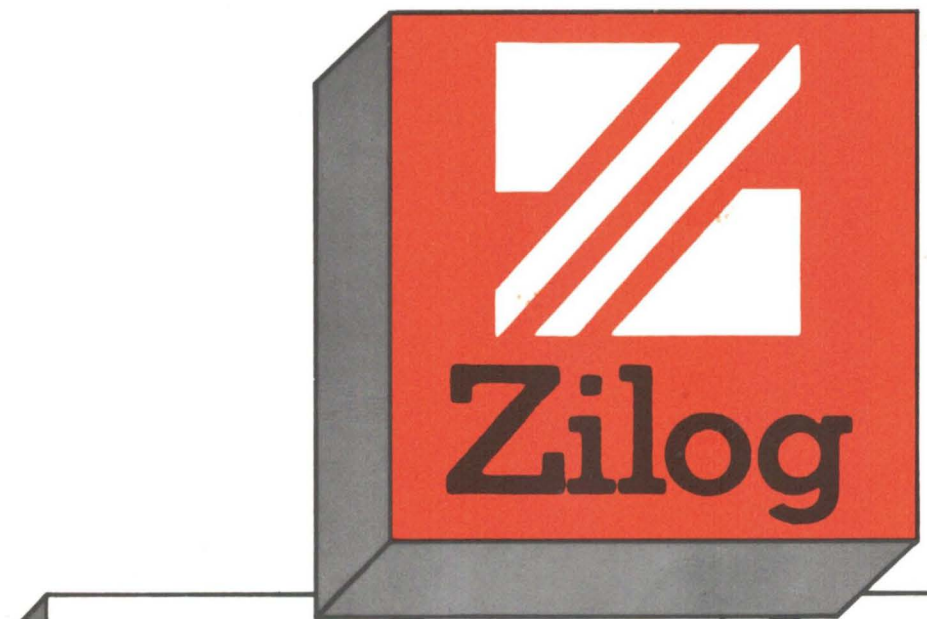
Mikro-computer 77/78

Applikations-sammlung



Ein neues Mikrocomputer-Konzept

Autoren:
Stephan Bauer
H. P. Blomeyer-Bartenstein
Bernhard Mindermann

Ein/Ausgabe und Interrupt in Z80-Mikro-computer-Systemen

Mikrocomputer-system mit eingebauter Tastatur und Billig-Anzeige

Universelle betriebsbereite Mikrocomputer-Anwendersysteme

Erweiterung des Adreßbereichs von Z80-Systemen über 64 kByte hinaus

ZILOG-APPLIKATIONSSCHRIFT

Dipl.-Ing. H.-P. Blomeyer Bartenstein

Ein neues Mikrocomputer-Konzept

Bausteine – Baugruppen – Entwicklungssystem

1 Systemkonzept

Bei der Schaffung des Mikroprozessors Z 80 der amerikanischen Firma Zilog setzte man sich die folgenden Entwicklungsziele:

– Schaffung eines marktgerechten Mikrocomputersystems, das die bisherigen Schwierigkeiten bei der Anwendung der „Mikros" soweit wie möglich ausschaltet, aber voll auf Vorhandenem (Investitionen, Know-how) aufbaut.

– Volle Software- und Applikations-Kompatibilität zum bisher weitest verbreiteten System 8080.

– Besondere Beachtung des Systemgedankens: Daher Koordination der Entwicklungs-Konzepte von Systemanalytikern, Softwarespezialisten, Halbleiterfachleuten und Gerätetechnikern. Ergebnis: Ein geschlossenes technisches Konzept.

– Zur Systemphilosophie gehört u.a. das Prinzip des minimalen Hardwareaufwands: Eine einzige Speisespannung und 1-Phasen-TTL-Takt, Unterbringung sämtlicher Nebenfunktionen (Systemsteuerung, Interruptsteuerung, Prioritätscodierer) in den Hauptbausteinen. Dadurch wird mit nur sechs verschiedenen Bausteinen (*Bild 1*) die gleiche Universalität erreicht wie bei anderen Systemen mit mehr als zwölf!

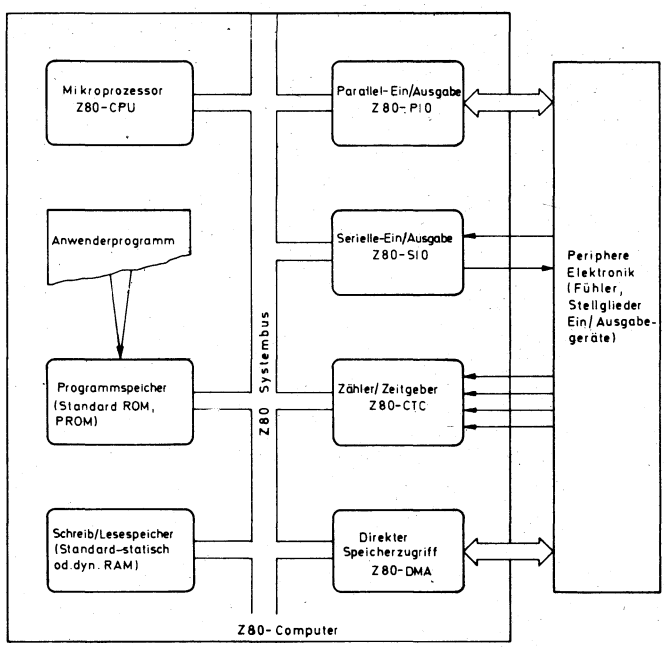


Bild 1. Das Mikrocomputersystem Z 80 erfüllt mit nur fünf Bausteinen alle normalerweise gestellten Anforderungen

– Bis zu fünfmal höhere Verarbeitungsgeschwindigkeiten als bei allen bisher verfügbaren MOS-Prozessoren durch kürzeren Befehlszyklus (standardmäßig 1,6 µs), fortgeschrittene Architektur und Befehlssatz, dadurch außerdem bis zu – 50 % Speicherplatzeinsparung.
– Erweiterung der Prozessorfähigkeiten in zwei Richtungen: Hin zu den *16-bit-Rechnern* (=Minicomputer und Prozeßrechner) durch Implementierung neuer 16-bit-Arithmetikbefehle mit zusätzlichen Adressierweisen (z.B. indiziert), eines kompletten unabhängigen zweiten Registriersatzes inklusive Akkumulator, blockweiser Datenbehandlung und wesentlich erweiterter Interrupt-Fähigkeiten.

In Richtung *Maschinensteuerung* (der bisherigen Domäne der 4-bit-„Mikros") durch Hinzufügen von Einzelbitoperationen (Bit testen, Bit setzen, Bit rücksetzen) und durch ein neues Hardwarekonzept, das Einzelbitbehandlung durch die „Intelligenz" der Parallel-Ein-/Ausgabe-Bausteine weiter vereinfacht.

– Direkte Anschlußmöglichkeit sämtlicher Standardspeicherbausteine, sogar dynamischer Speicher ohne jeglichen Hardwareaufwand (*Refreshcontroller*) oder Programmieraufwand (*Softwarefresh*) zur Einsparung von Speicherkosten.
– Lieferbarkeit sämtlicher Bausteine auch im militärischen Temperaturbereich.
– Mikrocomputer-Anfängern wird der Einstieg in die neue Technik durch den KONTRON-Z-80-Kit erleichtert, der einen „Kaltstart" mit minimaler Investition und geringer Einarbeitungszeit erlaubt.

2 Bausteine

2.1 Die Zentraleinheit (CPU)

Der gegenüber bisher verfügbaren Mikroprozessoren höhere Informations-Durchsatz (*throughput*) und die effizientere Programmspeicherausnutzung erlauben die Realisierung von Mikrocomputer-Systemen mit höheren Anforderungen als bei den bisher bekannten Systemen. Die CPU ist in N-Kanal-Silicon-Gate-Technik mit Ionenimplantation ausgeführt. *Bild 2* zeigt ihre Blockschaltung. In *Bild 3* ist die Registerorganisation mit insgesamt 208 bit dem Anwender zugänglicher Schreib-/Lesespeicher dargestellt. Hiervon sind zwei gleich aufgebaute Blöcke mit je sechs allgemeinen Registern, die jeweils wahlweise 8- oder 16-bit-Operationen erlauben. Hinzu kommen zwei Akkumulatoren und Status-Register (*Flag*).

Die Operationen der CPU laufen in einem der beiden Register/Akku-Blöcke 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 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.

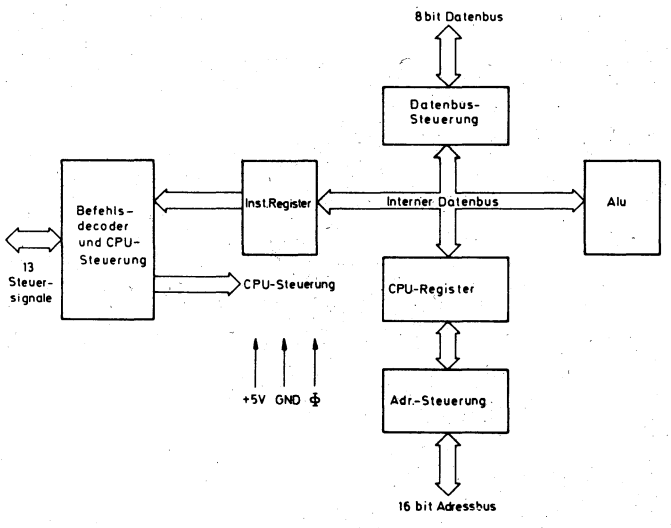
Zwei *16-bit-Indexregister* erlauben die Bearbeitung von Tabellen und relokativen (d. h. im Adreßraum des Arbeitsspeichers verschiebaren) Informationen. Ein eigenes *Refresh-Register* wurde für ein direktes, übersichtliches Arbeiten mit externen dynamischen Speichern ohne Software-Aufwand implementiert. Das *Unterbrechungsregister* (*I-Register*) schließlich liefert zur Realisierung einer besonders leistungsfähigen Art der Interrupt-Behandlung die höherwertigen acht Bits eines Zeigers, der über eine Tabelle den Sprung in Interrupt-Behandlungsprogramme ermöglicht; die niederwertigen acht Bits der Anfangsadressen werden in üblicher Weise von den anfordernden Schaltungen geliefert.

Zusammengefaßt ergeben sich die in *Tabelle 1* aufgelisteten Eigenschaften.

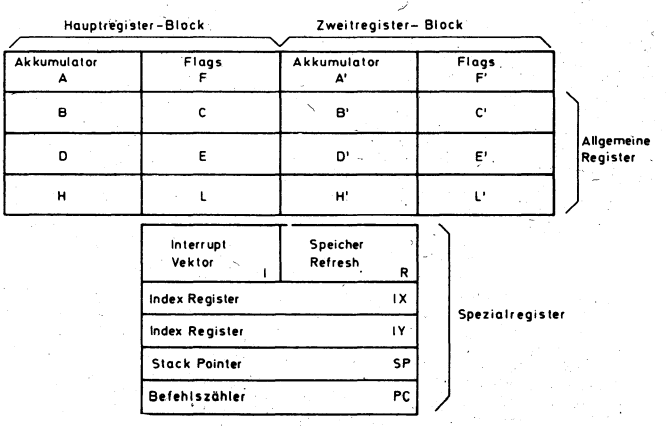
Als *Adressierweisen* kommen (auch Kombinationen) in Frage: Direkt (*immediate*); Erweitert direkt (*immediate extended*); Modifizierte Seite Null (*Modified Page Zero*); Relativ (*relative*); Erweitert (*extended*) und Indiziert (*indexed*). Ferner kann man über Register folgendermaßen adressieren: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

2.2 Der programmierbare parallele Ein-/Ausgabebaustein

Der Software-programmierbare Parallel-I/O (PIO) - Interfacebaustein Typ Z 80-PIO enthält zwei TTL-kompatible Ports, über die der Datenverkehr zwischen dem Mikroprozessor und der Umwelt (*Peripherie*) abgewickelt wird. *Bild 4* ist seine Blockschaltung, *Tabelle 2* gibt weitere Einzelheiten. In *Bild 5* wird die Gliederung eines Ports gezeigt; interessant ist (rechts unten) die Quittungs-Betriebslogik zur schnellen Anforderungsbearbeitung im Interrupt-Betrieb.



◄ Bild 2. Blockschaltung der Zentraleinheit (CPU)



▲ Bild 3. Die Register-Blöcke der CPU

2.3 Der serielle Ein-/Ausgabebaustein

Dieser unter der Bezeichnung Z 80-SIO laufende Baustein hat im Prinzip eine analoge Funktion wie der zuvor beschriebene PIO, wickelt jedoch den Datenverkehr zwischen Mikroprozessor und Außenwelt bit-seriell ab. Dadurch eignet er sich vor allem zum Anschluß von Floppy-Disks, Bildschirmgeräten, Fernschreibern und allgemein seriellen Daten-Übertragungskanälen.

Der Baustein arbeitet bidirektional (bedient also einen bit-seriellen Doppelkanal) mit 5- bis 8-bit-Zeichen, mit 1, $1^1/_2$ oder 2 Stop-Bits; CRC- und Paritätserzeugung erfolgt automatisch. Dadurch ist auch direkter Anschluß an IBM-Bi-Sync und SDL-Kanäle möglich.

2.4 Der Zähler/Zeitgeberbaustein

Dieser Baustein (Z 80-CTC) enthält vier Zähler/Zeitgebereinheiten und die zu jeder Einheit gehörigen Interrupt- und Prioritätsschaltungen.

Die Zeitgeber können zur Bearbeitung synchroner Real-Time-Aufgaben Interrupts in Zeitabständen von 8 µs bis 32 ms veranlassen. Die Zähler (je 8 bit) können zur Entlastung des Mikroprozessors ebenfalls bei Real-Time-Aufgaben benützt werden.

2.5 Der Baustein für direkten Speicherzugriff (DMA)

Dieser Baustein (Z 80 DMA) ist für ein-/ausgabeintensive Anwendungen und hohe Datenübertragungsraten gedacht; er gestattet externen „intelligenten" Peripheriegeräten, wie Floppy-Disks und Datenübertragungskanälen, den direkten Zugriff auf den Systemspeicher, wobei der Mikroprozessor für die Zeit des Zugriffs „kaltgestellt" wird.

Hierzu wurden in dem Baustein u.a. Blocklängenzähler, Speicheradressen-Zeiger und Kaskaden-Prioritätsschaltung integriert.
Dadurch lassen sich Systeme mit einer theoretisch unbegrenzten Anzahl von DMA-Kanälen aufbauen.

3 Entwicklungssystem

3.1 Bedeutung und Übersicht

Nach den bisherigen Erfahrungen mit Mikrocomputer-Systemen betragen die Software-Kosten 50...90 % der Entwicklungskosten des gesamten Systems unter der Voraussetzung, daß der Anwender vom Lieferanten hardware- und softwaremäßig in der üblichen Weise unterstützt wird; hardwaremäßig z.B. durch betriebsfertige Computerschaltungen, Schaltungsbeispiele und Applikationsberichte, softwaremäßig durch Bibliotheken von universell einsetzbaren Programmteilen, die es dem Anwender ersparen, häufig vorkommende Teilprobleme, wie Gleitkomma-Operationen, Sortierungen, Zeitschleifen usw. selbst lösen zu müssen. Nur den anwendungsspezifischen Rest des Programms muß der Anwender selbst entwickeln.

Da dies bei mittleren und größeren Systemen erhebliche Aufwendungen darstellt, ist einsichtig, wie stark die Leistungsfähigkeit von Programmierhilfen auf den Gesamt-Entwicklungsaufwand Einfluß nimmt. Um diesen zu senken, wurde daher, passend zum System Z80, ein eigenes Entwicklungssystem geschaffen.

Tabelle 1. Ausstattung und Befehle der Zentraleinheit (CPU)

17 interne Register
3 schnelle Interrupt-Behandlungsarten und ein zusätzlicher nichtmaskierbarer Interrupt
Direkter Anschluß von dynamischen oder statischen Standard-Speicherchips ohne zusätzlichen Bauteileaufwand; eingebaute dynamische Refresh-Hardware
Standard-Befehlsausführungsdauer: 1,6 µs
Stromversorgung über eine einzige 5-V-Versorgungsspannung
5-V-Einphasen-Takt
Alle Anschlüsse TTL-kompatibel
158 Befehle, darunter die 78
Instruktionen des Mikroprozessors 8080A (voll Software-kompatibel; darüber hinaus verfügt Z 80 über umfassende 16-, 8-, 4- und Einzel-Bit-Instruktionen und zusätzliche Adressierweisen (indizierte, relative und Bit-Adressierung).
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-Arithmetik- und Logik-Befehle (*8 bit arithmetic and logic*)
16-bit-Arithmetik- und Logik-Befehle (*16 bit arithmetic and logic*)
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

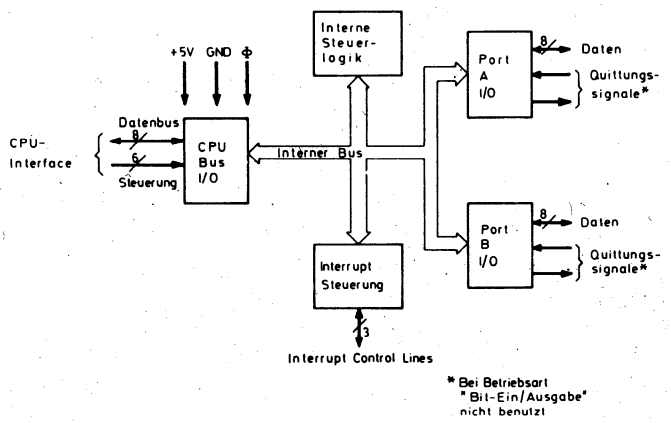


Bild 4. Blockschaltung des parallel arbeitenden Ein-/Ausgabebausteins PIO

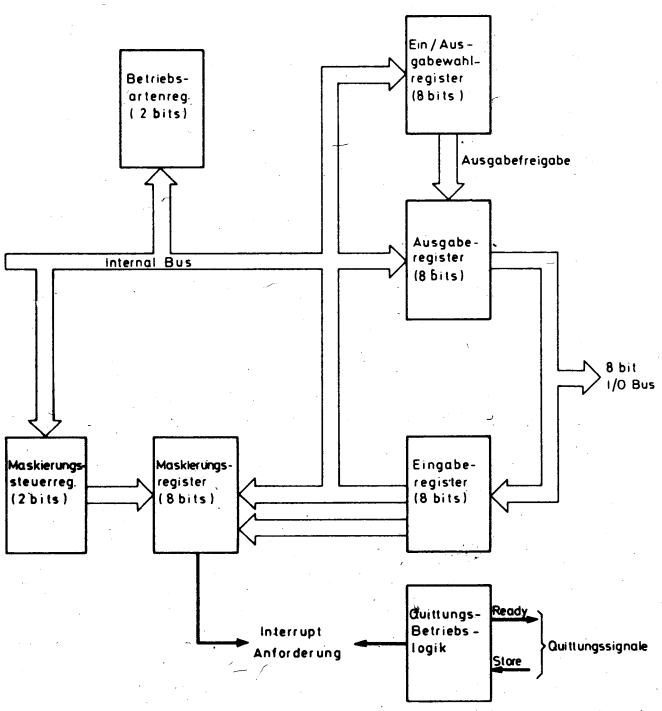


Bild 5. Blockschaltung eines Ports

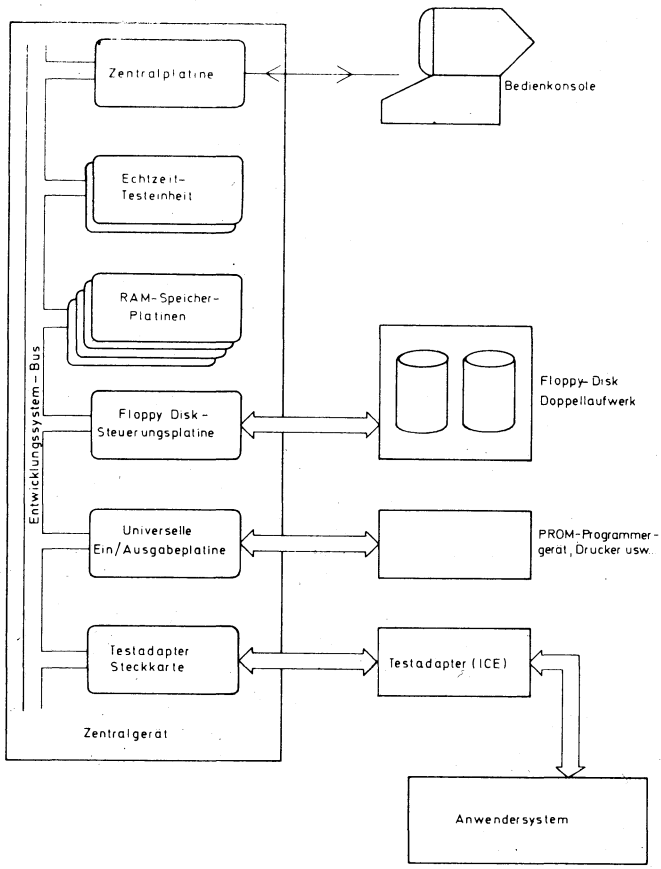


Bild 6. Bestandteile des Z 80-Entwicklungssystems

Das Z-80-Microcomputer-Entwicklungssystem ist so flexibel aufgebaut, daß es auch für alle zukünftigen Zilog-Mikroprozessoren und Mikrocomputer geeignet ist. Sein 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 Echtzeittest-Adapter (*In-Circuit-Emulator*) zur direkten Koppelung zwischen dem Entwicklungscomputer und dem zu testenden Anwendersystem, wodurch in kürzester Zeit sowohl Hardware- als auch Software-Tests durchgeführt werden können.

Ergänzt werden die Möglichkeiten des Echtzeittestadapters durch eine *Firmware-Echtzeit-Testeinheit*, mit der sich fragliche Anwendersystemzustände unter Echtzeitbedingungen in einen hierfür reservierten Speicher ablegen und anzeigen, oder aber als Rücksprungursache in den Monitor-Mode verwenden lassen. Dadurch wird die Anschaffung zusätzlicher Logik-Analysatoren mit Anzeige im Datenbereich unnötig.

Diese Möglichkeit, Vorgänge in Echtzeit ablaufen, aufzeichnen und nach Wunsch selektiv anzeigen zu lassen, garantiert hohe Zeitersparnis bei Hard- und Softwaretests und damit wesentlich verringerte Entwicklungskosten.

3.2 Technische Einzelheiten

Kernstück des Entwicklungssystems ist gemäß *Bild 6* der Mikroprozessor Z 80-CPU. Durch seine Multi-Task-fähige Architektur ist er besonders gut für das Mikrocomputer-Entwicklungssystem geeignet, das sowohl mit dem residenten Monitor (*Monitor-Mode*) als auch mit dem Anwendersystem (*Users-Mode*) operieren muß.

In der *Monitor-Betriebsart* arbeitet das Gerät als autonome Entwicklungshilfe, wobei Anwender-Programme in den Schreiblesespeicher des Systems eingegeben, editiert, übersetzt, korrigiert auf Diskette abgespeichert, von der Diskette geladen und zum Ablauf gebracht werden können.

All diese Funktionen werden mit einfach zu erlernenden Kommandos von der Bedienungskonsole aus angestoßen. Im *Users Mode* werden Speicher und Peripherie des Entwicklungssystems einer fertigen Anwenderschaltung verfügbar gemacht, wobei ein ins Entwicklungssystem-RAM eingelesenes Anwenderprogramm unter Echtzeitbedingungen ausgeführt wird. Dadurch ist in der Entwicklungsphase eine PROM-Programmierung überflüssig.

Tabelle 2. Ausstattung und Eigenschaften des PIO-Bausteins

Technologie: N-Kanal Silicon Gate (Depletion Load)	
Gehäuse: 40-Stift-DIP	
Stromversorgung: 5 V	
Takt: 5 V einphasig	
Der Baustein umfaßt:	– ein Interface zur CPU-Anschaltung
	– interne Steuerlogik
	– Logik für I/O-Port A
	– Logik für I/O-Port B
	– Interrupt-Steuerlogik
A und B sind zwei unabhängige bidirektionale 8-bit-Ports mit Einrichtungen für Quittungsbetrieb (*Handshaking*); 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	
Die I/O-Port-Logik selbst besteht nach Bild 5 aus 6 Registern mit Quittungsbetrieb-Steuerlogik und zwar:	
	ein 2-bit-Betriebsarten-Register
	ein 8-bit-Ausgabe-Register
	ein 8-bit-Eingabe-Register
nur bei Betriebsart Bit-I/O benutzt	ein 2-bit-Maskierungs-Steuerungsregister
	ein 8-bit-Maskierungs-Register
	ein-8-bit-Ein-/Ausgabe-Wahl-Register
Unter Zustandsbedingungen des peripheren Geräts programmierbare Interruptbearbeitung	
Automatische Interrupt-Vektorerzeugung und Prioritätscodierung ohne zusätzlichen Schaltungsaufwand durch Kaskadierung der Bausteine (*Daisy chain priority interrupt logic*)	
8 Ausgänge für den direkten Anschluß von Darlington-Transistoren	
Alle Ein- und Ausgänge voll TTL-kompatibel	

Tabelle 3. Daten und Ausstattung des Entwicklungssystems

Zentraleinheit:	Mikroprozessor Z-80 CPU
Speicher:	3 kByte ROM } für Betriebssystem
	1 kByte stat. RAM } reserviert
	16 kByte dyn. RAM für allgemeine Verwendung bis 60 kByte erweiterbar
Systemtakt:	Quarz 2 MHz (auf Wunsch 2,5 MHz)
Ein-/Ausgabekanäle:	– Standard-Interfaces Serienschnittstelle, Floppy-Disk-Schnittstelle Echtzeit-Testadapter („ICE"),
	– Optionale-Interfaces für PROM-Programmiergerät Zeilendrucker
	– Zwei zusätzliche Reservesteckverbinder
Ausstattung:	– Zentraleinheit-Steckkarte mit 4 kByte ROM/RAM und Betriebssoftware und RS-232 oder Strompräge-Serienschnittstelle
	– Eine 16-kByte-Dyn.-RAM-Steckkarte
	– Echtzeit-Testeinheit (bestehend aus 1 Echtzeitspeichersteckkarte und 1 Haltepunkt-Steckkarte)
	– Floppy-Disk-Steuerungssteckkarte für 1 Floppy-Disk-Doppellaufwerk
	– Steckkarte für Echtzeit-Testadapter
	– Echtzeittestadapter zur Direktverbindung zwischen Zentralgerät und Anwenderschaltung für Echtzeit Hard- und Software-Tests

Folgende Arten von Anwendersystem-Aktivitäten bzw. eine Kombination von ihnen lassen sich selektiv abspeichern und anzeigen: Lesezugriffe auf Speicher; Schreibzugriff auf Speicher; Lesezugriff auf I/O-Ports sowie Schreibzugriffe auf I/O-Ports.

