIER
NUMBER OF BYTES:    15
EXECUTION TIME:     134.4 - 176.0 MICROSEKUNDEN

REGISTERS USED:     A,B,D,E,H,L
MEMORY USED:        -

ON ENTRANCE:        MULTIPLIER IN HL
                    MULTIPLICAND IN DE
ON EXIT:            RESULT IN A,HL


-------------------------------------------------------
SUBROUTINE:         MHEX44
                    16 X 16 BIT INTEGER MULTIPLIER
NUMBER OF BYTES:    38
EXECUTION TIME:     488.4 - 674.0 MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L,D',E',H',L'
MEMORY USED:        -

ON ENTRANCE:        MULTIPLIER IN HL
                    MULTIPLICAND IN DE
ON EXIT:            RESULT IN HL,DE


-------------------------------------------------------
SUBROUTINE:         DHEX42
                    16/8 BIT INTEGER DIVISION
NUMBER OF BYTES:    29
EXECUTION TIME:     666.0 - 734.0 MICROSEKUNDEN

REGISTERS USED:     A,B,C,D,E,H,L,B'
MEMORY USED:        -

ON ENTRANCE:        DIVIDEND IN HL
                    DIVISOR IN A
ON EXIT:            RESULT IN DE
                    REST OF DIVISION IN HL


-------------------------------------------------------
SUBROUTINE:         DHEX84
                    32/16 BIT INTEGER DIVISION
NUMBER OF BYTES:    29
EXECUTION TIME:     1 469.2 - 1 601.2 MICROSEKUNDEN

REGISTERS USED:     B,C,D,E,H,L,B',C',H',L'
MEMORY USED:        -

ON ENTRANCE:        DIVIDEND IN HL,DE
                    DIVISOR IN BC
ON EXIT:            RESULT IN HL,DE
```

Z80-FPA Gleitkomma-Programm-Paket

Das Z80-FPA (= „**F**loating **P**oint **A**rithmetik Package") ist ein Softwarepaket zur Lösung arithmetischer Aufgaben in Gleitkommaformat.

Es umfaßt die vier Grundrechnungsarten, Quadratwurzel, Vergleich und Umwandlungsroutinen von binärem Format in BCD und von BCD-Format in binäre Zahlendarstellung. Die Zahlendarstellung erfolgt in einer 3Byte langen Mantisse mit einem 1Byte langem Exponenten.

Der arithmetische Teil des Programmpakets erfordert weniger als 1kByte Speicher; die einzelnen Operationen werden in typisch 2—3 msec ausgeführt (Z80-CPU) bzw. 1,2 . . . 1,8 msec (Z80A-CPU).

Der Datenkonvertierungsteil des Programmpakets umfaßt weniger als ½kByte. Konversionszeiten liegen bei ca. 10 msec (Z80-CPU) bzw. 6,6 msec (Z80A-CPU). Das gesamte Softwarepaket macht in umfangreicher Weise von den besonderen Eigenschaften der Z80-CPU bzw. Z80A-CPU Gebrauch, insbesondere der beiden Registersätze und des IX-Indexregisters. Unter gewissen Voraussetzungen wird auch das IY-Indexregister benützt.

Zur Ausführung des Programms ist ein RAM-Speicherbereich nötig, der über den üblichen Prozessor-Stackbereich einen sogenannten arithmetischen Stack von mindestens 15 Byte's erfordert.

```
0. PREFACE
   =======


  THE Z 80 - FPA IS A FLOATING POINT ARTITHMETIC AND
CONVERSION PROGRAM PACKAGE FOR THE Z 80 MICRO PROCESSOR.
WHICH USES THE UNIQUE POWERFUL Z 80 INSTRUCTIONS EXTENSIVELY.
IN GENERAL BOTH CPU INTERNAL REGISTER SETS AND THE IX REGISTER
ARE USED. WITH SPECIAL ENTRIES (SEE PAGE 7. BOTTOM)
THE IY REGISTER IS USED. TOO.
 THE Z 80 - FPA / VERSION 2.0 HAS TO BE EDITED AND ASSEMBLED
WITH OS 2.0, REVISION B.
 THE FULL ARITHMETIC PACKAGE CONTAINS TWO SECTIONS.
SECTION 1 (ARTH) ARE THE FOUR BASIC ARITHMETIC OPERATIONS, COMPARE
AND SQUARE ROOT, INCLUDING ALL BINARY FORMAT CONVERSION
AND LOAD ROUTINES. THIS SECTION REQUIRES
LESS THAN 1 KBYTE OF PROM MEMORY. TYPICAL EXECUTION TIMES
ARE 3 MILLISECONDS PER OPERATION.
SECTION 2 (FPCONV) ARE THE CONVERSIONS TO AND FROM
BCD FLOATING POINT PACKED FORMAT.
WHICH USE SUBROUTINES FROM SECTION 1. THIS SECTION
REQUIRES LESS THAN 512 BYTES OF PROM IN ADDITION
TO SECTION 1. CONVERSION TIMES ARE ABOUT 10 MILLISECONDS.
  RAM IS REQUIRED FOR THE USUAL PROCESSOR STACK AND FOR
THE USER'S ARITHMETIC STACK (SEE 4.).
THE PROCESSOR STACK FILLES LESS THAN 16 BYTES. THE ARITHMETIC
STACK MUST BE AT LEAST 15 BYTES DEEP.


