

Mit Hilfe von integrierten Mikroprozessoren lassen sich leistungsfähige Mikrorechner aufbauen, die in ihrem Einsatz sehr flexibel sind. Die Anwendung dieser Technik bringt es mit sich, daß die Entwicklungsarbeit an Geräten und Systemen sich vom Schaltungsentwurf entfernt und mehr der Programmerstellung zuwendet. Unter anderem wurde auf dem 6. Plenum des ZK der SED gefordert, daß sich die Ingenieure und Techniker in allen Industriezweigen mit dieser neuen Technik, die gegenwärtig das fortgeschrittene internationale technische Niveau repräsentiert, vertraut machen. Zu diesem Zweck beginnen wir in diesem Heft mit der Veröffentlichung eines Lehrganges über die Technik der Mikroprozessoren und Mikrorechner. In ihm sollen u. a. die Grundlagen, Fragen der Technologien, Mikrorechnerkonfigurationen und die Arbeit mit diesen behandelt werden. Die gesamte Reihe „Technik der Mikrorechner“ wurde in der Sektion Informationstechnik der TU Dresden erarbeitet.

1. Grundlagen

Die jüngste Etappe der Entwicklung integrierter Schaltungen wurde 1971 eingeleitet, als der erste Mikroprozessor angeboten wurde. Das Erscheinen dieses LSI-Schaltkreises war das Ergebnis des Überganges auf eine neue Technologie zur Schaltkreisherstellung. Gegenwärtig werden bereits mehrere leistungsfähige Mikroprozessorsysteme von verschiedenen Herstellern angeboten.

Mikroprozessoren erlangen eine immer größere Bedeutung für die Realisierung sowohl steuerungstechnischer als auch klassischer rechentechnischer Aufgaben. Damit wächst zwangsläufig die Zahl derer, die sich in die damit verbundenen Probleme einarbeiten müssen.

Die bei den Mikroprozessor-Realisierungen notwendigen Software-Entwicklungen stellen den Schaltungstechniker vor ihm bisher unbekannte Probleme. Deshalb sollte man in den Kreis der zu durchdenkenden Faktoren auch Fragen der Fähigkeit und Kapazität zur Anfertigung der notwendigen Software einbeziehen.

Andererseits benötigt man Informationen über die zur Verfügung stehenden Zusatzschaltkreise und Möglichkeiten zur Systemkonfiguration, denn allein mit dem Mikroprozessor-Schaltkreis ist eine Aufgabenrealisierung nicht möglich.

Von der Bauelementeindustrie der DDR wurde der sich international abzeichnenden Tendenz des verstärkten Einsatzes hochintegrierter Schaltkreise zur Prozeßüberwachung und -automatisierung Rechnung getragen. Für volkswirtschaftliche Anwendungen steht gegenwärtig bereits der erste in der DDR entwickelte Mikroprozessor-Schaltkreis U 808 D aus dem Kombinat VEB Funkwerk Erfurt zur Verfügung. Erste Einsatzbeispiele dieses Schaltkreises wurden auf der Leipziger Frühjahrsmesse 1977 vorgestellt.

In Erkenntnis einer verstärkt notwendigen Publikationstätigkeit soll diese Artikelserie in zusammenfassender Form sowohl einen Überblick über den erreichten Stand der Technik vermitteln als auch dem zukünftigen Anwender von Mikroprozessoren eine Hilfe bei der Einarbeitung in diese Problematik geben.

In einem ersten großen Abschnitt werden Begriffserklärungen und Erläuterungen zu den verschiedenen verwendeten Schaltkreistechniken sowie den einzelnen Schaltkreistypen gegeben. Weiterhin enthält dieser erste Abschnitt Aussagen zu unterschiedlichen Mikrorechnerkonfigurationen und zu

TECHNIK DER MIKRORECHNER

1

prinzipiellen Zusammenschaltungsmöglichkeiten von Mikroprozessoren bzw. Mikrorechnern. Ein zweiter Abschnitt ist detaillierteren Angaben zu den einzelnen Komponenten eines Mikrorechners vorbehalten (Mikroprozessoren, Halbleiterspeicher, Peripherie und Interface).

Ein dritter Abschnitt wird Fragen des Prozesses der Softwareerstellung einschließlich der dafür im allgemeinen zur Verfügung stehenden Hardware-Hilfsmittel behandeln. Im abschließenden vierten Abschnitt soll auf Einsatzbeispiele für Mikroprozessoren und Mikrorechner eingegangen werden.

1.1. Begriffe und Definitionen

Unter einem Mikroprozessor (Abkürzung μP) ist eine aus einem oder mehreren LSI-Schaltkreisen bestehende Einheit zu verstehen, die in der Lage ist, unter Steuerung durch eine extern zusammengestellte Befehlsfolge (Programm) eine Vielzahl von elektrischen Funktionen zur Verwirklichung gewünschter arithmetischer, logischer oder Transport- und Verschiebeoperationen sequentiell auszuführen.

Der Begriff „Mikroprozessor“ ist damit gleichbedeutend mit der aus der Rechentechnik bekannten Bezeichnung „Zentrale Verarbeitungseinheit“ (ZVE), die dort zur Benennung einer aus Steuer- und Rechenwerk bestehenden Einheit dient. Neben dem Kürzel μP sind auch die Abkürzungen CPU (engl. central processing unit) und MPU (engl. micro processing unit) gebräuchlich und gerechtfertigt. Der Umfang der von einer CPU realisierbaren Operationen bzw. die Möglichkeiten der Erteilung von gewünschten Anweisungen hängen vom Befehlssatz des verwendeten Mikroprozessors ab.

Der Befehlssatz (engl. instruction set, dt. oft auch Befehlsliste) beinhaltet sämtliche möglichen Anweisungen, die dem Mikroprozessor in kodierter Form erteilt werden können (Maschinenbefehle).

Wird ein Mikroprozessor durch Speichermoduln zur Aufnahme von Programmen und Daten sowie durch Ein- und Ausgabesteuerschaltungen zur Kommunikation mit der Umwelt (Austausch von Programmen, Daten, Steuersignalen) ergänzt, erhält man eine als Mikrorechner (Mikrocomputer, μC) bezeichnete, autonom arbeitsfähige Konfiguration. Wiederum in Anlehnung an die schon seit langem geprägten Begriffe der Rechentechnik kann eine solche Konfiguration auch als „Zentraleinheit“ (ZE) bezeichnet werden, da bekanntlich dieser Begriff im allgemeinen eine aus Steuerwerk, Rechenwerk, Hauptspeicher (Arbeitsspeicher) und Ein- und Ausgabesteuereinheit(en) bestehende Einheit charakterisiert. Die einzelnen Komponenten dieses Systems sind an ein gemeinsames Leitungsbündel, den sogenannten Bus, angeschlossen und so miteinander verbunden. Je nach Art der über bestimmte Leitungen dieses Bündels bevorzugt übertragenen Informationen unterscheidet man zwischen dem Adreß-Bus, dem Daten-Bus und einem Steuer-Bus.

Auf dem Adreß-Bus (z. B. 16 Leitungen) übermittelt die CPU den Speicherbausteinen die Adresse der Speicherzellen, auf die sie zurückgreifen möchte. Weiterhin dient er zur Adreßübermittlung an die Peripherie. Auf dem Daten-Bus (z. B. 8 Leitungen $\hat{=}$ 1 Wortbreite) erfolgt der Datenverkehr vom und zum Speicher sowie von der und zur Peripherie. Unter der Bezeichnung Steuer-Bus (engl. control bus) werden die Leitungen zusammengefaßt, auf denen die CPU (bzw. im erweiterten Sinn der Mikrorechner) und die Umwelt Signale zur Steuerung eines einwandfreien Funktionsablaufes entsprechend der zu realisierenden Aufgabenstellung austauschen.

Die Erweiterung der als Mikrorechner bezeichneten Konfiguration durch Gehäuse, Bedienkonsole, Peripherie und Software wird oft Mikrorechnersystem genannt.

Während Hardware als Sammelbegriff für alle eingesetzten Bauteile und Geräte gilt, umfaßt der Begriff Software ebenfalls als Sammelbegriff alle Arten von Programmen. Die Software eines Mikrorechnersystems ist im Minimalfall das Steuerprogramm, sie kann aber ebenfalls die Nutzerprogramme einschließen. Bewußt wird eine Verbindung zwischen beiden Sammelbegriffen durch den Begriff Firmware zum Ausdruck gebracht. Dabei handelt es sich um vom Systemhersteller in Form von Hardware dem Nutzer zur Arbeit mit seinem erworbenen System notwendige mitgelieferte Betriebsmittel (z. B. das in einem Festwertspeicher abgelegte Steuerprogramm).

Als Betriebssystem (engl. operating system) wird in diesem Zusammenhang ein Programmpaket bezeichnet, das die Steuerung rechnerinterner Aktivitäten übernimmt. Dazu gehört im allgemeinen die Organisation der Abarbeitung von Nutzerprogrammen, die Verwaltung aller Systemkomponenten, die Behandlung auftretender Hardware- und zum Teil Software-Fehler und die softwaremäßige Steuerung der Kommunikation des Rechners mit der Umwelt, allgemein auch als Peripherie bezeichnet. Besonders aus diesem letztgenannten Aufgabengebiet erwachsen an das Betriebssystem auf Grund der dabei verstärkt notwendigen Behandlung eintreffender Bearbeitungswünsche besondere Anforderungen. Die ganze Komplexität der als Unterbrechungsbehandlung (UB-Behandlung, engl. interrupt handling) bezeichneten Arbeit wird deutlich, wenn man die Rolle eines Mikrorechners im Einsatzfall der Prozeßsteuerung näher betrachtet. Für jeden am Mikrorechner einlaufenden zulässigen Unterbrechungswunsch ist zunächst der Status der aktuell bearbeiteten Aufgabe sicherzustellen, um später nach abgeschlossener UB-Behandlung am richtigen Programmpunkt mit gültigen Daten eine Abarbeitung fortsetzen zu können. Danach ist die Quelle der UB-Anmeldung zu ermitteln und aus dem Ergebnis bzw. sofort aus der Form der UB-Anmeldung gezielt eine Software-Routine zur Behandlung des UB-Wunsches einzuleiten. Die Größe der vom Zeitpunkt einer UB-Anmeldung bis zum tatsächlichen Beginn der Behandlung durch die CPU verstreichenden Zeit, auch als Reaktionszeit bezeichnet, kann als Gütekriterium der Arbeitsorganisation innerhalb der Rechnersysteme angesehen werden.

Für die Gestaltung der Kommunikation eines Mikrorechners mit seiner Umwelt wird eine sogenannte Schnittstelle (engl. interface) definiert. An diesem Punkt wird von einer Seite das Anschlußbild der Peripherie zur CPU und umgekehrt das der CPU an den Peripherieanschluß angepaßt. Im allgemeinen wird diese Aufgabe in erster Linie hardwaremäßig durch Einsatz spezieller elektronischer Schaltungen, sogenannter Interface-Bausteine, gelöst. Es sei jedoch bereits an dieser Stelle auf den Abschnitt 2.3. verwiesen, in dem gezeigt wird, daß auf der Basis einer universell gestalteten Hardware mit Software prinzipiell jede Anpassung Mikrorechner-Peripherie ausgeführt werden kann.

Soll ein Mikrorechner zur Prozeßüberwachung und -steuerung eingesetzt werden, so sind über ein entsprechendes Prozeß-Interface ständig Informationen zwischen Rechner und zu steuernder Anlage auszutauschen. Die dabei mit dem Mikrorechner realisierte Betriebsart wird als Echtzeitbetrieb bezeichnet (engl. real time processing).

Wird ein Mikrorechner zur Lösung klassischer rechentechnischer Aufgaben eingesetzt, so ist er über Interfaces im allgemeinen mit folgenden peripheren Geräten verbunden:

- Lochbandleser
- Lochbandstanzer
- Fernschreiber
- Drucker, Plotter
- Floppy Disk
- Bildschirmeinheit mit Tastatur.

Nicht zuletzt daran werden gleichzeitig die engen Verbindungen zur bereits bekannten Rechentechnik deutlich. Eine Gegenüberstellung der Architektur konventioneller Klein- und Prozeßrechner mit der von Mikrorechnern läßt ebenfalls keine prinzipiellen Unterschiede erkennen. So tauchen auch nahezu alle in der bisherigen Entwicklung der EDV aufgetretenen Probleme bei Mikrorechnersystemen wieder auf,

und umgekehrt sind für Fragen der Mikroprozessortechnik Analogielösungen im Bereich der Rechentechnik zu finden. Sofern man überhaupt von Besonderheiten der μ P-Technik sprechen will, ist vielleicht gegenwärtig nur die Tatsache hervorzuheben, daß zum Zweck einer effektiven Realisierung gestellter Aufgaben mit Mikroprozessoren die Software-Erstellung in der unmittelbaren Laboratmosphäre einen Vorrang besitzt. Dieser Umstand ist mit den für optimale Lösungen unbedingt notwendigen Hardware-Kenntnissen und -Eingriffsmöglichkeiten in die aktuelle Systemkonfiguration zu begründen.

Die folgende Zusammenstellung der wichtigsten Abkürzungen soll das Verständnis der weiteren Ausführungen erleichtern.

A, AC, ACC	Accumulator, Akkumulator
A.C.	
Characteristics	Alternating Current Characteristics, dynamische Kennwerte
ADC	Analog Digital Converter, Analog Digital-Umsetzer
ALU	Arithmetic Logic Unit, arithmetisch-logische Einheit
ASCII	American Standard Code for Information Interchange, Amerikanischer Standard-Kode für Informationsaustausch
BCD	Binary Coded Decimal, dual verschlüsselte Dezimalzahl
CCD	Charge Coupled Devices, Ladungskopplungsstrukturen
CFF	Carry Flip Flop, Übertrags-Flip-Flop
CLK	Clock, Takt
CMOS	Complementary MOS, komplementäre MOS-Technologie
CPE	Central Processing Element, Zentrales Verarbeitungselement (Aufbauelement der ZVE von Bit-Slice-Mikroprozessoren)
CPU	Central Processing Unit, Zentrale Verarbeitungseinheit (ZVE)
CRC	Cyclic Redundancy Check, zyklische Redundanzprüfung, zyklische Datenblockprüfung
CRT	Cathode Ray Tube, Katodenstrahlröhre (Bildschirm)
CTC	Counter Timer Circuit, Zähler- und Impulsgeber-Baustein
DAC	Digital Analog Converter, Digital/Analog-Umsetzer
D.C.	
Characteristics	Direct Current Characteristics, statische Kennwerte
DIL, DIP	Dual In Line Package, Schaltkreisgehäuse mit zwei Anschlußstiftreihen, DIL-Gehäuse
DMA	Direct Memory Access, direkter Speicherzugriff, direkter Speicherkanal (DSK)
DOS	Disk Operating System, Plattenbetriebssystem
DSK	direkter Speicherkanal (s. a. DMA)
E/A	Eingabe/Ausgabe (s. a. I/O)
EAROM	Electrically Alterable ROM, elektrisch veränderbarer Festwertspeicher
EBCDIC	Extended Binary Coded Decimal Interchange Code, erweiterter dualverschlüsselter Dezimalcode für Informationsaustausch (8-Bit-Kode)
ECL	Emitter Coupled Logic, emittergekoppelte Logik (Stromschaltlogik)
EPROM	Erasable Programmable ROM, lösch- und programmierbarer Festwertspeicher
FIFO	First In First Out, Speicherelementprinzip: Das zuerst in den Speicher übernommene Datenwort wird zuerst wieder ausgegeben (z. B. Schieberegister)
FPLA	Field Programmable Logic Array, elektrisch (vom Anwender) programmierbares logisches Feld
HS	Hauptspeicher, Arbeitsspeicher
IC	Integrated Circuit, integrierte Schaltung
ICE	In-Circuit Emulator, Firmware zur Kopplung und zum Test des Anwendersystems mit einem Entwicklungssystem
ICU	Interrupt Control Unit, Unterbrechungssteuereinheit
IIL, I ² L	Integrated Injection Logic, Integrierte Injektionslogik
I/O	Input/Output, Eingabe/Ausgabe (B/A)
IR	Instruction Register, Befehlsregister
IS	Integrierter Schaltkreis, integrierte Schaltung
K	Definition: $K = 2^{10} = 1\,024$, steht in Verbindung mit bit, Byte oder Wort
LCG	Look-Ahead Carry Generator, Einheit zur vorausschauenden Übertragsbildung
LIFO	Last In First Out, Speicherelementprinzip: Das zuletzt in den Speicher übernommene Datenwort wird zuerst wieder ausgegeben (Kellerspeicher)
LSB	Least Significant Bit, niederwertigstes Bit (niedrigster Stellenwert)
LSI	Large Scale Integration, hoher Integrationsgrad (mehr als 1 000 integrierte Elemente je Chip)
M	Definition: $M = 2^{20} = 1\,048\,576$, steht in Verbindung mit bit, Byte oder Wort
MCU	Microprogram Control Unit, Mikroprogramm-Steuereinheit
MISFET	Metal Isolator Semiconductor Field Effect Transistor, Metall-Isolator-Halbleiter-Feldeffekttransistor

Wird fortgesetzt

Dr.-Ing. STEFAN LINDNER

MNOS	Metal Nitride Oxide Semiconductor, Metall-Nitrid-Oxid-Halbleiter-Technologie
MODEM	Modulator/Demodulator
MOS	Metal Oxide Semiconductor, Metall-Oxid-Halbleiter-Technologie
MOSFET	Metal Oxide Semiconductor Field Effect Transistor, Metall-Oxid-Halbleiter-Feldeffekttransistor
MPU	Micro Processing Unit, Mikroverarbeitungseinheit ($\approx$ ZVE)
MSB	Most Significant Bit, höchstwertiges Bit (höchster Stellenwert)
MSI	Medium Scale Integration, mittlerer Integrationsgrad (100 bis 1 000 integrierte Elemente je Chip)
MTBF	Mean Time Between Failures, mittlere fehlerfreie Betriebszeit (zwischen zwei Fehlern)
MTC	Magnetic Tape Cassette, Magnetbandkassette
NMOS	n-Kanal-MOS-Technologie
O.C. Output	Open Collector Output, offener Kollektorausgang
Opcode	Operation Code, Operationskode
OS	Operating System, Betriebssystem
PC	Program Counter, Programmzähler, Befehlszähler
PCB	Printed Circuit Board, gedruckte Schaltung, Leiterplatte
PCI	Programmable Communication Interface, programmierbare Kommunikationsschnittstelle
PIA	Peripheral Interface Adapter, peripherer Schnittstellen-Anpaßbaustein
PIO	Parallel Input Output, Baustein zur Parallel-Ein- und Ausgabe von Daten
PLA	Programmable Logic Array, programmierbares logisches Feld, programmierbare logische Anordnung (s. a. FPLA)
PL/M	Programming Language for Microprocessors, Programmiersprache für Mikroprozessoren
PMOS	p-Kanal-MOS-Technologie
PPI	Programmable Peripheral Interface, programmierbare Peripherieschnittstelle
PROM	Programmable ROM, (vom Anwender) programmierbarer Festwertspeicher (s. a. ROM)
PSW	Program Status Word, Programmzustandswort
RALU	Register and Arithmetic Logic Unit, arithmetisch-logische Einheit mit Registersatz
RAM	Random Access Memory, Speicher mit wahlfreiem Zugriff, heute meist (Halbleiter-)Schreib-/Lesespeicher mit wahlfreiem Zugriff
REPROM	(Electrically Erasable and) Reprogrammable ROM, lösch- und wieder programmierbarer Festwertspeicher (s. a. ROM, EPROM)
ROM	Read Only Memory, Nur-Lese-Speicher, Festwertspeicher
SBC	Single Board Computer, Ein-Karten-Rechner
Sif, SI	Standard-Interface, Standardschnittstelle
SIO	Serial Input Output, Bausteine für serielle Ein- und Ausgabe von Daten
SOS	Silicon On Sapphire, MOS-Technologie, wobei als Halbleitersubstrat „Silizium auf Saphir“ verwendet wird
SP	Stack Pointer, Stapelzeiger, Kellerzeiger
SSI	Small Scale Integration, niedriger Integrationsgrad (bis zu 100 integrierten Elementen je Chip)
T.S. Output	Three State Output, tristabiler Ausgang, Dreizustandsausgang
TTL	Transistor-Transistor-Logik
TTY	Teletypewriter, Fernschreiber
UART	Universal Asynchronous Receiver/Transmitter, universeller Asynchron-Empfänger/Sender
UB	Unterbrechung, Interrupt
USART	Universal Synchronous/Asynchronous Receiver/Transmitter, universeller Synchron-/Asynchron-Empfänger/Sender
USRT	Universal Synchronous Receiver/Transmitter, universeller Synchron-Empfänger/Sender
VLSI	Very Large Scale Integration, sehr hoher Integrationsgrad (10^5 bis 10^6 integrierte Elemente je Chip)
W	Word, Wort
ZE	Zentraleinheit
ZVE	Zentrale Verarbeitungseinheit (s. a. CPU)
„C	Microcomputer, Mikrorechner
„P	Mikroprozessor

1.2. Schaltkreisfamilien

Da der künftige Anwender von Mikrorechnern gezwungen ist, sich aus dem vorhandenen Bausteinsortiment und meist zusätzlichen Schaltkreisen sein eigenes Mikrorechnersystem aufzubauen, sollte er neben den technischen Parametern auch den prinzipiellen Aufbau, die Wirkungsweise sowie Vor- und Nachteile der wichtigsten Schaltungstechniken kennen. Gegenwärtig sind folgende Techniken vorherrschend: TTL, Schottky-TTL, ECL und I^2L als bipolare Familien und p-MOS, n-MOS sowie CMOS als Vertreter der MIS-Strukturen.