Die gleichen Aktivitäten sind, wie schon erwähnt, als Haltepunktbedingungen zu verwenden, wobei als zusätzliche Bedingung Daten- und Adreßbusinformationen zu verwenden sind.

Dadurch lassen sich komplexe Ereignisse identifizieren, wie z. B. das Schreiben einer Eins auf Bitposition 6 des I/O-Ports Nummer F6H. *Tabelle 3* zeigt die wichtigsten Daten des Systems.

2.3 Software-Ausstattung

Mit dem Entwicklungssystem werden ein ROM-residentes Betriebssystem nebst Testprogramm mitgeliefert, ferner ein Floppy-Disk-Betriebssystem, ein Texteditor, ein Datei-Verwaltungsprogramm, ein Assembler sowie die nötige Dokumentation.

4 Mikrocomputer-Baugruppensystem

4.1 Bedeutung

Fertige Baugruppen ermöglichen den sofortigen Einstieg in die Praxis. Die Kontron Elektronik GmbH hält daher zunächst eine komplette CPU-Platine (Z-80-ECB/C) und eine Erweiterungsbaugruppe (Z-80-ECB/E) bereit. Als Ergänzung befinden sich eine dynamische RAM-Baugruppe, eine nichtflüchtige CMOS-RAM-Baugruppe und eine Stromversorgungsbaugruppe in Entwicklung, die ab 1. Quartal 1977 zur Auslieferung kommen sollen.

Mit diesem System kann der Anwender seine gesamte Computerschaltung einfach durch Auswahl und Zusammenstecken fertig bestückter, getesteter und in großer Stückzahl aufgelegter Standardplatinen realisieren, wobei die einzelnen Baugruppen in ihren Steckplätzen beliebig vertauscht werden dürften.

Zusammengefaßt beschränkt sich die Mikrocomputer-System-Entwicklung beim Anwender auf folgende Aktivitäten:

- Auswahl und Kauf der Platinen; in einfachen Fällen genügt auch die Minimalkonfiguration nach *Bild 7*
- Entwicklung der problemspezifischen Peripherie (Anschaltung von Leistungsschaltern, periph. Geräten usw.
- Entwicklung und Tests des Anwenderprogramms mit Hilfe des Z-80-Entwicklungssystems
- Systemtest mit dem Testadapter des Z-80-Entwicklungssystem

4.2 Ausstattung der Baugruppen

Neben dem Mikroprozessor Z-80-CPU, den Decoderbausteinen für die CPU-Karten-residenten Speicher und den Treibern für volle Programmspeicherausbaufähigkeit sind auf der CPU-Karte Z-80 ECB/C folgende Funktionseinheiten untergebracht:

- 1/4 kByte statisches RAM
- 1 kByte elektrisch programmierbarer Festwertspeicher (PROM)
- zwei 8-bit-Parallelschnittstellen mit voller Interrupt- und Quittungs (*Handshaking*)-Fähigkeit
- 1 Zweifach-Serienschnittstelle zum direkten Anschluß von Geräten für 24 V, RS 232 bzw. Stromschleifen-Schnittstelle (TTY-kompatibel)

Die problemlose Erweiterung dieser Karte bis zur vollen Ausbaufähigkeit der Z-80-CPU erlaubt die Erweiterungsbaugruppe Z-80-ECB/E, die mit

- 1 kByte statischem RAM

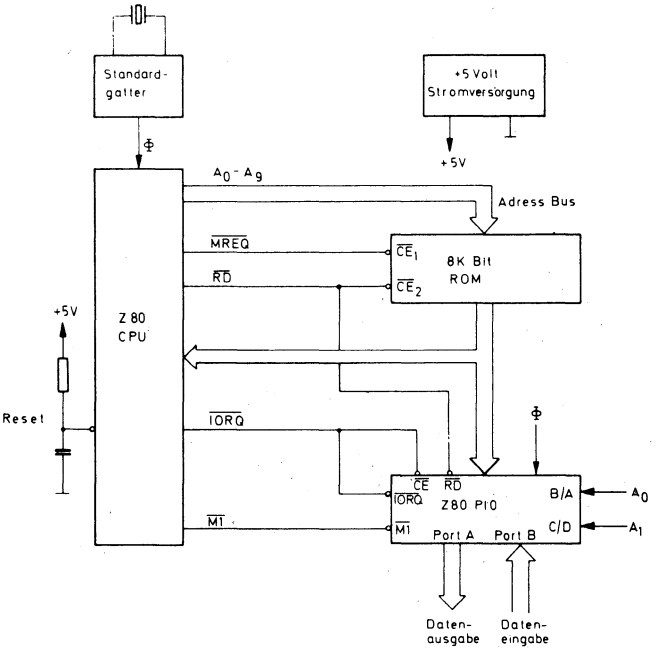


Bild 7. Minimalkonfiguration des Systems Z 80

- 4 kByte elektrisch programmierbarem Festwertspeicher (PROM)
- zwei 8-bit-Parallelschnittstellen mit voller Interrupt- und Quittungs-Fähigkeit
- 1 vierfach Zähler/Zeitgeber für programm- und zeitsparende Behandlung von Echtzeitaufgaben

eine vernünftige, stufenweise Erweiterung mit einem Baugruppentyp ermöglicht.

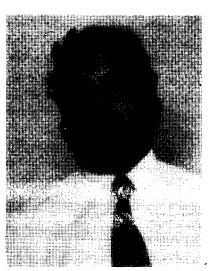
5 Standard-Software, Seminare

Selbstverständlich ist auch ein Spektrum von Standardprogrammen für das System Z 80 verfügbar, das dauernd erweitert wird.

Grundsätzlich werden diese Programme unter Angabe einer Freigabestufe abgegeben. Unter *Freigabestufe 1* fallen ausgetestete, gepflegte und dokumentierte Programme. *Freigabestufe 2* umfaßt ausgetestete Programme unterschiedlicher Herkunft, die jedoch nicht kontinuierlich gepflegt werden. Programme der *Freigabestufe 3* sind als Grundlage für eigene weitergehende Entwicklungsarbeit bzw. Denkanstöße zu verstehen.

Schulungskurse der Lieferfirma werden weiter dazu beitragen, die Einführung des Systems Z 80 zu erleichtern.

Dipl.-Ing. Hans-Peter Blomeyer-Bartenstein studierte Elektrotechnik und Wirtschaftswissenschaften an der TU München. Er leitet seit Oktober 1976 den Geschäftsbereich Mikrocomputer der Kontron Elektronik GmbH in Eching bei München. Seine Laufbahn begann er im Bauelementebereich der Siemens AG im Jahre 1970, wo er zunächst DV-Probleme bearbeitete, später das Applikationslabor für Digitalbausteine und zuletzt eine Mikrocomputer-Applikationsgruppe leitete; mit Mikroprozessoren befaßte er sich dort seit Ende 1974.



Dipl.-Ing. H.-P. Blomeyer-Bartenstein

Anwendung von Standard-Speicherbausteinen in Z 80 Mikrocomputersystemen

1 Aufgabenstellung

Die Zentraleinheit (CPU) des Mikrocomputersystems Z 80-CPU der Firma Zilog [1] ist so konzipiert, daß an sie direkt (d.h. ohne irgendwelche zusätzliche Interface-Hardware) alle gängigen statischen und dynamischen Speicherbausteine (ROMs, PROMs, EPROMs, EEPROMs [4], sowie RAMs [5]) anschließbar sind. Aufgrund der Architektur solcher Speicherbausteine (Komplexität, Anzahl der Chipauswahl-Eingänge, Adreßmultiplex usw.), ihrer maximalen Zugriffszeit und der endlichen Strom- und Kapazitätsbelastbarkeit der CPU-Ausgänge (1,8 mA) können jedoch – je nach Gesamtgröße, Architektur und Typ des Speichers – folgende Schaltmaßnahmen nötig werden:

- Pufferung von Adreß-, Daten- und Steuerbus der Z80-CPU;
- Adreßcodierung zur Aktivierung der einzelnen Bausteine;
- Seitenauswahl-Schaltung bei Speicherblöcken, deren Adreßbereich-Zuweisung umschaltbar („jumper"- oder „switch-selectable") bleiben soll;
- Adreßmultiplexen bei Verwendung von Speicherbausteinen mit gemultiplexten Adreßleitungen (z. B. dynamische 16-Kbit-RAMs mit 16 Stiften (*pins*);
- Schaltung zur Erzeugung eines WAIT-Signals bei Verwendung langsamer Speicherbausteine.

Für diese Schaltungsmaßnahmen werden nachfolgend Beispiele gezeigt; darüber hinaus wird näher auf das Zeitverhalten bei der Verwendung dynamischer RAMs eingegangen.

2 Grundsätzliches zum Anschluß von Festwert- und Schreib-/Lesespeichern

Es wurde bereits erwähnt, daß alle gängigen Typen von Festwertspeichern [4] mit der Z-80-CPU praktisch ohne zusätzliche Hardware verwendbar sind. Zu beachten ist dabei jedoch, daß sämtliche Mikrocomputerbausteine des Systems Z80 mit einer einzigen 5-V-Versorgungsspannung auskommen, heute verfügbare EPROMs und EEPROMs jedoch mehrere Versorgungsspannungen benötigen.

Daher kann im System Z80 die Verwendung von „fusible" PROMs besonders vorteilhaft sein, die ebenfalls mit einer einzigen 5-V-Spannung arbeiten (z. B. HARRIS 7641 oder 7781). Das Preisverhältnis zwischen „fusible" PROMs und EPROMs ist 1:2, so daß man bei der Verwendung von PROMs erst bei der dritten Version eines Festwertspeicherinhaltes höhere Kosten als bei der Verwendung und Neuprogrammierung von „UV"-EPROMs in Kauf nehmen müßte, wobei die zusätzlichen EPROM-Stromversorgungskosten noch nicht in die Rechnung einbezogen wurden.

2.1 Minimalkonfiguration

Obwohl das System Z80 aufgrund seiner hohen Leistungsfähigkeit und Verarbeitungsgeschwindigkeit im allgemeinen nicht zur Implementierung von Klein-Systemen gedacht ist, soll hier eine Minimalkonfiguration (vgl. [1] und [2]) gezeigt werden (*Bild 1*). Sie verwendet einen 2-KByte-Masken-ROM-Programmspeicher; vorausgesetzt ist dabei natürlich, daß Testschaltungen mit PROMs oder EPROMs implementiert waren oder wenigstens Hardware- und Systemtests mit dem Z80-Echtzeittestadapter (= *Users Hardware Interface*) durchgeführt wurden. Ein externes RAM wurde nicht vorgesehen, d. h. Interrupt- und Subroutinenbehandlung ist mit dieser Schaltung nicht möglich, da der Stapelspeicher (*Stack*) beim System Z80 im externen RAM liegt. Trotzdem ist der gezeigte Computer voll arbeitsfähig, da die Z80-CPU über einen internen zweiten Registersatz verfügt, der als RAM-Ersatz in beschränktem Umfang verwendet werden kann und die beiden Ports der Z80-PIO als Ein- und Ausgänge (inkl. Handshaking) verwendbar sind. Die Adreßleitungen A_0 und A_1 wurden zur Adressierung der beiden Ports innerhalb des Bausteins (*A/B-Select*) respektive zur Unterscheidung zwischen Steuerworten und -transfers oder Daten (*C/D-Select*) herangezogen.

Beim Hinzufügen weiterer Z80-PIO im Abfrage(= *Polling*)-Verfahren oder anderer Ein-/Ausgabebausteine kön-

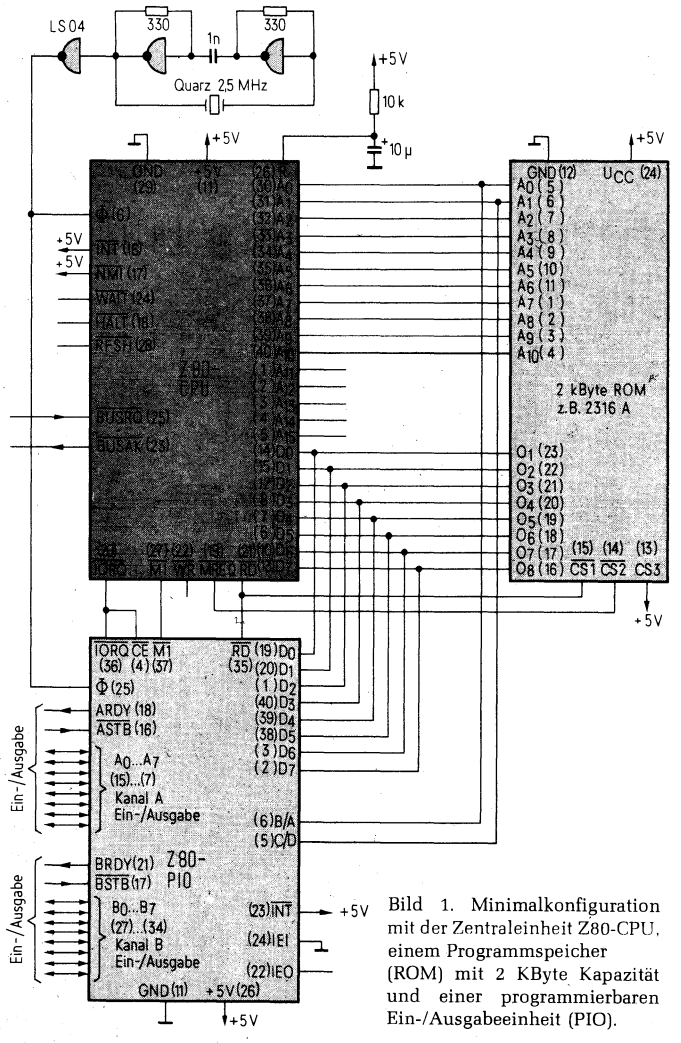


Bild 1. Minimalkonfiguration mit der Zentraleinheit Z80-CPU, einem Programmspeicher (ROM) mit 2 KByte Kapazität und einer programmierbaren Ein-/Ausgabeeinheit (PIO).

nen die CE-Eingänge der Bausteine natürlich nicht mehr direkt mit $\overline{IORQ}$ verbunden werden, sondern sie müssen geeignet decodiert werden.

Auf Details von Ein-/Ausgabesystemen beim Z80 kann hier nicht eingegangen werden; hierzu soll in Kürze eine gesonderte Applikationsschrift herausgegeben werden.

2.2 Zuschaltung statischer RAMs

Für den Fall, daß mit Interrupt und/oder Subroutinen gearbeitet werden soll oder einfach ein größerer Schreib-/Lesespeicher [5] für Daten nötig ist, kann man die Minimalkonfiguration in einfacher Weise entsprechend erweitern

2.3 Realisierung eines voll ausbaufähigen Computers

2.3.1 Pufferung

Die nötige Pufferung im Fall des Anschlusses bipolarer Bausteine ergibt sich einfach aus der Summe der „Fan In" sämtlicher auf einer Leitung liegenden Anschlüsse; bei Pufferung bidirektionaler Signale muß darüber hinaus den Pufferbausteinen „gesagt" werden, in welche Richtung die Datenübertragung momentan erfolgt.

Beim Anschluß von MOS-Speicherbausteinen ergibt sich die Grenze der höchstens anschließbaren Eingänge aus der kapazitiven Belastung, die diese Eingänge für die CPU (oder den Treiberbaustein) darstellen und die dadurch zusammen mit dem Ausgangsinnenwiderstand entstehende erhöhte Schaltzeit bzw. verminderte Flankensteilheit.

2.3.2 Bausteinauswahl

Zur Bausteinauswahl verwendet man Standard-Gatterbausteine, falls nur wenige „Chip-Select"-Signale erzeugt werden müssen, sonst 1 aus 8-Decodierer. Es soll auch noch an die Möglichkeit erinnert werden, komplexere Decodierungen durch PROMs vorzunehmen; dadurch läßt sich in vielen Fällen die erforderliche Bausteinstückzahl drastisch senken.

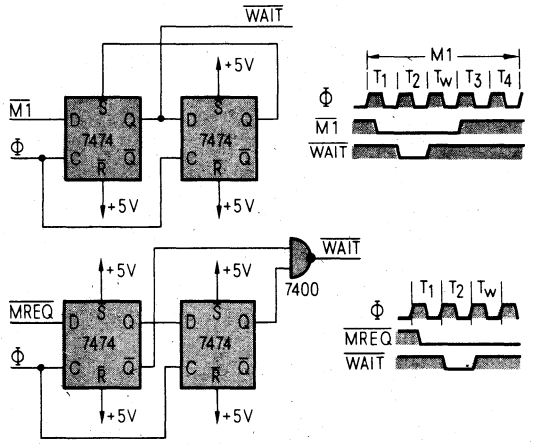
2.3.3 Schaltbeispiel

Bild 2 zeigt eine voll ausbaubare Schaltung; die zugehörige Adreßbelegung geht ebenfalls daraus hervor

2.4 Anschluß langsamer Speicherbausteine

Die meisten Speicherbausteine gibt es in mehreren Geschwindigkeitsversionen, wobei die Bausteine mit kürzerer Zugriffszeit häufig bedeutend teurer sind als die etwas langsameren Versionen. Da umgekehrt in vielen Anwendungen ein gewisser Verlust von Verarbeitungsgeschwindigkeit der CPU tragbar ist, kann man in diesen Fällen durch Erzeugung sogenannter WAIT-Zyklen in den Genuß niedriger Speicherbausteinkosten kommen.

Hierzu ist lediglich nötig, aus den Steuersignalen der Z80-CPU ein Warte-Signal WAIT zu erzeugen, das der CPU wiederum über den $\overline{WAIT}$-Eingang zugeführt wird und diese veranlaßt, den nächsten Maschinenzyklus erst einen Taktimpuls später als standardmäßig zu beginnen.



◀ Bild 3. Erzeugung eines WAIT-Signals zu jedem M 1-Zyklus

◀ Bild 4. Erzeugung eines WAIT-Signals zu jedem Speicherzugriff

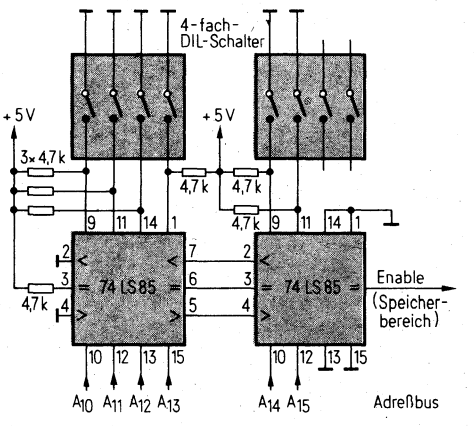


Bild 5. ▶ Speicher-Auswahlschaltung für 1 KByte-Speicherportionen in einem adressierbaren Speicherbereich von 64 KByte

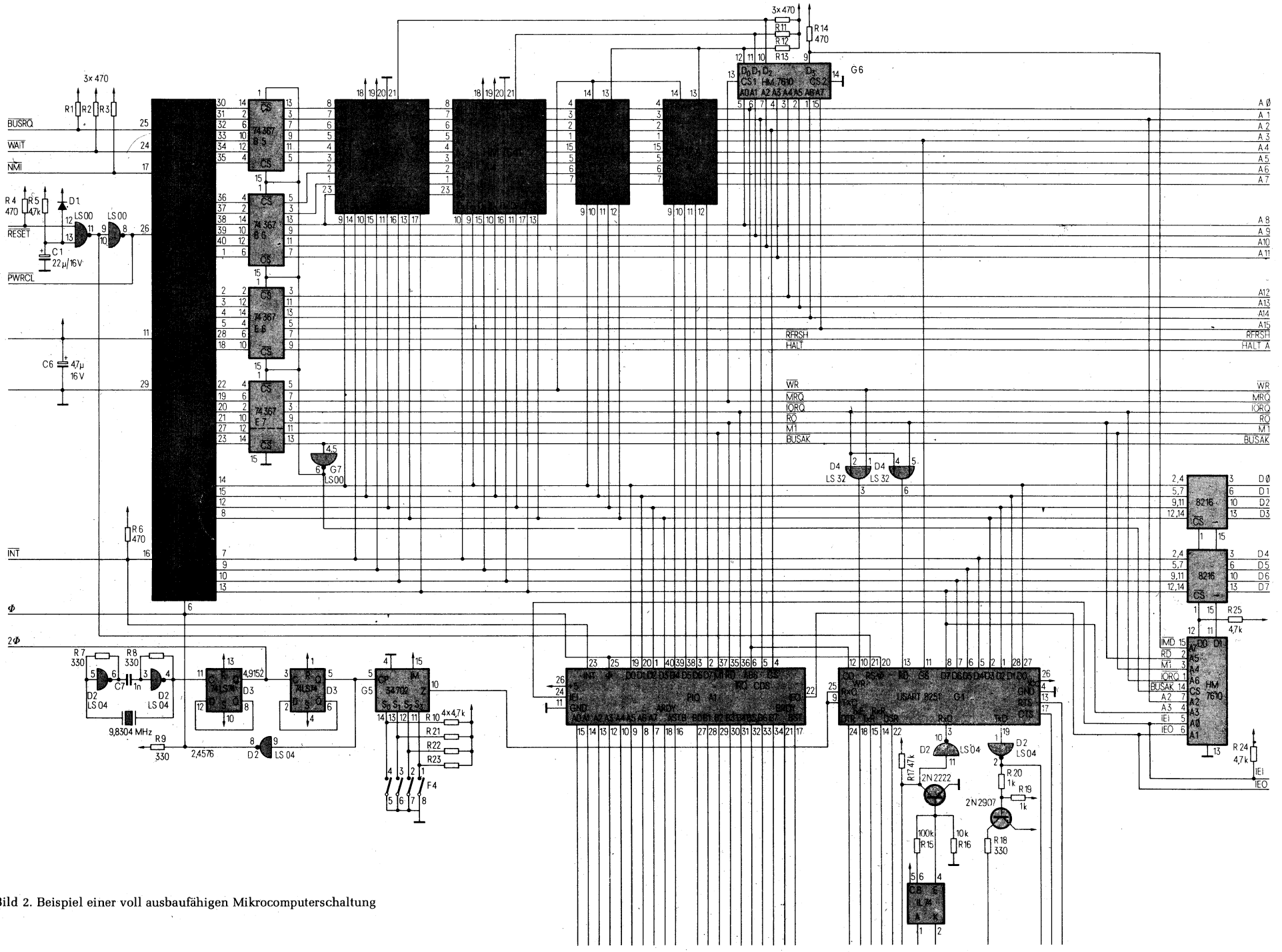


Bild 2. Beispiel einer voll ausbaufähigen Mikrocomputerschaltung

Je nachdem, ob lediglich dem besonders zeitkritischen M 1-Zyklus oder aber jedem Speicherzugriffszyklus ein „WAIT"-Zyklus T_w hinzugefügt werden soll, ist die Schaltung nach *Bild 3* respektive nach *Bild 4* zu verwenden; in den Bildern ist auch das entsprechende Zeitverhalten der Schaltung angegeben.

2.5 Schaltung zur Speicherbereichsauswahl

Bei modularem Aufbau von Mikrocomputern (d. h. in Fällen, wo die Gesamtschaltung aus einer Reihe von Standard-Steckkarten zusammengestellt wird) ist es wesentlich, die Adreßbereiche von Speicherkarten mit Schaltern (z. B. DIP-Switch) einstellbar zu halten. Eine einfache Schaltung hierfür zeigt *Bild 5*. Sie besteht im wesentlichen aus 4-bit-Vergleichern und den Speicherbereichsauswahl-(meist DIL)-Schaltern. Die Anzahl der nötigen Schalter und Komparatoren ist von der Größe des zu verschiebenden Speicherblocks und der Größe des Speicherbereichs abhängig, in dem der Speicherblock verschoben werden soll:

$$\text{Anzahl Schalter} = \text{ld}\left\{\frac{\text{Speicherbereich}^*}{\text{Speicherblockgröße}}\right\}$$

Anzahl Komparatoren = 1/4 Anzahl Schalter

3 Anschluß von dynamischen Speichern

3.1 CPU-Signale zum Betrieb von dynamischen Speichern

3.1.1 Zeitverhalten

Der Mikroprozessor Z80-CPU ist so ausgelegt, daß er mit minimalem Hardware-Aufwand die Verwendung von standardmäßig lieferbaren dynamischen Speicherbausteinen [5] erlaubt [3]. Die entsprechenden Steuersignale sind $\overline{\text{MREQ}}$, $\overline{\text{WR}}$ und $\overline{\text{REFR}}$ (*Bild 6*, *Bild 7* und *Bild 8*). Zu beachten ist, daß die Adressen 125 ns vor der Aktivierung des $\overline{\text{MREQ}}$-Signals stabil sind und die Information auf dem Datenbus 220 ns vor Aktivierung des $\overline{\text{WR}}$-Signals gültig ist. Nach Ende des $\overline{\text{WR}}$-Signals bleiben sie noch 150 ns lang gültig.

Bei der Verwendung von 4-Kbit-RAMs in 22-Stift-Gehäusen kann das Signal $\overline{\text{MREQ}}$ direkt als Steuerung des $\overline{\text{CE}}$-Takteingangs und das Signal $\overline{\text{WR}}$ als Steuerung des $\overline{\text{WRITE}}$-Eingangs der Speicherbausteine verwendet werden.

Bei der Verwendung von dynamischen 4-Kbit-RAMs mit 16 Stiften wird das $\overline{\text{MREQ}}$-Signal zur Erzeugung des $\overline{\text{RAS}}$-Signals direkt verwendet, während das Adreßmultiplexen und die Erzeugung des $\overline{\text{CAS}}$-strobe-Signals durch zusätzliche Standard-Hardware erfolgen. Das CPU-Signal $\overline{\text{WR}}$ liegt wie bei den 22-Stift-Bausteinen direkt am $\overline{\text{WRITE}}$-Eingang der Speicherbausteine an.

Der kritische Punkt für das Zeitverhalten (*timing*) beim Anschluß von dynamischen RAM-Bausteinen ist der erste Maschinenzyklus M1 (*fetch*-Zyklus). Hierbei geschieht der gesamte Speicherzugriff innerhalb der ersten 2 Taktimpulse (Bild 6). Man sieht, daß zwischen dem Zeitpunkt der Stabilisierung der Adressen und dem Signal $\overline{\text{MREQ}}$ 125 ns liegen, die bei weitem ausreichen, um die nötigen Decodier- und Einschwingzeiten abzudecken.

Kritischer ist die Situation in der High-Periode des Signals $\overline{\text{MREQ}}$, dessen Minimalzeit zu 170 ns garantiert ist. Die speicherseitig minimal erforderliche Dauer dieses Signals (*min. precharge-duration*) liegt bei 120...150 ns; da jedoch das $\overline{\text{MREQ}}$-Signal diverse Adreßdecoder und Puffer

* Der Speicherbereich ist in dieser Formel so zu wählen, daß er bei ganzzahligen Vielfachen der Speicherblockgröße beginnt und endet.