1. ARITHMETIC OPERATIONS
   =====================

     ENTRY     SUBROUTINES  COMMENT
     LABEL     IN ORDER
               OF USE
     ---------------------------------------

1.1  ADD       LDOPS        LOAD OPERANDS
               BIGAC        ADD IN REGISTERS
               ACXIT        STORE & CHECK RESULT

1.2  SUB       LDOPS        LOAD OPERANDS
               NEG          NEGATE 2ND OPERAND
               BIGAC        ADD IN REGISTERS
               ACXIT

1.3  MPY       LDOPS        LOAD OPERANDS
               MPENT        MULTIPLY IN REGISTERS
               ACXIT

1.4  DIV       LDOPS
               DVENT        DIVIDE IN REGISTERS
               ACXIT


     ENTRY     SUBROUTINES  COMMENT
     LABEL     IN ORDER
               OF USE
     ---------------------------------------

1.5  CMP       LDOP
               INCSP        INCREM.STACK POINTER
               CPENT        COMPARE (WITH SIGN)
                            STACK.POINT.NOT ALTERED

1.6  ACP       LDOPS        LOAD OPERANDS
               INCSP        INCREM.STACK POINTER
               MAGCP        COMPARE ABSOLUTE
                            STACK POINTER NOT ALTERED.
```

```
FLAG AND REGISTER SETTING
BY THE CMP AND ACP SUBROUTINES:

CONDITION                ZERO   CARRY   DE-
                         FLAG:  FLAG:   REGISTERS
-------------------------------------------------

OPERAND 2 > OPERAND 1     0      1       - 1 = FFFF HEX.

OPERAND 2 = OPERAND 1     1      0         0

OPERAND 2 < OPERAND 1     0      0       + 1 = 0001 HEX.


1.7 THE SQUARE ROOT SUBROUTINE (LABEL: SQRT) CONVERTS THE
    LAST OPERAND IN THE ARITHMETIC STACK INTO IT'S SQUARE ROOT,
    WITHOUT DECREMENTING THE ARITHMETIC STACKPOINTER.
    THE EXIT SUBROUTINE IS USED (SEE 1.8).


1.8 ACXIT SUBROUTINE
    ----------------

 THE ACXIT SUBROUTINE STORES THE RESULT OF A
COMPUTATION FROM THE CPU REGISTERS ONTO THE ARITHMETIC STACK.
ACXIT CONTINUES WITH THE EXIT SUBROUTINE,
WHICH CHECKS FOR DIFFERENT TYPES OF RESULTS
AND SETS ACCUMULATOR AND FLAGS. THERE ARE
FOUR POSSIBLE CASES:


1.8.1 RESULT EQUAL ZERO
      ACC. = 80 H
      ZERO = 1,  CARRY = 0
      PARITY = ODD = 0.  SIGN = 0

1.8.2 OVERFLOW HAS OCCURRED
      RESULT IS SET TO GREATEST NUMBER
      ACCUMULATOR:
      BIT 0 : INVERTED MANTISSA SIGN
      BIT 1 = 1 : OVER-UNDERFLOW BIT
      BIT 6 : EXPONENT SIGN
      BIT 7 : MANTISSA SIGN
      ZERO = 0.   CARRY = 1
      SIGN = MANTISSA SIGN
      PARITY = EVEN = 1