Bild 1.2.1. zeigt eine Einteilung der digitalen Schaltkreisfamilien. Schaltungen mit konstantem logischem Gehalt können mit bipolaren oder unipolaren (MOS-)Transistoren realisiert werden. Wesentliche Vertreter der bipolaren Übersteuerungstechnik sind die DCTL (Direktgekoppelte Transistorlogik), die RTL (Widerstands-Transistor-Logik), die DTL (Dioden-Transistor-Logik), TTL und Schottky-TTL sowie die I^2L . Für die Mikroprozessortechnik sind die letzten drei Familien bedeutsam. Die Stromschalttechnik wird durch die Familie der ECL (Emittergekoppelte Logik) vertreten. Für MOS-Schaltkreise kommt nur das Übersteuerungsprinzip zur Anwendung. Das Stromschaltprinzip bringt hier keinen Vorteil, sondern ist nur mit höherem Aufwand verbunden. Dynamische MOS-Schaltungen sind insbesondere für die Realisierung von Speichern, Schieberegistern u. ä. interessant. Dabei unterscheidet man zwischen den dynamischen Schaltungen in Ein- oder Mehrphasentechnik, die die Möglichkeit einer kurzzeitigen Ladungsspeicherung auf der Torelektrode eines MOSFET nutzen, und den Ladungstransfer-Stufen, deren Funktionsweise auf einem von MOS-Elektroden gesteuerten Transport von Minoritätsträgerladungen entlang der Oberfläche eines Halbleiters beruht. Ladungstransferstrukturen wie CCD (Charge Coupled Devices), SCT (Surface Charge Transistor) und BBD (Bucket Brigade Devices) werden im weiteren nicht betrachtet.

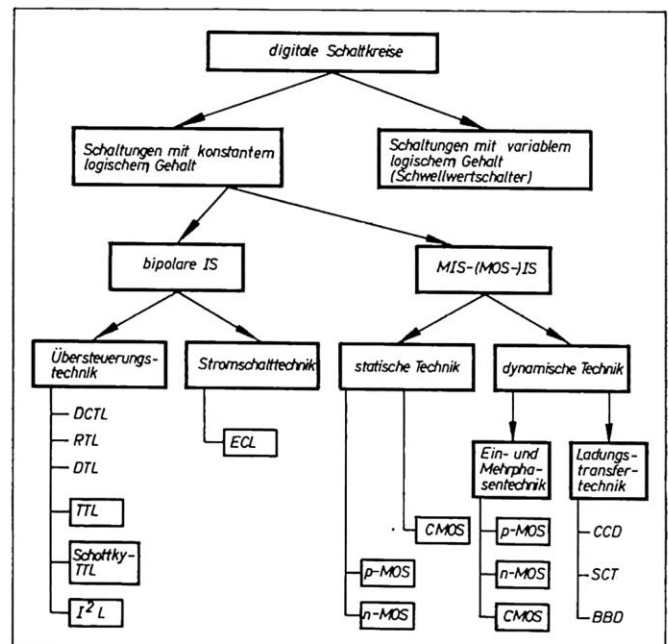


Bild 1.2.1.: Einteilung digitaler Schaltkreisfamilien

1.2.1. Bipolare Schaltkreise

1.2.1.1. Übersteuerungs- und Stromschalttechnik

Bild 1.2.2. zeigt das Grundprinzip des Übersteuerungsschalters. Er arbeitet als Ein-Ausschalter. Im Aus-Zustand ist der Transistor gesperrt. Bei Vernachlässigung des Kollektorreststromes ist $U_A = U_{CC}$. Im Ein-Zustand befindet sich der Transistor im Übersteuerungsbereich $U_A = U_{CEX}$ (U_{CEX} = Kollektor-Emitter-Restspannung). Damit ist der Spannungshub am Ausgang

$$\Delta U_A = \Delta U_{CE} = U_{CC} - U_{CEX} \approx U_{CC} \quad (1.2.1.)$$

annähernd gleich der Betriebsspannung U_{CC} . Die Folge ist eine gute statische Störsicherheit (Störabstand). Der Basisstrom im Ein-Zustand und damit der Übersteuerungsgrad wird wesentlich durch R_K bestimmt. Mit $-U_{BB}$ über R_B wird ein sicheres Sperren des Transistors bei $U_E \approx 0V$ erreicht.

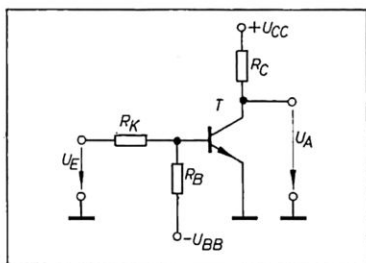


Bild 1.2.2.: Übersteuerungsschalter

Bei ausgangsseitigem Leerlauf wird im Aus-Zustand die Spannungsquelle U_{CC} praktisch nicht belastet ($I_C \approx 0$), im Ein-Zustand fließt der Strom

$$I_C \approx \frac{U_{CC}}{R_C} \quad (1.2.2.)$$

Die mittlere Verlustleistung (definiert für ein Tastverhältnis von 1 : 1) ergibt sich damit zu

$$P_V = \frac{1}{2} (P_{V_{Ein}} + P_{V_{Aus}}) \approx \frac{1}{2} \frac{U_{CC}^2}{R_C} \quad (1.2.3.)$$

Für das dynamische Verhalten des Übersteuerungsschalters sind der Übersteuerungsgrad

$$m = \frac{\text{Basisstrom bei Übersteuerung}}{\text{Basisstrom an der Übersteuerungsgrenze } (U_{CB} = 0)} \quad (1.2.4.)$$

und der Ausschaltfaktor

$$k = \frac{|\text{Basisstrom zum Ausschaltzeitpunkt}|}{\text{Basisstrom an der Übersteuerungsgrenze}} \quad (1.2.5.)$$

Tafel 1.2.1.: Schaltzeiten von k und m (s. Bild 1.2.3.)

Verzögerungszeiten	t_d	t_a	t_s	t_f
Übersteuerungsgrad m	$\sim 1/m$	$\sim 1/m$	$\sim m$	—
Ausschaltfaktor k	$\sim k$	—	$\sim 1/k$	$\sim 1/k$

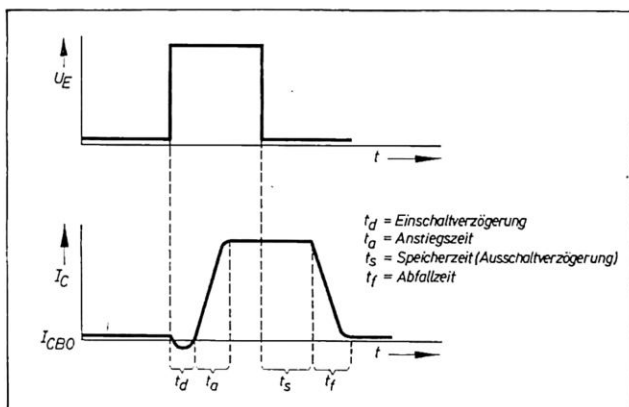


Bild 1.2.3.: Zeitliches Verhalten des Übersteuerungsschalters

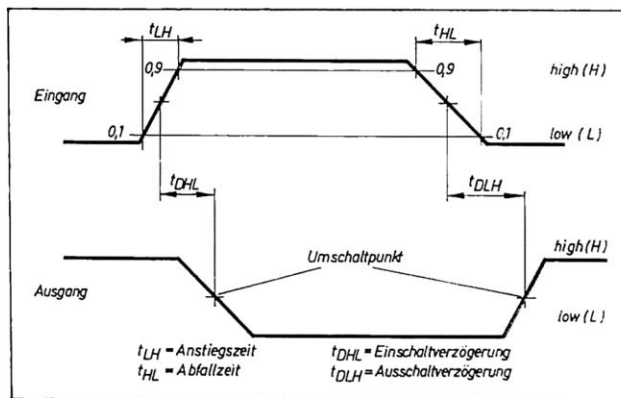


Bild 1.2.4.: Definition der Schaltzeiten

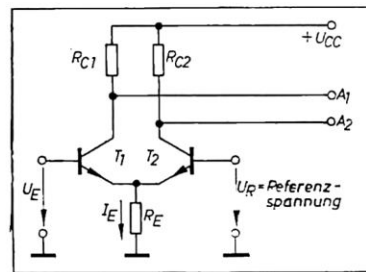


Bild 1.2.5.: Stromschalter

von Bedeutung. Bild 1.2.3. zeigt das zeitliche Verhalten des Kollektorstromes I_C in Abhängigkeit von U_E , Tafel 1.2.1. gibt einen Überblick über die Abhängigkeit der im Bild 1.2.3. dargestellten Schaltzeiten von k und m.

Für den Übersteuerungsschalter ist besonders die Speicherzeit t_s von Bedeutung, da in ihr die durch Übersteuerung hervorgerufene überschüssige Basisladung abgebaut werden muß. Die Einschaltverzögerung t_d ist dagegen vernachlässigbar.

Für reale technische Systeme werden diese Verzögerungszeiten meist entsprechend Bild 1.2.4. definiert. Die Anstiegszeit t_{LH} der L/H-Flanke und die Abfallzeit t_{HL} der H/L-Flanke werden zwischen 10 % und 90 % des Endwertes definiert. Dabei ist t_{DHL} die Verzögerung der H/L-Flanke, bezogen auf den Ausgang, also die Einschaltverzögerung des Übersteuerungstransistors, und t_{DLH} die Verzögerung der ausgangsseitigen L/H-Flanke, also die Ausschaltverzögerung des Übersteuerungstransistors. Die mittlere Verzögerungszeit oder die Durchlaufverzögerung eines Schaltkreises ergibt sich dann zu

$$t_v = \frac{1}{2} (t_{DHL} + t_{DLH}) \quad (1.2.6.)$$

Bild 1.2.5. zeigt die Grundsaltung des bipolaren Stromschalters. Sie entspricht der Schaltkreisfamilie der ECL. Charakteristisch für diese Familie sind die zwei emittergekoppelten Transistoren T_1 und T_2 , wobei T_1 leitend und T_2 gesperrt ist, wenn $U_E > U_R$ ist, T_1 ist gesperrt und T_2 leitend, wenn $U_E < U_R$ ist.

Für $R_{C1} = R_{C2} = R_C$ fließt in beiden Schaltzuständen annähernd der gleiche Speisestrom

$$I_E \approx \frac{U_{CC}}{R_E + R_C} \quad (1.2.7.)$$

Dadurch entsteht eine höhere Verlustleistung als in der Übersteuerungstechnik:

$$P_V = \frac{1}{2} (P_{V_{Ein}} + P_{V_{Aus}}) = U_{CC} I_E \quad (1.2.8.)$$

Wird fortgesetzt

Dr.-Ing. STEFAN LINDNER

Infolge der gegenkoppelnden Wirkung von R_E erhält man von den Transistorparametern (insbesondere Stromverstärkungsfaktoren) nahezu unabhängige Ausgangsgrößen U_{A1} und U_{A2} . Man braucht deshalb die Transistoren nicht im Übersteuerungsbereich zu betreiben. Daraus folgt der wesentlichste Vorteil der Stromschalttechnik, denn die Verzögerungszeiten sind infolge fehlender Speicherzeit t_s kleiner als in der Übersteuerungstechnik. Damit verbunden ist allerdings die Bedingung, daß der Eingangsspannungshub

$$\Delta U \leq U_{BEF} \quad (1.2.9.)$$

ist, wobei U_{BEF} die Flußspannung der Basis-Emitterdiode von T_1 ist ($\approx 0,7 \dots 0,8 \text{ V}$). Werden die Transistoren genau an der Übersteuerungsgrenze ($U_{CB} = 0$) betrieben, sind die Ausgangspegel spannungsmäßig genau um ΔU gegenüber den Eingangspegeln verschoben.

Der Vergleich von Übersteuerungs- und Stromschalttechnik liefert folgende Aussagen:

- größerer Bauelementeaufwand bei der Stromschalttechnik
- größere Verlustleistung bei der Stromschalttechnik
- größere Verzögerungs-, Anstiegs- und Abfallzeiten bei der Übersteuerungstechnik
- größerer Spannungshub, damit größerer statischer Störabstand der Übersteuerungstechnik
- Vorhandensein von negierten und nichtnegierten Ausgängen bei der Stromschalttechnik
- Impulsbelastung der Versorgungsspannung bei der Übersteuerungstechnik (Erzeugung von Störsignalen).

1.2.1.2. Transistor-Transistor-Logik

Die TTL ist auf Grund ihrer vielen Vorteile und ihrer Typenvielfalt die z. Z. am häufigsten angewendete digitale Schaltkreisfamilie. Wegen ihrer hohen Arbeitsgeschwindigkeit ist sie auch für die Mikroprozessortechnik von Interesse. Bild 1.2.6. zeigt ein TTL-Grundgatter. Charakteristisch für die TTL ist der Mehremittertransistor T_1 , der die UND-Verknüpfung der Eingangssignale vornimmt. Durch die folgende Negation mit T_2 wird die NAND-Funktion realisiert. Zur weiteren Verbesserung der Störsicherheit und der Aus-

gangsbelastbarkeit wird die Grundschialtung meist durch Endstufen ergänzt.

Den TTL-Schaltkreisen kommt gegenwärtig in der Mikroprozessor- und Mikrorechnertechnik insbesondere als Zusatzaufbausteine zum Aufbau kompletter Mikrorechnersysteme eine entscheidende Bedeutung zu. Auf Grund der weiten Verbreitung, der Typenvielfalt und der Existenz mehrerer TTL-Familien sind die Mikroprozessorerhersteller bei ihren LSI-Schaltkreisen insbesondere auf TTL-Kompatibilität bedacht.

Der Anwender sollte dabei aber immer auf die spezifischen Einschränkungen der TTL-Kompatibilität achten. So kann z. B. der Mikroprozessor U 808 nur Low-Power-TTL-Schaltkreise treiben (Lastfaktor $= 1/4$ der TTL-Last).

Eine weitere Erhöhung der Schaltgeschwindigkeit von TTL-Schaltkreisen erreicht man durch sogenannte Klemmdioden über der Basis-Kollektorstrecke des Übersteuerungstransistors (Bild 1.2.7.), indem ein Teil des überschüssigen Basisstroms zum Kollektor geleitet wird. Außerdem wird durch die Begrenzereigenschaft der Diode nur eine geringe Übersteuerung entsprechend der Flußspannung der Diode zugelassen. Die Klemmdiode muß schneller als der Transistor schalten. Man verwendet Schottky-Dioden mit einem Metall-Halbleiter-Übergang, die außerdem noch den Vorteil einer geringen Flußspannung ($U_F \approx 0,4 \text{ V}$) besitzen, so daß eine geringe Übersteuerung erreicht wird. Mit derartigen Dioden wird die Schottky-TTL realisiert, die sich durch extrem geringe Schaltzeiten auszeichnet.

Tafel 1.2.2. gibt einen Überblick über die wichtigsten Para-

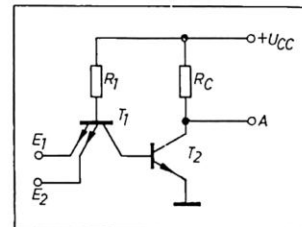


Bild 1.2.6.:
TTL-Grundgatter

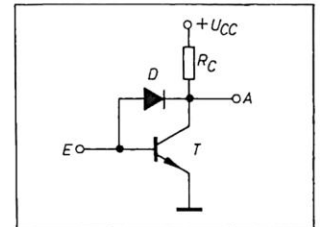


Bild 1.2.7.:
Schaltung mit Klemmdiode

Tafel 1.2.2.: Wesentliche Parameter der TTL-Familien bei $U_{CC} = 5 \text{ V}$, $\vartheta_a = 25^\circ \text{C}$

TTL-Familie	Impulsflankendauer in ns		mittlere Verzögerungszeit in ns	Meßbedingungen C_L	R_L	mittlere Verlustleistung	Geschwindigkeits-Leistungs-Produkt in pJ	Eingangslastfaktor ¹⁾	Ausgangslastfaktor ¹⁾	statischer Störabstand in V	Vergleichsreihen
	t_{LH}	t_{HL}			in pF	in mW					
74	10	5	10	15	400	10	100	1	10 (30)	L: 1,2 H: 1,9	D 10 (DDR) MH 74 (ČSSR) K 155 (UdSSR)
	(20...80 % bei 3,4 V)										
74 L	25	25	33	50	4 000	1	33	0,25	5	L: 1,2 H: 1,8	K 158 (UdSSR)
	(10...90 % bei 3 V)										
74 H	7	7	6	25	280	22	132	1,25	12,5	L: 1,2 H: 2,1	D 20 (DDR) K 131 (UdSSR)
	(10...90 % bei 3 V)										
74 S	2,5	2,5	3	15	280	19	57	1,25	12,5	L: (0,9) H: 2,0	MH 74 S (ČSSR) TL 74 S (UVR)
	(10...90 % bei 3 V)										
74 LS	15	6	9,5	15	2 000	2	19	0,5	5	L: 1,05 H: 2,0	—
	(10...90 % bei 3 V)										

¹⁾ bezogen auf Serie 74

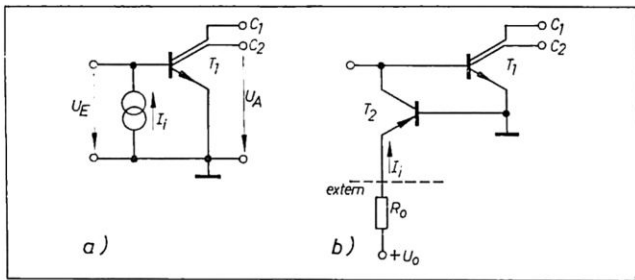


Bild 1.2.8.: I²L-Grundgatter. a) Prinzip; b) Ersatzschaltbild

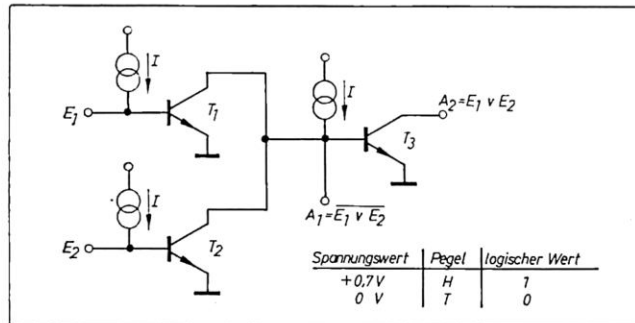


Bild 1.2.9.: NOR/OR-Gatter in I²L-Technik

meter von NAND-Gattern der fünf TTL-Familien (Standard-Serie 74, Low-Power-Serie 74 L, High-Speed-Serie 74 H, Schottky-Serie 74 S, Low-Power-Schottky-Serie 74 LS).