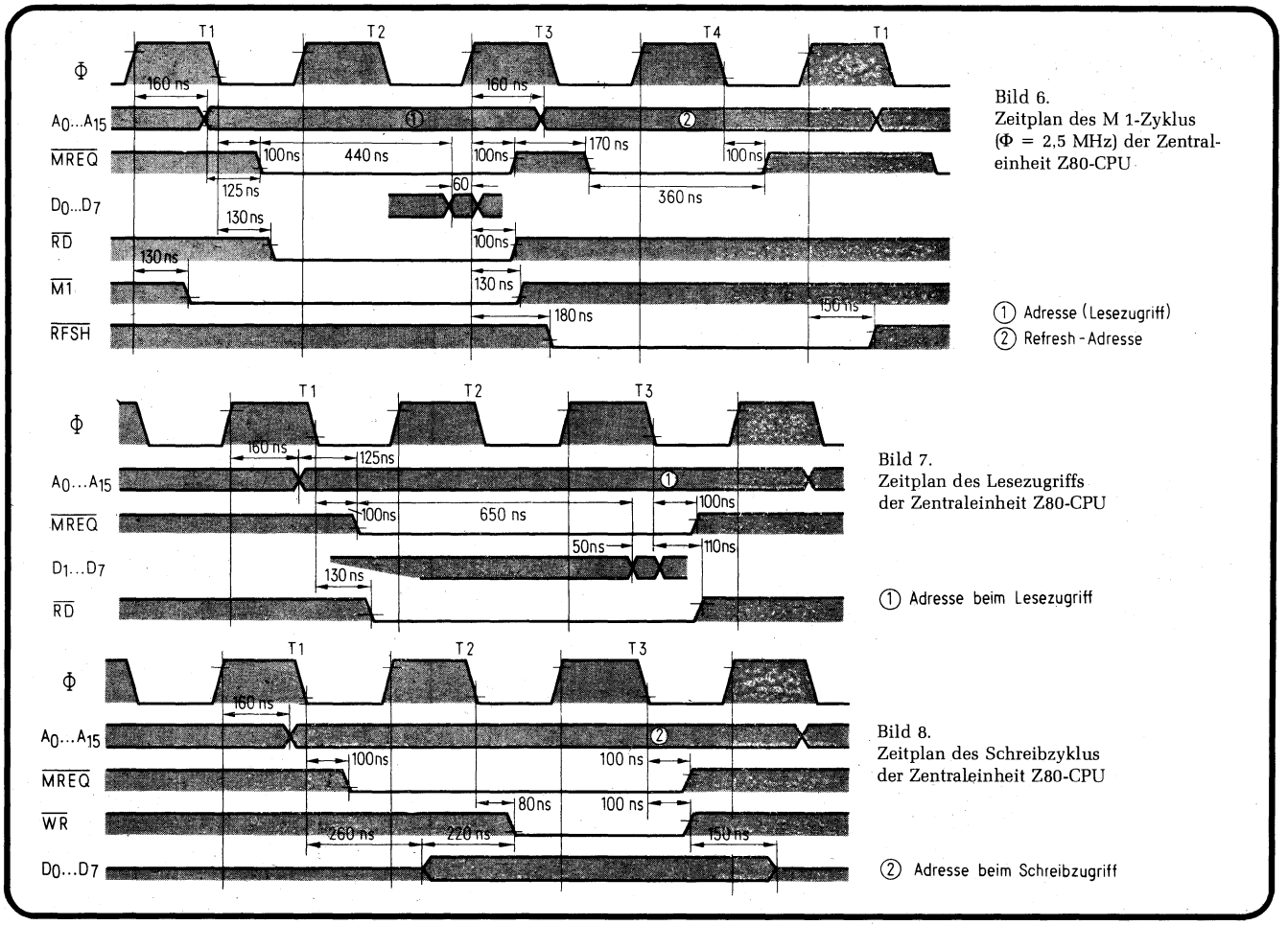
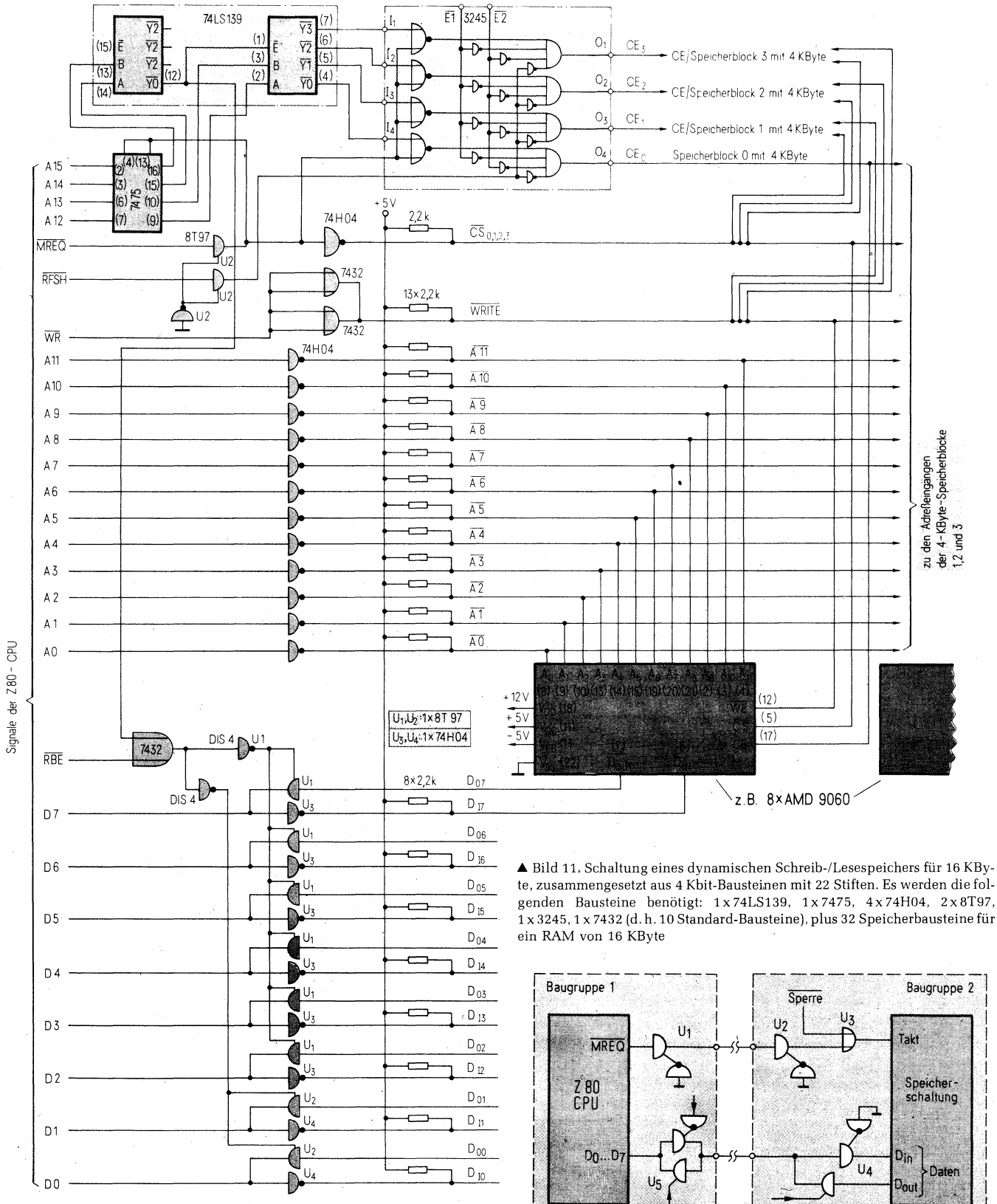


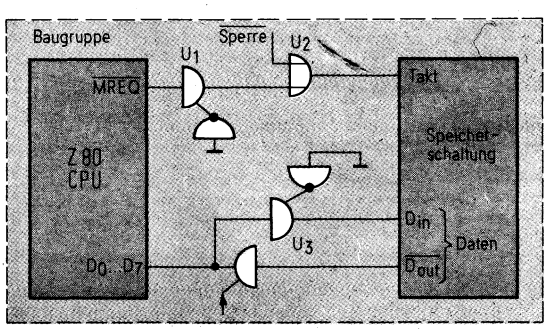
Bild 6.
Zeitplan des M 1-Zyklus
(Φ = 2,5 MHz) der Zentraleinheit Z80-CPU

Bild 7.
Zeitplan des Lesezugriffs der Zentraleinheit Z80-CPU

Bild 8.
Zeitplan des Schreibzyklus der Zentraleinheit Z80-CPU



▲ Bild 11. Schaltung eines dynamischen Schreib-/Lesespeichers für 16 KByte, zusammengesetzt aus 4 Kbit-Bausteinen mit 22 Stiften. Es werden die folgenden Bausteine benötigt: 1x74LS139, 1x7475, 4x74H04, 2x8T97, 1x3245, 1x7432 (d. h. 10 Standard-Bausteine), plus 32 Speicherbausteine für ein RAM von 16 KByte



◀ Bild 9. Prinzipschaltung bei „einfacher" Pufferung

Dyn. RAM-Bausteine 4 kbit					
16 Stifte			sinngemäß für 22 Stifte		
U1	8T97	12 ns	U1	8T97	12 ns
U2	74932	7,5 "	U2	3245	32 "
U3	8T28	17 "	U3	8T97	12 "
		36,5 ns			56 ns

Dyn. RAM-Bausteine 4 kbit					
16 Stifte			sinngemäß für 22 Stifte		
U1	8T97	12 ns	U1	8T97	12 ns
U2	8T97	12 "	U2	8T97	12 "
U3	74S32	7,5 "	U3	3245	32 "
U4	8T28	17 "	U4	8T97	12 "
U5	8T28	17 "	U5	8T28	17 "
		65,5 ns			95 ns
Verdrahtgs-Laufz.		25 ns	Verdrahtgs-Laufz.		25 ns
		90,5 ns			120 ns

▲ Bild 10. Prinzipschaltung bei „doppelter" Pufferung

durchläuft, muß beim Entwurf besonders darauf geachtet werden, daß an dieser Stelle vernünftige Schaltzeitreserven verbleiben. Während normaler Lesezyklen wird $\overline{MREQ}$ 650 ns aktiv, bevor die zu lesenden Daten gültig sein müssen.

Bei Schreibzyklen ist $\overline{MREQ}$ genauso lange aktiv wie bei Lesezyklen, wobei die einzuschreibenden Daten noch etwa 160 ns nach Ende des $\overline{MREQ}$-Signals gültig sein müssen und $\overline{WR}$ noch etwa 400 ns nach Beginn des $\overline{MREQ}$-Signals inaktiv bleibt. Daraus folgt, daß bei 4-Kbit-Speicherbausteinen mit 22 Stiften das $\overline{WR}$-Signal direkt die $\overline{WRITE}$-Eingänge der Speicher ansteuern kann. Bei RAM-Bausteinen mit 16 Stiften kann das invertierte $\overline{RD}$ anstelle von $\overline{WR}$ benutzt werden, wenn nur beim Schreibzyklus die gültige Dateninformation vor $\overline{CAS}$ verfügbar ist.

3.1.2 Pufferung

Bei Computerschaltungen, die auf einer einzigen Platine aufgebaut sind, kommt man mit einer einfachen Pufferung entsprechend *Bild 9* aus. Dort sind auch die Verzögerungszeiten, die in einer solchen Schaltung zu berücksichtigen sind, angegeben; man sieht, daß dynamische RAM-Bausteine mit 16 Stiften und mit 400 ns Zugriffszeit bzw. RAM-Bausteine mit 22 Stiften mit 380 ns Zugriffszeit in einem System mit der Z-80-CPU ohne weiteres bei einer Taktfrequenz von 2,5 MHz arbeitsfähig sind.

Bei Computerschaltungen, die auf mehrere Baugruppen verteilt sind, ist es aus Störsicherheitsgründen empfehlenswert, eine zweifache Pufferung der Busleitungen vorzunehmen, d. h. je einen Satz Pufferbausteine auf der Baugruppe der Zentraleinheit und auf den einzelnen Speicher-Baugruppen vorzunehmen (*Bild 10*). In diesem Fall werden selbstverständlich höhere Anforderungen an die Zugriffszeit der Speicherbausteine gestellt. 22-Stift-Bausteine dürfen eine Schreib-/Lese-Zykluszeit von höchstens 315 ns haben, während 16-Stift-Bausteine mit max. 340 ns auskommen. Sorgfältige Leitungsführung ist dabei selbstverständlich; insbesondere sollte dafür gesorgt werden, daß die Baugruppen mit den dynamischen RAMs möglichst nahe an der Zentraleinheits-Platine gelagert sind.

3.1.3 Das Auffrischen („Refresh")

Bei einer CPU-Taktfrequenz von 2,5 MHz erfolgen Speicherzugriffe im Mittel alle 1,6 µs. Obwohl bei der Verwendung eines 2,5-MHz-Taktes das Auffrischen (*refresh*) etwa 9 mal öfter als notwendig (alle 2 ms) erfolgt, würde durch Beschränkung auf die minimal nötigen Auffrisch-Zyklen an Leistungsverbrauch nur 100 mW eingespart werden. In Fällen, wo minimaler Leistungsverbrauch besonders wichtig ist, wird man daher mit Hilfe von Zeitsteuerungs-Schaltungen eine Beschränkung der Auffrisch-Zugriffe auf das absolute Mindestmaß durchführen.

3.2 Schaltung mit dynamischen 4-Kbit-RAMs mit 22 Stiften

Die Schaltung nach Bild 11 arbeitet mit praktisch jedem beliebigen dynamischen RAM-Baustein mit 22 Stiften, bei dem der Zeitabstand zwischen dem Beginn des $\overline{CE}$-Signals und dem Zeitpunkt, bei dem die Information auf dem Datenbus gültig ist, kleiner als 350 ns ist, wobei für Pufferbausteine und Verdrahtung eine Gesamtverzögerung von 30 ns übrig bleibt [6].

Wie unter 3.1.1 erwähnt, treten die zeitkritischsten Bedingungen während des M1-Zyklus auf. Bei einer Betriebsfrequenz von 2,5 MHz ist die minimale Zeit in diesem Lesezyklus 440 ns. Dabei müssen selbstverständlich die Verzögerungszeiten der Steuer- und Decodierschaltungen berücksichtigt werden.

Der obere Teil von Bild 11 zeigt die Schaltung eines solchen Systems. Ein Decodierer führt die Seiten-(*page*)-Auswahl durch, ein zweiter ist für das Umschalten zwischen den vier CE-Takten verwendet. Die Adreßleitungen A 12...A 15 werden den Speicherbausteinen über einen getakteten Pufferspeicher zugeführt, um eventuelle Störungen auf den Adreßleitungen während des $\overline{MREQ}$-Signals aufzufangen; diese Maßnahme ist bei den Adreßleitungen A 0...A 11 nicht nötig, da diese bereits in den Speicherbausteinen intern zwischengespeichert sind.

Das $\overline{WRITE}$-Signal für die Speicher wird direkt aus dem $\overline{WR}$-Signal der Z80-CPU erzeugt; es wurden Treiberbausteine mit Pull-up-Widerständen zwischengeschaltet, um einen ausreichenden Störabstand ($U_{il} \leqq 0{,}4$ V und $U_{ih} \geqq 2{,}6$ V) zu erreichen. Aus dem gleichen Grunde wurden auch Adreß- und Datenbus mit Treiberstufen mit Pull-up-Widerständen gepuffert.

3.3 Schaltungen mit dynamischen 4-Kbit-RAMs mit 16 Stiften

Die im folgenden zu besprechende Schaltung arbeitet mit praktisch jedem dynamischen Speicherbaustein mit 4 Kbit und 16 Stiften, der einen Lese-/Schreibzugriffszyklus von kleiner als 350 ns hat, wobei Laufzeiten zwischen der CPU und den Speichern (Treiber, Verdrahtung) unter 40 ns angenommen sind. Hier sind besonders folgende Punkte zu beachten:

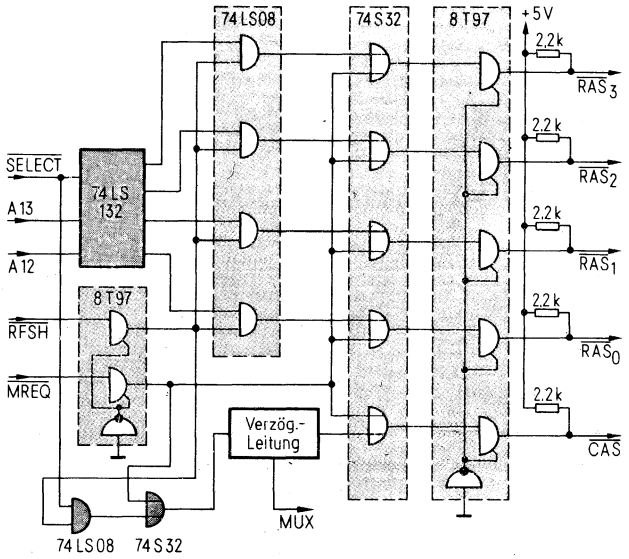
- Erzeugung des $\overline{RAS}$-Signals aus dem $\overline{MREQ}$-Signal und Decodierung zur Minimierung der Verlustleistung;
- Umschalten zwischen Zeilen- und Spalten-Adresse;
- Erzeugung des $\overline{CAS}$-Signals;
- Erzeugung des $\overline{WRITE}$-Signals, sobald die Information auf dem Datenbus gültig ist ($\overline{WRITE}$ kann vor oder nach $\overline{CAS}$ beginnen);
- Verbindung des $\overline{CAS}$-Signals mit sämtlichen Speicherbausteinen, soweit deren Datenausgangspuffer wired-OR-verknüpft sind, um sicherzustellen, daß nicht-angesprochene Bausteine gesperrt bleiben; und schließlich
- Erzeugung des $\overline{RAS}$/$\overline{CAS}$-Auffrisch-Signals.

Die *Bilder 12a, 12b* und *12c* zeigen drei verschiedene Möglichkeiten des Adreßmultiplexens (MUX) und der Erzeugung des $\overline{CAS}$-Signals. Die einfachste Methode ist die Verwendung einer digitalen Verzögerungskette mit Anzapfungen bei 125 und 175 ns (Bild 12a). Diese Methode ist natürlich recht aufwendig.

Eine zweite Möglichkeit ist, die steigende Flanke des Φ-Impulses, der auf das $\overline{MREQ}$-Signal folgt, als MUX zu verwenden und $\overline{CAS}$ über Gatterlaufzeiten zu erzeugen (Bild 12b). Bei dieser Methode entsteht allerdings eine starke Abhängigkeit der $\overline{CAS}$-Verzögerungszeit von der Taktfrequenz.

Die dritte Möglichkeit (Bild 12c) verwendet ein 2-Φ-Signal, erzeugt MUX aus Φ und $\overline{CAS}$ von der fallenden Flanke des 2-Φ-Impulses, der auf MUX folgt.

Eine vollständige Schaltung mit RAM-Bausteinen mit 16 Stiften ist als Sonderdruck erhältlich [6]. – Das Adreßmultiplexen erfolgt durch abwechselnde Freigabe zweier Tri-State-Treiber. Der Vorteil dieser Methode ist, daß dabei die RAM-Adreß-Eingangskapazitäten während des Zeilen-/Spaltenumschaltens wesentlich schneller umgeladen werden als dies bei der Verwendung normaler TTL-Treiberbausteine möglich wäre.



◀ Bild 12a. Adreßmultiplexen (MUX) und Erzeugung des $\overline{CAS}$-Signals mit einer angezapften digitalen Verzögerungskette

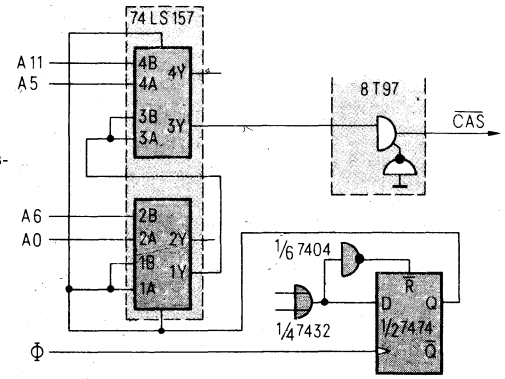


Bild 12b. ▶ Adreßmultiplexen und Erzeugung des $\overline{CAS}$-Signals über Gatterlaufzeiten

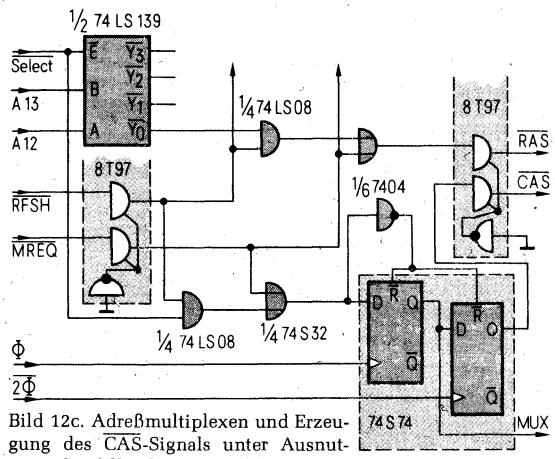


Bild 12c. Adreßmultiplexen und Erzeugung des $\overline{CAS}$-Signals unter Ausnutzung der fallenden Flanke des 2Φ-Impulses

4 Spezielle Hinweise zum Anschluß dynamischer 4-K-RAMs

Sollen andere als die in den gezeigten Schaltungen verwendeten Bausteine eingesetzt werden, muß eine Reihe spezieller Regeln beachtet werden, da selbst stiftkompatible dynamische RAM-Bausteine verschiedener Hersteller sich nicht exakt gleich verhalten.

4.1 Dynamische RAMs mit 4 Kbit und 22 Stiften

Die gesamte Zeitsteuerung der Bausteine ist vom CE-Takt abgeleitet. Dieses Signal (MOS-Eingang) hat einen Hub von U_{dd}–U_{ss} und wird im allgemeinen mit einem TTL/MOS-Pegelwandlergatter erzeugt; seine Verzögerungszeit liegt in der Größenordnung von 50 ns. Der Speicherbaustein zieht praktisch nur Strom, solange das CE-Signal „HIGH" ist.

Adressen und das Signal CS müssen vor der steigenden Flanke von CE eingeschwungen sein, d. h. daß zu diesem Zeitpunkt sämtliche Anwahlvorgänge abgeschlossen sein müssen. CS und die Adreßinformationen werden im allgemeinen intern zwischengespeichert und dürfen nach einer vorgegebenen Zeit wieder ungültig werden. – Des weiteren kann bei manchen Bausteinen der Leistungsverbrauch dadurch reduziert werden, daß man sämtliche Adreßeingänge auf einen bestimmten Logikpegel bringt; in vielen Fällen ist diese Tatsache in den Datenblättern nicht erwähnt.

Während des Lesezyklus muß der $\overline{WRITE}$-Eingang inaktiv („HIGH") sein, und zwar vor der steigenden Flanke von CE, um zu verhindern, daß die Information in den Speicherbausteinen zerstört wird. Während der Schreibzyklen kann das $\overline{WRITE}$-Signal vor oder auch nach Beginn dieser Zyklen aktiv werden, wobei selbstverständlich grundsätzlich gewisse Restriktionen zu beachten sind.

Der Ausgangspuffer ist vor und nach jedem Zyklus in hochohmigem Zustand. Er enthält nach der durch den Zugriff bedingten Verzögerungszeit die aus dem Speicher ausgelesenen Daten (während eines Lesezyklus, falls CS „LOW" ist) und geht nach Ende des CE-Signals wieder in den hochohmigen Zustand über. An dieser Stelle soll noch angemerkt werden, daß man wegen ihrer geringen Verzögerungszeit Tri-State-Treibern gegenüber Open-Collector-Treibern den Vorzug geben sollte; so würde sich beispielsweise bei einer Lastkapazität von 80...120 pF und einem 2,2-kΩ-Pull-up-Widerstand eine Verzögerungszeit von etwa 100 ns ergeben.

Die Eingangsspannungspegel der hier besprochenen Bausteine U_{il} und U_{ih} sind von Exemplar zu Exemplar unterschiedlich. In der Regel ist U_{il} gleich 0,6 V, während U_{ih} zwischen 2,2 und 3,0 V garantiert wird. Zur Sicherstellung eines Störabstandes von etwa 200 mV sowohl im H- als auch im L-Pegel müssen Treiberbausteine der Serie 74 oder 74 H mit Pull-up-Widerständen verwendet werden, so daß auch Bausteine mit U_{ih} = 2,2 V oder mehr den gewünschten Störabstand erreichen.

Während CE inaktiv ist, werden die internen Kapazitäten der Speicherbausteine vom $\overline{MREQ}$-Signal (H) aufgeladen. Die kritischsten Zeitverhältnisse treten, wie schon bemerkt, während des M1-Zyklus auf, und zwar zwischen dem Befehls-Lese-Zyklus (*fetch*-Zyklus) und dem Auffrischen (*refresh*). Da als Dauer des Signals $\overline{MREQ}$ der Z80-CPU 170 ns minimal garantiert sind und die entsprechenden Zeitbedingungen der dynamischen RAMs in der Regel bei 120...150 ns liegen, muß dafür Sorge getragen werden, daß sämtliche Puffer-, Decodierer- und Verdrahtungslaufzeiten unter der restlichen verfügbaren Verzögerungszeit bleiben. In kritischen Fällen darf daher an dieser Stelle auch die Temperaturabhängigkeit der Laufzeiten nicht vernachlässigt werden (genügende Laufzeitreserve!).

Da die CE-Eingänge der Speicherbausteine relativ große Eingangskapazitäten haben (25...30 pF), ist beim Schaltungsentwurf besonders große Sorgfalt auf die Erzeugung dieses CE-Signals zu legen. Da in einem 8-bit-System mindestens acht solche CE-Eingänge an einem CE-Takttreiber angeschlossen sind, kann vor allem durch ein unzureichendes Platinen-Layout eine Laufzeitkette entstehen, in der die Qualität des am letzten Baustein anlangenden CE-Signals nicht mehr ausreichend ist.

Die Versorgungsspannungen U_{dd} (+12 V ±5 %) und U_{bb} (–5 V ±5 %) werden von CE bausteinintern ein- und ausgeschaltet; dadurch entstehen bei den CE-Flanken Hochfrequenzstörungen auf diesen Speiseleitungen, die besonders sorgfältig über Hochfrequenz-Kondensatoren (Keramik) möglichst dicht an den U_{dd}- und U_{bb}-Anschlüssen gegen Masse (U_{ss}) kurzgeschlossen werden müssen. Darüber hin-

aus sollte man versuchen, den Speicherbereich auf der Leiterplatine *so kompakt wie irgend möglich* zu machen und die Adreß- und Datenpuffer räumlich möglichst nahe an die Speicherbausteine legen.