1.8.3 UNDERFLOW HAS OCCURRED
      RESULT IS SET TO ZERO
      SEE 1.8.2
      PARITY = ODD = 0 = UNDERFLOW

1.8.4 RESULT NOT ZERO AND NO OVER/UNDERFLOW
      ACCUMULATOR
      BIT 0 = 1
      BIT 6 = EXPONENT SIGN
      BIT 7 = MANTISSA SIGN
      ZERO = 0.   CARRY = 0
      SIGN = MANTISSA SIGN
      PARITY = ODD = 0 IF SIGNS EQUAL
      PARITY = EVEN = 1 IF SIGNS NOT EQUAL


2. DATA FORMATS
   ============

  THE ARITHMETIC PACKAGE EMPLOYS SEVERAL BINARY AND A BCD
DATA FORMAT. FOR FORMAT CONVERSIONS SEE CHAPTER 3.
BYTES OF MULTIPLE PRECISION QUANTITIES ARE DESIGNATED
NO. 0 THROUGH NO. N  FROM LS TO MS BYTE
(LOW TO HIGH ADDRESS).

2.1 FIXED POINT (INTEGER)
    ---------------------

 THESE NUMBERS ARE 16 BIT TWO'S COMPLEMENT BINARY.
THE "FLO3" SUBROUTINE ACCEPTS AN ADDITIONAL BYTE.
I.E. 24 BIT TWO'S COMPLEMENT NUMBERS.


2.2 BINARY FLOATING POINT UNPACKED
    ------------------------------

THESE NUMBERS ARE CONTAINED IN 5 BYTES, USING 33 DEC. BITS.
BYTES 0 THROUGH 2:
   NORMALIZED BINARY MANTISSA, I.E. BITS
   (FROM BYTE 2, BIT 7 = MS BIT) REPRESENT
   1/2, 1/4, 1/8, ...
   MS BIT MUST BE A "1" (NORMALIZATION),
   EXCEPT FOR A VALUE OF ZERO FOR THE FLOATING POINT NUMBER.
BYTE 3:
   BINARY EXPONENT (TWO'S COMPLEMENT NOTATION),
   USING BASE 2, I.E. EXPONENT = 3 MEANS 2 ** 3 = 8.
   BIT 7 IS EXPONENT SIGN.
   EVERY NUMBER WITH EXPONENT = 80 HEX.
   IS EQUAL ZERO, DISREGARDING THE MANTISSA.
BYTE 4:
   MANTISSA SIGN BYTE, BIT 7 IS THE ONLY SIGNIFICANT.
   BIT 7 = 0 MEANS POISITVE, = 1 MEANS NEGATIVE.

2.3 BINARY FLOATING POINT PACKED
    ----------------------------
THESE NUMBERS ARE CONTAINED IN FOUR BYTES.
BECAUSE OF THE NORMALIZATION, THE MS MANTISSA BIT,
WHICH ALWAYS MUST BE A "1", IS EXMITTED (HIDDEN BIT).
BYTES 0 AND 1:
   LOWER 16 BITS OF NORMALIZED MANTISSA (SEE 2.2)
BYTE 2, BITS 0 THROUGH 6:
   HIGH 7 BITS OF MANTISSA, WITHOUT THE 8TH,
   WHICH IS HIDDEN, AND MUST BE RE-INSERTED
   TO OBTAIN THE FULL UNPACKED MANTISSA.
BYTE 2, BIT 78
   LS BIT OF EXPONENT (SEE 2.2)
BYTE 3, BITS 0 THROUGH 6:
   HIGH 7 BITS OF EXPONENT (TWO'S COMPLEMENT)
   BIT 6 IS EXPONENT SIGN.
BYTE 3, BIT 7:
   MANTISSA SIGN, BIT 7 = 1 MEANS NEGATIVE.
```

```
2.4 BCD FLOATING POINT PACKED
    ------------------------
THESE NUMBERS ARE CONTAINED IN 5 BYTES, WITH
2 BCD DIGITS EACH.
SIGNS ARE NEGATIVE IF BITS = "1"
BYTE 0, BITS 0 THROUGH 5:
   BCD EXPONENT (2 DIGITS) IN THE RANGE 0 TO 39.
   BASE IS 10 DEC.
BYTE 0, BIT 6:
   EXPONENT SIGN
BYTE 0, BIT 7:
   MANTISSA SIGN
BYTES 1 THROUGH 4:
   NORMALIZED BCD MANTISSA (8 DIGITS)
   MS DIGIT (BYTE 4, BITS 4 .. 7)
   MUST NOT BE ZERO.
```