1.2.1.3. Integrierte Injektionslogik

Eine junge bipolare Schaltungstechnik ist die I²L-Technik (auch IIL = Integrated Injection Logic, integrierte Injektionslogik), die anstelle der in anderen bipolaren Techniken notwendigen passiven Widerstände mit aktiven Konstantstromquellen arbeitet. Auf Grund des einfachen Aufbaus (nur ein Transistor notwendig) wird in dieser Technik ein extrem hoher Integrationsgrad erreicht. Der Flächenbedarf eines I²L-Gatters beträgt etwa nur $\frac{1}{10}$ der Fläche eines entsprechenden TTL- oder CMOS-Gatters und ist auch noch geringer als der der hochentwickelten p-MOS- oder n-MOS-Technik. Aus diesem Grund ist sie für LSI-Schaltungen sehr interessant.

Bild 1.2.8. zeigt die I²L-Grundschialtung, einen Inverter. Der npn-Transistor T_1 arbeitet als Inverter, der von einer Konstantstromquelle (Injektor) gespeist wird. Die Konstantstromquelle kann extern durch Bestrahlung mit Licht (I_i = Fotostrom) oder intern durch einen lateralen pnp-Transistor T_2 realisiert werden (Bild 1.2.8.b), wobei der Strom I_i aus einer externen Spannungsquelle U_0 über einen externen Arbeitswiderstand R_0 gewonnen wird. Der Schalttransistor T_1 kann als Multikollektortransistor realisiert werden, dessen Ausgänge einfache Wired-OR-Verknüpfungen zu nachfolgenden Gattern ermöglichen. Je nach Geschwindigkeitsanforderungen können bis zu 50 Kollektorausgänge vorgesehen werden. Bild 1.2.9. zeigt eine einfache logische Verknüpfung in Form eines NOR/OR-Gatters. Liegt mindestens ein Eingang auf H-Potential ($\approx 0,7$ V), so ist einer der Transistoren (oder beide) leitend, wobei der Strom I in die Basis von T_1 bzw. T_2 eingespeist wird. Der gemeinsame Kollektorpunkt wird auf L-Potential gezogen, und am Ausgang A_1 wird die NOR-Funktion realisiert. Der Stromgenerator an der Basis von T_3 wird damit kurzgeschlossen. Der Strom fließt somit nicht in die Basis von T_3 , sondern durch den jeweils leitenden Transistor T_1 bzw. T_2 . Der Ausgangstransistor T_3 ist damit gesperrt, so daß A_2 die Negation von A_1 ergibt.

Den einfachen Aufbau eines RS-Flip-Flops in I²L-Technik verdeutlicht Bild 1.2.10. nach [1.2.6.]. Auf dem n^+ -Substrat sind nach Aufwachsen einer epitaxialen n -Schicht nur zwei Diffusionen für die p-Gebiete (Basis) und die n^+ -Gebiete (Kollektoren) erforderlich. Die Verbindung der Elemente erfolgt durch eine Metallisierungsschicht.

Außer dem Injektorstrom benötigen I²L-Gatter keine weiteren Strom- bzw. Spannungsquellen, wobei ein Injektor theoretisch eine unbegrenzte Anzahl von Transistoren versorgen

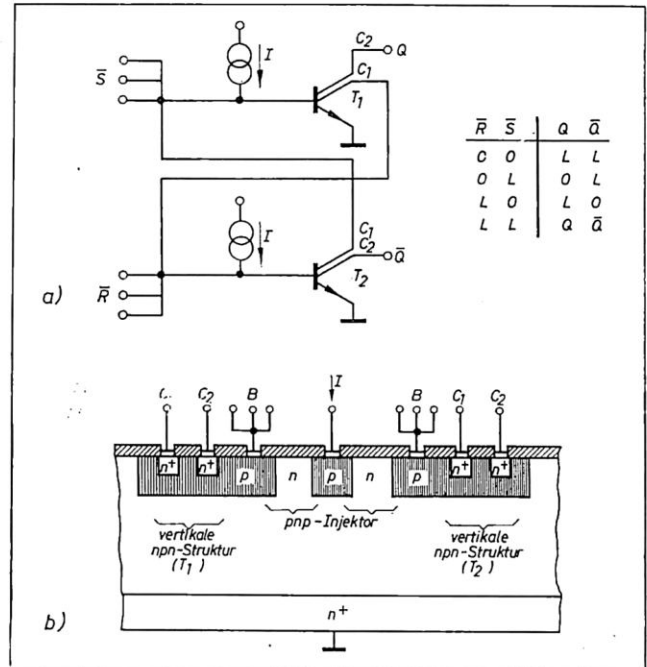


Bild 1.2.10.: RS-Flip-Flop in I²L-Technik. a) Schaltung; b) Querschnitt

kann. Die Strom-Spannungscharakteristik entspricht einer in Durchlaßrichtung betriebenen Silizium-Diode, so daß nur eine kleine Versorgungsspannung von etwa 0,85 V erforderlich ist. Durch geringe Änderung der Injektorspannung (um einige 100 mV) können der Injektorstrom und damit die Verzögerungszeiten um mehrere Größenordnungen variiert werden. Bei einem Geschwindigkeits-Leistungsprodukt von etwa 0,06...2 pJ können neueste I²L-Schaltungen sowohl im Kleinstleistungsbereich (einige nW) als auch im Hochgeschwindigkeitsbereich (5...20 ns) eine breite Anwendung finden.

Es ergeben sich folgende Vorteile der I²L-Technik:

- hohe Packungsdichte (100 bis 300 Gatter/mm²), bedingt durch das einfache Eintransistorgatter ohne Widerstände und Isolation der Einzelelemente
- wenige Herstellungsschritte (zwei Diffusionsschritte, vier Masken, eine Metallisierung)
- geringe Verlustleistung (6 nW...70 μ W) bei geringer Versorgungsspannung (0,85 V)
- hohe Geschwindigkeit (50 ns/Gatter bei 70 μ W) durch kleinen internen Spannungshub und hohe Ladungsträgerbeweglichkeit
- großer Verlustleistungs- und Geschwindigkeitsbereich (über fünf Dekaden) durch Steuerung des Injektorstromes
- einfach zu erreichende TTL-Kompatibilität durch offene Kollektorausgänge und Widerstandsteiler an den Eingängen.

Mit Hilfe der I²L wurde bereits ein 4-Bit-Mikroprozessorelement (Bit-Slice-Prozessor) realisiert [1.2.6.].

1.2.1.4. Emitttergekoppelte Logik (ECL)

Die ECL-Familie ist die gegenwärtig schnellste Schaltungstechnik. Bild 1.2.11. zeigt eine vollständige ECL-Schaltung mit Kollektoranpaßstufen. Die Transistoren T_2 und T_3 bilden den Differenzverstärker mit dem gemeinsamen Emittterwiderstand R_3 . Durch Parallelschaltung von Transistoren (hier T_1) zu T_2 erfolgt eine ODER-Verknüpfung der Eingangssignale. Das temperatur- und spannungskompensierte Netzwerk mit T_4 , D_1 , D_2 und den Widerständen R_6 , R_7 , R_8 sorgt für eine konstante Referenzspannung an T_3 unter den zulässigen Betriebsbedingungen. Die Transistoren T_5 und T_6 bilden zusammen mit den Eingangswiderständen der nachgeschalteten Stufen ($\triangleq R_1$, R_2) Emittterfolger, die der Anpassung zwischen Eingangs- und Ausgangspegeln dienen. Der Ausgang A_1 liefert die logische ODER-Verknüpfung von E_1 und E_2 , der Ausgang A_2 die NOR-Verknüpfung (positive Logik). Es werden typische Gatterverzögerungszeiten von 2 μ s bei

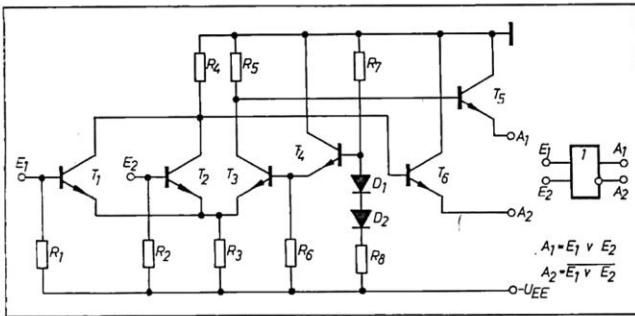


Bild 1.2.11.: Vollständige ECL-Schaltung

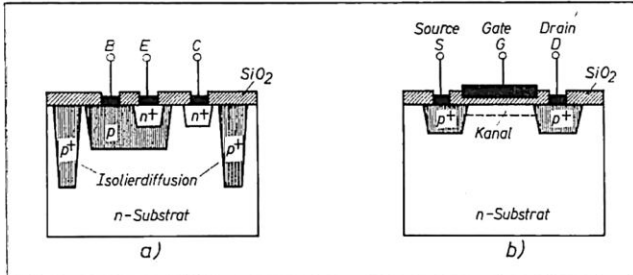


Bild 1.2.12.: Aufbau integrierter Transistoren.
a) Bipolar-Transistor; b) MOSFET

einer Verlustleistung von etwa 25 mW erreicht. Die Verlustleistung ist dabei im Gegensatz zu den TTL-Schaltkreisen frequenzunabhängig. Der Abstand zwischen beiden logischen Niveaus beträgt etwa 0,8 V. Damit verbunden sind eine relativ niedrige statische Störsicherheit ($< 0,4$ V) und keine direkte TTL-Kompatibilität. Auf Grund des großen Aufwands (kleiner Integrationsgrad) wird die ECL in der Mikroprozessortechnik nur bei Hochgeschwindigkeitsforderungen angewendet.

1.2.2. Unipolare Schaltkreise

1.2.2.1. Statische MOS-Schaltkreise

Im Unterschied zur Bipolartechnik wird der Stromtransport bei der MOS-Technik nur durch eine Ladungsträgerart, entweder Elektronen (n-Kanal) oder Defektelektronen (p-Kanal) bewirkt, wobei es sich um Majoritätsladungsträger handelt. MOS-Schaltkreise sind damit strahlungsresistenter als bipolare. Auf Grund des Feldeffektes erfolgt, statisch gesehen, eine leistungslose Steuerung der MOSFETs, wodurch ein geringerer Leistungsbedarf der MOS-Schaltkreise erzielt wird. Feldeffekttransistoren sind sehr hochohmig. Das hat einer-

seits den Nachteil, daß im Vergleich zu den niederohmigen Bipolartransistoren größere Schaltzeiten auftreten (die Schaltkapazitäten werden langsamer umgeladen), aber andererseits die Vorteile, daß eine direktgekoppelte Transistorlogik (DCTL) ohne zusätzliche Dioden oder Widerstände möglich ist und daß notwendige Lastwiderstände durch Feldeffekttransistoren ersetzt werden können, deren Flächenbedarf wesentlich geringer als der äquivalenter integrierter Widerstände ist. Hinzu kommt, daß ein MOSFET auf Grund des einfachen Aufbaues (nur Diffusion für Source und Drain erforderlich) mit geringem technologischen Aufwand hergestellt werden kann und etwa nur $1/5$ bis $1/10$ des Flächenbedarfes eines Bipolartransistors benötigt. Außerdem sind MOSFETs auf Grund ihres Aufbaues gegenüber der Umgebung selbst isolierend, während bei bipolaren Transistoren zusätzliche Isolierdiffusionen (z.B. von gesperrten pn-Übergängen beim Epitaxie-Doppeldiffusionsverfahren) notwendig sind, die den technologischen Aufwand weiter erhöhen.

Daraus ergibt sich der wesentlichste Vorteil der MOS-Schaltkreise, der insbesondere für die Mikroprozessortechnik von Interesse ist: Auf Grund des geringen Flächenbedarfs in Verbindung mit einer niedrigen Verlustleistung (Wärmeabfuhr!) ist eine sehr hohe Packungsdichte, ein hoher Integrationsgrad bei niedrigen Kosten je Schaltungsfunktion möglich.

Bild 1.2.12. verdeutlicht noch einmal den wesentlichen Unterschied im Aufbau zwischen einem Bipolar- und einem MOS-Feldeffekttransistor.

1.2.2.1.1. Einkanaltechnik

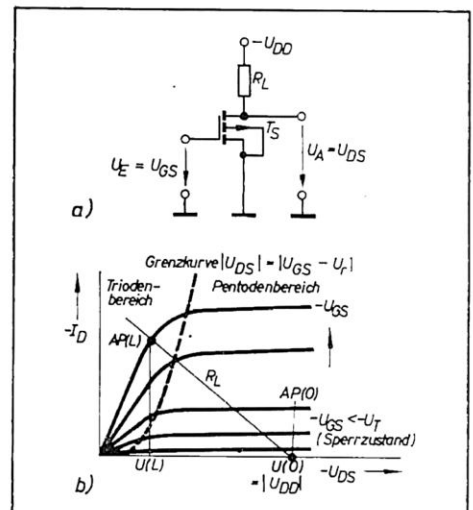
In der Einkanaltechnik werden nur MOSFETs eines Leitfähigkeitstyps (p- oder n-Kanal) verwendet. Außerdem muß man unterscheiden, ob bei einer Gate-Source-Spannung $U_{GS} = 0$ bereits ein leitender Kanal vorhanden ist (Verarmungstyp, engl. depletion type) oder nicht (Anreicherungstyp, engl. enhancement type).

Bild 1.2.13. gibt einen Überblick über die vier Grundtypen von MOSFETs. Physikalisch bedingt sind an der Grenzschicht Siliziumoxid – Siliziumsubstrat stets feste positive Ladungen im SiO_2 vorhanden, so daß bereits ohne Anliegen einer elektrischen Spannung Elektronen in das Kanalgebiet gezogen werden. Dadurch lassen sich sehr einfach n-Kanal-Verarmungs- und p-Kanal-Anreicherungsmosfet herstellen (nur ein Diffusionsschritt zur Dotierung von Drain und Source erforderlich), während bei den anderen beiden Typen zusätzliche Verfahren wie Kanaldotierung erforderlich sind. Bei der Betrachtung der Transferkennlinien und der Spannungspolaritäten erkennt man, daß die Schwellspannungen U_T bei den Verarmungstypen, d. h. die zur Sperrung

Typ	p-Kanal		n-Kanal	
Merkmal	Verarmungstyp	Anreicherungstyp	Verarmungstyp	Anreicherungstyp
Aufbau ($U_{GS} = 0V$)				
Symbol				
Spannungspolarität				
U_{DS}	-	-	+	+
U_{GS}	+ (-)	-	- (+)	+
U_T	+	-	-	+
Transferkennlinie				

Bild 1.2.13.: Vier Grundtypen der MOSFETs

Bild 1.2.14.: p-MOS-Inverter. a) Schaltung; b) Ausgangskennlinienfeld



der Transistoren notwendigen Gate-Source-Spannungen, die entgegengesetzte Polarität zur Betriebsspannung U_{DS} haben, während bei den Anreicherungstypen nur eine Speisespannungspolarität erforderlich ist. Daraus ergibt sich der wesentliche Vorteil der Anreicherungstypen:

Ausgangs- und Eingangsspannung haben die gleiche Polarität, so daß die Transistorstufen direkt gekoppelt werden können. Digitale Schaltkreise werden deshalb fast ausschließlich mit Schalttransistoren vom p- oder n-Kanal-Anreicherungstyp hergestellt. Das ist auch der Hauptgrund dafür, daß in digitalen Schaltungen keine Sperrschichtfeldeffekttransistoren (Verarmungstypen) eingesetzt werden. Auf Grund der hohen Packungsdichte von MOS-Schaltungen allgemein und der oben erwähnten einfachen Herstellungstechnologie wurden die ersten Mikroprozessoren in p-MOS-Technik hergestellt (z. B. Intel 4004, Intel 8008).

Bild 1.2.14. zeigt das Grundprinzip einer p-MOS-Schaltstufe (Inverter) und das dazugehörige Ausgangskennlinienfeld.

Bei einer Eingangsspannung $U_E = U_{GS} > U_T$ ($-U_{GS} < U_T$) ist der MOSFET gesperrt. Die Ausgangsspannung U_A ist gleich der Betriebsspannung U_{DD} . Durch eine negative Eingangsspannung $U_E = U_{GS} < U_T$ wird der MOSFET leitend. Damit die Ausgangsspannung genügend klein wird ($-U_A < -U_T$), um eine nachfolgende Stufe noch sicher zu sperren, muß der MOSFET im Triodenbereich ($|U_{DS}| < |U_{GS} - U_T|$) arbeiten (entsprechende Wahl von R_L). Dieser Arbeitsbereich zeigt ein etwas gleiches Übertragungsverhalten wie der Übersteuerungsbereich von Bipolartransistoren, aber ohne den dort auftretenden Speicher-effekt. Deshalb bringen MOS-Schaltungen nach dem Stromschaltprinzip keine Vorteile, sondern nur einen höheren Aufwand.

Die Lastwiderstände R_L werden durch Feldeffekttransistoren realisiert. Dabei gibt es nach Bild 1.2.15. drei prinzipielle Möglichkeiten. Technologisch sind die Schaltungen nach Bild 1.2.15 a) und b) am einfachsten zu realisieren, da Schalttransistor T_S und Lasttransistor T_L vom gleichen Typ sind. Der Vorteil einer Betriebsspannung (Variante a) wird dabei durch einen geringen Signalhub und niedrigere Arbeitsgeschwindigkeit gegenüber der Variante (b) erkauft. Die Realisierung des Lastwiderstandes durch einen Verarmungstyp (c) ist mit gezielter Ionenimplantation möglich geworden. Neben dem Vorteil von nur einer Betriebsspannung ist besonders die erhöhte Arbeitsgeschwindigkeit gegenüber den Schaltungen mit Anreicherungs-Lasttransistoren zu nennen.