Diese Art RAMs hat einen Leistungsverbrauch, der direkt proportional der Zugriffshäufigkeit und der „Ein"-Zeit des CE-Taktes ist. Aus diesem Grund kann man die Gesamtverlustleistung eines Speichersystems dadurch verringern, daß man CE decodiert und *nur* dem Speicherbereich zuführt, der momentan tatsächlich aktiv ist.

4.2 Dynamische RAMs mit 4 Kbit und 16 Stiften

Bei diesen Bausteinen wird die Adresse über sechs Adreßleitungen eingelesen, wobei diese Leitungen über zwei Taktsignale in Zeilen- und Spaltenadresse gemultiplext werden. Im Gegensatz zu den dynamischen 4-K-Bausteinen mit 22 Stiften sind die 16-Stift-Bausteine verschiedener Hersteller *fast in allen Fällen risikolos austauschbar*. Trotzdem soll hier noch eine Liste der wichtigsten Gesichtspunkte bei ihrer Verwendung gegeben werden:

Jeder Zugriffszyklus beginnt mit dem Zeilen-Adreß-Takt $\overline{RAS}$ (*row-address-strobe*), der sämtliche Zeilenadressen von den Adreßleitungen einliest und zwischenspeichert. Dabei wird eine der 64 Zeilen ausgewählt und sämtliche 64 Speicherstellen auf die zugehörigen Leseverstärker geschaltet, die die Informationen aufnehmen, zwischenspeichern und wieder in die Zelle einschreiben. Auf diese Weise erfolgt durch $\overline{RAS}$ ein von $\overline{CAS}$ völlig unabhängiges Auffrischen.

Der zweite Teil eines jeden Zugriffs wird vom Spaltenadreßtakt $\overline{CAS}$ (*column-address-strobe*) eingeleitet, der die Spaltenadressen einliest und abspeichert. Die Signale $\overline{WRITE}$ und $\overline{CS}$ müssen nicht unbedingt bis zum Auftreten von $\overline{CAS}$ anliegen. Entsprechend dem Zeilenauswahltakt wird auch $\overline{CAS}$ decodiert, aktiviert einen von 64 Leseverstärkern/Zwischenspeichern und schaltet den ausgewählten Zwischenspeicher auf die Ausgangstreiberstufe. – Bei den meisten lieferbaren Bausteinen werden die Daten in Zwischenspeicher geladen und bleiben gültig, bis der nächste $\overline{CAS}$-Impuls auftritt.

Da der $\overline{RAS}$-Zyklus beginnen kann, sobald die Zeilenadresse stabil anliegt, jedoch bevor der Speicherbereichsauswahlvorgang abgeschlossen ist, sparen 16-Stift-RAMs gegenüber 22-Stift-RAMs etwa 50...100 ns. Da hier jedoch die Verlustleistung in erster Linie von $\overline{RAS}$ abhängt, wird man in den meisten Fällen $\overline{RAS}$ decodieren, wodurch ein Teil dieses Geschwindigkeitsvorteils verlorengeht.

Die Eingangspegel von $\overline{RAS}$ und $\overline{CAS}$ liegen beim L-Pegel (U_{il}) zwischen 0,6 und 0,8 V, bei H-Pegel (U_{ih}) zwischen 2,4 und 2,7 V. Die Ansteuerung dieser Eingänge über TTL-Gatter erfolgt daher am besten unter Verwendung von Pull-up-Widerständen. – Schottky-Treiberbausteine sollten bei Speichern mit U_{il} kleiner als 0,7 V nicht verwendet werden, da sie ein U_{ol} kleiner oder gleich 0,5 V haben und damit nicht der geforderte Störabstand von 200 mV erreichbar ist. – Praktisch resultiert aus der Möglichkeit, bei 16-Stift-RAMs TTL-Treiber anstelle von TTL/MOS-Pegelwandlern zu verwenden, eine Zeitersparnis von 20...40 ns.

Die sechs Adreßleitungen müssen zwischen Zeile und Spalte nach der Eingabe der Zeilenadresse umgeschaltet werden. Die Kapazitäten der Adreßeingänge liegen bei 5...10 pF, so daß ein 16-KByte-Speicherbereich für den Treiber eine Last von 150...250 pF pro Adreßleitung darstellt. Da die meisten TTL-Ausgänge nur für eine kapazitive Belastung unter 50 pF spezifiziert sind, muß man einen Bustreiber, wie die Typen 74S50 oder 7428 (für 150 pF Lastkapazität spezifiziert), verwenden. – Wegen des Eingangspegels $U_{il} \leqq 2{,}4$ V, den die meisten Bausteine haben, ist zur Erreichung eines ausreichenden Störabstandes ein Pull-up-Widerstand erforderlich.

Der $\overline{WRITE}$-Eingang wird erst abgefragt, wenn das $\overline{CAS}$-Signal aktiv wird. Wird $\overline{WRITE}$ vor $\overline{CAS}$ aktiv, so führt das RAM einen Schreibzyklus in der kürzestmöglichen Zykluszeit aus. Wird $\overline{WRITE}$ erst nach Beginn von $\overline{CAS}$ aktiv, so führt das RAM einen Lese-/Änderungs-Schreibzyklus durch.

Bei einem Schreibzugriff werden die eingehenden Daten durch $\overline{CAS}$ oder $\overline{WRITE}$ übernommen, und zwar von dem Signal, das als letztes auftritt. Man kann somit durch Verzögerung von $\overline{WRITE}$ gegenüber $\overline{CAS}$ erreichen, daß die Information erst nach Auftreten des $\overline{WRITE}$-Signals übernommen wird.

Der Ausgangspuffer der 16-Stift-RAMs speichert die ausgelesenen Daten zwischen, sobald $\overline{CAS}$ aktiv wird und während $\overline{CS}$ aktiv ist. Erscheint ein $\overline{CAS}$-Signal ohne $\overline{CS}$, werden die Ausgangspuffer/Zwischenspeicher gesperrt und gehen in den hochohmigen Zustand über.

Die Ausgänge können wired-OR-verknüpft werden, soweit $\overline{CAS}$ bei Auswahl eines Speicherbereiches sämtlichen Bausteinen zugeführt wird, während $\overline{RAS}$ decodiert und nur dem angesprochenen Speicherbereich zugeführt wird. Sobald ein $\overline{CAS}$-Signal empfangen ist, gehen sämtliche Ausgangstreiber der Speicher in den hochohmigen Zustand über, so daß keine Buskonflikte auftreten können.

Das Auffrischen (*refresh*) wird ausschließlich über das $\overline{RAS}$ bewirkt ($\overline{CAS}$-Signal nicht nötig) und hat alle 2 ms zu erfolgen. Dies läßt sich für DMA-Anwendungen (direkten Speicherzugriff) ausnutzen, bei denen bekanntlich das Auffrisch-Signal der Z80-CPU während des M1-Zyklus unterbunden wird [2]. Man läßt dann eine Serie von 64 aufeinanderfolgenden Zugriffen zu, die alle 2 ms lediglich $\overline{RAS}$ erzeugen; dadurch wird ein „transparentes" Auffrischen durchgeführt.

Da die interne Schaltung dynamisch aufgebaut ist, muß bei jedem Einschalten der Versorgungsspannung eine Reihe von „leeren" $\overline{RAS}$/$\overline{CAS}$-Zyklen erzeugt werden, um eine definierte Vor-Aufladung zu garantieren. Auch sollten $\overline{RAS}$-Zyklen allein nicht über einen längeren Zeitraum als 2 ms gegeben werden, da sonst die Ausgangstreiber unter Umständen unkorrektes Verhalten zeigen. Entsprechend sollten auch in einem solchen Fall mehrere „leere" $\overline{CAS}$-Zyklen gegeben werden, bevor Daten von den Bausteinen ausgewertet werden.

Genauso wie bei den 22-Stift-Speichern ist bei den 16-Stift-Ausführungen die Verlustleistung proportional der Anzahl der Zugriffe pro Zeiteinheit. Anders als bei den 22-Stift-Bausteinen ist jedoch bei vielen 16-Stift-Bausteinen der Leistungsverbrauch *nicht* von der Dauer der einzelnen Zugriffe abhängig, da hier in sämtlichen internen Schaltungen der Bausteine konsequent dynamische Speicherzellen verwendet werden.

Literatur

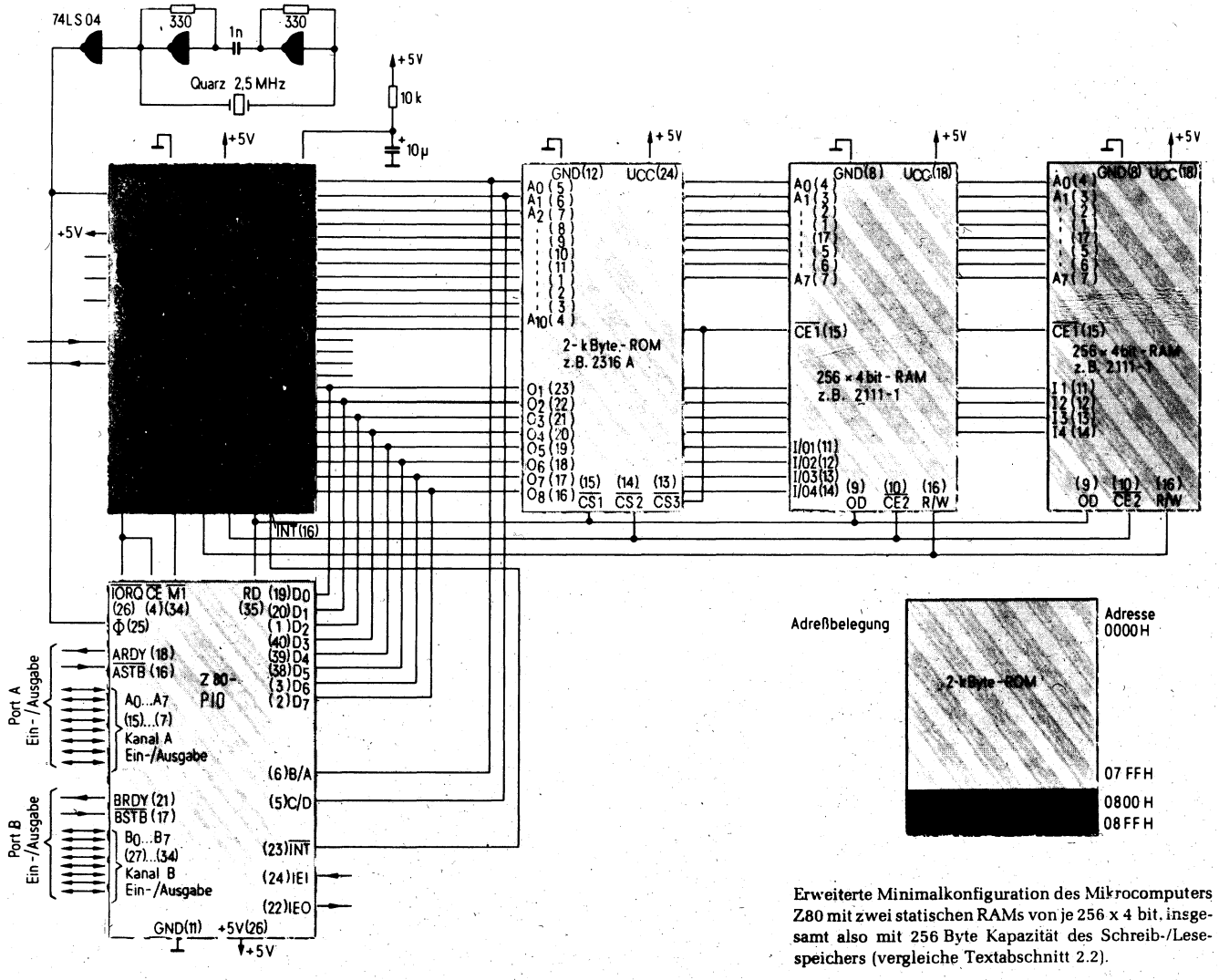
[1] *Blomeyer-Bartenstein, H.-P.*: Ein neues Mikrocomputer-Konzept. ELEKTRONIK 1976, H. 11, S. 83...87.

[2] Z-80-CPU-Technical Manual (Zilog-Druckschrift, beziehbar von der Kontron Elektronik GmbH).

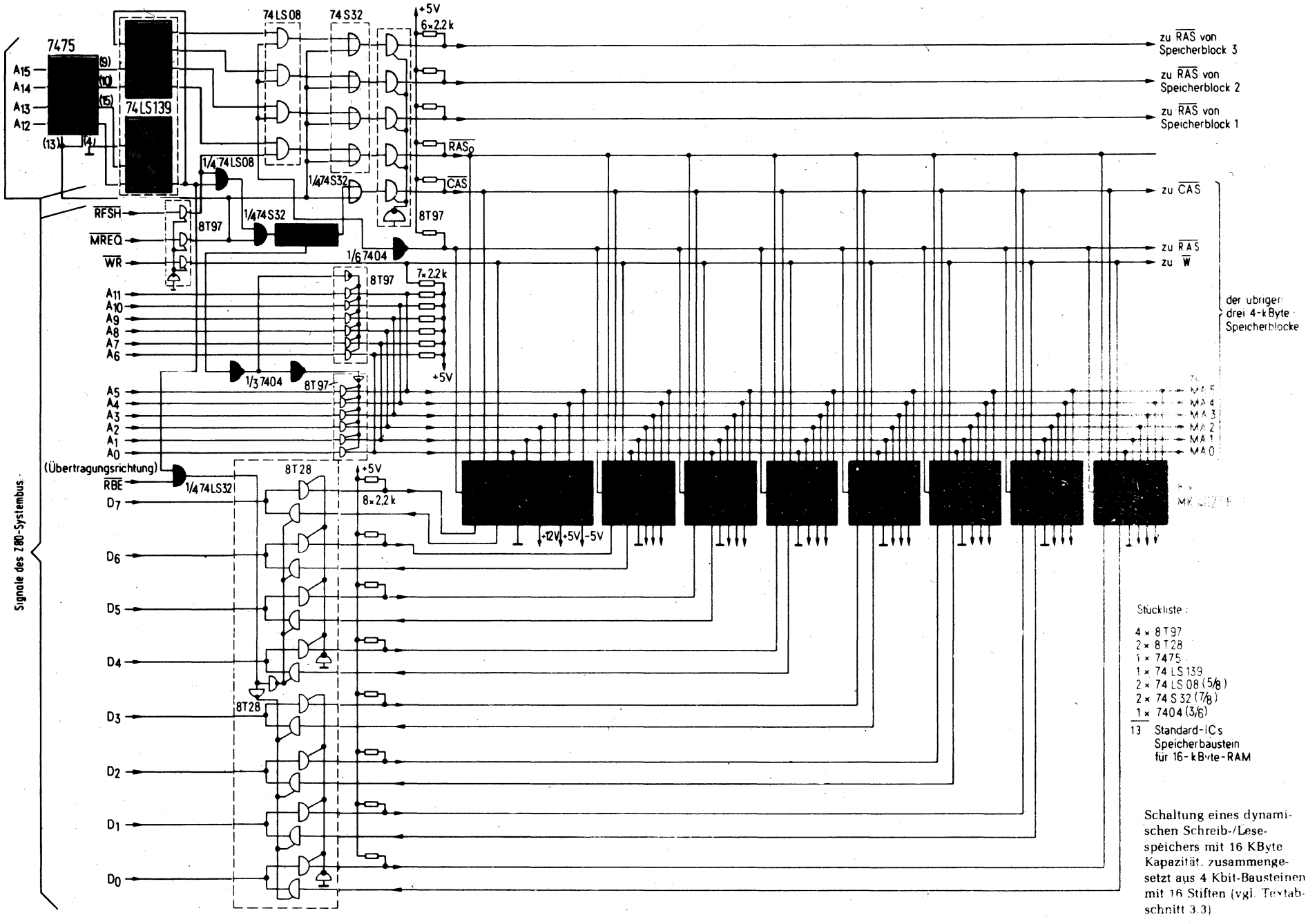
[3] *Elliot, M.*: Interfacing dynamic RAMs to the Z-80-Microprocessor (Zilog-Applikationsschrift).

[4] *Timm, V.*: Im Blickpunkt: ROMs, PROMs und PLAs. ELEKTRONIK 1976, H. 5, S. 38...47.

[5] *Schlenther, M.*: Im Blickpunkt: Bipolare und MOS-Schreib-/Lesespeicher (RAMs). ELEKTRONIK 1976, H. 10. S. 57...67.



Erweiterte Minimalkonfiguration des Mikrocomputers Z80 mit zwei statischen RAMs von je 256 x 4 bit, insgesamt also mit 256 Byte Kapazität des Schreib-/Lesespeichers (vergleiche Textabschnitt 2.2).



Stückliste:

4 ×	8T97
2 ×	8T28
1 ×	7475
1 ×	74LS139
2 ×	74LS08 (5/8)
2 ×	74S32 (7/8)
1 ×	7404 (3/6)
13	Standard-ICs Speicherbaustein für 16-kByte-RAM

Schaltung eines dynamischen Schreib-/Lesespeichers mit 16 KByte Kapazität, zusammengesetzt aus 4 Kbit-Bausteinen mit 16 Stiften (vgl. Textabschnitt 3.3)

Mikrocomputer-Entwicklungssystem ZILOG Z80

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 sein

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.

Der residente BASIC-Interpreter und der residente PL/Z-Compiler erlauben Ihnen sogar das Verwenden bekannter, höherer Programmiersprachen! Fortran und Cobol sind in Vorbereitung . . . .

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 das alles in Echtzeit — sodaß Sie nicht riskieren, daß die eigentliche Fehlersuche erst dann anfängt, wenn die Fähigkeiten Ihres Entwicklungssystems ausgeschöpft sind.

Und warum beziehen Sie Ihr Z80-Entwicklungssystem von KONTRON?

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 KONTRON stellt für Sie einen Partner in Deutschland 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!

KONTRON bietet 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: KONTRON bietet 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!

Und wenn Ihnen 27473,— DM für das komplette System zuviel sind — kontaktieren Sie uns!

Wir können Ihnen noch preiswertere Systeme liefern — allerdings dann mit eingeschränktem Leistungsumfang.

KONTRON ELEKTRONIK GMBH
GESCHÄFTSBEREICH MIKROCOMPUTER

8057 Eching b. München
Oskar-von-Miller-Str. 1
Tel. (0 81 65) 77-1
Telex 05 27 531

TECHNISCHE BÜROS

4000 Düsseldorf
Pempelforter Str. 10
Tel. (02 11) 35 01 51
Telex 08 582 675

6000 Frankfurt-Süd
Kennedy-Allee 34
Tel. (06 11) 63 60 61
Telex 04 14 881

2000 Hamburg 70
Königsreihe 2
Tel. (0 40) 68 23 21
Telex 02 11 998

7000 Stuttgart 30
Heilbronner Str. 380
Tel. (07 11) 81 46 21
Telex 07 23 061

1000 Berlin 41
Albrechtstraße 34
Tel. (0 30) 7 92 30 31-3
Telex 01 85 484

3167 Burgdorf/Hannover
Postfach 162
Tel. (0 51 36) 56 00

Hans Peter Blomeyer-Bartenstein und Bernhard Mindermann

Ein/Ausgabe und Interrupt in Z 80-Mikrocomputer-Systemen

Hans Peter Blomeyer-Bartenstein und Bernhard Mindermann

Ein/Ausgabe und Interrupt in Z 80-Mikrocomputer-Systemen

Für Anwender, die nicht auf Standardbaugruppen zurückgreifen wollen oder können, ist zur vollen Ausnutzung der Vorteile der Architektur des Mikroprozessors Z 80 die Kenntnis von Hardware-Details der peripheren Bausteine wichtig. In diesem Beitrag werden entsprechende Hinweise zu Fragen der Ein-/Ausgabe und der Interrupt-Behandlung zusammen mit den zugehörigen Programmieranleitungen gegeben.

Das System Z 80 wurde so ausgelegt, daß mit minimalem Schaltungsaufwand eine möglichst hohe Leistungsfähigkeit erreicht wird. Dies gilt besonders für die Interrupt-Behandlung, deren Flexibilität und Leistungsfähigkeit über das bei Mikrocomputern übliche hinausgeht und damit dem Z 80 Möglichkeiten eröffnet, die bisher teuren Prozeßrechnern vorbehalten waren.

Interrupt-Architektur der Z 80-Peripheriebausteine

Die Z 80-CPU verfügt über zwei grundsätzlich getrennte Interrupt-Behandlungsweisen, den „nichtmaskierbaren" und den „normalen" Interrupt.

Beim „nichtmaskierbaren" Interrupt gelangen die Interruptanforderungen über den hardwaremäßig separaten Eingang $\overline{\text{NMI}}$ an die CPU. Die Anfangsadresse der zugehörigen Interrupt-Bedienroutine liegt fest auf der Programmspeicheradresse 0066. H. Es muß hier angemerkt werden, daß beispielsweise das System 8080 A, auf dem das System Z 80 softwaremäßig aufbaut, nicht über einen solchen gesonderten Interrupt verfügt. Dieser nichtmaskierbare Interrupt wird vom Interrupt-Freigabe-Flipflop der Z 80-CPU nicht beeinflußt, d.h. eine Interrupt-Anforderung an diesem Pin läßt sich nicht durch einen *Interrupt-Disable*-Befehl per Programm sperren.

Sinn dieser Einrichtung ist es, Alarmfälle, wie z.B. kritische Bedingungen in einem Prozeß oder Stromausfall, die vom momentanen Zustand der CPU und des Anwenderprogramms unabhängig sind, in möglichst kurzer Zeit behandeln zu können. Das ist deshalb möglich, weil die Verzweigung auf die Interrupt-Bedienroutine hard- statt softwaremäßig realisiert ist. Dieser Interrupt hat grundsätzlich höchste Priorität.

Der „normale" Interrupt (Eingang $\overline{\text{INT}}$) kann durch die Befehle *Interrupt Disable* und *Interrupt Enable* per Anwenderprogramm gesperrt bzw. freigegeben werden. Hierbei sind das Verzweigen auf verschiedene Interruptbedienroutinen und das Priorisieren aller Interrupt-Anforderungen möglich. Aus Kompatibilitätsgründen zum System 8080 A verfügt die Z 80-CPU über folgende drei Betriebsarten *(Interrupt modes)*, die sich bei Programmbeginn durch den Befehl *Set Interrupt Mode* = IM mode (mit mode = 0, 1 oder 2) vorwählen lassen:

- Mode 0 entspricht genau der Interrupt-Behandlungsweise der 8080 A. Dabei sind in gängigen Hardwareimplementierungen max. 8 Interrupt-Bedienroutinen möglich, deren Startadressen im Bereich 0 bis 56 (dezimal) liegen, und die über eine „Restart"-Bedingung auf dem Datenbus angesprungen werden können. Die Priorisierung muß durch gesonderte Hardware erfolgen, z.B. mit Prioritäts-Enkoderbausteinen.
  Wegen seiner eingeschränkten Möglichkeiten wird man dieses Interrupt-Behandlungsverfahren in Z 80-Systemen nur dann verwenden, wenn bereits vorhandene 8080 A-Programme auf Z 80-Hardware unverändert ablaufen sollen.
- Mode 1 ermöglicht nur die Verwendung einer einzigen Interruptebene („Restart 7").
  Ähnlich wie Mode 0 ist auch Mode 1 zur Herstellung der Kompatibilität zum 8080 A gedacht; er entspricht der Verwendung des 8228 (mit 1 kΩ gegen +12 V am „INTA"-Stift) als Sender einer einzigen Restart-Adresse (0038 H).
- Mode 2 beinhaltet die für das System Z 80 charakteristische Betriebsart, die eine praktisch unbegrenzte Anzahl an Interruptebenen zuläßt, wobei die gesamte Priorisierung in den Z 80-Ein-/Ausgabebausteinen ohne zusätzlichen Schaltungsaufwand geschieht.

Der entsprechende Informationsfluß ist in Bild 1 schematisch dargestellt.

Vor dem erstmöglichen Auftreten eines Interrupts werden CPU und Peripherie initialisiert. Das geschieht üblicherweise zu Programmbeginn nach folgendem Schema, um undefinierte Verhältnisse beim Einschalten zu vermeiden:

```
DI    ; Sicherstellung definierter Interrupt-Verhältnisse
      ; es folgen sämtliche CPU- und I/O-Initialisierungen
IM 2  ; Interrupt-Mode = 2
EI    ; jetzt kann INT freigegeben werden.
```

Dabei ist zweierlei zu tun:

- In das I-Register der Z 80-CPU (8 bit) wird der höherwertige Teil einer 16-bit-Adresse (Interrupt-Service-Routinen-Zeiger) geladen.
- In das Vektor-Register jedes Z 80-Peripheriebausteins wird der gewünschte niederwertigere Teil einer 16-bit-Adresse (Interrupt-Service-Routinen-Zeiger) geladen.

Wie aus Bild 1 ersichtlich, sind die Interrupt-Anforderungsausgänge „$\overline{\text{INT}}$" sämtlicher Peripheriebausteine (Open-Drain-Ausgänge) miteinander und mit dem Interrupt-Anforderungseingang $\overline{\text{INT}}$ der Z 80-CPU verbunden. Aufgrund der internen Prioritätsfestlegung kann nur *der* $\overline{\text{INT}}$-Ausgang aktiv (Low) werden, dessen Tor die momentan höchste

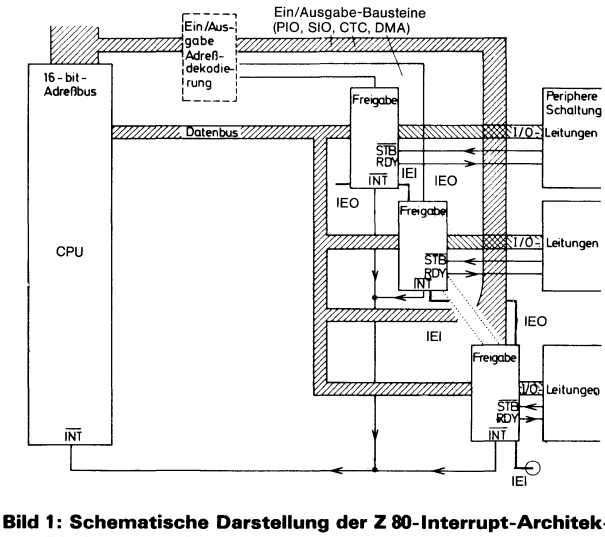


Bild 1: Schematische Darstellung der Z 80-Interrupt-Architektur. Die Interrupt-Anforderungsausgänge sämtlicher Peripheriebausteine sind miteinander und mit dem Interrupt-Anforderungseingang der CPU verbunden

Priorität hat. Es ist also eindeutig definiert, welches Tor eine Interrupt-Anforderung ausgestellt hat.
Dieses Port legt nun die in seinem anfangs geladenen Vektorregister gespeicherte untere Adressenhälfte auf den Datenbus.
Von der CPU wird dieser Wert gelesen, sobald die Interrupt-Anforderung der Peripherie eingetroffen und der momentan in Ausführung befindliche Befehl abgearbeitet ist.
Der zu Anfang im I-Register der Z 80-CPU abgespeicherte Wert wird nun von der CPU als höherwertiges Byte, der von der Peripherie her eingelesene Vektor als niederwertigeres Byte einer 16-bit-Adresse interpretiert, unter der die CPU die Anfangsadresse der der anfordernden Peripherie in eindeutiger Weise zugeordneten Interruptbedienroutine erwartet.