```
3. DATA MOVEMENT AND CONVERSION
   ============================

 THERE ARE SEVERAL SUBROUTINES TN CONVERT DATA FROM
ONE FORMAT TO ANOTHER (SEE 1.) AND TO MOVE THEM
TO OR FROM THE ARITHMETIC STACK.
ABBREVIATIONS:
FPBI = FLOATING POINT BINARY,   FPBC = FLOATING POINT BCD
FX = FIXED POINT,   PCK = PACKED,   UNP = UNPACKED
(DE) = DE REGISTERS CONTAIN THE DATA ADDRESS
DE = DATA ARE CONTAINED IN DE REGISTERS

SUB-     DATA      CONVERTS DATA          DATA
ROUTINE  SOURCE    FROM       TO          DESTINATION
-------------------------------------------------------

LOAD     (DE)      FPBI PCK   FPBI UNP    ARTH.STACK

STORE    STACK     FPBI UNP   FPBI PCK    (DE)

FIX      STACK     FPBI UNP   FX          DE

FLO      DE        FX         FPBI UNP    ARTH.STACK

FLO3     DEH       FX         FPBI UNP    ARTH.STACK

OPMOV    STACK     FPBI UNP   FPBI UNP    "STACK - 5"

LDOPS    STACK(IX) FPBI UNP               CPU REGISTERS

LDOPY    STACK     FPBI UNP               CPU REGISTERS
          (IX, IY)

POWER    (DE)      2 BCD DIG. FPBI UNP    ARTH.STACK

FBNBC    STACK     FPBI UNP   FPBC PCK    (DE)

FBCBN    (DE)      FPBC PCK   FPBI UNP    ARTH.STACK
```

```
4. ARITHMETIC STACK
   ================

 THE FLOATING POINT ARITHMETIC PACKAGE FEATURES
THE ARITHMETIC STACK PRINCIPLE
EQUIVALENT TO RPN (REVERSE POLISH NOTATION),
WHICH AVOIDS THE USE OF ANY BRACKETS.
 ALL OPERANDS, WHICH ARE USED IN A CALCULATION,
MUST BE ENTERED INTO THE ARITHMETIC STACK PRIOR TO THE FIRST
MATHEMATICAL OPERATION.
THE ARITHMETIC STACK IS A CONTIGUOUS MEMORY AREA,
PLACED ANYWHERE IN THE READ/WRITE MEMORY BY THE USER.
THE PROGRAM OPERATES ON THIS STACK, USING THE
ARITHMETIC STACK POINTER, WHICH IS DEFINED AS THE ADDRESS
CONTAINED IN THE CPU'S IX REGISTER.
 THE STACK POINTER IS MOVED IN INCREMENTS OF 5 UP AND DOWN,
WHICH IS THE LENGTH OF UNPACKED BINARY FLOATING POINT NUMBERS.
THE POINTER (IX REGISTER) CONTAINS ALWAYS
THE ADDRESS OF THE NEXT FREE STACK LOCATION,
I.E. THE ADDRESS OF THE BYTE RIGHT ABOVE
THE MANTISSA SIGN OF THE LAST OPERAND.
 EVERY OPERATION, EXCEPT COMPARE AND SQUARE ROOT,
MOVES THE STACK POINTER BY ONE OPERAND LOCATION
UP OR DOWN (FOR ABREVIATIONS SEE CHAPTERS 1 AND 3) :

UP:
   LOAD, FLO, FLO3, FBCBN, POWER, INCSP

DOWN:
   STORE, FIX, MPY, DIV, SUB, ADD, LDOPS, FBNBC, DECSP

 EVERY OPERATION SUBROUTINE FETCHES ITS OPERANDS FROM THE
CURRENT STACK POSITIONS (BYTES NO.0 ARE IX - 5 AND IX - 0AH)
INTO THE CPU'S REGISTERS BY THE LDOPS SUBROUTINE,
OR STORES AN OPERAND INTO THE NEXT STACK POSITION
BY THE ACXIT SUBROUTINE.
WHEN TWO OPERANDS ARE FETCHED, THE STACKPOINTER
IS DECREMENTED BY ONE PLACE, AND THE RESULT OF
A COMPUTATION IS STORED IN PLACE OF THE
LOWER (FIRST) OPERAND.
 A SECOND SET OF OPERATION ENTRIES (MPYY, DIVY, ADDY, ...)
EMPLOYS A SPECIAL ARITHMETIC STACK POINTER (IY REGISTER)
FOR THE SECOND OPERAND. WHEN USING THESE ENTRIES
NEITHER IX NOR IY ARE ALTERED.
```

Z80-EAP Erweiterungs-Programmpaket für Arithmetische Operationen