Das dynamische Verhalten einer MOS-Schaltung läßt sich am besten anhand einer Ersatzschaltung nach Bild 1.2.16. erläutern.

Obwohl Feldeffekttransistoren sehr hohe Grenzfrequenzen zulassen, erreichen reale MOS-Schaltungen nur relativ geringe Schaltgeschwindigkeiten. Infolge der Hochohmigkeit der MOSFETs und der wirksamen Kapazitäten (bis zu

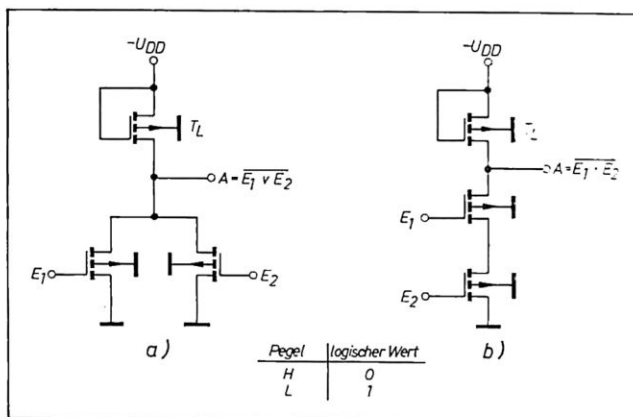


Bild 1.2.17.: p-MOS-Grundgatter. a) NOR-Gatter; b) NAND-Gatter

100 pF) lassen sich nur langsame Umschaltvorgänge erzielen. Die Ersatzkapazität C ergibt sich dabei aus der Ausgangskapazität, der Kapazität der Verbindungsleitungen und der Eingangskapazität der nachfolgenden Stufen. Da der Aufladevorgang von C über den hochohmigen Lastwiderstand R_L (bzw. Lasttransistor T_L) erfolgt, ist die Aufladezeitkonstante größer als die Entladezeitkonstante, wobei die Entladung (Übergang von L- auf H-Pegel am Ausgang) über dem relativ niederohmigen, leitenden Schalttransistor T_S erfolgt. Der Aufladevorgang kann durch einen im Triodenzustand arbeitenden Lasttransistor (zwei Betriebsspannungen oder Verarmungstyp) verkürzt werden. Praktische Verzögerungszeiten liegen in der Größenordnung von 300 bis 1 000 ns.

Auf Grund der direkten Koppelbarkeit der MOSFETs erhält man sehr einfache Verknüfungsprinzipien. Bild 1.2.17. zeigt die beiden Grundgatter NOR und NAND für negative Logik. Die ODER-Verknüpfung wird dabei durch Parallelschaltung und die UND-Verknüpfung durch Reihenschaltung der Schalttransistoren erzielt. Zu beachten ist allerdings die Tatsache, daß mit zunehmender Anzahl von Eingängen (Schalttransistoren) weitere Kapazitäten entstehen, die die Schaltzeiten bedeutend vergrößern können.

Ein wichtiger Gesichtspunkt bei Verwendung von MOS-Schaltungen ist die Anpaßbarkeit an andere Schaltkreisfamilien, besonders an die weit verbreitete TTL-Technik. Eventuell notwendige Anpaßstufen erfordern immer einen relativ großen Aufwand und Platzbedarf, da sie meist in diskreter Technik realisiert werden müssen. Deshalb sollte gerade in der Mikroprozessortechnik TTL-kompatiblen Schaltkreisen grundsätzlich der Vorzug gegeben werden.

Die Anpaßfähigkeit wird dabei hauptsächlich durch das technologische Herstellungsverfahren bestimmt. In [1.2.3.] ist eine Zusammenfassung und Erläuterung der wichtigsten Technologien zu finden. Die Metall-Gate-Technik (auch Hochvolt- oder Standardtechnik) ist das billigste und auch heute noch ein weit verbreitetes Verfahren zur Herstellung von MOS-Schaltungen. Es weist die wenigsten Herstellungsschritte auf. Die Gate-Elektroden bestehen meist aus Aluminium. Infolge der hohen Schwellspannung ($U_T \approx -5$ V bei p-Kanal-Anreicherungstypen) ist eine hohe Betriebsspannung notwendig, so daß TTL-Schaltkreise nur über Anpassungsstufen angeschaltet werden können. Bei der Silizium-Gate-Technik wird eine polykristalline Siliziumschicht als selbstjustierende Gate-Elektrode und auch für Verbindungsleitungen verwendet. Das bedeutet neben den Vorteilen von höheren Packungsdichten und Arbeitsgeschwindigkeiten (infolge Verringerung der Streukapazität durch Selbstjustage) niedrigere Speisespannung und Verlustleistung. Eine direkte TTL-Ansteuerung ist heute in der n-Kanal-Technik möglich. Eine weitere Verbesserung der Bauelementeparameter wird durch Dotierung mit Hilfe der Ionenimplantation erzielt. Hierbei sind insbesondere die wählbare Schwellspannung und damit der weite Speisespannungsbereich zu nennen.

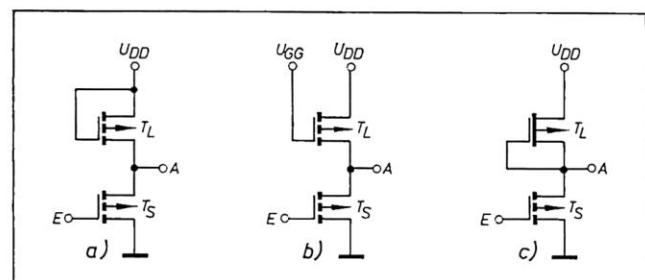


Bild 1.2.15: Lastwiderstände in MOS-Schaltungen.

a) Anreicherungstyp mit einer Betriebsspannung;
b) Anreicherungstyp mit zwei Betriebsspannungen;
c) Verarmungstyp

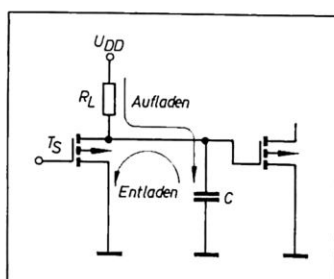


Bild 1.2.16.: Ersatzschaltbild für das dynamische Verhalten von MOS-Schaltungen

Wird fortgesetzt

Dr.-Ing. STEFAN LINDNER

Durch Integration spezieller Anpaßstufen läßt sich aber auch in der p-Kanal-Technik für den Anwender TTL-Kompatibilität erreichen, wobei aber bestimmte Kompromisse eingegangen werden müssen (z. B. mehrere Speisespannungen, Schutzmaßnahmen gegen negative Spannungen).

Bild 1.2.18. zeigt als ein solches Beispiel die Realisierung der Eingangs- und Ausgangsstufen des p-MOS-Mikroprozessors U 808 D vom Kombinat VEB Funkwerk Erfurt [1.2.8.] in Silizium-Gate-Technik. Die Eingangsstufe (a) ist ein Inverter mit dem Schalttransistor T_1 und dem Lasttransistor T_2 . T_3 arbeitet als gesteuerter Widerstand, der den TTL-H-Pegel auf die erforderliche Eingangsspannung anhebt. Die Eingänge sind damit TTL-kompatibel. Die Ausgangsstufe (b) wird als Gegentaktendstufe mittels T_4 und T_5 realisiert (siehe auch Abschnitt 1.2.3.). Ist T_5 leitend, wird eine minimale Ausgangsspannung von 2,4 V bei einem Laststrom $I_{aH} = 0,2$ mA erzielt. Ist T_4 leitend (L-Zustand mit $U_{aL} \leq +0,4$ V bei $I_{aL} = 0,4$ mA), können negative Ausgangsspannungen auftreten, da T_4 gegen -9 V arbeitet. Nachfolgende TTL-Schaltkreise müssen diese durch Eingangsdiolen begrenzen. Die angegebenen Stromwerte lassen erkennen, daß die Ausgänge Low-power-TTL-kompatibel sind.

Modernere Mikroprozessoren (z. B. Intel 8080) werden in n-Kanal-MOS-Technik realisiert. In der prinzipiellen Arbeitsweise gibt es keine Unterschiede zur p-Kanal-Technik. Nach Bild 1.2.13. arbeiten n-Kanal-Anreicherungs-MOSFETs mit positiver Schwellspannung und positiver Betriebsspannung $+U_{DD}$. Die logischen Grundsaltungen für NOR und NAND sind nach Bild 1.2.19. die gleichen wie für die p-MOS-Technik. Die Logik ist positiv. Die Vorteile der n-Kanal-Technik liegen in den erreichbaren Parametern. Zwar

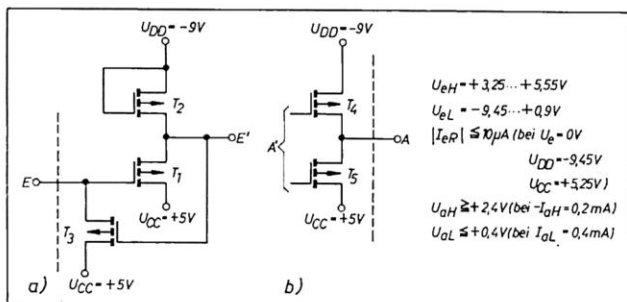


Bild 1.2.18.: Teile des U 808 D. a) Eingangsstufe; b) Ausgangsstufe

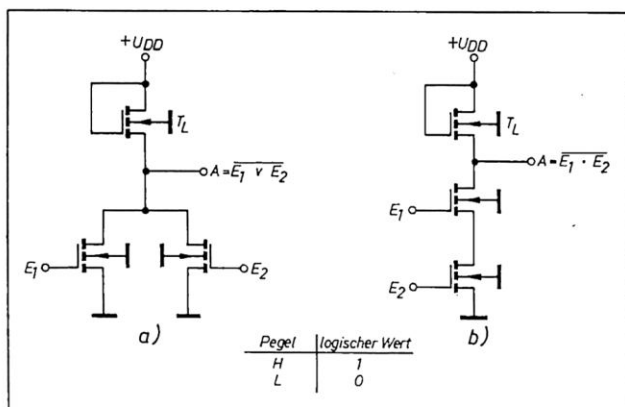


Bild 1.2.19.: n-MOS-Grundgatter. a) NOR-Gatter; b) NAND-Gatter

ist eine aufwendigere Technologie (größere Anzahl von Diffusionsschritten und Masken) erforderlich, aber es werden höhere Packungsdichten und Arbeitsgeschwindigkeiten erreicht. Ein n-Kanal-MOSFET ist etwa zwei- bis dreimal schneller und kann zwei- bis viermal kleiner als ein p-Kanal-MOSFET hergestellt werden. Praktische Verzögerungszeiten liegen etwa bei 100...200 ns. Außerdem haben n-Kanal-Transistoren kleinere Schwellspannungen (≈ 1 V), so daß kleine Betriebsspannungen möglich sind. So benötigt z. B. der Mikroprozessor Valvo 2650 [1.2.9.], der in n-Kanal-Silizium-Gate-Technologie mit zweifacher Ionenimplantation realisiert ist, nur eine Betriebsspannung von $+5$ V. Der nicht zu unterschätzende Vorteil von nur einer Betriebsspannung, die genau der TTL-Versorgungsspannung entspricht (Aufwand im Netzteil gering!) wird durch niedrigen Leistungsverbrauch und einfache Realisierung der TTL-Kompatibilität ergänzt.

1.2.2.1.2. Komplementärtechnik

In der komplementären MOS-Technik (CMOS-Technik) werden im Gegensatz zur Einkanaltechnik sowohl p-Kanal- als auch n-Kanal-MOSFETs eingesetzt. CMOS-Schaltungen weisen ein sehr gutes statisches und dynamisches Verhalten auf. Ihre Herstellung ist allerdings aufwendig und schwierig, so daß etwa gleiche Preise wie bei TTL-Schaltungen vorliegen.

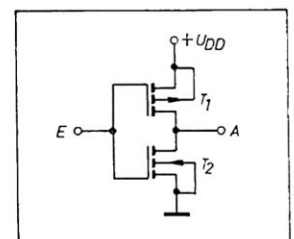


Bild 1.2.20.: CMOS-Grundsaltung

Bild 1.2.20. zeigt die häufigste CMOS-Grundsaltung, die COSMOS-Schaltstufe (Komplementärsymmetrische MOS-Schaltstufe). Sie stellt einen Inverter aus zwei komplementärsymmetrischen Anreicherungs-MOSFETs dar. Beide Transistoren werden vom Eingangssignal E angesteuert. Bei einer Eingangsspannung $U_E = U(0) = 0$ V ($< U_T$) sperrt der n-Kanal-MOSFET, der p-Kanal-MOSFET ist leitend, und am Ausgang liegt die Spannung $U_A = U(L) = U_{DD}$. Bei einer Eingangsspannung $U_E = U(L) \geq U_{DD} - U_T$ ist der p-Kanal-Transistor gesperrt und der n-Kanal-Transistor leitend, die Ausgangsspannung ist $U_A = U(0) = 0$ V. Die Betriebsspannung wird also voll als Signalhub am Ausgang wirksam. Der besondere Vorteil aller CMOS-Schaltungen ergibt sich aber aus der Tatsache, daß in beiden Schalterstellungen jeweils ein Transistor gesperrt ist. Die statische oder Ruherverlustleistung ist damit fast null (in realen Schaltungen nur einige Nanowatt infolge von parasitären gesperrten Dioden). Im dynamischen Betrieb steigt allerdings die Verlustleistung infolge der notwendigen Umladevorgänge vor allem der Lastkapazität linear mit der Frequenz und der Lastkapazität sowie dem Quadrat der Versorgungsspannung an. Nach [1.2.10.] beträgt sie etwa 0,1 mW bei $f = 10$ kHz und etwa 10 mW bei $f = 1$ MHz ($U_{DD} = 10$ V, $C_L = 30$ pF).

CMOS-Schaltungen erreichen relativ hohe Schaltgeschwindigkeiten (etwa 25...30 ns bei $U_{DD} = 10$ V und $C_L = 30$ pF).

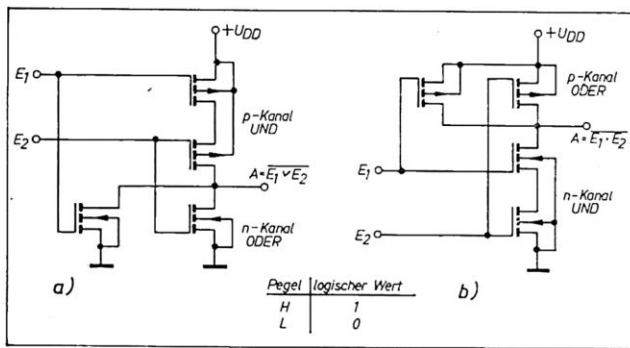


Bild 1.2.21.: CMOS-Grundgatter für positive Logik. a) NOR-Gatter; b) NAND-Gatter

Weitere Vorteile der CMOS-Schaltungen sind der weite Speisespannungsbereich (3...18 V), der ein breites Einsatzgebiet (TTL-Kompatibilität) erschließt und keine Stabilisierungsprobleme mit sich bringt, weiterhin der hohe statische Störabstand (bis zu 45 % von U_{DD}) und die Strahlungsresistenz. Daraus ergaben sich besonders Anwendungsgebiete in der Militärtechnik und Raumfahrt, aber auch wegen der extrem niedrigen Verlustleistung in anderen transportablen Geräten (z. B. elektronische Uhren).

Ein Nachteil aller CMOS-Schaltungen besteht darin, daß in Verknüpfungsschaltungen gleichzeitig ODER- und UND-Schaltungen vorkommen müssen. Daraus resultiert ein relativ hoher Flächenbedarf (etwa wie bei TTL). Bild 1.2.21. zeigt die beiden Verknüpfungsschaltungen NOR und NAND. Mit größer werdender Anzahl von Eingängen kommt es durch die Parallel-Serienschaltung der Transistoren (gesteuerte Widerstände) zur Verschiebung der Übertragungskennlinie (Verringerung der Störsicherheit) und zur Verschlechterung des dynamischen Verhaltens infolge größerer Kapazitäten. Die Eingänge werden deshalb meist auf vier begrenzt.

Sinnvoller ist die CMOS-Technik, besonders in Hinsicht auf die Anwendung in der Mikroprozessortechnik, zur Realisierung von Speicherschaltungen.

1.2.2.2. Dynamische MOS-Schaltungen

Im Gegensatz zu den statischen MOS-Schaltkreisen mit konstanten Versorgungsspannungen und Steuerpegeln werden in der dynamischen Technik eine oder mehrere Impulsspannungen als Steuer- und z. T. auch als Versorgungsspannungen eingesetzt.

Die statische Technik bietet zwar sehr einfache Schaltungslösungen, sie hat aber auch einige wichtige Nachteile:

- Es fließt in einem der beiden statischen Arbeitspunkte stets ein Gleichstrom (statische Verlustleistung).
- Die notwendige Dimensionierung der Schaltkreise führt zu ungünstiger Flächenverwertung und Nichtausnutzung der möglichen Schaltgeschwindigkeiten der MOSFETs.
- Die parasitären Kapazitäten verschlechtern das dynamische Verhalten der Schaltstufen.

Es wurden dynamische Einkanal-MOS-Schaltkreise entwickelt, die diese Nachteile weitgehend überwinden. Die dynamische Technik ist durch folgende Eigenschaften gekennzeichnet:

- Die Lastkapazitäten werden als dynamische Speicherelemente verwendet. Dem Schwinden gespeicherter Ladungen durch Leckströme muß allerdings durch Auffrischzyklen (engl.: refresh cycles) entgegengewirkt werden.
- Die dynamischen Speicherelemente gestatten die Auftrennung der Schaltstufen in einen Auflade- und einen Entladepfad, so daß statische Gleichstromwege in beiden Arbeitspunkten vermieden werden können.
- Diese Auftrennung wird durch Tastung des Auf- und Entladepfades durch sich nicht überlappende Impulsspannungen (Phasen) erreicht.

Üblicherweise werden in einer Schaltstufe eine oder zwei Phasen zur Steuerung verwendet. Man spricht dann von Ein- oder Zweiphasentechnik.

Da in einer komplexen Schaltung (z. B. Schieberegister) den

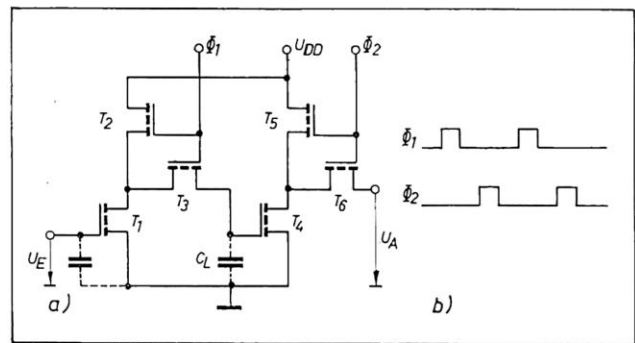


Bild 1.2.22.: Dynamische Zweiphasenzelle in MOS-Technik. a) Schaltung; b) Phasendiagramm

einzelnen aufeinanderfolgenden Schaltstufen (Kettenschaltungen) in Ein- oder Zweiphasentechnik u. U. mehrere unterschiedliche Phasen zugewiesen werden müssen, spricht man auch von Zwei-, Drei-, Vier- und Sechsphasensystemen oder -zellen.