Auch diese Adreßinhalte werden vor Auftreten der Interrupt-Anforderung per Programm definiert.
Es wird also eine echte speicherindirekte Adressierung der Interrupt-Bedienroutinen mit Hilfe einer Sprungtabelle durchgeführt.
Dadurch lassen sich folgende Vorteile realisieren:

- Die Interrupt-Bedienroutinen können auf jeder beliebigen Speicheradresse beginnen.
- Es können beliebig viele verschiedene Interrupt-Bedienroutinen benutzt werden.
- Es ist eine dynamische Zuweisung zwischen Peripherie und Interrupt-Bedienroutinen durch programmgesteuerte Zuweisung der Zeiger an die Peripherie möglich.
- Da die Sprungtabelle wahlweise in einem Festwert- oder in einem Schreib-/Lesespeicher-Bereich abgelegt werden kann, ist es möglich, in einfachster Weise (nämlich durch Hochzählen eines Sprungzeigers) ein und derselben Peripherie im Verlauf des Programms verschiedene Bedienroutinen zuzuweisen.

Dabei ist darauf zu achten, daß bei der letztgenannten Möglichkeit die Änderung der Zuweisung durch Modifizieren der Sprungtabelle und bei dem davor genannten Punkt durch Neuladen des Interruptvektors in der Peripherie erfolgt.
Es muß jetzt noch klargestellt werden, wie Anforderungskonflikte der Ein-/Ausgabebausteine vermieden werden. Im System Z 80 geschieht dies durch eine sogenannte Hardware-Priorisierungskette *(Daisy chain priority interrupt)*. Sie ist bereits standardmäßig in jedem Z 80-Peripheriebaustein implementiert und bewirkt, daß das Port einen Interrupt anfordern kann, dessen Priorisierungseingang „High" ist (Bild 2). Sein Priorisierungsausgang (IEO) ist mit dem Priorisierungseingang des nächstniedriger zu priorisierenden Tores verbunden und immer dann „High", wenn gerade keine Interruptanforderung vorliegt, bzw. „Low", wenn dieses höher priorisierte Tor gerade mit der Behandlung eines Interrupts beschäftigt ist.
Der Priorisierungseingang des „ersten" Tores wird fest auf „High" gelegt. Da sein Priorisierungseingang niemals gesperrt werden kann, hat er grundsätzlich die höchste Priorität. Beinhaltet ein Z 80-Baustein mehrere Tore (Z 80-PIO, -CTC, -SIO), so ist die Priorität zur Einsparung von Stiften chipintern festgelegt.
Gegenüber einer softwaremäßigen Priorisierung hat diese Methode folgende Vorteile:

- Die gesamte Interrupt-Priorisierungs-Hardware verschwindet, da sie in den Ein-/Ausgabe-Bausteinen des Systems unterzubringen ist; ein „Interrupt-Controller" oder ein spezieller Priorisierungsbaustein wird überflüssig.
- Eine schnelle Interrupt-Bedienung ist möglich, da die höchste Priorität nicht erst von der CPU ermittelt werden muß.

Das hardwaremäßige Festlegen der Priorität wird manchmal als unangenehme Restriktion empfunden. Dazu ist zu bemerken, daß eine Änderung der Priorisierung während der Programmausführung nur in sehr seltenen Fällen wirklich erforderlich und im übrigen aus Zuverlässigkeitsgründen recht problematisch ist und daß im System Z 80 zusätzlich zur Interrupt-Freigabe der CPU auch die Peripheriebausteine per Programm für die Ausstellung von Interrupt-Anforderungen freigegeben und gesperrt werden können. Dadurch ist eine Umschichtung der Interrupt-Priorität der einzelnen Peripheriebausteine möglich.
An dieser Stelle muß angemerkt werden, daß die Anzahl der ohne Einfügen zusätzlicher „WAIT"-Zyklen und einem damit verbundenen Verlust an Verarbeitungsgeschwindigkeit kaskadierbaren Peripheriebausteine zunächst einmal durch die Laufzeiten in der Priorisierungskette eingeschränkt ist. In einem 2,5-MHz-System sollten zur Sicherstellung der einwandfreien Funktion max. 4 periphere Z 80-Standard-Bausteine kaskadiert werden. Ist eine höhere Anzahl von Bausteinen erforderlich, muß die in Bild 3 gezeigte Zusatzschaltung eingeführt werden, die aus der Addierwerktechnik als *„Look Ahead Carry Logic"* bekannt ist. Dabei wird einfach der Übertrag (bzw. in unserem Beispiel die Meldung, daß ein höherpriorisierter Kanal einen Interrupt angefordert hat) den folgenden Einheiten über parallel geschaltete Gatter zugeführt, ohne erst die gesamte Bausteinkette durchlaufen zu müssen.

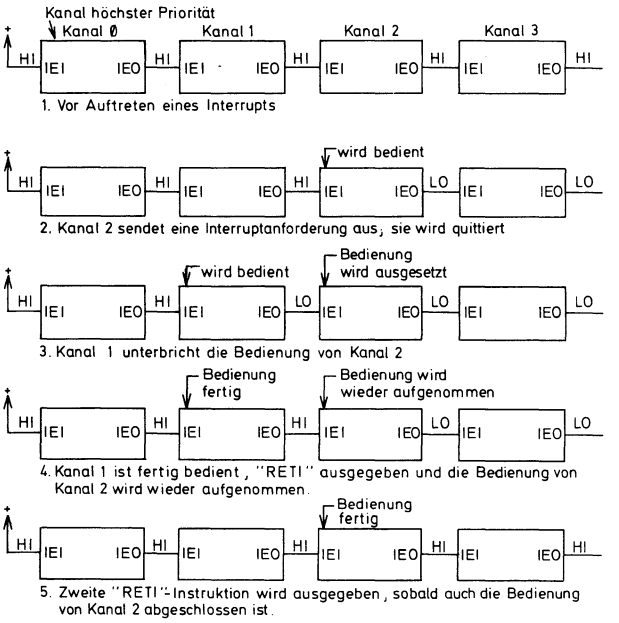


Bild 2: Beispiel für eine Interrupt-Priorisierung

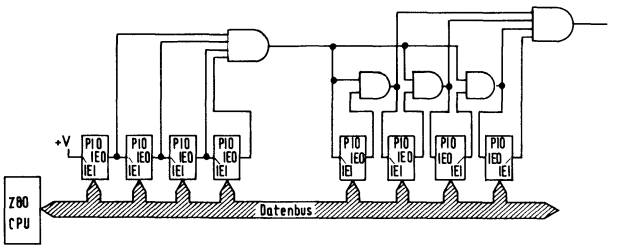


Bild 3: Beim Kaskadieren mehrerer Z 80-Standard-Bausteine wird eine Zusatzschaltung nach dem Look-Ahead-Carry-Prinzip nötig

Programmierung des PIO-Bausteins

Sämtliche Funktionen des Schaltkreises werden über Befehle des Z 80-Anwenderprogramms festgelegt [2, 3]. Zum Verständnis der Funktionsweise ist die Kenntnis der Systemarchitektur (Bilder 4 und 5) nötig.
Ein Z 80-PIO-Baustein besteht im wesentlichen aus zwei 8-bit-Toren, deren Funktion (Byte-Ausgabe, Byte-Eingabe, Byte-Ein-/Ausgabe, Bit-Ein-/Ausgabe) durch das Programm festgelegt wird, und je zwei Quittungsleitungen je Tor, die für die Interrupt-Abwicklung maßgeblich sind. Die beiden Tore werden automatisch über den Adreßeingang *Port A/Port B-Select* unterschieden, der im allgemeinen mit A_0 des Adreßbus verbunden wird.
Jedes Tor wird über zwei Adressen angesprochen. Die Unterscheidung dieser beiden Adressen geschieht automatisch durch den Anschluß

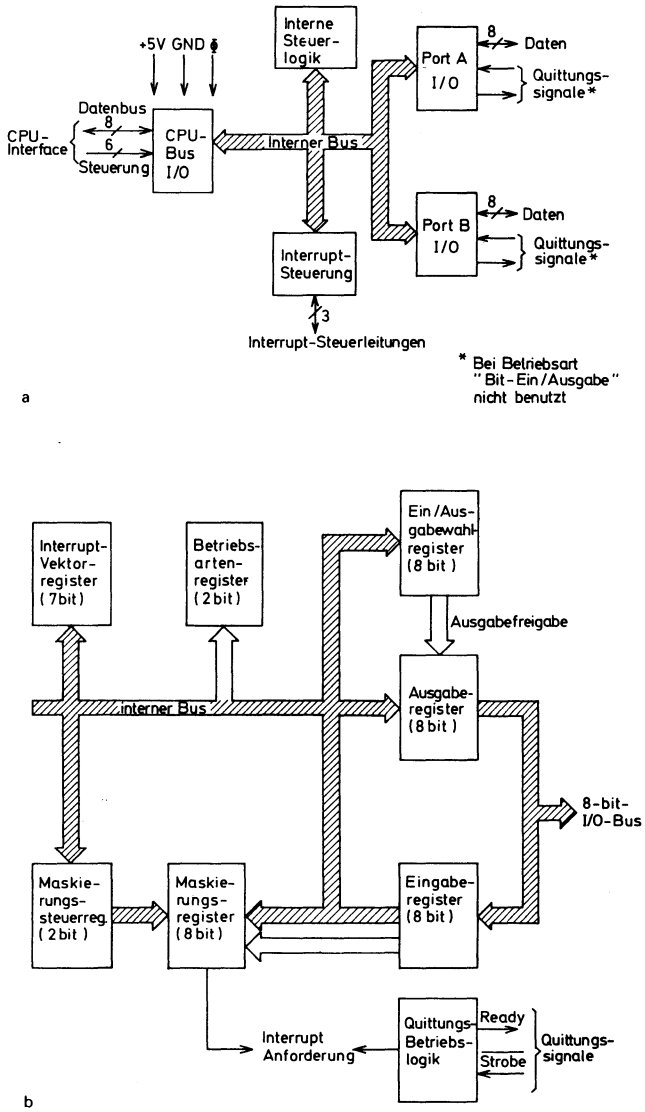


Bild 4: Blockschaltbild eines PIO-Bausteins (a) und die Schaltung eines Kanals (b)

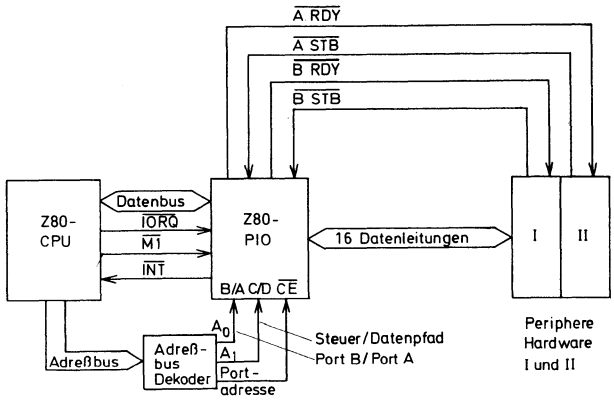


Bild 5: Adressierung der PIO-Kanäle

Control/Data-Select, der im allgemeinen mit A_1 des Adreßbus verbunden wird.
Der Eingang $\overline{CE}$ sperrt einen ganzen Baustein bzw. gibt ihn frei. Er wird im allgemeinen aus den niederwertigen 8 bit des Adreßbusses über einen Dekoder abgeleitet (während C/D-Select und A/B-Select direkt mit A_1 bzw. A_0 des Adreßbus verbunden sind) und repräsentiert sämtliche 4 Adressen des PIO-Bausteins, nämlich:

- Steuerwort-Pfad von Tor A (B/A-SEL = Low, C/D-SEL = Low)
- Daten-Pfad von Tor A (B/A-SEL = Low, C/D-SEL = High)
- Steuerwort-Pfad von Tor B (B/A-SEL = High, C/D-SEL Low)
- Daten-Pfad von Tor B (B/A-SEL = High, C/D-SEL = High)

Über die Steuerwort-Pfade erfolgt die Übergabe der Steuerworte an die Tore über die üblichen Z 80-Ausgabe-Befehle:

OUT (n), A
OUT (C), r
OUTI
OTIR
OUTD oder
OTDR.

Folgende Steuerwörter können oder müssen dem Z 80-PIO übergeben werden:

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

D_7	D_6	D_5	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	0

identifiziert das Steuerwort als Interrupt-Vektor

Auswahl der gewünschten Betriebsart
Hierfür wird das 2-bit-Betriebsartauswahl-Register über die höchstwertigen 2 bit (M 1 und M 0) eines Steuerworts, das von der CPU an das betreffende Tor ausgegeben wird, gesetzt.
Format dieses Steuerworts:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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

In der Betriebsart Byte-Ein-/Ausgabe kann das Tor B nur in der Betriebsart 3 betrieben werden, da alle 4 Handshake-Leitungen beider Tore für die Steuerung des bidirektionalen Datentransfers auf Tor A verwendet werden.

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 Tores 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 als Ausgang benutzt werden kann.

Format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Interrupt-Freigabe	UND/ ODER	High/ Low	nächstes Steuerwort ist Maske	0	1	1	1
	nur in Betriebsart 3 benutzt			bedeutet Interrupt-steuerwort			

Falls in diesem Steuerwort das Bit D_4 = „High" war (= „nächstes Steuerwort ist Maske"), muß ein weiteres Steuerwort zur Maskierung an das Port ausgegeben werden.
Sein Format ist:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
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.
Ein Programmierbeispiel zeigt Bild 6. Hieraus ist ersichtlich, daß die Port-Adressen (Steuerwort- und Datenpfad) bei Verwendung des residenten Z 80-Assemblers in symbolischer Form angegeben werden können; selbstverständlich müssen ihnen vorher über die Pseudobefehle EQU oder DEFB Hexadezimalwerte zugewiesen werden.

Programmierung des Z 80-CTC

Ebenso wie der Baustein Z 80-PIO wird auch der Z 80-CTC vom Anwenderprogramm gesteuert [4]. Es soll auch hier zunächst kurz auf die Hardwarearchitektur eingegangen werden (Bild 7).
Der Baustein verfügt über vier getrennte Zähler-/Zeitgeber-Kanäle. Anders als beim Z 80-PIO wird hier zwischen Steuerwörtern und

Laden der PIO-Register (1 Port, nämlich Port A. Eine Datenübertragung würde danach durch den Befehl OUT (PAD,A) vorgenommen).

```
LD A, VECT
OUT (PAC),A
LD A, MODE 3
OUT (PAC),A
LD A,I/O
OUT (PAC),A
LDA, INTEIV
OUT (PAC),A
LDA,MASK
OUT (PAC),A
```

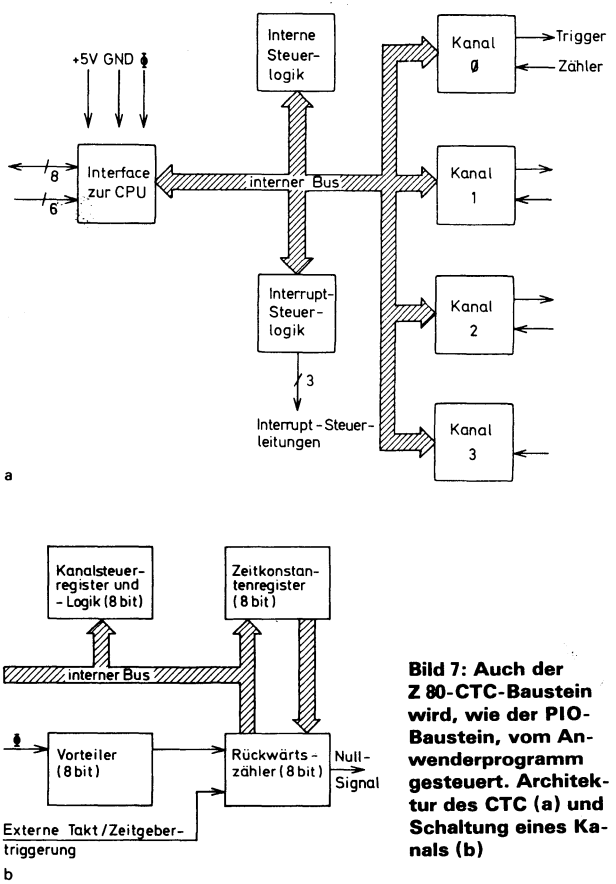


Bild 6: PIO-Programmierbeispiel

Bild 7: Auch der Z 80-CTC-Baustein wird, wie der PIO-Baustein, vom Anwenderprogramm gesteuert. Architektur des CTC (a) und Schaltung eines Kanals (b)

Datenwörtern zum Setzen der Zeitkonstantenregister nicht durch einen eigenen *„Control/Data"*-Anschlußstift unterschieden, sondern durch ein vorangehendes Steuerwort. Das bedeutet, daß der Baustein Z 80-CTC grundsätzlich ein Steuerwort erwartet, wenn er über $\overline{CE}$ freigegeben wird. Die Adressierung der vier verschiedenen Ports erfolgt über die beiden Anschlüsse *„Channel Select 0"* und *„Channel Select 1"*.
Grundsätzlich wird zwischen zwei verschiedenen Steuerworten unterschieden;

- Dem Steuerwort zur Betriebsartfestlegung, das durch eine „1" im niederwertigsten Bit gekennzeichnet ist.
  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.

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

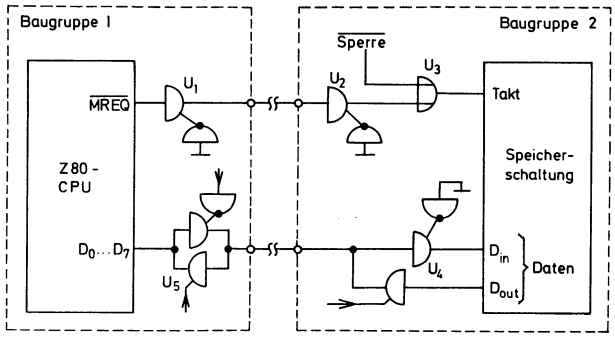


Bild 8: In komplexen Mikrocomputersystemen müssen die CPU-Busleitungen gepuffert werden. Das Bild zeigt das Prinzip der doppelten Pufferung

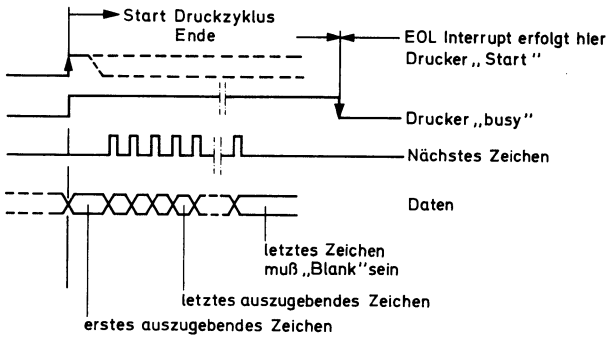


Bild 9: Zeitplan für den Anschluß des OEM-Druckers

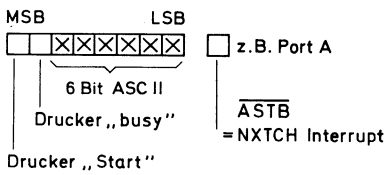


Bild 10: Verknüpfung zwischen Z 80-PIO und Drucker

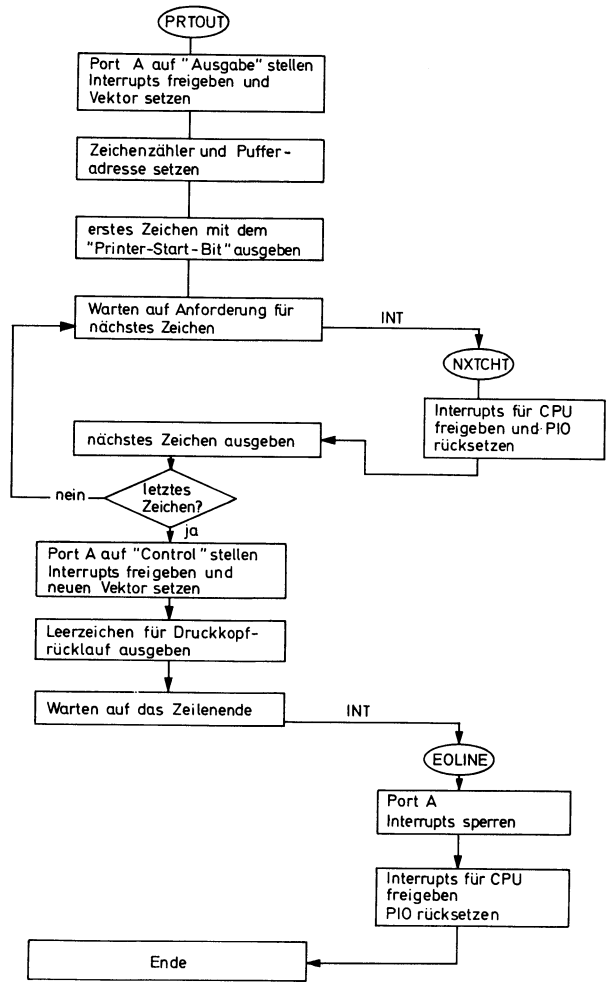


Bild 11: Flußdiagramm des Drucker-Steuerprogramms

- Dem Steuerwort zur Eingabe des Interrupt-Vektors, das vom Z 80-CTC durch eine „0" im niederwertigsten Bit erkannt wird.
  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 0 und Datenbit D_0 gleich Null vorgegeben werden.
  $D_7 \cdots D_3$ 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 Binärkode 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

Die Aussteuerung von Datenbus-Puffern

In Mikrocomputersystemen höherer Komplexität muß man die CPU-Busleitungen puffern, vor allem, wenn die Funktionseinheiten auf mehrere Baugruppen verteilt sind [1]. Im allgemeinen kommt man dann zu einer Konfiguration nach Bild 8.

Man sieht, daß hier — wie allgemein üblich — die Datenleitungen der auf der gleichen Baugruppe wie die CPU untergebrachten Speicher- und Ein-/Ausgabe-Bausteine direkt mit dem Datenbus verbunden sind.

Da der Datenbus bidirektional ist, muß dann den Datenbus-Pufferbausteinen gesagt werden, in welcher Richtung die Übertragung stattfindet; dies geschieht über das Steuersignal CPUIN. Damit beim Aufteilen der Gesamtschaltung über mehrere Karten keine Konflikte zwischen den Puffern auf den verschiedenen Karten entstehen, muß außerdem festgelegt werden, welcher Pufferbaustein überhaupt aktiv sein darf (die übrigen sind abzuschalten). Hierzu dient das Steuersignal $\overline{CS}$.
Da jeder der Pufferbausteine als Empfänger (von der Bus-Seite her gesehen) oder Sender arbeiten kann, können folgende Fälle des Datenflusses auftreten:

- Der Datentransfer findet innerhalb der Zentralbaugruppe statt.
- Der Datentransfer findet statt zwischen Zentraleinheit und einer der beiden Zusatzeinheiten.

Im ersten Fall ist die Behandlung einfach: Hier müssen lediglich die Datenbuspuffer auf der Zentralbaugruppe abgeschaltet werden, damit der „interne" Datentransfer nicht gestört wird.
Im zweiten Fall sind folgende Unterscheidungen zu treffen:

- Datentransfer zu Speicherbausteinen auf einer anderen Baugruppe
- Datentransfer von Speicherbausteinen auf einer anderen Baugruppe

- Datentransfer von Speicherbausteinen auf einer anderen Baugruppe zur CPU
- Datentransfer von Ein-/Ausgabebausteinen auf einer anderen Baugruppe zur CPU
- Interrupt-Behandlung

Mit Ausnahme des letztgenannten Falles müssen die Datenbuspuffer auf jeder beteiligten Baugruppe aktiviert und ihre Datenübertragungsrichtung entsprechend der Transferrichtung festgelegt werden.
Sind in einem System, wie dem anfangs geschilderten, mehrere Baugruppen, die durch Datenbuspuffer voneinander getrennt sind, berechtigt, einen Interrupt anzufordern, so muß der Datenfluß vom anfordernden Baustein zur CPU gewährleistet sein. Fordert ein auf der Zentralbaugruppe befindlicher Ein-/Ausgabebaustein einen Interrupt an, so muß der Datenbuspuffer auf der Zentralplatine gesperrt sein, damit der Interrupt-Vektor ungestört übertragen werden kann.
Fordert dagegen ein Ein-/Ausgabebaustein auf einer anderen Baugruppe einen Interrupt an, so muß zum Zeitpunkt INTA (Koinzidenz von $\overline{M1}$ und $\overline{IORQ}$) die Übertragung des Vektors von dort über den Datenbuspuffer der Zentralbaugruppe möglich sein.
Dazu ist es notwendig, nur *die* Datenbuspuffer zu aktivieren, die auf dem Weg zwischen der den Interrupt auslösenden Baugruppe und der CPU liegen, und die Richtung dieser Puffer *so* festzulegen, daß die Datenübertragung zur CPU *hin* erfolgt.
Die hierbei notwendige Unterscheidung, welcher Ein-/Ausgabebaustein einen Interrupt angefordert hat, ist durch dessen Interrupt-Enable-In- und Interrupt-Enable-Out-Leitung möglich: Nur *der* Baustein, an dessen Interrupt-Enable-in-Leitung eine „logische" 1 *und* an dessen Interrupt-Enable-Out-Leitung eine „logische" 0 liegt, kann einen Interrupt angefordert haben.
Daraus ist ersichtlich, daß die Verknüpfung mehrerer Steuerungssignale nötig ist, um die Datenbuspuffer richtig zu steuern.
Die Verknüpfung der einzelnen Signale läßt sich nach den folgenden Formeln mit Standard-Gatter-Bausteinen — oder besser mit einem einzigen 1-Kbit-PROM — durchführen. Dabei erhöht sich durch die Festlegung der Übertragungsrichtung die im System erforderliche Bausteinstückzahl nicht, da die Freigabe der Pufferbausteine in jedem Fall eine Standard-Steuerlogik je Baugruppe erfordert.
Die Verknüpfung wird nach folgenden Regeln vorgenommen:

Datenbuspuffer auf der CPU-Karte

$$\text{CPUIN} = \underbrace{(\overline{IMD} \wedge \overline{RD})}_{\text{EXT MEM READ}} \vee \underbrace{(\overline{IORQ} \wedge \overline{M1})}_{\text{INTA}} \vee \underbrace{(\overline{IORQ} \wedge \text{IIDEC} \wedge \overline{RD})}_{\text{EXT I/OREAD}}$$

$$\overline{CS} = \underbrace{(\overline{IORQ} \wedge \overline{M1} \wedge \text{IIEI} \wedge \text{IIEO})}_{\text{EXT INTA}} \vee \underbrace{\overline{IORQ} \vee \overline{\text{IIODEC}}}_{\text{EXT MEM TRANSF}}$$

Dabei sind:

CPUIN	:	Richtungssteuersignal (High: Richtung zur CPU)
$\overline{CS}$	:	Aktivierungssignal (High: Puffer gesperrt)
$\overline{M1}$	:	Maschinenzyklus 1
$\overline{IORQ}$	:	I/O-Request
$\overline{\text{IIODEC}}$	:	ECB/C Internal I/O-Anforderung
$\overline{RD}$	:	Read
IIEO	:	ECB/C/Internal Interrupt Enable Out
IIEI	:	ECB/C Internal Interrupt Enable In
$\overline{IMD}$	:	ECB/C Intern Speicheranforderung

Datenbuspuffer auf weiteren I/O-Karten

$$\overline{\text{IOIN}} = \underbrace{(\overline{IORQ} \wedge \overline{IODEC} \wedge \overline{RD})}_{\text{IO WRITE}} \vee \underbrace{(\overline{IORQ} \wedge \overline{M1} \wedge \text{IEI} \wedge \text{IEO})}_{\text{INTA}}$$

$$\overline{CS} = 0$$

Dabei sind:

IOIN	:	Richtungssteuersignal (High = zur betreffenden Karte hin)
$\overline{CS}$	:	Aktivierungssignal (High = Puffer gesperrt)
$\overline{M1}$	:	Maschinenzyklus 1
$\overline{IORQ}$	:	I/O-Anforderung
$\overline{IODEC}$	:	I/O-Auswahl
$\overline{RD}$	:	Read
IEI	:	Interrupt Enable IN
IEO	:	Interrupt Enable OUT

Anschluß eines OEM-Druckers an den Z 80-ECB/C-Computer

Der OEM-Drucker 5010 von Kontron ist ein Streifendrucker auf Metallpapierbasis. Er kann auf 50 mm breitem Papier bis zu 30 ASCII-Zeichen je Zeile bei einer Geschwindigkeit von max. 2 Zeilen/s drucken und eignet sich zum Einsatz in Mikrocomputer-Anwendersystemen, in denen kleinere Datenmengen zu dokumentieren sind.