```
A. ALLGEMEINES
   ===========


 DAS Z 80 - EAP IST EIN ERWEITERTES ARITHMETISCHES
PAKET FUER DEN Z 80 MIKROPROZESSOR.
ZUR BERECHNUNG DER FUNKTIONEN WERDEN SUBROUTINEN AUS SEC-
TION 1 DES FLOATING POINT PAKETS VERWENDET; DABEI WERDEN
BEIDE CPU - INTERNEN REGISTERSAETZE UND DIE IX, IY RE-
GISTER BENUTZT

 DER FUER DEN ARITHMETISCHEN STACK ZUR VERFUEGUNG STEHEN-
DE SPEICHERPLATZ SOLLTE MINDESTENS 20 BYTES UMFASSEN.
FUER DEN PROZESSOR STACK WERDEN NICHT MEHR ALS 30 BYTES
BENOETIGT. DES WEITEREN SIND FUER DIE ABSPEICHERUNG VON
ZWISCHENERGEBNISSEN BZW. ZUR BERECHNUNG DER STANDARDAB-
WEICHUNG UND DES MITTELWERTS 4 BZW. 12 WEITERE BYTES AR-
BEITSSPEICHERBEREICH ERFORDERLICH. FUER DAS PAKET WER-
DEN WENIGER ALS 2 1/4 KBYTE PROM BENOETIGT.
 DIE ARGUMENTE DER FUNKTIONEN WERDEN DEN FUNKTIONSROU-
TINEN AUSSER SDEV UND MEAN IM ARITHMETIC STACK UEBERGE-
BEN. DIE ADRESSE DER BETREFFENDEN SPEICHERPLAETZE ENT-
HAELT DAS IX REGISTER (ENTSPRECHEND DER KONVENTION IM
FLOATING POINT PAKET). DIE ERRECHNETEN FUNKTIONSWERTE
ERSETZEN DIE ARGUMENTE IM ARITHMETIC STACK.
BEI AUFRUF DER FUNKTION "Y HOCH X" STEHT IM IX REGISTER
DIE ADRESSE VON Y; DIE ADRESSE VON X IST IX - 5. NACH BE-
RECHNUNG DES ERGEBNISSES WIRD ES AUF DEN SPEICHERPLATZ
DES X WERTES GELADEN.
BEI DEN FUNKTIONEN SDEV UND MEAN SIND DIE EINGANGSGROESSEN
DIE INHALTE DER SPEICHERPLAETZE NUMOFX, SUMOFX UND SQSOFX.
DAS ERGEBNIS ERSCHEINT IM ARITHMETIC STACK.
 DIE TYPISCHEN RECHENZEITEN UND -GENAUIGKEITEN WERDEN IN
C,D ANGEGEBEN.


B. DIE FUNKTIONEN
   ==============


1. TRIGONOMETRISCHE FUNKTIONEN

EINSPRUNGSMARKE   BEMERKUNGEN          ARTH.-STACK-POINTER
                                       IX
-------------------------------------------------------------

1.1.1 TAN         TAN(X) FUER          NICHT VERAENDERT
                  IXI =< PI/2

1.1.2 TANG        TAN(X) FUER          NICHT VERAENDERT
                  IXI < 16384*PI

1.2   SIN         SIN(X) FUER          NICHT VERAENDERT
                  IXI < 16384*PI

1.3   COS         COS(X) FUER          NICHT VERAENDERT
                  IXI < 16384*PI

2. INVERSE TRIGONOMETRISCHE FUNKTIONEN

EINSPRUNGMARKE    BEMERKUNGEN          ARTH.-STACK-POINTER
                                       IX
-------------------------------------------------------------

2.1  ARCTAN       ARCTAN(X)            NICHT VERAENDERT

2.2  ARCSIN       ARCSIN(X)            NICHT VERAENDERT

2.3  ARCCOS       ARCCOS(X)            NICHT VERAENDERT

3. HYPERBOLISCHE FUNKTIONEN

EINSPRUNGMARKE    BEMERKUNGEN          ARTH.-STACK-POINTER
                                       IX
-------------------------------------------------------------