Bild 1.2.22. zeigt eine dynamische Zweiphasenzelle, wobei die einzelne Schaltstufe in Einphasentechnik realisiert ist. Ist die Eingangsspannung $U_E = 0 \text{ V}$ ($< U_T$), ist der Transistor T_1 gesperrt. Mit der Phasenspannung Φ_1 werden die Transistoren T_2 und T_3 aufgesteuert, so daß sich die Lastkapazität C_L (Gate-Source-Kapazität von T_4) auf U_{DD} aufladen kann. Nach Weggang von Φ_1 sind T_2 und T_3 wieder gesperrt, so daß C_L seine Ladung behält. Mit Φ_2 werden T_5 und T_6 leitend. Da auch T_4 infolge der positiven Ladungen auf der Gateelektrode leitend ist, erscheint am Ausgang bei geeigneter Dimensionierung des Widerstandsverhältnisses T_5/T_4 eine genügend kleine Spannung U_A , die dem Wert 0 entspricht.

Auf diese Weise wird also eine Information am Eingang in zwei Taktphasen zum Ausgang „verschoben“. Die Anwendung als Schieberegister liegt nahe. Bei Verbindung des Ausgangs mit dem Eingang erhält man eine dynamische Speicherzelle (Ringschieberegister).

Mit dieser Zweiphasenzelle ist noch keine klare Trennung zwischen Auf- und Entladepfad und keine Vermeidung von Gleichstrompfaden möglich. Der Vorteil liegt in der Verringerung der Verlustleistung durch den getakteten Betrieb.

Die beiden Nachteile dieser Zelle können durch die im Bild 1.2.23. gezeigte Vierphasenzelle beseitigt werden, wo-

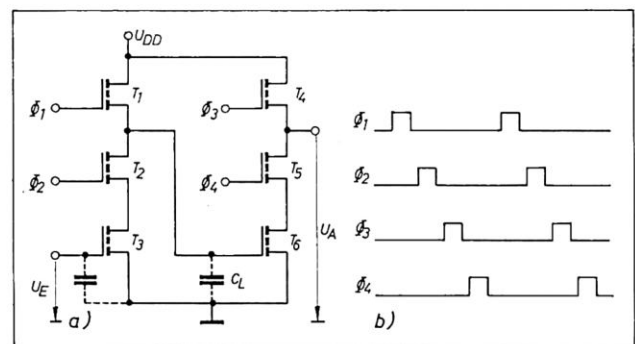


Bild 1.2.23.: Dynamische Vierphasenzelle in MOS-Technik. a) Schaltung; b) Phasendiagramm

bei jede der beiden Schaltstufen in Zweiphasentechnik realisiert ist. Mit Φ_1 wird C_L von T_6 unabhängig von U_E aufgeladen (Vorladetechnik). Bei Erscheinen von Φ_2 wird die Information an U_E auf C_L weitergegeben (bei $U_E = 0 \text{ V}$ Entladen von C_L). Mit Φ_3 wird die Kapazität einer evtl. nachfolgenden Stufe vorgeladen und mit Φ_4 die Information von C_L auf den Ausgang gelegt. Es ist also zu sehen, daß zu keinem Zeitpunkt ein Gleichstrompfad zwischen U_{DD} und Masse besteht. Die statische Verlustleistung ist damit praktisch null.

Daneben gibt es noch die Gruppe der dynamischen Zellen ohne konstante Speisespannung, wobei eine Taktphase die Funktion der Speisespannung übernimmt.

Vorteile der dynamischen Technik

- größere Packungsdichte infolge weitgehend unabhängiger Optimierung des statischen und des dynamischen Verhaltens (Minimierung des Flächenbedarfs durch gleichgroße, quadratische MOSFETs)
- Verringerung der Verlustleistung durch Vermeidung von Gleichstrompfaden
- Erhöhung der Schaltgeschwindigkeit durch unabhängige Optimierung von Auf- und Entladezeit sowie durch minimale MOSFET-Strukturen mit geringen parasitären Kapazitäten.

Nachteile der dynamischen Technik

- relativ hoher Aufwand zur Erzeugung des Phasenschemas und der Verbindungsleitungen
- Notwendigkeit einer streng synchronen dynamischen Betriebsweise in einem weitestgehend festen Zeitraster.

Deshalb werden dynamische Schaltungen vorwiegend für große MOS-Speicher und Schieberegister eingesetzt. In einigen Mikroprozessoren selbst (U 808 D, Intel 8080) werden dynamische Schaltungen zur Realisierung von Registersätzen verwendet. Daraus ergibt sich aber die Notwendigkeit einer dynamischen Betriebsweise des Mikroprozessors in einem sehr engen Frequenzbereich.

1.2.3. Ausgangsstufen

Der Aufbau komplexer digitaler Schaltungen erfordert die Zusammenschaltung vieler Einzelschaltkreise. Dabei ergibt sich erstens die Notwendigkeit, daß ein Schaltkreisausgang n parallelgeschaltete Eingänge der nachfolgenden Stufe treibt. Zweitens erfordert der Aufbau eines bidirektionalen Busses (Leitungsbündel, das Informationen in beiden Richtungen übertragen kann) die Parallelschaltung von Schaltkreisausgängen. Gleichzeitig soll eine hohe Störsicherheit erreicht werden. Dazu sind i. allg. spezielle Endstufen (Treiber- und Leistungsstufen) erforderlich. Darüber hinaus können diese Endstufen zur Realisierung der meist geforderten TTL-Kompatibilität genutzt werden.

1.2.3.1. Gegentaktausgangsstufe

Die Gegentaktausgangsstufe (engl.: push-pull output stage) ist die in der Übersteuerungstechnik gebräuchlichste Endstufe. In der Stromschalttechnik sind wegen der Niederohmigkeit der üblichen Ausgänge meist keine derartigen Endstufen erforderlich.

Bild 1.2.24. zeigt die übliche TTL-Schaltung für einen Ausgangslastfaktor (Ausführung) $F_{LA} = 10$. Der Ausgang erlaubt also die Parallelschaltung von zehn Eingängen mit je einem Eingangslastfaktor $F_{LE} = 1$. Nach der Verknüpfungsschaltung mit dem Multiemittertransistor T_1 folgt die Treiberstufe (T_2), die die gegenphasige Ansteuerung der Transistoren T_3 und T_4 der Endstufe ermöglicht. Die Diode D_1 dient zur sicheren Sperrung von T_3 bei ausgangsseitigem L-Potential. Der 130- Ω -Widerstand begrenzt den Ausgangskurzschlußstrom auf den zulässigen Wert bei H-Potential. Mit dieser Endstufe werden folgende Eigenschaften erreicht:

1. In beiden Arbeitspunkten ist nur ein Strompfad leitend, der sehr niederohmig ist. Dadurch wird einerseits ein großer

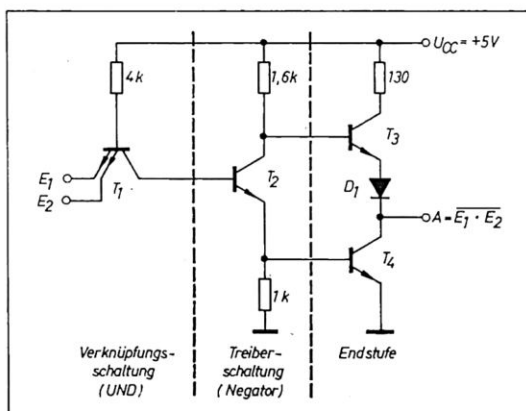


Bild 1.2.24.: TTL-Schaltung mit Ausgangslastfaktor $F_{LA} = 10$

Ausgangslastfaktor möglich, andererseits werden kapazitive Störeinflüsse weitgehend wirkungslos.

2. Auch im Umschaltbereich ist die Schaltung sehr niederohmig (große Stromaufnahme- bzw. Stromabgabemöglichkeit). Das ermöglicht ein schnelles Umladen bei kapazitiven Belastungen und damit eine hohe Schaltgeschwindigkeit. Allerdings tritt beim Umschalten (beide Transistoren leitend) eine Speisestromspitze auf, die zu Störungen im System führen kann und bei hohen Arbeitsfrequenzen (> 2 MHz) eine spürbare Erhöhung der Verlustleistung hervorruft.

3. Durch die Symmetrie der Übertragungskennlinie wird in beiden Arbeitspunkten eine gute Störsicherheit erreicht.

Bild 1.2.25. zeigt eine Gegentakstufe in MOS-Technik. Der Nachteil aller Gegentaktschaltungen besteht darin, daß eine ausgangsseitige Parallelschaltung mehrerer Gatter nicht zulässig ist. Befänden sich nämlich ein Gatterausgang im L-Zustand und n parallelgeschaltete Ausgänge im H-Zustand, so würde praktisch n-mal der Ausgangskurzschlußstrom in den L-führenden Ausgang hineinfließen, was zu einer Zerstörung des Gatters führen kann. Für diesen Fall werden zwei andere Endstufen eingesetzt.

Bild 1.2.25.: MOS-Gegentaktausgangsstufe

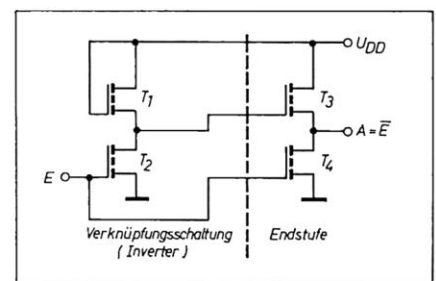
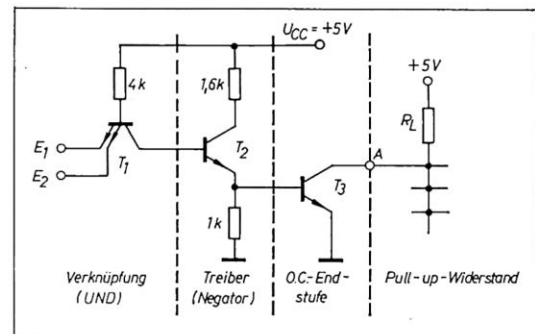


Bild 1.2.26.: Gatter mit offenem Kollektorausgang



1.2.3.2. Ausgangsstufe mit offenem Kollektor

Der bidirektionale Bus wird in der Mikroprozessortechnik vor allem wegen der Pin-Einsparung an den μ P-Schaltkreisen benutzt. Die Verwendung eines Anschlusses sowohl als Dateneingang als auch als Datenausgang führt beim Zusammenschalten von zwei und mehr Einheiten an einen gemeinsamen Bus zwangsläufig zur Parallelschaltung von Gatterausgängen.

Eine Möglichkeit dazu bieten die Gatter mit offenem Kollektorausgang (engl.: Open Collector Output, O.C. Output). Bild 1.2.26. zeigt ein solches Gatter (D 103). Der Transistor T_3 mit offenem Kollektor erlaubt die Parallelschaltung mehrerer solcher Ausgänge und die Belastung dieses Knotens mit mehreren Eingängen.

Führt mindestens ein Gatterausgang L-Potential, so setzt sich dieses im Verbindungsknoten durch. Bezüglich der Ausgänge wird damit eine Wired-AND-Verknüpfung (verdrahtetes UND), bezüglich der Eingänge mehrerer Gatter ein Wired-NOR ($A = E_1 E_2 \vee E_3 E_4 \vee \dots$) realisiert.

Die Parallelschaltung erfordert allerdings einen zusätzlichen Widerstand R_L gegen $+U_{CC}$ (Pull-up-Widerstand, „Hochzieh“-Widerstand), der das H-Potential liefert.

Die Nachteile dieser Schaltung liegen in einer Verschlechterung des dynamischen Verhaltens insbesondere der L/H-Flanke, in der begrenzten Zusammenschaltbarkeit und im zusätzlichen Aufwand durch externe, nichtintegrierbare Widerstände.

Eine Verbesserung bzw. Vermeidung dieser Punkte wird durch die nachfolgend beschriebene Endstufe erreicht.

1.2.3.3. Tristabile Ausgangsstufe

Wurde im vorherigen Abschnitt die ausgangsseitige Parallelschaltbarkeit durch nur einen Ausgangstransistor erreicht, der bei H-Potential gesperrt ist, so wird bei den tristabilen Ausgangsstufen (engl.: Three State Output, T.S. Output, Dreizustandsausgang) eine Gegentaktendstufe verwendet, bei der beide Ausgangstransistoren gesperrt werden können (Bild 1.2.27.). Der obere Teil der Schaltung entspricht einem herkömmlichen NAND-Gatter mit zwei Eingängen, wobei zur Erhöhung des Lastfaktors der obere Transistor der Gegentaktendstufe durch einen Darlington-Emitterfolger (T_3, T_4) ersetzt wurde. Der untere Teil der Schaltung dient zur Sperrung beider Ausgangstransistoren. Ist $S = 1$, werden T_7 und T_8 leitend. Damit wird am dritten Eingang des Multiemittertransistors T_1 L wirksam, wodurch T_2 und T_5 gesperrt werden. Außerdem wird über D_1 das Basispotential von T_3 auf U_{BEF} heruntergezogen. Dadurch werden auch T_3 und T_4 gesperrt. Auf diese Weise wird der dritte Zustand, die Hochohmigkeit des Ausgangs, er-

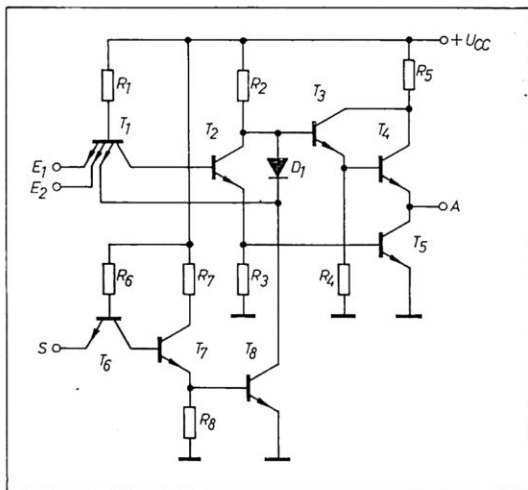


Bild 1.2.27.: Gatter mit tristablem Ausgang nach [1.2.2.]

reicht, so daß ein solches Gatter praktisch von einem Bus abgetrennt wird. Der Nachteil einer Parallelschaltung dieser Gatter liegt also darin, daß durch eine zusätzliche Logik gesichert werden muß, daß immer nur ein Gatterausgang am Bus wirksam ist.

Die Realisierung eines bidirektionalen Busses zeigt Bild 1.2.28. Im Teilbild a) werden die nichtsendenden Gatter mit offenen Kollektorausgängen durch das Torsignal 2 auf H gelegt. Ein L-Signal des sendenden Gatters setzt sich auf dem Bus durch. Bei der Variante b) werden die nichtsendenden Gatter mit Dreizustandsausgängen durch das Torsignal 2 in einen hochohmigen Zustand gebracht und damit praktisch vom Bus getrennt. Die Eingangsgatter bleiben in beiden Varianten am Bus angeschlossen, wobei aber die Eingangssignale erst mit dem Torsignal 1 intern wirksam werden.

In der Praxis ist es oft notwendig, beide Varianten zu kombinieren. Neben den notwendigen Pull-up-Widerständen ist besonders auf die Einhaltung der zulässigen Schaltkreisparameter (Strom, Spannung) zu achten.

1.2.4. Speicherschaltkreise

Die sehr hohe Speicherdichte, der niedrigere Bitpreis, geringere Verlustleistungen und höhere Arbeitsgeschwindigkeiten waren ausschlaggebend, daß Halbleiterspeicher in modernen Rechenanlagen dominieren. Grundsätzlich kann man zwischen Schreib-/Lesespeichern und Festwertspeichern unterscheiden.

- Schreib-/Lesespeicher (RAM) sind Speicher, die sowohl das Einschreiben als auch das Auslesen einer Information entsprechend der Aufrufbreite des Speichers in je einem Speicherzyklus (Zykluszeit) gestatten.

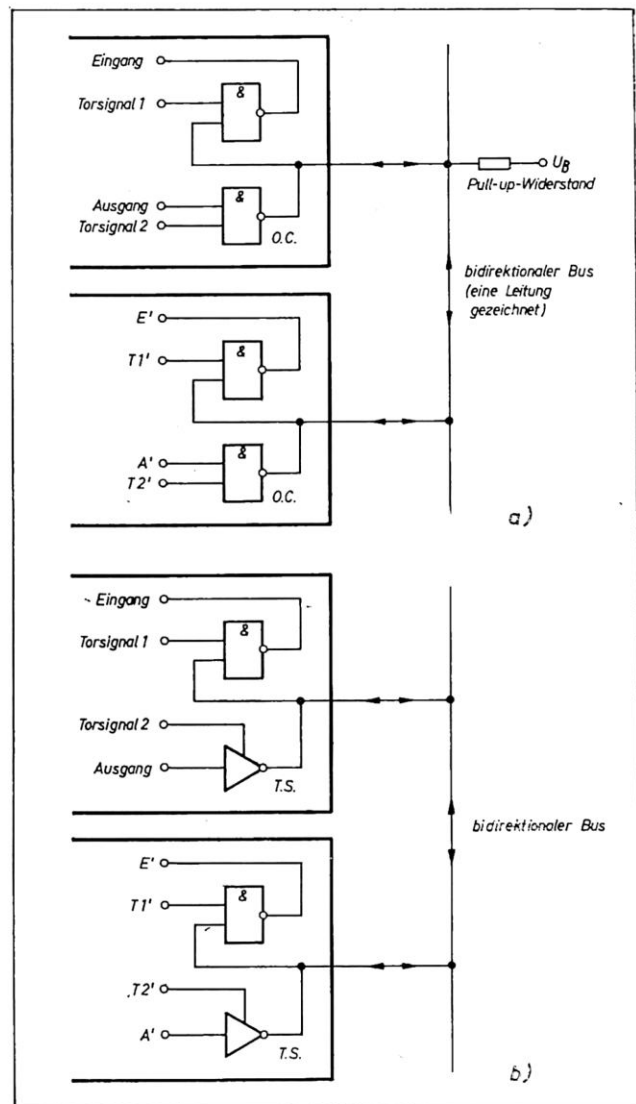


Bild 1.2.28.: Bidirektionaler Bus. a) Mit O.C.-Ausgängen; b) mit T.S.-Ausgängen

Halbleiterspeicher können zerstörungsfrei gelesen werden. Dadurch entfällt der beim Kernspeicher sowohl beim Lesen als auch beim Schreiben erforderliche vollständige Schreib-/Lese-Zyklus. Die für den Einsatz als Arbeitsspeicher vorgesehenen Schreib-/Lesespeicher besitzen grundsätzlich wahlfreien Zugriff. Deshalb hat sich die Bezeichnung RAM (Random Access Memory, Speicher mit wahlfreiem Zugriff) durchgesetzt, die aber nicht ganz exakt ist, da auch Festwertspeicher wahlfreien Zugriff haben.