Schaltung der Hardwareanpassung

Zum Anschluß des Druckers genügt ein halber Baustein Z 80-PIO. Der Druckvorgang beim 5010 läuft nach dem Zeitplan in Bild 9 ab.
Der Anschluß der einzelnen Druckerleitungen an den Mikrocomputer läßt sich dementsprechend in einfacher Form mit dem Z 80-PIO nach dem in Bild 10 angegebenen Schema durchführen.

Zugehöriges Steuerprogramm (Handling Routine)

Ein Flußdiagramm des Steuerprogramms zeigt Bild 11. Es enthält folgende Hauptaktivitäten:

- Initialisierung des PIO *(Output-Mode)*
- Übertragung des ersten Zeichens
- Warten auf das nächste Zeichen (Interrupt NXTCH)
- wird wiederholt, bis Zeile zu Ende ist
  Nach Ausgabe des letzten Zeichens Bereitschaft für „Neue Zeile" (Umschalten des Interrupts auf EOL-Signal).

Hierbei wird von zwei wichtigen Fähigkeiten des Systems Z 80 Gebrauch gemacht:

- Die „Betriebsart 3" der Z 80-PIO *(Control Mode)*, die die vom Anwenderprogramm kontrollierte Festlegung einzelner Leitungen eines Ports als Ein- oder Ausgabeleitungen erlaubt (wobei unter den Eingabeleitungen zusätzlich festgelegt werden kann, welche einen Interrupt auslösen sollen), in Verbindung mit der Möglichkeit, die Datenübertragungsart (Ein-, Ausgabe, bidirektional, Control) während des Programmablaufs umzudefinieren.
- Die Interrupt-Architektur des Systems Z 80, die es erlaubt, ein und dasselbe Port von verschiedenen Interrupt-Bedienroutinen behandeln zu lassen. Die dadurch in unserem Beispiel erzielte Hardware-Ersparnis ist evident; man kommt ohne ein oder mehrere Ports zur Übergabe der Drucker-Steuersignale aus.

Aufruf von Interrupt-Service-Routinen über Schaltkontakte

Der Anwender sieht sich häufig mit folgenden Problemen konfrontiert:

- Es gibt *n* Schalter (Einfach-Kontakte), deren Betätigung jeweils einen Interrupt bzw. den Sprung zu einer Interrupt-Service-Routine auslöst.
- Alle diese Schalter sind in vorgegebener Weise zu priorisieren.
- Jedem Schalter ist in eindeutiger Weise eine Interrupt-Service-Routine zugeordnet, deren Anfangsadresse an einer beliebigen Stelle im max. 64 KByte großen Programmspeicher steht.

Dieses Problem läßt sich mit einem Baustein Z 80-CTC je 4 Schaltern/Interruptservice-Routinen ohne weitere Logik lösen. Jeder Eingang CLK/TRG ist mit einem der Schalter verbunden (Low- oder High-Aktiv). Das Entprellen des Schalters wird hardwaremäßig über ein Flipflop oder softwaremäßig über eine entsprechende Warteschleife nach Bearbeitung des Interrupts durchgeführt.
Jedem der 4 Kanäle ist eine Speicheradresse zugeordnet, in der jeweils die Anfangsadresse der zum Interrupt zugehörigen Service-Routine steht. Die erste dieser 4 Adressen wird mit einem Steuerwort (letztes Bit = „1") gewöhnlich zu Programmbeginn von der CPU in den CTC geladen.
Die 4 Kanäle eines jeden CTC sind untereinander hardwaremäßig priorisiert. Eine Erweiterung auf mehr als 4 Kanäle ist durch Kaskadierung mehrerer Bausteine über deren *„Daisy Chain"* (IEI/IEO) möglich.

Eine Änderung dieser Priorisierung ist jederzeit durch die Möglichkeit gegeben, die Interrupt-Meldung der Bausteine individuell zu sperren, wobei trotzdem die Interrupt-Meldung der Peripherie nicht verloren geht, sondern so lange gespeichert bleibt, bis sie durch die CPU bearbeitet wird.
Es muß selbstverständlich sichergestellt sein (z.B. durch Quittungslogik), daß inzwischen keine neue Interruptanforderung erfolgt.
So würde beispielsweise der Kanal hardwaremäßig niedrigster Priorität die höchste Priorität bekommen, indem per Programm alle übrigen Kanäle vorübergehend gesperrt werden.
Zu Programmbeginn wird jedem Kanal ein Steuerwort übermittelt, das ihn in Betriebsart „Ereigniszähler" bringt und seinen Rückwärtszähler über das Zeitkonstantenregister auf den Wert „1" setzt (vorher ist ein Steuerwort mit Bit 2 = 1 auszugeben).

Die Betätigung eines der Schalter löst nun ein Dekrementieren des Rückwärtszählers des CTC aus, der dadurch den Wert 0 erreicht und einen Interrupt auslöst, soweit er nicht softwaremäßig gesperrt ist. Die Programmausführung wird nun nach Abarbeitung des momentan in Ausführung befindlichen Befehls unterbrochen und bei der Anfangsadresse der Interrupt-Service-Routine fortgesetzt. Diese ist gleich dem Inhalt der Arbeitsspeicheradresse, die sich aus Inhalt des CTC-Interrupt-Vektorregisters als niederwertiges Byte und dem Inhalt des Interrupt-(I-)Registers der CPU als höherwertiges Byte ergibt. Der Rücksprung ins Hintergrundprogramm erfolgt in üblicher Weise mit einer RETI-Anweisung. Vorher muß jedoch dafür gesorgt werden, daß das CTC-Zeitkonstantenregister wieder auf „1" gesetzt wird.
Falls eine Priorisierung der einzelnen interruptanfordernden Schalter bereits in der Peripherie mit einer *Daisy-Chain*-ähnlichen oder sonstigen Hardware durchgeführt wurde, kann die Auslösung der Interrupts über einen oder mehrere PIO-Bausteine erfolgen.
Die betreffenden PIO-Ports arbeiten dabei in der Betriebsart „Bit-Ein-/-Ausgabe" *(Control Mode)*, wobei über Steuerworte die Möglichkeit der Interruptanforderung an jede einzelne Leitung freigegeben oder gesperrt werden kann; darüber hinaus sind Bedingungen zum Erzeugen von Interrupts aufgrund logischer Verknüpfungen programmierbar [2, 3].
Der Vorteil dieses Verfahrens gegenüber der Verwendung eines CTC ist, daß mit einem einzigen PIO-Baustein bis 16 Interruptbedingungen verknüpft, maskiert und geprüft werden können. Nachteilig kann hierbei sein, daß man den einzelnen Schaltern die Einsprungadressen der Bedienroutinen über ein Verzweigungsprogramm zuordnen muß, da ja hardwaremäßig pro 8 bit (= 1 PIO-Kanal) zunächst nur eine einzige Bedien-Routinenadresse zur Verfügung steht.

Schrifttum

[1] Blomeyer, P.: Anwendung von Standard-Speicherbausteinen in Mikrocomputersystemen, Elektronik 3 (1977), S. 75–82.
[2] o.V.: Z 80-PIO-Produktspezifikation in deutscher Sprache, herausgegeben von Kontron Elektronik GmbH Eching/München.
[3] o.V.: Z 80-PIO-Technical Manual, Zilog-Druckschrift, zu beziehen über Kontron Elektronik GmbH, Eching/München.
[4] o.V.: Z 80-CTC-Produktspezifikation in deutscher Sprache, herausgegeben von Kontron Elektronik GmbH, Eching/München.
[5] o.V.: Zilog Z 80-Assembly-Language Programming Manual, Zilog-Druckschrift, zu beziehen über Kontron Elektronik GmbH, Eching/München.

Stephan Bauer und Hans Peter Blomeyer-Bartenstein

Mikrocomputersystem mit eingebauter Tastatur und Billig-Anzeige

Vorbemerkung

Die wachsende Verbreitung der Mikrocomputertechnik macht ein billiges Hilfsmittel erforderlich, das

- ☐ die autodidaktische Ausbildung von Mikrocomputer-Anfängern durch praktisches Arbeiten mit Mikroprozessoren ermöglicht. Dabei ist es unerheblich, ob diese Arbeit aufgrund persönlicher Initiative des Einzelnen (etwa als Hobby) betrieben wird, oder im Rahmen des Ausbildungsprogramms eines Arbeitgebers oder sogar einer Studienarbeit finanziert wird,
- ☐ sich als gut vorbereitetes oder sogar voll funktionsfähiges und modular erweiterbares Anwendersystem benützen läßt.

Für den Drittgenerations-Mikroprozessor Z80 der amerikanischen Firma ZILOG (Cupertino/Ca und Sunnyvale/Ca) ist nun ein solches Hilfsmittel unter der Bezeichnung Z80-KIT als Bausatz (Einzelstückpreis: DM 799,—) und in zusammengebauter, ausgetesteter Form unter der Bezeichnung Z80-KIT/Z (Einzelstückpreis: DM 1346,—) lagermäßig erhältlich. (Hersteller: KONTRON Elektronik-GmbH, Eching/München).
Mitgeliefert wird mit dem KIT eine ausführliche Zusammenbauanleitung wahlweise in deutscher oder englischer Sprache, ein Programmierhandbuch und ein Programmier-Einsteckkärtchen.
Darüber hinaus (nicht im Preis inbegriffen) ist ein grundlegendes Einführungsbuch in die Mikrocomputerentwicklung („Mikrocomputertechnik" von P. Blomeyer, ca. 200 Seiten) zu einem Preis von ca. DM 38,— durch die KONTRON-Elektronik-GmbH zu beziehen.

Dieser 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. Das zugehörige Blockschaltbild zeigt Bild 1.
Zur Inbetriebnahme zusätzlich notwendig ist lediglich eine 5 V Stromversorgung (2 A, ± 5% Stabilität).

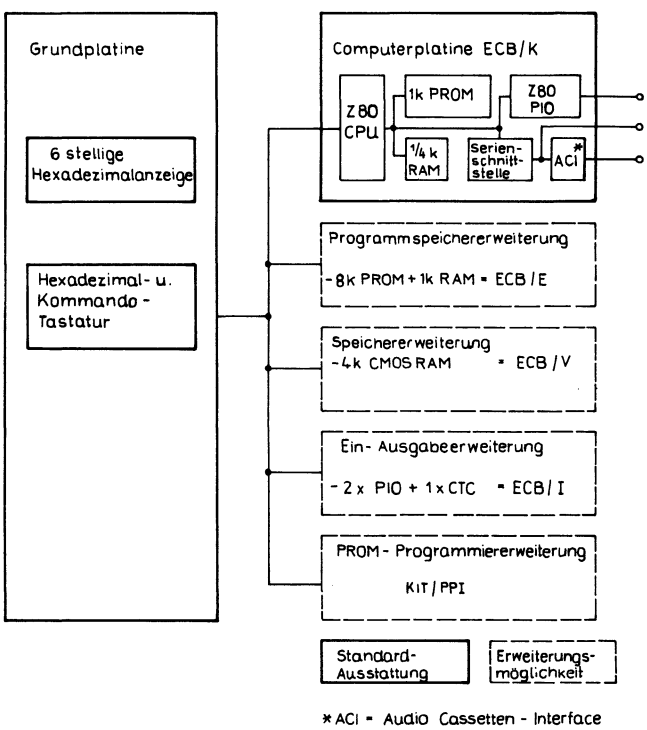
Mit dem Z80 KIT wurde ein Mikrocomputersystem geschaffen, das in seiner Grundform keinerlei periphere Geräte zu Bedienungszwecken benötigt und dadurch die Verwendung eines funktionsfähigen Mikrocomputers bei minimalen Kosten erlaubt.

Basis dieses Mikrocomputers ist die Grundplatine, auf dem die Bedienungstastatur, 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.

Die serienmäßige Grundausstattung (Lieferumfang des Z80 KIT) umfaßt folgende Systemkomponenten:

Grundplatine

- ☐ Eingabetastatur: 26 Tasten zur Bedienung des KIT
- ☐ Anzeigeeinheit
  - ● 4 hexadezimale Anzeigen für Adressen und
  - ● 2 hexadezimale Anzeigen für Daten



* ACI = Audio Cassetten - Interface

Einplatinencomputer

☐ Mikroprozessor	1 × Z80 CPU/PS
☐ Taktfrequenz	2,5 MHz
☐ Programmspeicher	Z80-KIT-Betriebssoftware (= 1 KByte)
☐ 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

Erweiterungsmöglichkeiten

Ohne Veränderung des bestehenden KIT ist eine Erweiterung durch Zusatzplatinen möglich. Angeboten werden folgende Erweiterungspakete:

☐ Z80-KIT/PPI — **P**rogrammier- und **P**eripherie-**I**nterface (PROM-Programmierbaugruppe)

☐ sämtliche Baugruppen der ECB-Serie:
- ECB/E (8 kPROM + 1 k stat. RAM)
- ECB/I (4 Achtbit-Parallel-Ein/Ausgabe-Ports + 4 Zähler/Zeitgeber-Kanäle)
- ECB/V (4 kByte CMOS-RAM, nicht-flüchtig, mit Speicherschutzschalter)

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.

Erläuterungen zur Arbeitsweise

Der ECB/K-Einplatinencomputer

Schaltungsbeschreibung

Die Baugruppe umfaßt folgende Funktionseinheiten
☐ Z80-CPU
☐ Decoder und Puffer für Speicher- und I/O-Ausbau
☐ 1 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 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.

CPU

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.

Takterzeugung

Aus Stabilitäts- und Kostengründen wird der Systemtakt über einen 9,8304 MHz-Quarz erzeugt, der zusammen mit ⅓ 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 2Φ -Signal abgeleitet und auf den Bus herausgeführt wird. Dieses Signal wird häufig in dynamischen RAM-Speichersystemen benötigt (siehe entsprechende Applikationsschriften!)

Programmspeicher

Umschaltung auf verschiedene Typen.
Das Monitorprogramm ist in zwei ½ kByte großen Speicherbausteinen untergebracht. Anstelle dieser Bausteine können im Bedarfsfall Typen ähnlicher Bauart eingesetzt werden:

a)	HARRIS HM 7641	512×8 bit organisiert bipolar
b)	HARRIS HM 7681	1 k×8 bit organisiert bipolar
c)	2708	1 k×8 bit organisiert EPROM
d)	2716	2 k×8 bit organisiert EPROM.

Im vorliegenden KIT ist die Anpassung an die eingesetzten Programmspeicher durchgeführt. Sollen andere, unter a, b, c oder d genannte Bausteine eingesetzt werden, sind teilweise geringfügige Änderungen (Jumper) vorzunehmen.

Schreib/Lese-Speicher

Der 256 Byte große Schreib/Lese-Speicher besteht aus 2 statischen 2112/A-RAM-Bausteinen, die 256 × 4-bitweise organisiert sind.

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.

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.

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 besteht aus einer einfachen Modulations/Demodulationsschaltung. 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.

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.

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 den 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 RS232 herausgeführt, wobei allerdings die Normpegel nicht im vollen Umfang eingehalten werden. Dies hat jedoch für Datenübertragungen auf Strecken < 20 m praktisch keine Bedeutung. Analog ist auch der Senderausgang des USART 8251 über einen TTL-Puffer und einen PNP-Transistor geschaltet, wobei der Ausgang des TTL-Puffers wieder über eine separate Leitung als RS 232 ähnliche Schnittstelle auf den Serieninterface-Steckverbinder herausgeführt ist. Der PNP-Transistor prägt den Strom der Datenübertragungs-Stromschleife.
Die restlichen Ausgänge des 8251 sind ungepuffert auf den Serienstecker herausgeführt.

Takterzeugung für ACI und TTY

Die für die Serienschnittstelle nötige Bezugsfrequenz wird durch eine, aus drei Teilerbausteinen bestehende Kette erzeugt. Um ein möglichst großes Spektrum von Übertragungsraten zu ermöglichen, können durch entsprechend gesetzte Jumper verschiedene Teilverhältnisse für die Systemfrequenz erzeugt werden. Mit Jumpern wird eine Übertragung mit 110 Baud (= 110 Hz) erreicht, was sowohl für die Cassettenspeicherung (ACI), als auch für einen Fernschreiber (TTY) passende Bedingungen bietet.

Ein/Ausgabe

Die Ein/Ausgabe-Bausteine auf dem Z80-ECB/K sind im 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).

Die Bedienungs- und Anzeigeeinheit

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.

Man definiert 2 Tasten-Puffer von je 4 Wörtern: Tastenneu-Puffer TNP und Tastenalt-Puffer TAP. Am Anfang werden alle Puffer gelöscht. Danach wird eine Programmschleife dadurch eingeleitet, daß die nicht mehr aktuelle TNP in den TAP transferiert wird.

Anschließend ist TNP zu reaktualisieren.

Man untersucht jetzt die beiden Tasten-Puffer auf „Neuzugänge", indem man gleiche Bitpositionen in beiden Puffern miteinander vergleicht. Damit die Tasten voneinander programmtechnisch leicht unterscheidbar sind, ordnet man jeder Taste einen eigenen Tastencode zu. Dieser setzt sich aus einer Zeilen- und einer Spaltenklemmung zusammen. Vorausgesetzt, daß nicht mehrere Tasten gleichzeitig gedrückt sind, sucht eine Routine die „Neuzugänge", ermittelt den Tastencode und legt ihn im „Tastenregister" ab.

Wiederholt man diese Routine zyklisch für immer neue TNP, so gewinnt man in dem Tastenregister alle Tastencodes in der Reihenfolge, in der die Tasten gedrückt worden waren. Nun können die eingegebenen Werte (= Tastencodes) weiterverarbeitet werden.

Ein Tastenfeld von max. 32 Tasten ist in 4 Reihen und 8 Spalten angeordnet. Die Reihen (= Eingänge) sind durch im Decoder Da decodierte 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.

Die Anzeigeneinheiten

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.
Den anzuzeigenden 6stelligen Zahlenwert speichert man zunächst in binärer Form. Die Umwandlung auf 7-Segment-Darstellung erfolgt über den Zugriff auf die 7-Segment-Tabelle SEGM, die zu jeder Zeile das entsprechende 7-Segment-Äquivalent enthält. Die Port-Adressen werden über die „PORT-Liste" angesprochen.

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, während das Signal LWR (LED's Write) aktiv ist. Dabei ist LWR = (0C ∨ 0D ∨ 0E ∨ 0F ∨ 1C ∨ 1D) ∧ WR ∧ IORQ. Die Helligkeit der LED's läßt sich über die Zeitkonstante τ beeinflussen. Das Zeitglied erzeugt mit der Konstante τ ein WAIT-Signal, das die Dauer des LWR-Signals entsprechend dehnt.

Bedienung des Z80-KIT

Allgemeines

Das Betriebsprogramm, das als Firmware in den beiden Programmspeichern A2 und A3 untergebracht ist, ermöglicht Ihnen die Inbetriebnahme Ihres Computers. Es liest, decodiert und verarbeitet die über die Tastatur eingetasteten Kommandos und Daten und ist für die Ansteuerung der Anzeigeelemente verantwortlich. Dadurch ist es möglich, die im folgenden beschriebenen Aktivitäten des Computers auszulösen.
Eingaben des Benutzers, die den Anforderungen der Kommandosprachen genügen, werden durch die gewünschten Aktionen quittiert. Bei Nichteinhaltung dieser Anforderungen leuchtet die Lampe ERROR auf, ohne daß das System irgendeine Aktion ausführt. Die Lampe ERROR erlischt bei der nächsten korrekten Eingabe.
Die Eingabe von numerischen Parametern erfolgt im hexadezimalen Format, d.h. durch die Dezimalziffern und die zusätzlichen Hex-Ziffern A, B, C, D, E und F.

Beschreibung des Tastenfeldes

Das Tastenfeld umfaßt 29 Tasten, die entsprechend ihrer Funktion in verschiedenen Farben gehalten sind:

— graue Tasten: DATEN-TASTEN
mit ihnen können Daten oder Adressen in hexadezimaler Form angegeben werden. Gleichzeitig ist mittels der Tasten mit Doppelbedeutung die Adressierung bestimmter CPU-Register möglich

— weiße Tasten: FUNKTIONSTASTEN
mit ihrer Hilfe werden Kommandos gegeben, die bestimmte Aktivitäten einleiten

— orange Tasten: FUNKTIONSTASTEN MIT BESONDERER BEDEUTUNG. Farbe, Lage und Größe dieser Tasten wurde zur Vermeidung von Fehlbedienungen in vorliegender Weise gewählt.

Kommandotastatur

RESET — Die Rücksetzung (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 betätigen.

EX — Ausführungskommando (EXECUTE)
veranlaßt das Mikrocomputersystem zu Ausführung der zuvor eingetippten Anweisung.
Es läßt das Bedienprinzip deutlich werden, daß in der gesamten Computertechnik üblich ist:
- ☐ die Eingabe der gewünschten Anweisung und
- ☐ die Nachricht an den Computer, daß das „Eingetippte" auch wirklich vom Computer ausgeführt werden soll.

Durch diese Zweiteilung wird der Computer davor geschützt, falsche Aktivitäten auszuführen, die durch Tippfehler des Anwenders zustande kommen. Die Bedienperson hat stattdessen die Möglichkeit, versehentlich gegebene Anweisungen durch Drücken einer Rücknahme-(„Storno"-, s. dort)-Taste zu annullieren.

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 Hingabemodus für die Eingabe eines Anwenderprogramms

IDM Ausgabekommando (INCREMENT AND DISPLAY MEMORY)
versetzt den Mikrocomputer in den Ausgabemodus für die Ausgabe zusammenhängender Speicherbereiche

STORN Stornokommando
dient zum Löschen falscher, oder irrtümlich eingegebener Werte und Anweisungen. Kann gegeben werden **bevor** die EX Taste gedrückt wurde. Danach ist das **gesamte** Kommando erneut einzutasten.

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

Programmbeispiele

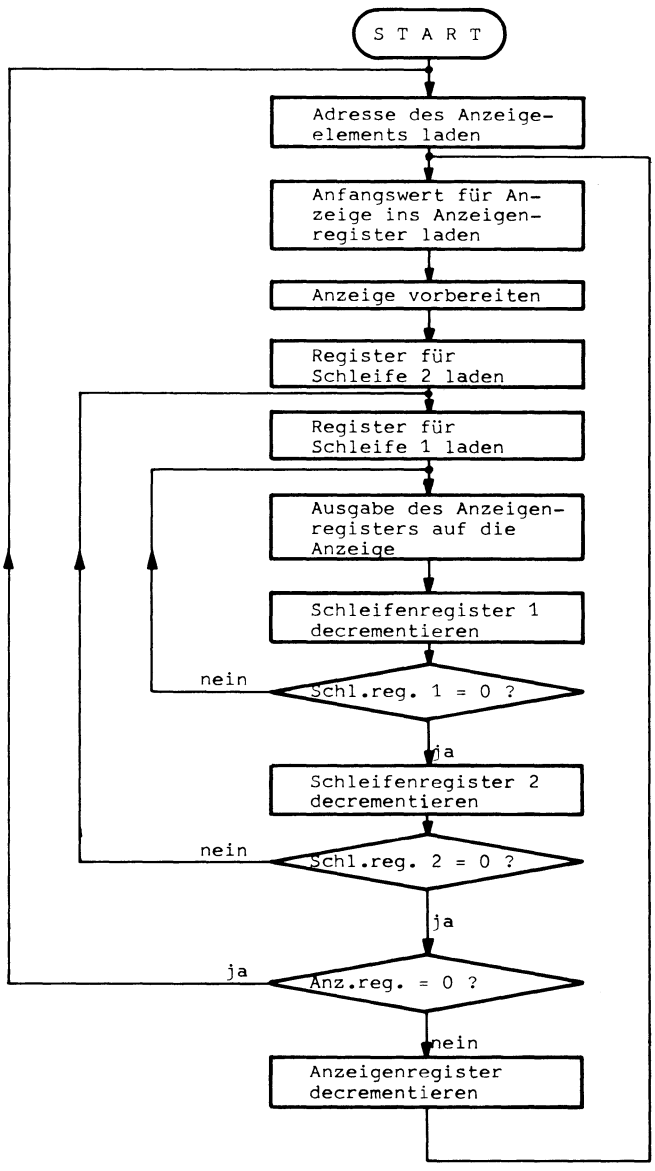
Um einen ersten Einstieg für den Gebrauch von USER-Programmen auf dem KIT zu geben, seien an dieser Stelle einige Programmbeispiele aufgeführt.