3.1  TANH         TANH(X)              NICHT VERAENDERT

3.2  ARTANH       ARTANH(X)            NICHT VERAENDERT

4. LOGARITHMISCHE FUNKTIONEN

EINSPRUNGMARKE    BEMERKUNGEN          ARTH.-STACK-POINTER
                                       IX
-------------------------------------------------------------

4.1  LOGN         LN(X)                NICHT VERAENDERT

4.2  EXP          EXP(X)               NICHT VERAENDERT

4.3  YPOWRX       Y HOCH X             UM 5 ERNIEDRIGT
```

5. STATISTISCHE FUNKTIONEN

EINSPRUNGMARKE	BEMERKUNGEN	ARTH.-STACK-POINTER IX
5.1 MEAN	ARITHMETISCHER MITTELWERT	UM 5 ERHOEHT
5.2 SDEV	STANDARDAB-WEICHUNG	UM 5 ERHOEHT

C. RECHENGENAUIGKEIT

DIE GENAUIGKEIT DER BERECHNETEN FUNKTIONSWERTE HAENGT TEILWEISE VON DER GROESSENORDNUG DES ARGUMENTS AB:

FKT.	BEREICH	MAX. FEHLER, ABSOLUT (= A) ODER RELATIV % (= R)	
TANG	IXI =< 1.47	5 * 10E -5	R
	IXI =< 1.57	1 * 10E -2	R
	IXI =< 1.57079	1 * 10E 0	R
	IXI =< PI/2	1 * 10E +1	R
SIN	IXI =< PI/2	2 * 10E -5	R
COS	IXI =< PI	2 * 10E -7	A

DIE FUNKTIONEN TANG, SIN, COS WERDEN FUER ARGUMENTWERTE BIS 16384*PI AUSGERECHNET. ERREICHT ODER UEBERSTEIGT DAS ARGUMENT DIESE GRENZE, WIRD DAS ERGEBNIS MIT 0 AUSGEGEBEN. FUER WERTE VON X, FUER DIE GILT: IXI > PI/2 BZW. IXI > PI (BEIM COS), NIMMT DIE GENAUIGKEIT IM ALLGEMEINEN JE DEKADE UM EINE ZEHNERPOTENZ AB.

FKT.	BEREICH	MAX. FEHLER, ABSOLUT (= A) ODER RELATIV % (= R)	
ARCTAN	.1=<IXI=<.13	5 * 10E -4	R
	AUSSERHALB	5 * 10E -5	R
ARCSIN	.999999 =< IXI =< 1	1 * 10E -2	R
	.999 < IXI < .999999	5 * 10E -4	R
	IXI =< .999	2 * 10E -5	R
ARCCOS	.999999 =< IXI =< 1	3 * 10E -5	A
	.9 =< IXI < .999999	5 * 10E -6	A
	IXI < .9	3 * 10E -7	A
TANH	GANZER BEREICH	2 * 10E -5	R
ARTANH	0 =< IXI =< .99	5 * 10E -5	R
	.99 < IXI =< .999	1 * 10E -4	R
	.999 < IXI =< .9999	2 * 10E -3	R
	.9999 < IXI =< .99999	1 * 10E -2	R
	.99999 < IXI =< .999999	2 * 10E -1	R
	.999999 < IXI < 1	2 * 10E 0	R
	1 =< IXI	NICHT ERLAUBT	

FKT.	BEREICH	MAX. FEHLER, ABSOLUT (= A) ODER RELATIV % (= R)	
LOGN	0 < X =< 1+/-.1	5 * 10E -5	R
	1+/-.1 < X =< 1+/-.01	1 * 10E -4	R
	1+/-.01 < X =< 1+/-.001	5 * 10E -3	R
	1+/-.001 < X =< 1+/-.0001	1 * 10E -2	R
	1+/-.0001 < X =< 1+/-.00001	1 * 10E -1	R
	1+/-.00001 < X =< 1+/-.000001	5 * 10E 0	R
	1+/-.000001 < X =< 1	2 * 10E +1	R
	1.1 =< X	5 * 10E -5	R
EXP	88.029690 < IXI =< 50	1 * 10E -3	R
	50 < IXI =< 5	1 * 10E -4	R
	5 < IXI =< 0	5 * 10E -5	R

YPOWRX DIE GENAUIGKEIT BETRAEGT IN WEITEN BEREICHEN 1 * 10E -4 R. HAT Y ANNAEHERND DEN WERT 1, SO SOLLTE X NICHT WESENTLICH GROESSER ALS 1000 SEIN, UM INNERHALB DIESER GENAUIGKEITSGRENZE ZU BLEIBEN.

D. RECHENZEITEN

FKT.	TYP.	MAX.	(MILLISEKUNDEN)
TAN	7	11.5	GEMESSEN BEI
TANG	9	15	PROZESSORTAKT-
SIN	18	23	FREQUENZ VON
COS	18	23	4.0 MHZ
ARCTAN	6	7.5	
ARCSIN	10	12.5	
ARCCOS	10	12.5	
TANH	12.5	15	
ARTANH	6	7.5	
LOGN	10	12.5	
EXP	15	17.5	
YPOWRX	30	35	
MEAN	5	6	
SDEV	10	11	

E. VERBOTENE ARGUMENTWERTE

FKT.	VERBOTENER BEREICH
LOGN	X =< 0