- Festwertspeicher (ROM, Read Only Memory, Nur-Lese-Speicher) sind Speicher, die nur das Auslesen von Informationen entsprechend der Aufrufbreite des Speichers in einem Speicherzyklus gestatten.

Der Hauptnachteil aller Halbleiter-Schreib-/Lesespeicher ist das flüchtige Speicherprinzip, d. h., bei Ausfall der Versorgungsspannung geht die gespeicherte Information verloren. Deshalb werden die bisher aus der Mikroprogrammtechnik bekannten Festwertspeicher auch zur Speicherung wichtiger System- und Nutzerprogramme (auf Maschinenbefehlsebene) sowie häufig verwendeter Konstanten benutzt. Der Hauptspeicher (HS) eines Mikrorechners besteht dann sowohl aus ROMs als auch aus RAMs. Man unterscheidet grob zwischen Programm- und Datenspeichern.

Wird fortgesetzt

Dr.-Ing. STEFAN LINDNER

1.2.4.1. Schreib-Lese-Speicher

1.2.4.1.1. Statische Schreib-Lese-Speicher

Das Grundelement aller statischen Schreib-Lese-Speicher ist die bistabile Kippstufe, das Flip-Flop. Es besteht aus mindestens zwei Transistoren und zwei Arbeitswiderständen (evtl. auch Zenerdioden) sowie meist zusätzlichen Bauelementen zur Ankopplung der Adreß-, Daten- und Steuerleitungen. Das Flip-Flop besitzt zwei stationäre Arbeitspunkte, in denen jeweils einer der beiden Transistoren leitend ist. Den Arbeitspunkten werden die logischen Werte 0 und 1 zugeordnet, so daß in einem Flip-Flop genau eine Binärstelle einer Information gespeichert werden kann. Diese statische Speicherzelle behält ihren Inhalt, wenn sie nicht neu beschrieben wird, so lange, wie die Versorgungsspannung anliegt. Mit Verschwinden der Betriebsspannung geht die Information verloren. Nach Wiederkehr der Spannung stellt sich ein beliebiger Zustand ein. Bei einigen Schaltungstechniken (z. B. CMOS), die eine sehr niedrige Ruheverlustleistung besitzen, ist es möglich, bei Ausfall der Versorgungsspannung auf eine Hilfsspannungsquelle (z. B. Batterie) umzuschalten, so daß die Information nicht verlorengeht. Die Batterie wird meist mit auf der Leiterplatte des Speichers untergebracht.

Die Vorteile der statischen Speicherzelle liegen in ihrem einfachen Aufbau und in der einfachen externen Beschaltung. Hinzu kommt der Vorteil aller Halbleiterspeicher, daß die zum Betrieb notwendigen Hilfsschaltungen (Adreßdekodierer, Treiber, Schreib-Lese-Verstärker sowie evtl. Daten- und Adreßregister) in den Speicherbaustein integriert werden können.

Der Nachteil liegt in der relativ hohen Verlustleistung

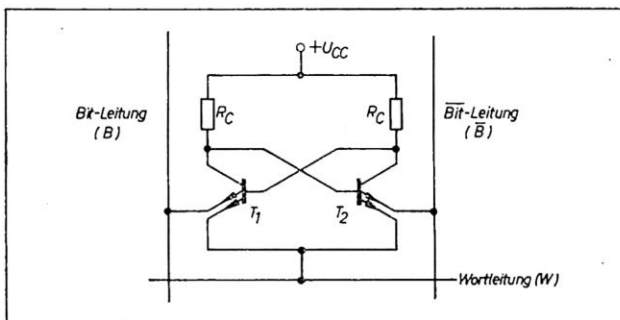


Bild 1.2.29.: Bipolare Speicherzelle mit Multiemittertransistoren

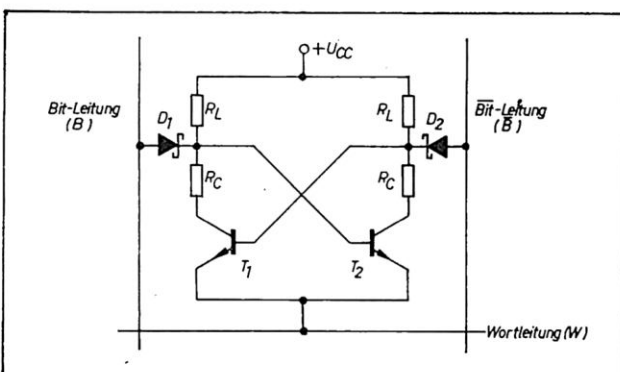


Bild 1.2.30.: Bipolare Speicherzelle mit Schottky-Dioden

(außer CMOS-Speicher). Da immer einer der beiden Transistoren leitend ist, besteht ein Gleichstrompfad zwischen Betriebsspannung und Masse.

Statische Speicherzellen gibt es sowohl in Bipolar- als auch in MOS-Technik. Bild 1.2.29. zeigt die gebräuchlichste bipolare Speicherzelle (in 2-D-Organisation) mit Multiemittertransistoren. Im Ruhezustand liegt auf der Wortleitung 0-Potential, so daß der Strom über den jeweils leitenden Transistor T_1 oder T_2 nach Masse abfließt. Beim Lesevorgang wird 1-Potential auf die Wortleitung gelegt, so daß der Strom durch den leitenden Transistor über den zweiten Emitter auf die jeweilige Bitleitung B oder $\bar{B}$ verdrängt wird, die über Widerstände an Masse liegen und durch Leseverstärker ausgewertet werden.

Beim Schreibvorgang wird zusätzlich zur Wortleitung noch die entsprechende Bitleitung B oder $\bar{B}$ auf 1-Potential gelegt. Der bisher leitende Transistor wird dadurch (bei Informationswechsel) gesperrt, was zur Anhebung des Kollektorpotentials führt. Über den Rückkopplungsweig wird somit der andere Transistor aufgesteuert.

Eine Erhöhung der Arbeitsgeschwindigkeit wird mit der Schaltung nach Bild 1.2.30. erreicht. Die sehr schnellen Schottky-Dioden verhindern, daß die Transistoren in die Sättigung gelangen. Ebenso wirken sich die niederohmigen Widerstände R_C positiv auf die Schaltgeschwindigkeit aus, während die hochohmigen Widerstände R_L eine niedrige Ruheverlustleistung ermöglichen. Die Schottky-Dioden benötigen keinen zusätzlichen Platz, da sie innerhalb der Kollektorregionen der Transistoren realisiert werden können. Die Zelle wird adressiert, indem die Wortleitung auf 0-Potential gelegt wird. Beim Lesen werden beide Bitleitungen auf ein positives Potential gelegt, so daß ein auswertbarer Strom auf der Bitleitung fließt, deren Transistor leitend ist. Beim Schreiben wird auf eine Bitleitung 1- und auf die andere 0-Potential gelegt.

Bipolare Speicherschaltkreise sind in ihrer Herstellung sehr aufwendig und damit teuer. In der Standardtechnik werden Speicherkapazitäten bis zu 256 bit/Chip ($1/4$ K), in der Schottky-Technik bis zu 1024 bit/Chip (1 K) erreicht. Die Verlustleistung liegt bei einigen mW/bit. Die bipolaren Speicher werden deshalb nur für Hochgeschwindigkeitsanforderungen verwendet (Zugriffszeit 30...100 ns).

Eine Erhöhung des Integrationsgrades und eine Verringerung der Verlustleistung werden durch statische MOS-Speicher erzielt. Bild 1.2.31. zeigt den prinzipiellen Aufbau einer solchen Zelle. Das typische statische Flip-Flop wird durch die Transistoren T_1 , T_2 gebildet. Die Transistoren T_3 und T_4

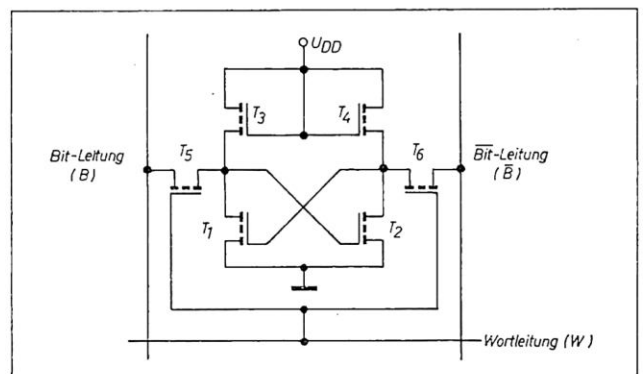


Bild 1.2.31.: Statische MOS-Speicherzelle

sind die Arbeitswiderstände. Über T_5 und T_6 werden die Wort- und Bitleitungen angekoppelt.

Bei der Annahme von n-Kanal-Transistoren arbeitet die Speicherzelle so, daß der Lesevorgang durch ein positives Signal auf der Wortleitung eingeleitet wird. Dadurch werden T_5 und T_6 leitend, und die Drains von T_1 und T_2 werden mit B bzw. $\bar{B}$ verbunden. Auf der Seite des gesperrten Transistors (hohes positives Potential am Drainpunkt) wird damit ein Strom auf der Bitleitung registriert. Das Einschreiben einer Information erfolgt nach Aktivierung der Wortleitung (T_5 , T_6 leitend) durch Anlegen von zueinander negierten Signalen auf der B- und $\bar{B}$ -Leitung.

Diese Zelle kann in p-MOS-, n-MOS- oder CMOS-Technik realisiert werden, wobei jetzt die Transistoren T_1 und T_2 oft n-Kanal- und die Transistoren T_3 bis T_6 p-Kanal-Typen sind. Die bereits dargestellten Vorteile der MOS-Technik gestatten einfachere Herstellungsverfahren, höhere Packungsdichten und geringere Verlustleistungen gegenüber der Bipolar-technik. Dafür ist die Arbeitsgeschwindigkeit der MOS-Speicher fünf- bis zehnmal kleiner als die der bipolaren Zellen. Heute wird fast ausschließlich die Silizium-Gate-Technologie verwendet. p-MOS-Speicher sind am einfachsten herzustellen. Ihre Hauptnachteile sind die geringe Arbeitsgeschwindigkeit und die hohe Schwellspannung (hohe Speisespannung). Moderne Speicherschaltkreise sind deshalb in n-MOS-Technik realisiert. Durch die zwei- bis dreimal größere Ladungsträgerbeweglichkeit der Elektronen gegenüber den Löchern werden höhere Geschwindigkeiten erzielt. Der Silizium-Gate-Prozeß liefert bei n-Kanal-Transistoren niedrige Schwellspannungen, man benötigt daher niedrige Speisespannungen und erhält TTL-Kompatibilität. Mit der Ionenimplantation können selbstsperrende n-Kanal-MOS-FETs (Anreicherungstypen) erzeugt werden, so daß eine zusätzliche Substratvorspannung entfallen kann (nur eine Betriebsspannung notwendig). Statische n-MOS-Speicherschaltkreise mit Speicherkapazitäten bis zu 4 K bit/Chip, Zugriffszeiten von 150...300 ns und Verlustleistungen von 100 μ W/bit sind gegenwärtiger Stand der Technik.

CMOS-Speicher (1 K bit/Chip, 50...100 ns, 20 μ W/bit) sind insbesondere wegen ihrer extrem niedrigen Ruheverlustleistung von Interesse.

1.2.4.1.2. Dynamische Schreib-Lese-Speicher

Die Funktionsweise, Vor- und Nachteile der dynamischen Schaltungen wurden ausführlich im Abschnitt 1.2.2.2. dargestellt. Der höhere Integrationsgrad erlaubt größere Speicherkapazitäten je Chip. 16 K bit/Chip werden gegenwärtig in dynamischer n-MOS-Technik erreicht. Die beiden heute verwendeten dynamischen Speicherzellen, die Dreitransistorzelle (bis 4-K-bit-RAM) und die Eintransistorzelle (bis 16-K-bit-RAM), zeigt Bild 1.2.32. Bei der Dreitransistorzelle wird die Gate-Kapazität C_L von T_1 als Speicherelement verwendet. Zu Beginn eines Lesezyklus wird die Lese-Bitleitung auf die Versorgungsspannung gelegt. Danach wird der Transistor T_3 über die Lese-Auswahlleitung aktiviert. Ist C_L geladen, so ist T_1 leitend, und auf der Bitleitung wird der Strom registriert. Bei entladener C_L bleibt T_1 gesperrt, und es fließt kein Strom. Beim Schreiben wird T_2 über die Schreib-Auswahlleitung aktiviert. Die auf der Schreib-Bitleitung liegende Information wird auf C_L übernommen. Ein hohes positives Potential (bei n-Kanal-Transistoren) führt dazu, daß C_L geladen wird und beim Lesen ein niedriges Signal (0) auf der Lese-Bitleitung festgestellt wird. Die Speicherzelle arbeitet also invertierend. Zum Auffrischen muß die Information gelesen, invertiert und wieder neu eingeschrieben werden.

Ein höherer Integrationsgrad wird mit der Eintransistorzelle erreicht. Sie besteht aus dem Schalttransistor T und der in

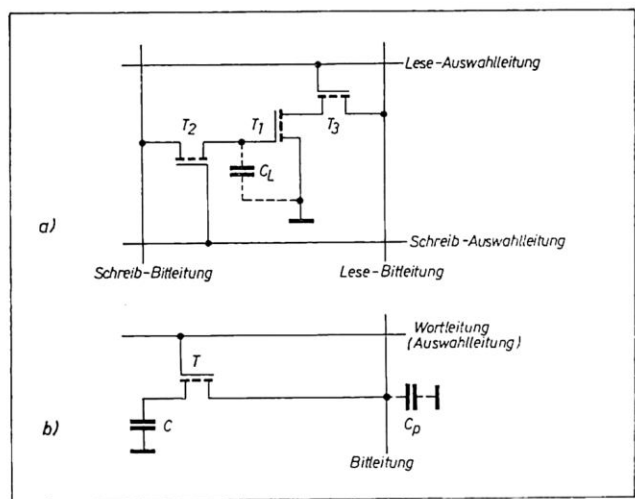


Bild 1.2.32.: Dynamische Speicherzellen. a) Dreitransistorzelle; b) Eintransistorzelle

Reihe geschalteten Kapazität C. Die Speicherzelle wird über die Wortleitung ausgewählt. Beim Schreiben wird C über die Bitleitung und T auf die gewünschte Information geladen. Beim Lesen wird die durch den auf die Bitleitung geschalteten Kondensator C hervorgerufene Spannungsänderung mit einem empfindlichen Leseverstärker ausgewertet. Die Größe der Lesespannung ist dabei proportional dem Verhältnis C/C_p , wobei C_p die parasitäre Kapazität der Bitleitung ist. In realisierten Bausteinen ist $C < C_p$, so daß die Information beim Lesen zerstört wird. Ähnlich wie beim Kernspeicher muß deshalb nach jedem Lesezyklus ein Wiedereinschreiben erfolgen.

Der Nachteil aller dynamischen Speicher besteht in der Notwendigkeit der periodischen Auffrischung (etwa nach 2 ms) der Speicherkapazitäten, die durch Leckströme entladen werden. Je nach Speichertyp erfolgt das Auffrischen von außen als Lese- oder Schreibzyklus. Dazu müssen alle Zeilen oder Spalten der Speichermatrix innerhalb von 2 ms mindestens einmal angesteuert werden.

Dynamische Schaltungen gab es bisher nur in MOS-Technik. Inzwischen ist auch eine dynamische bipolare Speicherzelle in I²L-Technik bekanntgeworden, die zur Informationsspeicherung eine pn-Sperrschichtkapazität ausnutzt [1.2.11.]. Eine noch höhere Speicherkapazität erreicht man mit dynamischen Speichern in CCD-Technik. Das serielle Speicherprinzip ist insbesondere für Massenspeicheranwendungen interessant geworden. Der Speicher besteht dabei aus einer Vielzahl von wahlfrei adressierbaren Schieberegistern, in denen die Informationen ständig umlaufen und aufgefrischt werden. Der Speicher TMS 3064 hat eine Speicherkapazität von 64 K bit und arbeitet mit einer Datenrate von 5 Mbit/s bei einer typischen Verlustleistung von 300 mW [1.2.12.].

Zusammenfassend gibt Tafel 1.2.3. einen Überblick über wichtige Daten von Halbleiter-Schreib-Lese-Speichern der verschiedenen Speichertechnologien, die etwa den gegenwärtigen Stand der Technik darstellen.

1.2.4.2. Festwertspeicher

Der Festwertspeicher ist ein nichtflüchtiger Speicher (non-volatile memory), der nur einen Informationsfluß in einer Richtung gestattet. Vom logischen Standpunkt ist er ein Zuordner, der einem Eingangswort (Adresse) ein Ausgangswort (Datenwort) zuordnet. ROMs gibt es sowohl in Bipolar- als auch in MOS-Technik (ROMs auf magnetischer Basis werden nicht betrachtet).

Tafel 1.2.3.: Wesentliche Daten von Schreib-Lese-Speichern verschiedener Technologien

	Standard-TTL	Schottky-TTL	ECL	I ² L	p-MOS	n-MOS	CMOS	CCD
Speicherkapazität in bit/Chip	256	1 024	1 024	4 096	4 096	16 384	1 024	65 536
Zugriffszeit in ns	50...100	30...50	10...20	100	300...400	150...300	100...250	5 Mbit/s ¹⁾
Leistung in μ W/bit	2 000	500	500	100	100	100	20...100	5

¹⁾ Datenrate

1.2.4.2.1. Bipolare Festwertspeicher

Die älteste Festwertspeichertechnik, die sich im kleinen Umfang auch diskret realisieren läßt, wird durch eine **Diodenmatrix** nach Bild 1.2.33. verkörpert. In integrierter Form werden die Brücken mit einer Nickel-Brom-Legierung realisiert, die an den ausgewählten Stellen von außen durch hohe Ströme weggeschmolzen werden. Auf diese Weise entsteht das gewünschte Bitmuster. Praktische Bedeutung haben diese Bausteine nur für kleine, schnelle Speicher (Zugriffszeiten etwa 10...20 ns) mit hohen Pegeln.

Der Aufbau der **bipolaren maskenprogrammierten Festwertspeicher** entspricht dem der Diodenmatrix. Die Kopplung zwischen Spalten- und Zeilenleitungen besorgen bipolare Transistoren und z. T. Schottky-Dioden. Das gewünschte Bitmuster wird vom Hersteller mit Hilfe einer Maske erzeugt, die den Metallisierungsprozeß der Verbindungsleitungen so steuert, daß die Brücken in den Speicherzellen gesetzt oder nicht gesetzt werden. In dieser Technik werden ROMs mit einer Speicherkapazität bis zu 16 K bit und Zugriffszeiten von 30...120 ns realisiert.

Bipolare programmierbare Festwertspeicher (PROM) haben den Vorteil, daß sie erst beim Anwender programmiert werden können. Im unprogrammierten Zustand sind alle Bits je nach Typ 0 und 1. Durch einen elektrischen Programmiervorgang wird dann das gewünschte Bitmuster erzeugt. Dieser Vorgang ist irreversibel. Gegenwärtig gibt es drei technologische Verfahren zur Herstellung der Programmier-elemente.

Bild 1.2.34. zeigt den prinzipiellen Aufbau einer PROM-Speicherzelle mit Ausbrennelement. Ein dünner Nickel-Brom-Film (15...35 nm) dient als Brückenelement in der Speicher-matrix, das beim Programmieren durch einen hohen Strom (etwa 20 mA) durchgebrannt wird. Die Programmierzeit beträgt etwa 1 ms/bit.