Für jedes Anwenderprogramm ist vor dem Starten ein definierter Befehlszählerstand (= Startadresse) anzugeben. Wird der Stapelspeicher (Stack) benutzt, ist auch die Angabe eines Stack-Pointers notwendig. Der vom Anwenderprogramm aufgebaute Stack darf sich dabei nicht mit dem vom Monitorprogramm benutzten Stack überdecken, sondern es sind zwei verschiedene Stapelspeicher notwendig! Dies wird erreicht, indem vor dem Programmstart ein entsprechender Wert eingegeben wird (siehe Beispiele, SP auf 3CDD).

Prüfprogramm für den Z80-KIT

Um eine Möglichkeit zu schaffen, den KIT ohne eigenes und evtl. fehlerbehaftetes Userprogramm zu testen, sei an dieser Stelle ein kleines Programmbeispiel gegeben.
Das Programm besteht aus zwei ineinander verschachtelten Zählschleifen, an deren Enden eine Ausgaberoutine für eines der Hexadezimalen Anzeigeelemente steht. Ausgegeben wird der Inhalt von Registerpaar DE, der nach jedem Durchgang decrementiert wird.
Beginnend mit F werden so alle hexadezimalen Elemente in absteigender Folge nacheinander angezeigt. Nach 0 folgt wieder F, so daß sich eine unendliche Folge ergibt.

Programmablaufplan



Programmerläuterung

Wichtigster Teil des Testprogramms ist die Ausgaberoutine, die es ermöglicht, bestimmte Werte auf der Anzeige darzustellen. Diese Routine, die auch vom Monitorprogramm benutzt wird, basiert auf folgendem Gedanken:

Ein anzuzeigender Wert (hexadezimal!) wird in das DE Registerpaar eingelesen, wobei das D Register stets nur 0 enthalten darf. Das DE Registerpaar wird nun zum HL Registerpaar addiert (deshalb war es notwendig, das DE Registerpaar und nicht nur das E Register zu verwenden), welches die Anfangsadresse einer Tabelle (SEGM) enthält. Nach der Addition enthält das HL Register eine neue Adresse und zwar den Wert:

Anfangsadresse + Inhalt DE Registerpaar

Unter dieser Adresse ist **der** Wert abgespeichert, der bei Ausgabe auf die Anzeigeeinheit den im DE Registerpaar abgespeicherten Wert darstellt.

z.B.: Ist in DE der Wert 8 gespeichert, so ergibt sich:

	D	E
DE =	00	08

Im HL Register steht die Anfangsadresse der Tabelle SEGM:

	H	L
HL =	3C	72

Nach der Addition (hexadezimal) ergibt sich:

	H	L
HL =	3C	7A

Unter dieser Adresse findet man den Wert

3C 7A = 7F

Bei Ausgabe dieses Wertes zeigt das Anzeigeelement den Wert 8.

Die Ansteuerung der einzelnen Anzeigeelemente geschieht mit Hilfe des Port-Registers C und einer zugehörigen Port-Tabelle, die dem Listing des KIT-Programms entnommen werden kann. Soll die Ausgabe z.B. auf das obere Datenanzeigeelement erfolgen, ist C nicht mit 0C sondern mit 0D zu laden.

Die Zeitkonstanten sind jeweils im B Register untergebracht. Nach dem erstmaligen Laden mit dem Wert für Schleife 2 (00F), wird der Inhalt des B Registers in den Stack ausgelagert (und damit „konserviert"). B wird erneut geladen und zwar mit dem Wert für Schleife 1 (0FF). Wird Wert 1 wieder gebraucht, so wird er aus dem Stack geholt.

Durch Veränderung der beiden Werte kann ein verschieden schneller Durchlauf der Anzeige erreicht werden.

Z.B.: Lädt man statt 0FF den Wert 00F, so ergibt sich ein wesentlich schnellerer Durchlauf.

Eingabe und Starten des Programms

Setzen Sie den Stack Pointer (SP) auf 3CDD, anschließend den Befehlszähler (PC) auf den Wert 3C50 (damit haben Sie ausreichenden Speicherplatz für den Stapelspeicher und das nun folgende Programm) und gehen Sie in den Input-Mode. Tippen Sie nun das Programm ein. Die Zahlenfolge finden Sie im Listing unter OBJ CODE.

Sie tippen demnach:

0	E
0	C
1	1
0	F

usw. bis Sie bei 71 angelangt sind. Verlassen Sie den Input-Mode und setzen Sie den Befehlszähler wieder auf die Startadresse (=3C50).

Wenn jetzt die Taste Start gedrückt wird, läuft Ihr Programm ab, was sich durch ein entsprechendes Verhalten der Anzeige bemerkbar macht.

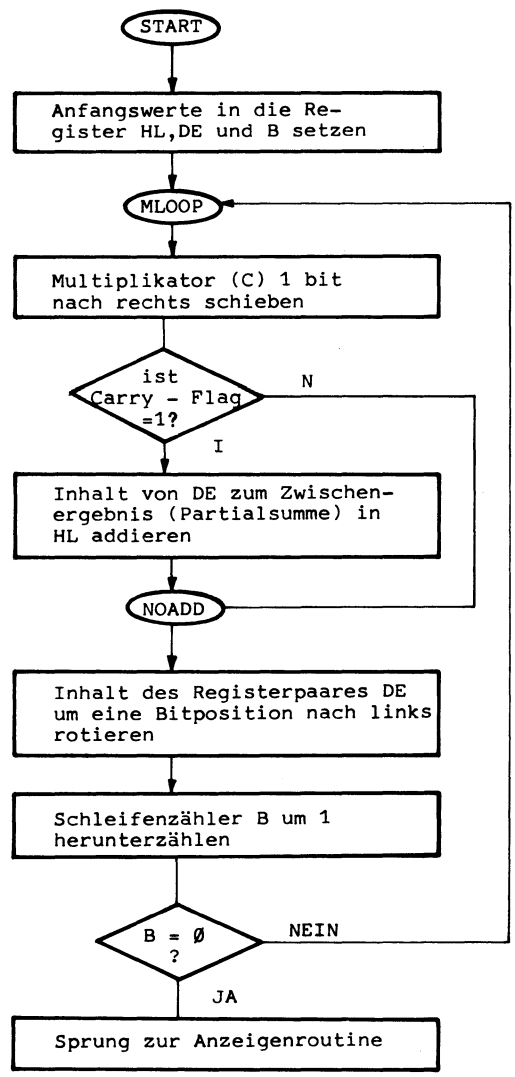
Ein Verlassen des Usersprogramms ist (in diesem Fall der unendl. Schleife) nur über einen Reset möglich.

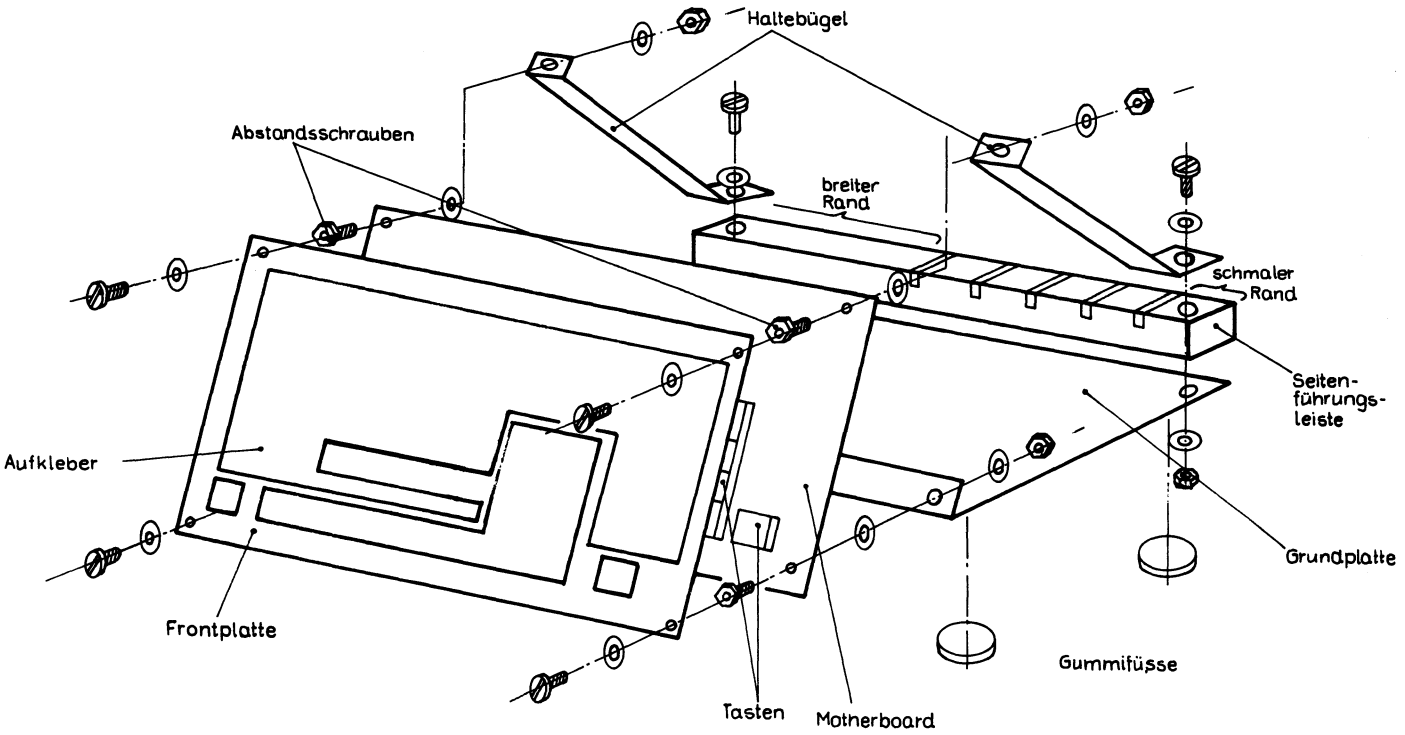
Multiplikationsprogramm

Hier handelt es sich um eine arithmetische Rechenfunktion, die es dem Anwender ermöglicht, zwei hexadezimale Zahlen miteinander zu multiplizieren. Das Ergebnis wird nach vollendeter Operation bei ADDRESS, ebenfalls in hexadezimaler Form angezeigt.

Im Gegensatz zum in 5.1 gezeigten Prüfprogramm, das in einer undendlichen Userprogrammschleife verbleibt, wird hier der Rücksprung ins Monitorprogramm ausgeführt. Dies hat zur Folge, daß ohne Reset sofort eine neue Aktivität ausgelöst werden kann.

Programmablaufplan





Programmerläuterung

Bekanntlich verfügen heute handelsübliche 8bit-Mikroprozessoren lediglich über festverdrahtete 8bit-Additionsbefehle, etwas weiterentwickelte Maschinen wie Z80 auch über festverdrahtete 8- und 16bit-Additions- **und** Subtraktionsbefehle. Benötigt man in einem solchen Mikrocomputeranwendersystem andere Rechnungsarten, so muß man diese mit Hilfe von Software implementieren. Da die Vorgehensweise dabei für den Anfänger interessant ist, soll hier eine 8bit-Multiplikationsroutine angegeben und erläutert werden.

Als Register für den 8bit-Multiplikator wurde C gewählt; das Ergebnis kann 16bit lang sein, da die Multiplikation der zwei höchstmöglichen 8stelligen Zahlen $255 \times 255 = 65025$ groß werden kann. Aus diesem Grund wurde für Zwischenergebnis und Ergebnis ein 16bit-Registerpaar verwendet (HL). Ebenso mußte für den Multiplikanden ein Register**paar** (DE) reserviert werden, da das Verfahren ja darin besteht, den um eine Potenzstelle nach **links** verschobenen Multiplikanden sukzessive auf das Zwischenergebnisfeld aufzuaddieren.

Selbstverständlich muß bei der Datenübergabe der Inhalt von $D = 0$ sein. Nach den erforderlichen Initialisierungen verschiebt man den 8bit-Multiplikator nach rechts. War das zu diesem Zeitpunkt im Register C stehende niederwertigste bit = 1, so wird das Überlaufbedingungsbit gesetzt und aufgrund einer darauffolgenden Abfrage „Carry = 1?“ der Inhalt des Registerpaares DE zum Inhalt des (Zwischen-)Ergebnisregisterpaares HL addiert.

War das Carry-Flag = 0, so entfällt die Addition; in beiden Fällen wird jedoch der Inhalt von DE um eine Binärstelle erhöht, d.h. um ein bit nach links geschoben.

Am Ende des Durchlaufs wird geprüft, ob bereits der ganze Multiplikator abgearbeitet ist, d.h. ob das Durchschieben des Multiplikators 8mal stattgefunden hat. Ist der Zähler = 0, so steht im Registerpaar HL das fertige Multiplikationsergebnis, sonst ist der Programmablauf ab der Marke MLOOP zu wiederholen.

Die Codierung (s. Listing) ist recht einfach. Bemerkenswert die einfache Realisierung eines 16bit-Schiebevorganges mit Hilfe der zwei Befehle SLA r und RL r. Dabei wird das höchstwertige Bit des Registers E durch SLA E ins Carry-Flag geschoben und die unterste Stelle mit einer Null aufgefüllt.

RL D „holt“ dann den Übertrag aus dem E-Register über das Carry-Flag ab.

Das ganze Programmstück kann selbstverständlich auch als Unterprogramm geschrieben werden.

In diesem Fall wäre bei Programmbeginn und -Ende ein Retten bzw. Wiederherstellen der Registerinhalte über 4 Push und 4 POP-Befehle zu besorgen.

Hans Peter Blomeyer-Bartenstein

Universelle betriebsbereite Mikrocomputer-Anwendersysteme

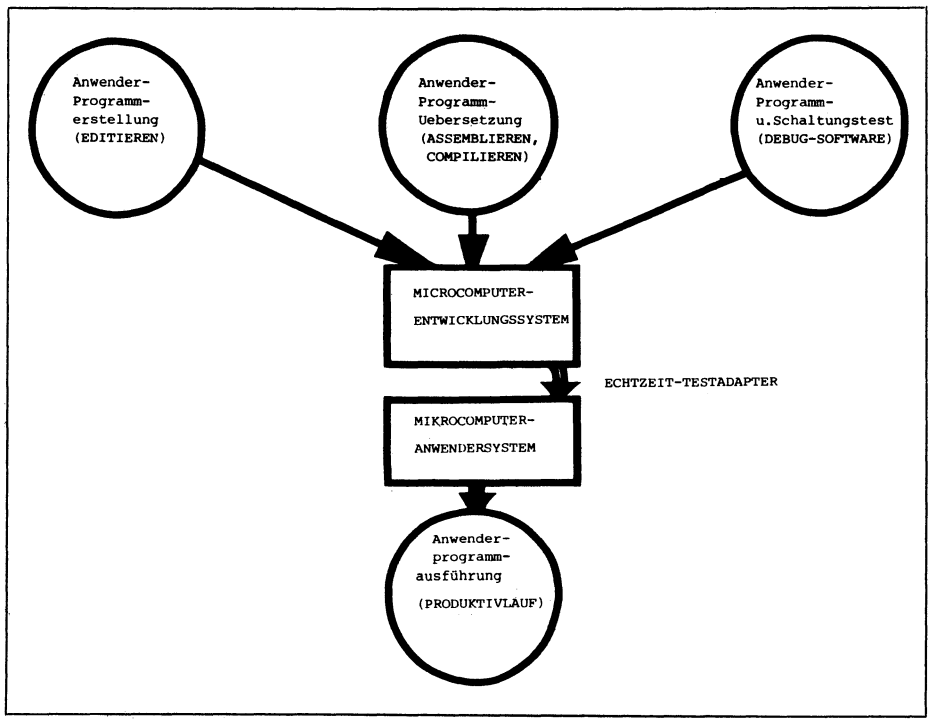
Ab der Verfügbarkeit der ersten Mikrocomputer im Jahre 1971 bis heute unterschied sich die Mikrocomputertechnik von der herkömmlichen Mini- und Megacomputertechnik, abgesehen von den bekannten hardwaremässigen Gegebenheiten (kleineres körperliches Volumen, niedrigerer Preis, höhere Universalität und weitergehende Adaptionbarkeit), insbesondere durch folgende organisatorische Besonderheit:

Es war grundsätzlich ein sogenanntes Entwicklungssystem nötig – das ist ein Computer, der seinerseits einen Mikroprozessor beinhaltet und darüber hinaus über eine Reihe von Hardwarezusätzen zur Erreichung höherer Effizienz beim Systemtest und die nötige Betriebssoftware verfügt. Auf ihm wird dann die eigentliche Programmentwicklung und -korrektur durchgeführt *(Bild 1)*, während für das Mikrocomputer-Endprodukt (z.B. Maschinensteuerung) eine eigene, davon unterschiedliche Hardware nötig ist, die man sich beispielsweise aus fertigen Computer- und eigenen Zusatz-Platinen zusammenstellt.

Der Test erfolgt dann unter der Kontrolle der Test-Software des Entwicklungssystems, wobei evtl. auch Hardware des Entwicklungssystems vom zu testenden Anwendersystem «mitbenutzt» wird und die CPU des Anwendersystems durch einen sogenannten Testadapter ersetzt ist, der Entwicklungs- und Anwendersystem miteinander verbindet.

Das Anwenderprogramm wird also in diesem Fall in der Testphase von der CPU des Entwicklungssystems ausgeführt. Da im allgemeinen die CPU beider Systeme gleichen Typs sind, kann ein solcher Test in Echtzeit erfolgen.

1 Mit Entwicklungssystem. Hard- und Softwareentwicklung mit Mikrocomputer-Entwicklungssystem.



Anders verläuft die Benutzung von Mini- und Grossrechnern *(Bild 2)*; hier erfolgen sowohl die ganze Programmentwicklung (Eingabe, Übersetzung und Test) als auch die vom Anwender gewünschten Abläufe (= «Produktivläufe») auf dem gleichen Rechner. Dies ist ohne weiteres möglich, da solche Maschinen im allgemeinen nicht selbst Gerätefunktionen realisieren, sondern (wie im Falle der Prozessrechner) höchstens Ergebnisse von Geräten verarbeiten bzw. Rechenergebnisse an Geräte weitergeben.

Zur Realisierung von Gerätefunktionen selbst sind solche Computer wegen ihres hohen Platzbedarfs und der hohen Kosten in den meisten Fällen nicht akzeptabel. Der stürmische Fortschritt auf dem Gebiet der Mikrocomputertechnologie hat es jedoch inzwischen möglich gemacht, Computer zu realisieren, die zwar die Leistungsfähigkeit und die übrigen Eigenschaften gängiger Minirechner haben, preislich und volumenmässig jedoch weit unter diesen liegen.

Diese Eigenschaften sind insbesondere:

- Befehlsausführungszeit knapp über 1 µs
- Zugriff auf Massenspeicher mit wahlfreiem Zugriff
- transparentes, leistungsfähiges Betriebssystem
- Verfügbarkeit höherer Sprachen.

Es liegt auf der Hand, dass Mikrorechnersysteme, die diese genannten Eigenschaften besitzen, folgende Möglichkeiten eröffnen:

- Realisierung einer Gerätefunktion unter der Verwendung der Vorteile von nichtflüchtigen Massenspeichern mit wahlfreiem Zugriff in einem einzigen Gehäuse zu niedrigstem Preis;
- Erstellung, Übersetzung und Test von Anwenderprogrammen auf eben dieser Anlage, die auch nachher die Produktivbearbeitung dieser Programme übernimmt.

Aus diesen Fähigkeiten resultiert folgende erstaunliche Feststellung:

In vielen Fällen ist durch solche Geräte der Einsatz von Mikrocomputerentwicklungssystemen nicht erforderlich – Programmentwicklungshilfe und Anwendersystem sind ein und dieselbe Hardware!

Diese Möglichkeit ist natürlich insbesondere für kleinere und mittlere Systemhäuser interessant, die grundsätzlich vor der Anschaffung eines Entwicklungssystems zurückschrecken oder aber die aus technischen Gründen einen bestimmten Prozessor einsetzen wollen, im allgemeinen jedoch mit einem Prozessor anderen Typs arbeiten; sie können auf der gleichen Hardware Programme entwickeln, die sie selbst bei ihren Kunden installieren!

Ein System, das unter anderem diese genannten Möglichkeiten bietet, hat die amerikanische Mikrocomputerfirma *Zilog* jetzt unter dem Namen Z80-MCS auf den Markt gebracht; dieses System verbindet

die Vorzüge der Mikrocomputertechnik (geringe Abmessungen, niedriger Preis) mit denen der Minirechner (leistungsfähiges Betriebssystem, höhere Sprachen, Massenspeicherzugriff).

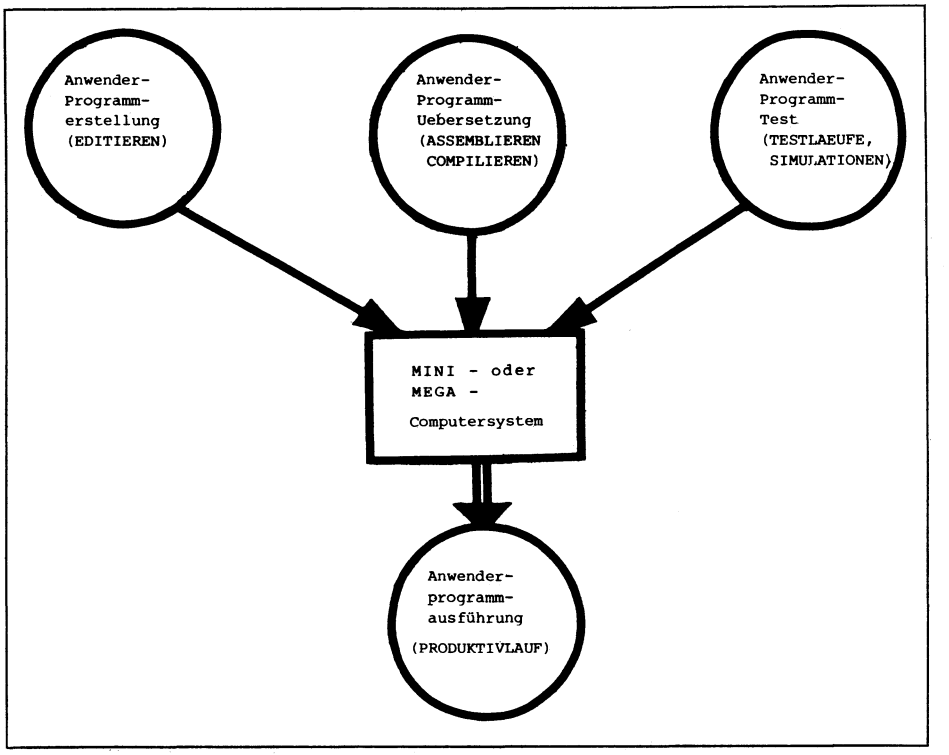
Es ist auch zur Entwicklung der Anwendersoftware in all den Fällen geeignet, bei denen kein Hardware-Echtzeittest erforderlich ist, da die hierfür nötigen Echtzeittestadapter aus architekturellen Gründen an das MCS nicht anschliessbar sind; für solche Entwicklungen sollte dem Z80-Entwicklungssystem der Vorzug gegeben werden.

Die Z80-MCS-Hardware

Das Zilog-Z80-MCS-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. 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 MCS-Systems ist der Z80-CPU-Mikroprozessor, der maschinencodekompatibel zum 8080A-System ist und darüber hinaus 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 MCS beinhaltet 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.

Kernstück des Z80-MCS 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 8-Bit-Akkumulator, ein Flagregister und sechs allgemeine 8-Bit-Register, die auch als 16-Bit-Speicheradressregister verwendet werden können.



2 Ohne Hilfe. Bei Benutzung von Mini- und Grossrechnern erfolgen Programmentwicklung und Produktivläufe auf dem gleichen Rechner.

- 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 Refreshcontroller 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-Datenwörtern
  - 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/Ausgabe mit einer Ein/Ausgaberate von 125 kByte/sec, Setzen, Testen oder Rücksetzen irgendeines einzelnen Bits in einem der CPU-Register oder einer Speicherstelle).

Das Z80-MCS ist eine nach Anlegen der Versorgungsspannung (220 V, 50 Hz) sofort arbeitsfähige Anlage mit 2 Floppy-Disk-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 38 600 Baud einstellbar. Zusätzlich zu diesem Anschluss der Bedienkonsole sind an der Rückwand des Systems 7 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.

Insgesamt sind im MCS neun 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-/Ausgabe-Erweiterungen 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 MCS 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 PIOs verdrahtet; auf diese Weise kann die Anschlussbelegung vom Anwender durch

Legen von Drahtbrücken festgelegt werden. Optional ist das Z80-MCS jedoch auch in für einen Schnelldrucker fertig verdrahteter Form lieferbar. Für Echtzeitaufgaben wurde ein Z80-CTC-Baustein eingesetzt, der die Festlegung der Übertragungsrate der Bedienungskonsole 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 Pufferung der Zentralbaugruppe im MCS und die sieben freien Kartensteckplätze erlauben einfach durch Einstecken von Standard- oder anwendungsspezifischen Baugruppen eine Erweiterung der Fähigkeiten des MCS.

Eine Erweiterung ist um folgende Baugruppen möglich:

- Z80-RMB: Schreib/Lesespeicher-Erweiterung mit 16 kByte Kapazität, aufgebaut mit 4-kByte-dynamischem RAM-Speicherbaustein. Zusätzlich auf dieser Baugruppe untergebracht sind acht Fassungen für Festwertspeicherbausteine.
- Z80-IOB: Parallel-Ein/Ausgabe-Erweiterung zur Erweiterung des MCS-Systems um acht Ein/Ausgabeports à 8-Bit inkl. Steuerleitungen.
- Z80-SIB: serielle Ein/Ausgabe-Erweiterung des Z80-MCS um vier serielle Ein/Ausgabe-Duplexschnittstellen.
- Z80-VDB: ermöglicht den direkten Anschluss des Z80-MCS 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 und 2 Parallel-Ein/Ausgabe-Bausteine (Z80-PIO).
- Z80-PPB: PROM-Programmierbaugruppe. Diese Baugruppe wird mit der zugehörigen Betriebssoftware geliefert und erlaubt das Programmieren von bipolaren und MOS-Festwertspeicherbausteinen (PROMs und EPROMs). Folgende Festwertspeichertypen lassen sich elektrisch direkt ohne Eingriff in der Schaltung des Z80-PPB programmieren: 2704, 2708, Harris 7620, 7621, 7640 und 7641.
- Z80-SCC: Im MCS-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-MCS-Bus eingesteckt werden, dass sie von aussen her zu Testzwecken zugänglich sind.
- AD-DA-Wandler-Baugruppen sind in Vorbereitung

MCS-Standardsoftware

Zu dem Standardlieferumfang des Z80-MCS gehört ein Standardbetriebssystem, das folgende Programme umfasst:

- Bootstrap- und Monitor-Firmware
- Platte-Betriebssystem, Ein/Ausgabe-Treiber-Routinen
- Editorprogramm
- Makroassembler
- Fehlersuchprogramm

Das Z80-MCS 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 MCS-Executive-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 aufgebauten Systemen drastisch erweitert (600 000 Byte).