ARCSIN	IXI > 1
ARCCOS	IXI > 1
ARTANH	IXI >= 1
MEAN	NUMOFX = 0
SDEV	NUMOFX = 0 ODER 1

LIEGEN DIE ARGUMENTE DER OBIGEN FUNKTIONEN IM VERBOTENEN BEREICH, IST DAS ERGEBNIS:
0 MIT NEGATIVEM VORZEICHEN, CARRY-FLAG GESETZT

13. Z80-Anwendersoftware

Mikro-Computer-Kursprogramm

Die KONTRON ELEKTRONIK GmbH setzt die seit Dezember 1976 laufenden Mikrocomputer-Schulungen in deutscher Sprache fort.

Ziel der Schulung ist, Elektro-Ingenieure in die Technik der Mikrocomputeranwendung einzuführen.

Folgende Kurse werden derzeit angeboten:

I. **Grundkurs**
1-tägiger Einführungskurs in die Grundlagen der Mikrocomputertechnik

II. **Z80-Intensiv-Kurs**
2-tägige Intensivschulung im Konzept ZILOG-Z80

III. **Z80-Systemkurs**
2-tägige Schulung an den ZILOG-Systemen MCZ und ZDS

IV. **PLZ-Kurs**
2-tägiger Einführungskurs in PLZ

V. **Z8000-Intensivkurs**

Grundkurs (Kurs I)

Lernziel des 1-tägigen Kurses:

a) Vorbereitung auf Kurs II für Ingenieure der Elektrotechnik und verwandter Gebiete, die keine oder geringe Erfahrung auf dem Mikrocomputersektor haben.
b) Gewährung eines grundlegenden Einblicks in den Themenkreis Mikrocomputer für Manager.
c) Einführung solcher Interessenten, die sich auf diese Weise vorab informieren möchten.

Folgende Themenkreise werden behandelt:

- Übersicht über den grundsätzlichen Aufbau eines uC-Systems
- Der Befehlsaufbau eines uP
- Programmabarbeitung im uC-System
- Vom Problem zum Programm
- Gebrauch des Stapelspeichers (Stack)
- Unterprogrammtechnik
- Programmaufbau mit Interrupts
- Betriebssystem und Anwenderprogramm
- Massenspeicher (Floppy Disks)

Z80-Intensivkurs (Kurs II)

Ziel von Kurs II (2 Tage) ist die spezifische Ausbildung von Anwendern **mit** Mikrocomputererfahrung bzw. Besuchern des Kurses I in der Anwendung des Systems Z80 (Hard- und Software).

Folgende Themenkreise werden behandelt:

- Das Z80-Konzept
- Überblick über ein Z80-uC-System
- Mikroprozessor Z80-CPU
- Periphere Bausteine des Z80-Systems
- Schaltungstechnik von Z80-Systemen
- Interrupt-Technik im Z80-System
- Einführung in den Z80-Befehlssatz
- Erläuterung vorgegebener Programmbeispiele

Zur Erreichung dises Lernzieles ist die Teilnehmerzahl von Kurs II auf max. 20 begrenzt.

Ein Teil des Kurses wird durch praktische Übungen eingenommen. Der Zeitpunkt der Veranstaltungen wurde so gewählt, daß Kurs II unmittelbar nach Kurs I besucht werden kann, wobei zwischen Kurs I und Kurs II ein Wochenende zum evtl. Überdenken der einführenden Information von Kurs I liegt.

Z80-Systemkurs (Kurs III)

Ziel von Kurs III (2 Tage) ist die Ausbildung von Teilnehmern in der Anwendung der Systeme MCZ bzw. ZDS. Neben der eingehenden Erläuterung des Betriebssystems RIO erfolgt eine umfangreiche Unterweisung in der Handhabung der Entwicklungssysteme (ZDS) und Minicomputer (MCZ).

Folgende Themenkreise werden behandelt:

- MCZ-Mini- bzw. Prozeßrechner auf Mikrocomputerbasis