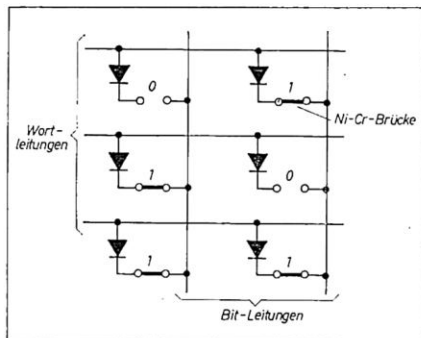


Bild 1.2.33.: Dioden-Festwertspeicher

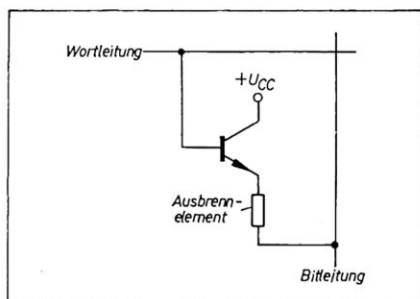


Bild 1.2.34.: Bipolare PROM-Speicherzelle mit Ausbrennelement

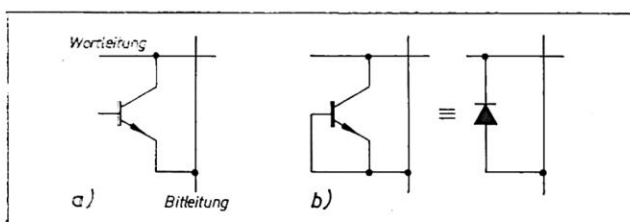


Bild 1.2.35.: AIM-PROM-Speicherzelle. a) Unprogrammiert; b) programmiert

Der Aufbau einer Speicherzelle mit Silizium-Programmier-element entspricht ebenfalls dem Bild 1.2.34., wobei aber eine polykristalline Siliziumschicht (≈ 300 nm) als Ausbrennelement verwendet wird. Der Programmierstrom erzeugt an der Kerbstelle (dünnste Stelle des Programmier-elements) Temperaturen bis zu 1400 °C, wobei das Silizium oxydiert und eine isolierende Lücke erzeugt. Die Programmierung benötigt ebenfalls eine Zeit von etwa 1 ms/bit.

Beim AIM-Verfahren (Avalanche Induced Migration, durch den Avalanche-Durchbruch hervorgerufener Materietransport) wird die Basis-Emitter-Diode eines Transistors kurzgeschlossen (Bild 1.2.35.). Im unprogrammierten Zustand ist der Transistor gesperrt. Im programmierten Zustand besteht über der Basis-Kollektor-Diode eine leitende Verbindung zwischen Wort- und Bitleitung.

Etwas problematisch ist die Dosierung der Programmierimpulse, damit kein Durchbruch zum Kollektor erfolgt. Bei diesem Verfahren werden sehr kurze Programmierzeiten (etwa 0,2 ms/bit) erreicht.

Bipolar PROMs gibt es mit einer Speicherkapazität bis zu 4 K bit/Chip und Zugriffszeiten von 40...90 ns, wobei zur Erhöhung der Geschwindigkeit Schottky-Klemmdioden verwendet werden.

Maskenprogrammierte ROMs erscheinen zwar billiger als PROMs, sind aber erst bei großen Stückzahlen (> 1000) ökonomisch herstellbar. Zur Erprobung, für Prototypen und kleine Serien werden fast ausschließlich PROMs eingesetzt. Grundsätzlich gibt es für jeden PROM einen pin-kompatiblen ROM.

1.2.4.2.2. MOS-Festwertspeicher

MOS-Festwertspeicher sind in ihrer Herstellung einfacher als bipolare ROMs und damit billiger. Dafür liegen die Zugriffszeiten um eine Größenordnung über denen der Bipolartechnik. MOS-ROMs gibt es in p-Kanal-, n-Kanal- und Komplementärtechnik.

Die Koppellemente zwischen Wort- und Bitleitung bilden bei den **maskenprogrammierten MOS-Festwertspeichern** invertierende Feldeffekttransistoren (Bild 1.2.36.). Die Maskenprogrammierung, d. h. die Festlegung des gewünschten Bitmusters, erfolgt durch einen Ätzprozeß, in dem die Dicke des Siliziumoxids zwischen Gate und Kanal eines MOSFETs verändert wird. Bei dünnem Oxid (etwa 10 nm) entsteht ein Transistor, so daß eine Kopplung zwischen Wort- und Bitleitung möglich ist. Bei dickem Oxid (etwa 1 μ m) entsteht kein Transistor.

In dieser Technik werden ROMs mit einer Speicherkapazität

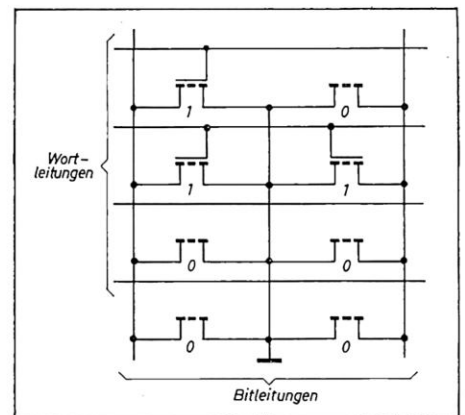


Bild 1.2.36.: MOS-ROM-Speichermatrix

bis zu 16 K bit/Chip und mit Zugriffszeiten zwischen 350 und 1000 ns realisiert.

Einmalig, d. h. irreversibel programmierbare MOS-Festwertspeicher (MOS-PROM) gibt es gegenwärtig noch nicht.

Die teuersten Festwertspeicher sind die **programmier- und löschbaren ROMs (EPROM = Erasable Programmable ROM oder REPROG = Electrically Erasable and REProgrammable ROM)**, die es nur in MOS-Technik gibt. Nach jeder Programmierung können sie wieder gelöscht (z. B. mit UV-Licht) und anschließend neu elektrisch programmiert werden.

Das typische Speicherelement ist ein FAMOS-Transistor (Floating Avalanche-Injection MOS Transistor). Bild 1.2.37. zeigt den Querschnitt durch einen FAMOS-Transistor und eine damit aufgebaute Speicherzelle. Ist die isolierte Gate-elektrode (polykristallines Silizium) ungeladen, sperrt der Transistor. Bei der Programmierung können mit einer hohen negativen Spannung (etwa 50 V) durch den Avalanche-Effekt Elektronen auf das Gate gebracht werden, so daß ein leitender p-Kanal entsteht.

Auf diese Weise erfolgt eine praktische permanente Speicherung der Information. Die Adressierung des FAMOS-

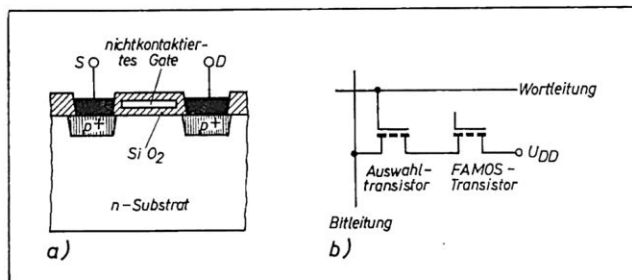


Bild 1.2.37.: a) FAMOS-Transistor; b) FAMOS-Speicherzelle

Transistors erfolgt mit einem in Reihe geschalteten Auswahltransistor. Die Programmierung wird hier wortweise durchgeführt, wobei alle Wortadressen nacheinander durchlaufen werden. Dieser Adreßzyklus wird n-mal durchgeführt. Bei einer maximalen Programmierimpulsdauer von etwa 1 ms sind etwa 100 Zyklen notwendig. Die Programmierung eines Bausteins dauert dann etwa 2 min. Das Löschen der mit einem Quarzdeckel versehenen Bausteine erfolgt mit einer UV-Quelle mit hoher Energie ($\approx 10 \text{ Ws/cm}^2$), wobei Fotoströme das Gate gegen das Substrat entladen. Der Löschvorgang dauert etwa 20...30 min.

Gegenwärtig werden Speicherkapazitäten bis zu 8 K bit/Chip und Zugriffszeiten zwischen 450 und 1 000 ns erreicht. Die elektrisch veränderbaren MOS-Festwertspeicher (EAROM, Electrically Alterable ROM) lassen sich Bit für Bit elektrisch programmieren und wieder zurücksetzen. Als Speicherelemente werden MNOS-FETs oder MAOS-FETs (MAFET) verwendet, die einen geschichteten Gate-Isolator aus Siliziumnitrid (MNOS = Metal Nitride Oxide Semiconductor) bzw. Aluminiumoxid (MAOS = Metal Aluminium Oxide Semiconductor) und Siliziumoxid besitzen. An der Grenzfläche zwischen beiden Dielektrika bilden sich sogenannte Haftstellen aus, auf die ähnlich wie beim FAMOS-Transistor Elektronen getunnelt und wieder abgezogen werden können.

Die mit zunehmender Anzahl der Schreiboperationen immer kürzer werdende Speicherzeit und die relativ großen Zugriffszeiten (etwa 25 μs) sind gegenwärtig die Haupthindernisse auf einen breiten Einsatz von EAROMs.

Zusammenfassend gibt Tafel 1.2.4. einen Überblick über wichtige Daten von Halbleiter-Festwertspeichern.

1.2.4.2.3. Programmierbare logische Felder (PLA)

Eine spezielle Form eines Festwertspeichers oder besser eines Zuordners stellen die programmierbaren logischen Felder dar. Die bisher beschriebenen ROMs können bei n Eingangssignalen (Adresse der Länge n) 2^n Ausgangsworte beliebiger Länge (entsprechend den vorgegebenen Bausteinen) bilden. Bei den PLA-Bausteinen kann nur ein beschränkter Teil der möglichen Eingangskombinationen genutzt werden. PLAs gestatten aber die Verwendung von „Don't-care“-Termen, so daß die Ausgangsworte nicht un-

bedingt von allen Eingangsvariablen abhängig sein müssen. Das erlaubt die Realisierung von minimierten kombinatorischen Funktionen.

Bild 1.2.38. zeigt den prinzipiellen Aufbau eines PLA. Die n-Eingangssignale werden durch die Eingangspuffer in negierter und nichtnegierter Form m UND-Gattern mit je 2ⁿ Eingängen zur Verfügung gestellt. Auf diese Weise können m Minterme gebildet werden, die durch k ODER-Gatter zu k minimierten disjunktiven Normalformen verknüpft werden können. Schließlich liefern k gesteuerte Inverter k Ausgangssignale in negierter disjunktiver Normalform. Hat ein Steuereingang 0, so erscheint unabhängig von den Eingangssignalen 1 am entsprechenden Ausgang.

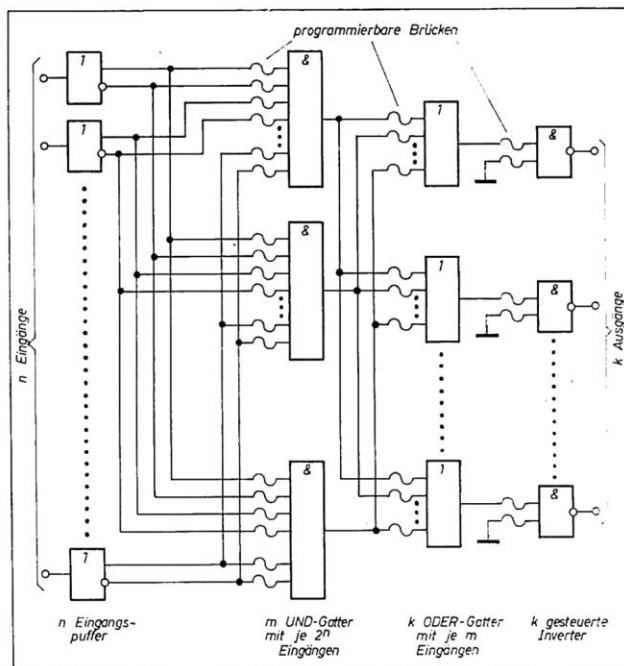


Bild 1.2.38.: PLA-Baustein

Tafel 1.2.5.: Wesentliche Daten von PLAs

Zahl der Eingänge n	12 bis 16
Zahl der Ausgänge k	6 bis 8
Zahl der Minterme m	48 bis 96
Verzögerungszeit	35...100 ns
Verlustleistung	550...900 mW

Die Programmierung, d. h. die Festlegung der zu bildenden logischen Gleichungen, erfolgt mit programmierbaren Brücken, die wie bei den ROMs entweder vom Hersteller durch Masken (Mask Programmable Logic Array, PLA) oder vom Anwender elektrisch (Field Programmable Logic Array, FPLA) gesetzt oder nicht gesetzt werden können.

Tafel 1.2.5. gibt einen Überblick über wichtige Daten von PLAs.

Wird fortgesetzt

Tafel 1.2.4.: Wesentliche Daten von Festwertspeichern

	Bipolarer ROM	Bipolarer PROM	Bipolarer ECL-PROM	MOS-ROM	CMOS-ROM	EPROM	EAROM
Speicherkapazität in bit/Chip	16 K	4 K	1 K	16 K	1 K (12 K)	8 K	1 K
Zugriffszeit in ns	30...120	40...90	12...20	350...1 000	250...1 800	450...1 000	25 000
Leistung in mW/Chip	400...1 000	400...1 000	500...750	250...1 000		700...800	

1.2.5. Entwicklungstendenzen der Schaltkreistechnik

Die Tendenz der Weiterentwicklung geht zu immer höheren Integrationsgraden, größeren Arbeitsgeschwindigkeiten und geringerem Energieverbrauch, zur weiteren Erhöhung der Zuverlässigkeit und Wirtschaftlichkeit. Bei dieser Entwicklung stehen drei Gesichtspunkte im Vordergrund:

1. Mit steigendem Integrationsgrad kommt man zu intelligenteren Systemlösungen mit weniger Zusatzschaltkreisen und größerer Anwenderfreundlichkeit.
2. Durch das Absinken der Bauteilanzahl und damit der notwendigen Verbindungsleitungen erreicht man eine höhere Zuverlässigkeit und Systemverfügbarkeit.
3. Die Erhöhung der Anzahl der Bauelemente je Chip führt neben einer Steigerung der Arbeitsgeschwindigkeit zu einer drastischen Kostenreduzierung der einzelnen Bauelemente eines Chips.

Dieser Tendenz stehen zwei Probleme entgegen:

1. Die technischen Verfahren und Anlagen zur Steigerung des Integrationsgrades werden immer komplizierter und damit teurer, so daß sehr hohe Mindeststückzahlen für eine ökonomische Fertigung erforderlich sind. Das verlangt standardisierte, universell einsetzbare LSI-Schaltkreise wie CPU, RAM, ROM, PLA.
2. Andererseits sinkt der Anwenderkreis mit steigender Komplexität der Bausteine (denken wir nur an die breiten Anwendungsmöglichkeiten des D-100-Gatters). Außerdem steigen die Anforderungen bezüglich Qualifikation und gerätetechnischer Ausrüstung (Meßmittel, Rechnerkapazität u. ä.) beim Anwender. Darüber hinaus wünschen sich viele Nutzer für ihre speziellen Aufgaben sog. kundenspezifische Schaltkreise bei relativ niedrigen Stückzahlen.

Gerade deshalb ist gegenwärtig noch keine Verringerung des Entwicklungstempos der integrierten Halbleitertechnik zu erkennen. Im Gegenteil, die Entwicklung von hoch- und höchstintegrierten Schaltkreisen wird in den nächsten Jahren noch rascher voranschreiten. Dabei wird es zunächst darum gehen, die Siliziumtechnologie auf der Grundlage der bekannten Bipolar- und MIS-Technik weiter zu vervollkommen.

Der Integrationsgrad wird durch mehrere technologische Maßnahmen gesteigert.

Die Verkleinerung der auf einem Chip realisierten Strukturen durch Anwendung minimaler Linienbreiten (etwa 0,1...0,3 μm) kann erreicht werden, indem der Belichtungsprozeß bei der Fotolithografie mit Hilfe kurzwelliger Elektronen- oder Röntgenstrahlung durchgeführt wird oder ein Elektronenstrahl schreiben direkt auf die Siliziumscheibe (Wafer) erfolgt, wodurch eine maskenlose Dotierung möglich ist. Mit Hilfe der Plasmaätzung werden hinreichend steile Flanken ohne Unterätzung erreicht. Die Dotierung

erfolgt durch superflache Diffusionen oder Ionenimplantation. Mit dünnen epitaktischen Schichten und selbstjustierenden Strukturen werden Höchstfrequenzschaltungen ($f > 5 \text{ GHz} \triangleq \text{Mikrowellengebiet}$) angestrebt.

Die Vergrößerung der Chipfläche hängt von der ökonomisch vertretbaren Ausbeute ab, da mit wachsender Fläche die Anzahl der Fehlstellen auf dem Chip steigt. Die Entwicklung wird also zur weiteren Verbesserung der Herstellungsverfahren für Siliziumscheiben führen. Verwendet man gegenwärtig Chipflächen von etwa 5 mm $\times$ 5 mm, so werden in Zukunft Flächen von 10 mm $\times$ 10 mm und mehr erreicht werden. Eine wachsende Bedeutung erlangen auch Multi-Chip-Bausteine, bei denen mehrere Chips in einem Gehäuse untergebracht werden.

Schließlich ergeben sich Möglichkeiten zur Steigerung des Integrationsgrades durch geschickte Ausnutzung der inner-elektronischen Wirkprinzipien und ihres Zusammenspiels.

Die Weiterentwicklung der Schaltkreise ist durch einen ständigen „Wettlauf“ der verschiedenen Techniken gekennzeichnet. Dabei geht es neben der Verbesserung der Parameter darum, den technologischen Prozeß ständig zu vereinfachen, effektiv und reproduzierbar zu gestalten, um so zu höherer Fertigungsausbeute und niedrigeren Herstellungskosten zu kommen.

Bei der Bipolartechnik wird insbesondere in Richtung höherer Packungsdichte und noch höherer Arbeitsgeschwindigkeit gearbeitet. In dieser Hinsicht sind vor allem die Low-Power-Schottky-TTL- und I^2L -Technik von Interesse, wobei letztere nach Low-Power-Schottky- oder ECL-Prozessen hergestellt wird.

Technologische Verfahren, wie superflache Diffusion (1 μm -Epitaxie), Oxidisolations- und Ionenimplantation, rücken dabei immer mehr in den Vordergrund. Wegen des kleinen Geschwindigkeits-Leistungsproduktes dürften besonders die I^2L -Technik, aber auch neuere ECL-Strukturen in naher Zukunft eine wesentliche Rolle spielen.

In der MOS-Technik werden vor allem Schritte zur Erhöhung der Arbeitsgeschwindigkeit unternommen. Niedrige Schwellspannung und nur eine Speisespannung ($\pm 5 \text{ V}$, TTL-Kompatibilität) sind heute bereits Stand der Technik. Die dominierende Rolle spielt dabei gegenwärtig die n-MOS-Technik (Silizium-Gate-Prozeß mit Ionenimplantation). Die CMOS-Technik findet ihre Anwendung vor allem auf den Gebieten, wo hohe Störsicherheit, Strahlungsresistenz und extrem niedrige Verlustleistung gefordert werden. Die Weiterentwicklung der MOS-Technik geht in Richtung der Integration von Anreicherungs- und Verarmungstypen auf einem Chip (höhere Packungsdichte und kleinere Verzögerungszeiten) und der Entwicklung neuer Strukturen wie DMOS und VMOS [1.2.15.] mit weiter verbesserten Parametern.