Optional zu diesem Betriebssystem ist ein BASIC-Interpreterprogramm verfügbar, das in einem Minimalspeicher von 32 kByte ablauffähig ist.

Höhere Betriebssoftware für MCS

Das transparente Betriebssystem Z80-RIO für das MCS ist eine modular aufgebaute, dem Anwender zugängliche und transparente Betriebssoftware, die einen relokativen Assembler und logisches I/O-Management umfasst; sie ist gleichermassen zur rationellen Entwicklung von Anwendersoftware wie zur Benutzung durch den Programmierer in dessen anwenderspezifischer Software geeignet.

Hierzu wurden Binde/Lade-Routinen und die Möglichkeit zur Erweiterung des Betriebssystems entsprechend den anwendungsspezifischen Erfordernissen implementiert. Diese Möglichkeiten sind u.a. durch eine flexible Exekutiv-Routine garantiert, die sämtliche Ein/Ausgabe- und System-Anforderungen verwaltet. Logische Dateien und Gerätebedienungsroutinen erlauben die Abwicklung aller Ein/Ausgabe-Aktivitäten unter Verwendung dynamischer Gerätezuweisung, wobei natürlich auch die Systemkonsole als logisches Gerät behandelt wird. Eine spezifische Kommandosprache erlaubt das Anfügen anwenderspezifischer Erweiterungen des Betriebssystems.

Schliesslich ist noch anzumerken, dass auf das Dateiverwaltungsprogrammpaket von jedem Anwenderprogramm ohne weiteres zugegriffen werden kann.

Der Datentransfer von und zu sämtlichen peripheren Geräten erfolgt in einem standardisierten Format. Das Exekutivprogramm akzeptiert Kommandos von der Systemkonsole ebenso wie Informationen von den Floppy Disks; dadurch lassen sich ganze System-Operationsfolgen über eine Disk-Datei festlegen, die in diesem Fall sozusagen als «automatischer Operator» fungiert. Auf diese Weise ist es möglich, das System in einer bestimmten, vorgegebenen Weise vollautomatisch ohne eine Bedienungsperson arbeiten zu lassen.

Der im RIO implementierte Floppy-Disk-Handler ZDOS II ist übrigens in der Lage, auf Disketten-Dateien wahlfrei zuzugreifen, was gegenüber dem Verfahren des indexsequentiellen Zugriffs den Vorteil einer kürzeren System-Reaktionszeit hat.

Übersetzungsprogramme für höhere Sprachen
Für das Mikrocomputersystem Z80-MCS ist eine Reihe Übersetzungsprogramme verfügbar oder in Entwicklung.

Zum Ablauf mit dem Standardbetriebssystem ist ein Interpreter für die interaktive Sprache BASIC lieferbar; sie benötigt einen 32-kByte-Arbeitsspeicher. Alle übrigen Übersetzungen laufen unter dem Betriebssystem RIO ab:

- Erweiterter BASIC-Interpreter, der gegenüber dem oben genannten Standard-BASIC eine wesentlich höhere Arbeitsgeschwindigkeit hat, das Rechnen mit doppelter Genauigkeit und das Einbinden von Assembler-Routinen ermöglicht.
- COBOL-Compiler (enthält eine Untermenge des Standard-COBOL-Befehlssatzes)
- FORTRAN Compiler (in Vorbereitung)
- PL/Z-Compiler. PL/Z ist eine neue prozedurorientierte Sprache, die Elemente der Sprachen PL/I, ALGOL '68

und PASCAL in sich vereint und speziell für den Einsatz mit Mikrocomputern geschaffen wurde. Sie ist nicht zu verwechseln mit dem bekannten PL/M, demgegenüber es eine wesentlich höhere Effizienz (fast die gleiche wie ein ASSEMBLER-Programm) bezüglich Speicherplatzausrüstung und Verarbeitungsgeschwindigkeit aufweist. PL/Z erlaubt sowohl die Ausnützung der Vorteile der Verwendung einer höheren Programmiersprache (leichtere, weniger zeitaufwendige und weniger fehleranfällige Codierung) als auch die Ausnützung der Prozessorarchitektur (z.B. Register).

Anwendungsbereiche des Z80-MCS

Die bisher beschriebenen Hard- und Software-Eigenschaften und die zugehörigen Erweiterungsmöglichkeiten machen das Z80-MCS zu einem universellen, sehr preisgünstigen Computer, dessen Einsetzbarkeit praktisch nur durch die Phantasie des Anwenders beschränkt ist. Wirtschaftlich ist es sicher in allen Fällen, in denen ein Rechner oder ein flexibles Steuergerät benötigt wird, das über Massenspeicher verfügt.

Erweitert wird diese Flexibilität noch durch die Tatsache, dass alle seine elektronischen Einzelteile (auch die Firm- und Software!) einzeln lieferbar sind, so dass der Entwickler seinen Prototyp mit dem MCS erstellen, für die Grossserie jedoch ohne Adaptionsaufwand seine «massgeschneiderte» Elektromechanik (Gehäuse, Anzahl und Typ der Ein/Ausgabe-Geräte, Baugruppenformat usw.) verwenden kann.

Die folgenden Applikationen können nur als Denkanstösse aufgefasst werden – wie gesagt, die tatsächlichen Möglichkeiten sind unbegrenzt.

In all diesen Fällen ist die komplette Rechnerhardware und in einigen Fällen sogar ein Teil der Software standardmässig verfügbar:

- Analysentechnik mit graphischer Ausgabe
- Messdatenerfassungssystem in Labors und Kliniken
- Verwaltung von Patientenkarteien in Arztpraxen
- Patientenüberwachung in Intensivstationen
- Steuerung und Regelung von Maschinen industrieller Prozesse
- Zeichnungsherstellung 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
- Steuerheinheit für automatische Testsysteme

Hans Peter Blomeyer-Bartenstein

Erweiterung des Adreßbereichs von Z 80-Systemen über 64 kByte hinaus

In Systemen, die Zugriff zu Massenspeichern haben (z. B. Z 80), kann es durchaus sinnvoll oder nötig sein, den durch die 16 Adreßleitungen des Mikroprozessors begrenzten Speicherbereich zu erweitern. Dies kann man relativ einfach mit der von Minirechnern her bekannten „Mapping-Technik" machen, bei der einfach über Ein-/Ausgabebefehle zwischen verschiedenen Speicherblöcken (*Partitions*) umgeschaltet wird. Im System Z 80 beispielsweise kann eine solche Partition maximal 64 KByte umfassen. In dieser Arbeit soll nun die Hardware- und Software-Architektur seines solchen Systems beschrieben werden, das 1 Mio.-Byte-Arbeitsspeicher adressiert. Besonders interessant sind derartige Konfigurationen im Zusammenhang mit Multiprocessing und direktem Speicherzugriff.

1 Grundprinzip

Der direkt, d.h. von *einem* Befehl adressierbare Speicherbereich ist (soweit man keine Adreßmultiplextechnik anwendet) durch die Anzahl der Adreßleitungen der CPU begrenzt. Im Falle der Z80-CPU mit einem 16 bit „breiten" Adreßbus sind dies 64 K (≙ 65 536) mögliche Adressen.

Die Adressierung geschieht im Rahmen der Wortkapazität der verwendeten Speicherbausteine direkt durch die von der CPU auf den Adreßbus gelegte Speicheradresse (= „niederwertiger Adreßteil"); bei den heute verfügbaren Speicherbausteinen (ROM: 4 Kbit x 8; dynamische RAM: 16 Kbit x 1) ist dieser vom Speicherbaustein unmittelbar zu verwertende Adreßteil 12 bit (≙ 4 K) bzw. 14 bit (≙ 16 K). Die übrige Adressierung muß durch Auswahl bzw. Freigabe desjenigen physikalischen Speicherblocks erfolgen, der angesprochen werden soll (*Bild 1*).

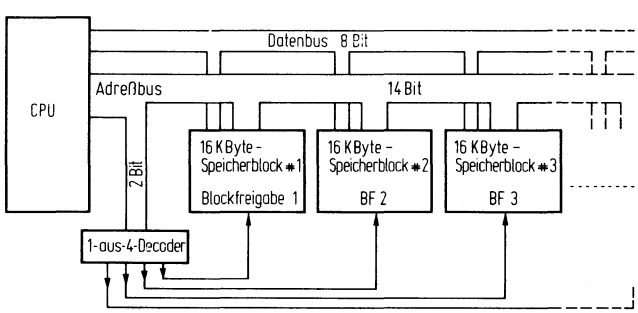


Bild 1. Speicherorganisation in einem herkömmlichen Z 80-Mikrocomputersystem mit max. 64 KByte

Man sieht sofort, daß eine praktisch beliebige Erweiterung der Speicherkapazität einfach dadurch zu realisieren ist, daß man weitere Freigabe-Signale „künstlich" erzeugt, die zusätzliche Speicherbereiche freigeben und sperren, während der niederwertige Teil des Adreßbus und der Datenbus aller Bausteine „wie gehabt" miteinander galvanisch verbunden sind.

Statt Erzeugung eines Signals pro Speicherblock kann man natürlich die sowieso schon erzeugten Signale des Adreßdecoders mit benutzen; die jetzt noch verbleibenden und zur Anwahl der zusätzlichen Speicherbereiche herangezogenen Freigabesignale nennt man *Partition-Adressen*. Dieses Verfahren ist schaltungstechnisch am einfachsten; die entsprechende Blockschaltung zeigt *Bild 2*.

Es ist jetzt noch im einzelnen zu klären, wie die Partitionadressen hardwaremäßig zu erzeugen sind und welche

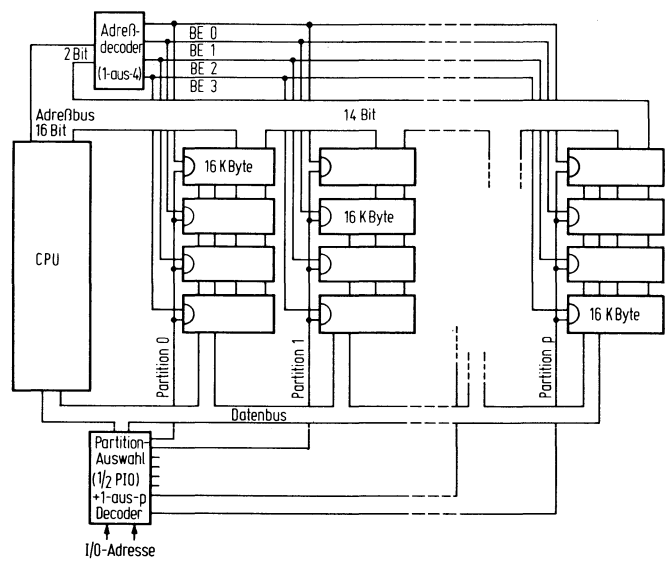


Bild 2. ▶
Blockschaltung für Mikrocomputersysteme über 64 KByte

Softwaremaßnahmen nötig werden, um mit der vorliegenden Architektur arbeiten zu können.

2 Hardware-Gesichtspunkte der „Mapping"-Technik

Die einfachste und von der Bausteinanzahl am wenigsten aufwendige Struktur ergibt sich, wenn man die einzelnen Speicherblöcke zusätzlich zur Blockfreigabe mit einem weiteren Freigabeeingang zur Partition-Freigabe ausstattet, wobei beide Freigabe-Signale über eine UND-Funktion verknüpft sind. Man hat dann lediglich dafür zu sorgen, daß

a) die Partition-Adreßsignale in einem Pufferspeicher (*Latch*) rechtzeitig vor einem Speicherzugriff zur Verfügung gestellt werden. Dies geschieht einfach durch einen Ausgabebefehl, bei dem die CPU eine Konstante auf den Datenbus legt, der über das I/O-Request und WRite-Signal in das Page-Adreß-Latch eingeschrieben wird.

b) die kapazitive und/oder Strom-Belastbarkeit der Bustreiber (Adreß-, Daten- und Steuerbus) nicht überschritten wird.

c) bei Verwendung dynamischer Speicherbausteine einwandfreies Refresh-Verhalten sichergestellt ist.

Zu Punkt a) ist zu bemerken, daß hier zwei Lösungsalternativen existieren:

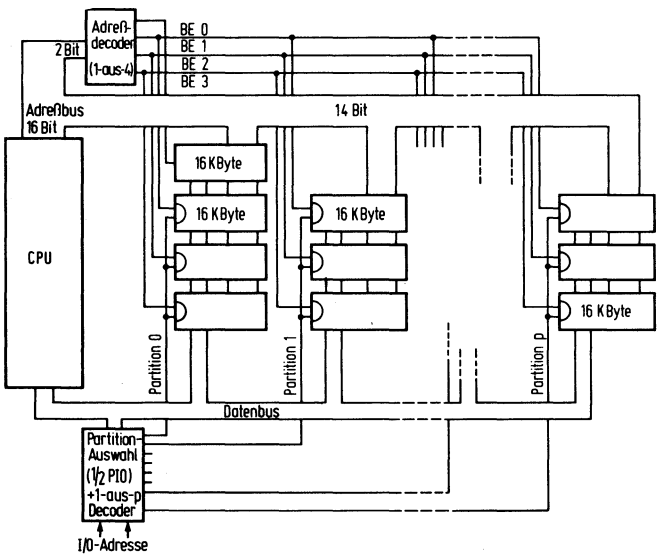


Bild 3. Blockschaltung wie Bild 2, jedoch mit gemeinsamem Supervisor-Bereich

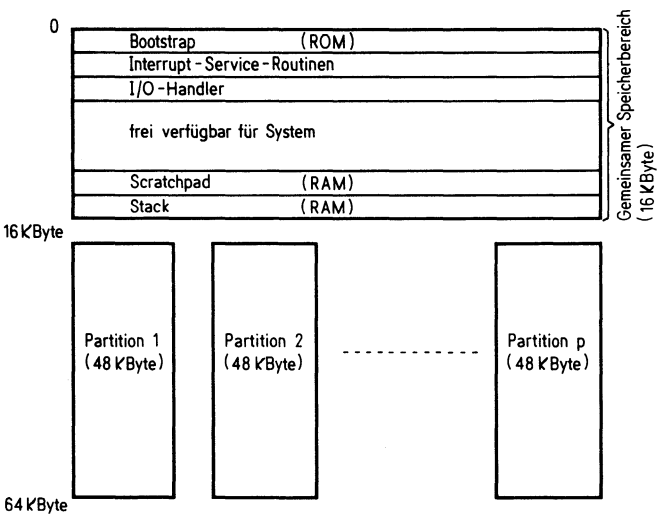


Bild 4. Speicherorganisation mit Supervisor-Bereich (schematisch)

- Verwendung von einem Flipflop pro Partition-Adreß-Leitung (n Flipflops)
- Verwendung eines 1-aus-n-Decoders **nach** dem Flipflop-Satz, was die Anzahl der nötigen Flipflops auf ld [n] reduziert.

Welches Verfahren angewendet wird, bleibt dem Systemdesigner überlassen; in beiden Fällen kann für den Flipflop-Satz ein Standard-Baustein mit entsprechenden Strobe-Leitungen oder aber ein Ein-/Ausgabe-Port-Baustein wie der Z80-PIO verwendet werden. Will man z. B. 1 MByte adressieren, genügen ein Z80-PIO-Port (von dem dann nur 4 Leitungen benützt sind) und ein 1-aus-16- bzw. zwei 1-aus-8-Decoder-Bausteine.

Zu b): Die Probleme der Speicherpufferung sind in [1] eingehend beschrieben, ebenso die bei c) erwähnten Refresh-Fragen; ein grundsätzliches Problem besteht beim Refresh nicht, da nur die niederwertigen Adressen angesprochen werden müssen.

Anders ist die Situation bei DMA; hier müssen Vorkehrungen getroffen werden (z.B. Beschränkung der DMA-Blocklänge und -Häufigkeit), um einen Verlust des RAM-Inhalts sicher zu vermeiden.

3 Speicherorganisation

Die Arbeitsweise des Z80-Mikroprozessors (Interrupt, Stack, DMA) und die Notwendigkeit von Betriebs-Grundprogrammen (z.B. Bootstrap) erfordert bei der Verwendung der „Mapping"-Technik das Einrichten eines physikalischen Speicherbereichs, der unabhängig vom momentan benutzten Speicherbereich angesprungen werden kann. Aufgrund der eingangs gezeigten technologischen Gegebenheiten macht man diesen Speicherbereich bei Verwendung von 4 Kbit-Bausteinen 4 KByte groß und bei Verwendung von 16 Kbit-Bausteinen 16 KByte.

Den gemeinsamen Zugriff realisiert man einfach, indem man diesen gemeinsamen Speicherblock ausschließlich mit der Adreßdecoderleitung BE0 freigibt, also ohne eine zusätzliche Partition-Freigabe einzubauen (*Bild 3*).

Hieraus folgt selbstverständlich, daß ein Speicherblock der Blockadresse 0 im gesamten Speicher physikalisch nur ein einziges Mal vorkommen darf und der tatsächliche realisierbare Speicherumfang nicht Kges = 64 KByte · p
ist, sondern nur K = (64 KByte – b) · p + b,
wobei b die Größe des gemeinsamen Speicherbereichs in KByte ist und p die Anzahl der Partitions.

In unserem Beispiel (Bild 3) wäre die tatsächliche Kapazität dann (64-16) · p + 16 = 736 KByte. *Bild 4* zeigt die resultierende Speicherorganisation.

Man wird folgende Aufteilung dieses allen Partitions gemeinsamen Speicherbereichs vornehmen:

- die „untersten" Speicherstellen (z.B. 1KB) werden als Festwertspeicher realisiert und enthalten die Bootstrap-Routine für die Systeminitialisierung („Kaltstart"), die Interrupt-Bedienroutine und das Unterprogramm zum Übergang von einer Partition auf eine andere für den nichtmaskierbaren Interrupt (NMI) und Programme für Alarmfälle wie Stromausfall usw. Soweit hier auch die Startadressen der Interruptservice-Routinen passend gelegt werden, ist der Prozessor in Interrupt-Mode 0 oder 1 (8080 A-kompatibel) zu betreiben.

● Die „obersten" Speicherstellen werden als Schreib-/Lesespeicher realisiert und sind als Stapelspeicher (*Stack*) und zur Ablage von Momentanwerten reserviert.
● Die übrigen Speicherstellen können beliebig als Schreib-/Lese- oder Festwert-Speicher ausgeführt werden und müssen enthalten: Die Sprungtabelle und die Interrupt-Service-Routinen (Interrupt-Mode 2) sowie Betriebs- und Hilfs-Programme, die von allen Partitions benützt werden sollen (z.B. Unterprogramm-Bibliothek, Programme zum Austausch von Daten zwischen den Partitions, System-Supervisor usw.).

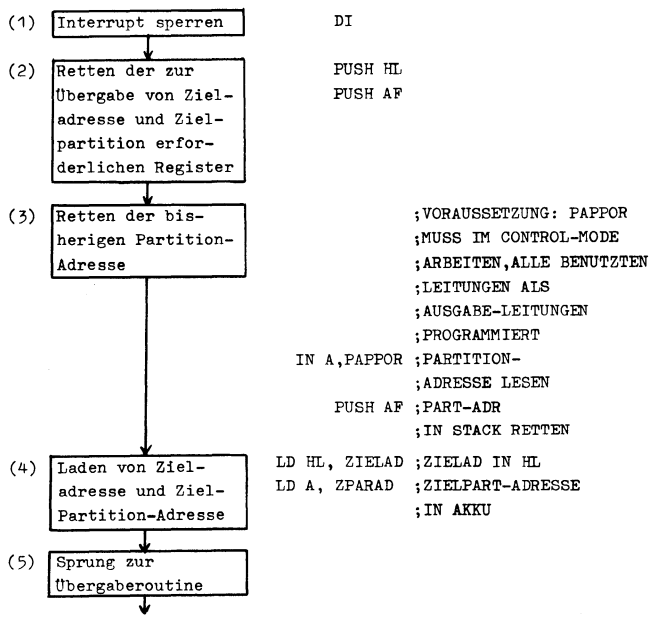
Durch diese organisatorischen Maßnahmen ist es möglich, die Interrupt-Fähigkeit des Z80 uneingeschränkt und ohne Software-Overhead auszunützen. Interrupt kann also bei Arbeit in jeder Partition zugelassen sein; ebenso ist der Aufruf von Unterprogrammen, die im Speicherblock 0 liegen, ohne zusätzlichen Aufwand durchzuführen. Dies beides ist gegeben, da jeweils der Befehlszähler (zuzüglich der evtl. zu rettenden Registerinhalte) im Stapelspeicher, der allen Partitions gemeinsam ist, abgespeichert wird und die Partition-Adresse bei dieser Operation nicht beeinflußt wird. Muß die Partition während der Interrupt-Bedienroutine gewechselt werden, so ist sie am Ende der Routine zu regenerieren.

4 Software

4.1 *Der Übergang von einer Partition auf eine andere*

Dieser Übergang erfolgt durch Laden der Partition-Adresse. Da hierfür i. A. sowohl der Befehlszähler als auch die Partition-Adresse vorzugeben ist, kann der Übergang nur über eine kleine Hilfsroutine geschehen, die in Speicherblock 0 liegen muß.

Der Aufruf dieser Routine muß, je nachdem, ob eine spätere Rückkehr in die zu verlassende Partition gewünscht ist oder nicht, etwa folgendermaßan aussehen:




```
        DI
        PUSH HL
        PUSH AF

                ;VORAUSSETZUNG: PAPPOR
                ;MUSS IM CONTROL-MODE
                ;ARBEITEN,ALLE BENUTZTEN
                ;LEITUNGEN ALS
                ;AUSGABE-LEITUNGEN
                ;PROGRAMMIERT
IN A,PAPPOR     ;PARTITION-
                ;ADRESSE LESEN
        PUSH AF ;PART-ADR
                ;IN STACK RETTEN

LD HL, ZIELAD   ;ZIELAD IN HL
LD A, ZPARAD    ;ZIELPART-ADRESSE
                ;IN AKKU
```

Soll ein Sprung aus einer Partition in eine andere erfolgen, ist Pkt. (5) im Programmablaufplan durch den Befehl
JP CHGPAR
zu realisieren; soll aus einer Partition ein Unterprogramm in einer anderen Partition aufgerufen werden, ist dort der Befehl
CALL CHGPAR
zu verwenden, der automatisch den Inhalt des Befehlszählers in den Stack rettet. Die in Block 0 gelegene Routine CHGPAR hat dann folgendes Aussehen:

```
CHGPAR: PUSH BC       ;NUR BEI AUFRUF EINER ROUTINE
        PUSH DE       ;IN EINER ANDEREN
        PUSH IX       ;PARTITION
        PUSH IY       ;NOETIG
        OUT PAPPOR,A  ;SETZEN DES PARTITION-PORTS AUF
                      ;DIE GEWUENSCHTE PARTITION-ADRESSE
        JP (HL)       ;LADEN DES BEFEHLSZAEHLERS MIT
                      ;DER ZIELADRESSE
```

In dem Programm, das in der neuen Partition bei ZIELAD beginnt, ist nun für die Wiederherstellung der zuvor geretteten Registerinhalte und evtl. für die Interruptfreigabe zu sorgen. Falls dieses Programm eine Subroutine ist, erfolgt der Rücksprung über eine der Routine CHGPAR analog aufgebaute Rücksprung-Routine, die natürlich statt der „PUSHs" nun POP-Befehle, statt dem JP (HL) einen RET-Befehl beinhaltet und die die in den Stack gerettete alte Partition-Adresse wiederherstellt.

4.2 *Statusinformationen*

Unbenützte Bits des Partition-Ports lassen sich zur Ausgabe von Programm-Statusinformationen (z. B. Interrupt-Zustand, Fehlermeldungen) oder Bereitstellung der vorhergehenden Partition-Adresse benützen. Diese Informationen lassen sich aus den PIO-Registern im Control-Mode in die CPU lesen, wie es in Punkt 4.1 bereits vorgeführt wurde.

4.3 *Übergabe von Parametern und Daten*

Parameter und Daten lassen sich grundsätzlich auf folgende Arten von Partition zu Partition übergeben:

● über die beiden Registersätze der Z80-CPU,
● über den Stapelspeicher,
● über den übrigen Schreib/Lesespeicher in Block 0.

4.4 *Erzeugung des Maschinen-Codes von Programmen, die in verschiedenen Partitions ablaufen sollen*

Da der Z80-Makroassembler das Arbeiten mit mehreren Partitions nicht automatisch handhabt, müssen hier einige (allerdings einfache) Besonderheiten beachtet werden.

Insbesondere ist zu berücksichtigen, daß von der CPU nicht zwischen den einzelnen Partitions unterschieden wird.

Die Source-Programme sind daher in getrennten Assembliervorgängen zu übersetzen, wobei natürlich geeignete ORG-Anweisungen zu setzen sind. Diese Tatsache hat den Vorteil, daß jedes absolute Maschinenprogramm in jeder Partition ablaufen kann; lediglich an den Stellen, an denen die Partition gewechselt wird, muß für Übergabe der wichtigen Ziel-Partitionsadresse gesorgt werden. Dadurch eignet sich diese Speicherarchitektur besonders gut für Multiprozessorsysteme, in denen mehrere Prozessoren Zugriff auf eine einzige Anwenderprogrammbibliothek haben und dynamische Zuweisung von Partitions an die einzelnen Prozessoren erwünscht ist.

(3) Nur, falls späterer Rücksprung zur bisherigen Speicherstelle und/oder Partition erfolgen soll (CALL-Äquivalent).

Literatur

[1] *Blomeyer-Bartenstein, H.-P.*: Anwendung von Standard-Speicherbausteinen bei Mikrocomputersystemen. ELEKTRONIK 1977, H. 3, S. 75...82.

DEUTSCHLAND

KONTRON ELEKTRONIK GmbH
8057 Eching b. München
Oskar-von-Miller-Str. 1
Telefon (0 81 65) 77-1
Telex 05 26 512

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 05 754655
Telex 54070

EUROPA

ZILOG (UK) LTD.
Nicholson House,
Maidenhead, Berkshire,
UNITED KINGDOM
Telefon 0628-36131/2/3
Telex 848 609

USA

ZILOG INC.
10460 Bubb Road,
Cupertino, Cal. 95014
Telefon (408) 446-4666
Telex 910-338-7621