- uC-Entwicklung mit ZDS-Entwicklungssystemen
- Z80-RIO, transparentes Betriebssystem für MCZ und ZDS
- Einführung in die Kommandosprache des Betriebssystems RIO
- Einsatz von höheren Sprachen in uC-Systemen

Zur Erreichung dieses Lernzieles ist die Teilnehmerzahl von Kurs III auf max. 20 begrenzt. Ein Teil des Kurses wird durch praktische Übungen in kleinen Gruppen an mehreren Systemen eingenommen.

PLZ-Kurs (Kurs IV)

Ziel des 2-tägigen PLZ-Kurses (Kurs IV) ist die Einführung in die höhere Programmiersprache PLZ.

Wie in Kurs II wird auch hier ein großer Teil der Zeit von praktischen Übungen eingenommen.

Vorraussetzung für den Besuch von Kurs IV ist Besuch von Kurs II und III oder äquivalente Kenntnisse.

Zur Erreichung dieses Lernzieles ist die Teilnehmerzahl auf max. 20 begrenzt.

Z8000-Intensivkurs (Kurs V)

Ziel von Kurs V (2 Tage) ist die spezifische Ausbildung von Anwendern **mit** Mikrocomputererfahrung bzw. Besuchern des Kurses I in der Anwendung des Systems Z8000 (Hard- und Software).

Folgende Themenkreise werden behandelt:

- Das Z8000-Konzept
- Überblick über ein Z8000-uC-System
- Mikroprozessor Z8000-CPU
- Periphere Bausteine des Z8000-Systems
- Schaltungstechnik von Z8000-Systemen
- Interrupt-Technik im Z8000-System
- Einführung in den Z8000-Befehlssatz
- Erläuterung vorgegebener Programmbeispiele

Zur Erreichung dieses Lernzieles ist die Teilnehmerzahl von Kurs V auf max. 20 begrenzt.

Ein Teil des Kurses wird durch praktische Übungen eingenommen. Der Zeitpunkt der Veranstaltungen wurde so gewählt, daß Kurs V unmittelbar nach Kurs I besucht werden kann, wobei zwischen Kurs I und Kurs V ein Wochenende zum evtl. Überdenken der einführenden Information von Kurs I liegt.

Unterlagen zum einführenden Selbststudium

Für Führungskräfte und Entwicklungsingenieure mit Digital-Grundkenntnissen steht ein leicht verständliches Hilfsmittel zur Einarbeitung in die Mikrocomputertechnik zur Verfügung, das Fachbuch

„Mikrocomputer-Technik“ von
P. Blomeyer-Bartenstein

Es vermittelt allgemeine Grundlagen und bietet Schaltungs- und Programmierbeispiele für eines der modernsten und leistungsfähigsten Mikrocomputerkonzepte, das System Z80A von ZILOG und gibt praktische Tips für die Anwendung.

Der Band umfaßt ca. 230 Seiten und 80 Abbildungen.

Bei Belegung von Kurs I ist dieses Buch in den Kursunterlagen enthalten. Es wird den einzelnen Teilnehmern zur Vorarbeitungsmöglichkeit vor Kursbeginn zugesandt.

DEUTSCHLAND

KONTRON ELEKTRONIK GmbH
8057 Eching b. München
Breslauer Straße 2
Tel. (089) 3 19 01-313
Telex 05 22 122

ÖSTERREICH

Kontron Ges mbH & Co KG
A-1140 Wien
Ameisgasse 49
Telefon 94 56 46
Telex 11 699

SCHWEIZ

STOLZ AG
CH 8968 Mutschellen
Telefon 057 54655
Telex 54070

DEUTSCHLAND

ZILOG GmbH
8011 Vaterstetten
Zugspitzstr. 4
Telefon 0 81 06-4035
Telex 529 110

USA

ZILOG INC.
10460 Bubb Road,
Cupertino, Cal. 950 14
Telefon (408) 446-4666
Telex 910-338-7621