Tafel 1.2.6. gibt noch einmal einen Überblick über Aufwand und wichtige Parameter der heute gebräuchlichen Schalt-

Tafel 1.2.6.: Aufwand und Parameter gebräuchlicher Schaltkreistechniken

	Standard-TTL	Low-Power-Schottky-TTL	ECL ¹⁾	I^2L	p-MOS	n-MOS	CMOS
Flächenbedarf in μm^2	34 100	12 800		3 100	6 800	3 600	32 100
Komponenten	3	3	3	1	2	2	3
Maskenschritte	7	7	7	4	4	7	6
Diffusionen	4	4	4	2	1	3	3
Verlustleistung in mW	10	2	25	0,07 ²⁾	0,1	0,2	0,1...10 ³⁾
Verzögerungszeit in ns	10	5	1 2	50 ²⁾	300 ..1 000	100 ..200	25
Geschwindigkeits-Leistungsprodukt in pJ	100	10	25 30	3,5	30 ..100	20 ..40	2,5...250

¹⁾ NOR-Gatter; ²⁾ injektorstromabhängig; ³⁾ spannungs- und frequenzabhängig

Tafel 1.2.7.: Grenzen und gegenwärtiger Stand der Halbleitertechnologie

	Grenzen	Stand der Technik
minimale Größe von Einzeltransistoren	$\approx 1 \mu\text{m}^2$ ($1 \mu\text{m}^2$)	$1\,000 \mu\text{m}^2$
maximale Integrationsdichte (geometrisch)	$\approx 2,5 \cdot 10^7$ Gatter/cm ²	10^4 Gatter/cm ²
	$\approx 2,5 \cdot 10^7$ bit/cm ²	$4 \cdot 10^4$ bit/cm ²
minimale Schaltzeit (bei Kopp- lung „nächster Nachbarn“ in dichtester thermisch zulässiger Packung)	≈ 10 ps	1 ns
maximale Taktfrequenz eines syn- chronen Netzwerkes mit 10^5 Gattern	≈ 3 GHz	30 MHz
minimales Geschwindigkeits- Leistungsprodukt	$\approx 10^{-15}$ Ws	10^{-12} Ws
minimale Zugriffszeit für Speicher mit einigen 10^4 bit	≈ 100 ps	10 ns

kreistechniken (nach [1.2.6.]). Alle Werte gelten für ein NAND-Gatter mit vier Eingängen. Mit der ständigen Weiterentwicklung der Halbleitertechnik insbesondere mit dem rasch wachsenden Integrationsgrad, ergibt sich natürlich die Frage nach den physikalisch-technischen Grenzen der integrierten Halbleitertechnik. Beherrscht man heute die Größenordnung von 10^4 Gattern/cm², so strebt die VLSI-Technologie in den nächsten Jahren 10^5 Gatter/cm² an. Damit sind aber die Grenzen der Halbleitertechnik noch nicht erreicht. In [1.2.16.] werden entsprechende Abschätzungen über die physikalisch-technischen Grenzen der Miniaturisierung vorgenommen. Danach beträgt der maximale, geometrisch begrenzte Integrationsgrad etwa $2,5 \cdot 10^7$ Gatter/cm², wobei aus thermischen Gründen die Schaltzeiten zwischen 0,6 ns und 6 ns (je nach Kühlungsart) liegen. Will man ein Netzwerk mit einer Frequenz von 10^9 Hz betreiben, so darf die Chipfläche wegen der begrenzten Fortpflanzungsgeschwindigkeit der Verbindungsleitungen nicht größer als $0,1 \text{ cm}^2$ sein. Daraus ergibt sich eine maximale Anzahl von $2,5 \cdot 10^6$ Gattern je Netzwerk. Bei 10^{10} Hz erhält man ein Netzwerk von maximal $2,5 \cdot 10^4$ Gattern. Will man noch schneller arbeiten, muß man aus thermischen Gründen weniger dicht packen. Der maximale, thermisch begrenzte Integrationsgrad liegt bei $2,5 \cdot 10^5$ Gattern/cm², wobei eine Schaltzeit von etwa 10 ps

erreicht wird. Betreibt man aber ein Netzwerk mit einer Taktfrequenz von 10^{11} Hz, so darf die Chipfläche nicht größer als 10^{-5} cm^2 sein. Daraus folgt die völlig unbefriedigende Zahl von 2,5 Gattern je Netzwerk.

Ein Netzwerk mit 10^5 Gattern kann also mit einer Taktfrequenz von höchstens einigen Gigahertz betrieben werden. Tafel 1.2.7. gibt einen zusammenfassenden Überblick über die Grenzen der Halbleitertechnik im Vergleich zu den heutigen Spitzenwerten. Trotz der Einschränkungen und der Voraussetzung, daß die physikalischen Grenzen technisch und ökonomisch sicher nicht erreicht werden, verbleiben noch mindestens ein bis zwei Größenordnungen zur Weiterentwicklung.

Literatur

- [1.2.1.] Goser, K.: Die Entwicklung zur Großintegration. Elektronik 26 (1976) H. 10, S. 53–56
- [1.2.2.] Eckart, D.; Groß, W.: Grundlagen der digitalen Schaltungstechnik. 2. Aufl. Berlin: Militärverlag der DDR 1975
- [1.2.3.] Möschwitzer, A.: Integration elektronischer Schaltungen. 2. Aufl. Berlin: VEB Verlag Technik 1976
- [1.2.4.] Kühn, E.; Schmied, H.: Integrierte Schaltkreise. 3. Aufl. Berlin: VEB Verlag Technik 1976
- [1.2.5.] The TTL Data Book. Supplement to CC-401. TI 1973
- [1.2.6.] Huse, H.: I²L: Die Technologie und ihre Anwendung in einem schnellen 4-bit-Mikroprozessorelement. Elektronik 26 (1976) H. 2, S. 79–82
- [1.2.7.] Teichmann, J.: I²L-Schaltungstechnik. radio fernsehen elektronik 26 (1977) H. 8, S. 245–247
- [1.2.8.] Höhne, H.: Der Mikroprozessor U 808 D. radio fernsehen elektronik 26 (1977) H. 5, S. 145–150; H. 6, S. 187 und 188, 197 und 198
- [1.2.9.] Valvo-Signetics-Mikroprozessoren. Valvo-Brief Nov. 1976
- [1.2.10.] CMOS-Handbuch. 3. Aufl. Motorola Inc. 1975
- [1.2.11.] Sander, W. B.; Shepherd, W. H.; Schinelle, R. D.: Dynamic I²L RAM competes with MOS designs. Electronics 49 (1976) H. 17, S. 99–102
- [1.2.12.] Nine new memories from the leader... Texas Instruments. Electronics 50 (1977) H. 7, S. 50 und 51
- [1.2.13.] Schlenker, M.: Im Blickpunkt: Bipolare und MOS-Schreib-/Lesespeicher (RAMs). Elektronik 26 (1976) H. 10, S. 57–68
- [1.2.14.] Timm, V.: Im Blickpunkt: ROMs, PROMs und PLAs. Elektronik 26 (1976) H. 5, S. 38–47
- [1.2.15.] VLSI – Neue Technologien für digitale integrierte Schaltungen der Zukunft. ntz 30 (1977) H. 3, S. 212–216
- [1.2.16.] Folberth, O. G.; Bleher, J. H.: Grenzen der digitalen Halbleitertechnik. ntz 30 (1977) H. 4, S. 307–314

1.3. Grundlegendes Schaltkreissortiment

1.3.1. Internationale Entwicklungstendenzen

Die Unterstützung, die der Bauelementehersteller dem Anwender bei der Entwicklung von Hard- und Software gibt, hat großen Einfluß auf die Senkung der Entwicklungskosten und der Entwicklungszeit.

Erforderte in den frühen 70er Jahren der Aufbau funktionsfähiger Mikrorechner außer der CPU eine umfangreiche Zusatzelektronik, die durch spezielle Ergänzungsschaltkreise reduziert werden konnte, so hat sich dank der neuen Technologien und des damit gewachsenen Integrationsgrades die Situation inzwischen verändert. Ein Teil der früher notwendigen externen Zusatzfunktionen wird überflüssig, z. B.

- Wegfall von Dekodierungsaufwand durch zweckmäßige Steuersignale
- Wegfall von Pegelanpassungen und Bustreibern, zumindest bei kleinen Konfigurationen
- Wegfall von Demultiplexern von Daten- und Adreßbus durch Vergrößerung der Anschlußzahl der Gehäuse und damit gleichzeitig Bereitstellung von Daten- und Adreßsignalen.

Ein anderer Teil der Zusatzfunktionen wird in die CPU, in periphere Schaltkreise und in Speicher integriert:

- Taktgeneratoren, interne Synchronisation von Steuersignalen in der CPU

- Einspeisung der Unterbrechungs-Vektoradressen auf den Datenbus durch Peripherieschaltkreise
- Integration von Ein-Ausgabe-Einrichtungen in Speicherschaltkreisen (z. B. enthält der Schaltkreis 8155 von Intel neben einem statischen 2-K-RAM drei bidirektionale Ein- und Ausgaberegister von 8; 8 und 6 bit Breite sowie einen programmierbaren Zeitgeber).

Minimalkonfigurationen von Mikrorechnern sind so mit wenigen Schaltkreisen realisierbar, und sehr leistungsfähige Konfigurationen benötigen nur wenige Leiterkarten. Moderne Schaltkreissortimente bestehen im wesentlichen aus

- CPU
- Speicherschaltkreisen (RAM, ROM, PROM)
- überwiegend programmierbaren und damit in ihrer Funktion flexiblen Peripherieschaltkreisen (paralleles und seriell Interface, Steuerbausteine für Unterbrechungssteuerung, direkter Speicherzugriff, für Tastaturen, Bildschirmgeräte, Floppy-Disc's, Zeitgeberbausteine, DA- und AD-Wandler)
- Treiberschaltkreisen für Buserweiterungen bei größeren Systemen (oft werden dafür auch Standard-MSI-TTL-Schaltkreise verwendet).

Wird fortgesetzt

Dr.-Ing. RUDOLF KÜBART

1.2.3. Zentrale Verarbeitungseinheit (CPU)

Die Leistungsmerkmale von Mikrorechnern werden wesentlich von der verwendeten CPU bestimmt. Neben

- Verarbeitungsgeschwindigkeit
- Synchronisationsmöglichkeit für langsamere Speicher und E/A-Geräte
- Verarbeitungsbreite (evtl. Erweiterung durch Kaskadierung)
- Befehlsumfang (evtl. Variationsmöglichkeit durch Mikroprogrammierung)
- Adressierungsarten (Einfluß auf Speicherplatzbedarf)
- Adressierungsumfang
- verfügbare interne Register
- Unterbrechungsmöglichkeiten
- Ein- und Ausgabe-Organisation

spielen Fragen der notwendigen Versorgungsspannungen und Ströme, Toleranzforderungen an die Versorgungsspannungen, Störfestigkeit, Temperaturbereich, Eingangs- und Ausgangspegel, Belastbarkeit der Ausgänge, Taktversorgung, interne Arbeitsweise (statisch/dynamisch), Möglich-

keiten des DMA-Betriebes und der zusätzliche Elektronik-aufwand eine große Rolle.

Aus der großen Zahl der realisierten Mikroprozessoren [1.3.1.] [1.3.2.], für die bipolare Technologien wie Schottky-TTL, ECL und I²L verwendet werden, sollen im folgenden wegen der doch mehr verbreiteten Unipolartechnologien und des Übergewichtes der 8-Bit-Technik folgende CPU-Bauelemente verglichen werden (s. a. Tafel 1.3.1.):

U 808 D vom Kombinat VEB Funkwerk Erfurt
8080 A von Intel
Z 80 von Zilog.

Folgende Gründe spielten bei der Auswahl dieser drei Prozessoren eine Rolle:

- Der U 808 D ist der erste in der DDR produzierte Mikroprozessor und daher für den Anwender in der DDR von großem Interesse.
- Der Intel 8080 A hat große Verbreitung gefunden und wird von mehreren anderen westlichen Firmen als Zweiterhersteller produziert; es werden umfangreiche Hard- und Software-Unterstützungen angeboten.

Tafel 1.3.1.: Technische Daten der Mikroprozessoren U 808 D, Intel 8080 A und Zilog Z 80

	U 808 D	Intel 8080 A	Zilog Z 80-CPU
Technologie	p-MOS, dynamisch	n-MOS, dynamisch	n-MOS, ionenimplantiert, statisch
Gehäuse	DIL 24	DIL 40	DIL 40
Betriebsspannungen in V	-9; +5	-5; +5; +12	+5
Betriebsströme in mA	< 60; < 60	< 1; < 80; < 70	< 150
Taktversorgung	2 Phasen (TTL)	2 Phasen (12 V)	1 Phase (TTL)
Taktperiode in µs	3 > t > 2	2 > t > 0,48 (A) 2 > t > 0,32 (A-1) 2 > t > 0,38 (A-2)	t > 0,4 (Z 80) t > 0,25 (Z 80-A)
Datenbus	8 bit bidirektional	8 bit bidirektional	8 bit bidirektional
Adreßbus	14 bit = (8 + 6) bit (zeitmultiplex über Datenbus)	16 bit getrennt	16 bit getrennt
Adressierungsbereich	16 K Byte	64 K Byte	64 K Byte
Belastbarkeit der Ausgänge	1 Low-Power-TTL	1 TTL	1 TTL
Pegel	TTL	eingeschränkt TTL (U _{IH} = 3,3 V, U _{OL} = 0,45 V)	TTL
Möglichkeit der Busabschaltung	nein	ja	ja
UB-Möglichkeit	ja, nicht maskierbar	ja, maskierbar	ja, maskierbar; 1 nichtmaskierbarer UB-Eingang, 3 durch Befehl setzbare UB-Betriebsarten
dem Nutzer zugängliche Register, Registerpaar kann zur Speicher- adressierung verwendet werden	A, B, C, D, E, (H, L) je 8 bit F 4 bit PC 14 bit	A, (B, C), (D, E), (H, L) je 8 bit F 5 bit PC, SP je 16 bit	(A, F), (B, C), (D, E), (H, L), (A', F'), (B', C'), (D', E'), (H', L') I, R je 8 bit PC, IX, IY, SP je 16 bit
Befehlsformat	1; 2; 3 Byte	1; 2; 3 Byte	1; 2; 3; 4 Byte
Operationscode	1 Byte	1 Byte	1; 2 Byte
Basisbefehle	48	72	158
unterschiedlich modifizierte Befehle	229	244	696
Befehlsabarbeitungszeiten in µs	12...44	2...9	1,6...9,2
Adressierungsarten			
direkt	ja	ja	ja
erweitert direkt	nein	ja	ja
modifizierte Seite Null	ja	ja	ja
relativ	nein	nein	ja
erweitert	nein	ja	ja
indiziert	nein	nein	ja
Register impliziert	ja	ja	ja
Register indirekt	ja	ja	ja
Einzelbit	nein	nein	ja
über Kellerspeicher	ja	ja	ja
Kellerspeicher	8 Register zu je 14 bit in CPU integriert	im Arbeitsspeicher, adressiert durch Stackpointer (SP)	im Arbeitsspeicher, adressiert durch Stackpointer (SP)
Startvorgang	mit Einschalten automatisch in HALT-Zustand, Befehlszähler auf 0, Start nur durch UB möglich	externes RESET-Signal, Befehls- zähler auf 0, UB-Erlaubnis- Flip-Flop zurückgesetzt	externes RESET-Signal, Befehls- zähler auf 0, UB-Mode 1, UB-Er- laubnis-Flip-Flop zurückgesetzt
Synchronisationsmöglichkeit für Arbeit mit langsameren Speichern und Geräten	ja	ja	ja

- Der Z 80 ist der gegenwärtig leistungsfähigste 8-Bit-MOS-Prozessor.

Detailliertere Angaben zu diesen Mikroprozessoren sind in [1.3.3.] bis [1.3.7.] und im Abschnitt 2. dieser Artikelserie zu finden.

U 808 D und 8080 A

Die meisten Befehle werden infolge der Vorteile der n-MOS-Technik gegenüber der p-MOS-Technik vom 8080 A etwa zehnmal schneller ausgeführt. Ausnahmen sind die Restart-, Return-, Halt- und Ein-Ausgabe-Befehle, die nur etwa fünfmal schneller verarbeitet werden. Grund dafür ist die unterschiedliche Art des Kellerspeichers und der Realisierung der Ein- und Ausgabe-Befehle und des Haltbefehls. Vorteilhaft sind beim 8080 A vor allem die größere Belastbarkeit der Ausgänge, der getrennte Daten- und Adreßbus, die Möglichkeit, durch (allerdings 2 Byte) Ein- und Ausgabebefehle je 256 Geräteadressen anzusprechen und der durch den Stackpointer adressierte Kellerspeicher (Stack) als Teil des Arbeitsspeichers, der praktisch unbegrenzt verschachtelte Unterprogramme zuläßt. Die Erweiterung des Befehlsschlüssels beim 8080 A resultiert überwiegend aus den Stackpointeroperationen und der Verwendung der Registerpaare B, C und D, E zur Speicheradressierung.

8080 A und Z 80

Der Geschwindigkeitsvorteil auf Grund der Ionenimplantations-n-MOS-Technik beim Z 80 ist unbedeutend. Der Z 80 realisiert alle 8080 A-Befehle (bei gleicher Befehlsdarstellung) meist mit gleicher Taktzahl. Wesentliche Vorteile des Z 80 sind, daß nur eine Betriebsspannung (+5 V) benötigt wird (nur echter Vorteil, wenn alle Systemkomponenten ausschließlich mit dieser Spannung arbeiten), eine integrierte Refresh-Erzeugung für dynamische Speicher, die im Zeitschatten der Befehlsausführung arbeitet, sowie die Verdopplung des Arbeitsregistersatzes. Die Umschaltung zwischen diesen Registersätzen erfolgt sehr schnell (1,6 µs), so daß bei Unterbrechung ohne das sonst notwendige, zeitaufwendige Abspeichern der Registerinhalte im Kellerspeicher gearbeitet und sehr kurze Reaktionszeiten auf Unterbrechungsanforderungen realisiert werden können. Durch die drei UB-Betriebsmoden des Z 80 ist außerdem eine sehr flexible Gestaltung der UB-Routinen und eine große Zahl von UB-Ebenen möglich.

Die bedeutendsten Vorteile der Z-80-CPU bestehen in der umfangreichen Erweiterung des Befehlssatzes gegenüber dem 8080 A:

- umfangreichere Adressierungsmöglichkeiten (relative Adressierung, indizierte Adressierung mit den Indexregistern IX und IY)
- Einzelbit-Test-Setz- und Rücksetzbefehle
- komplexere Ein- und Ausgabe-Befehle
- zusätzliche Rotationsbefehle
- Blocksuch- und Blocktransferbefehle.

Damit ist eine wesentlich effektivere Programmierung möglich, weil durch die Verwendung der komplexen Befehle Speicherplatz und Programmlaufzeit eingespart werden.

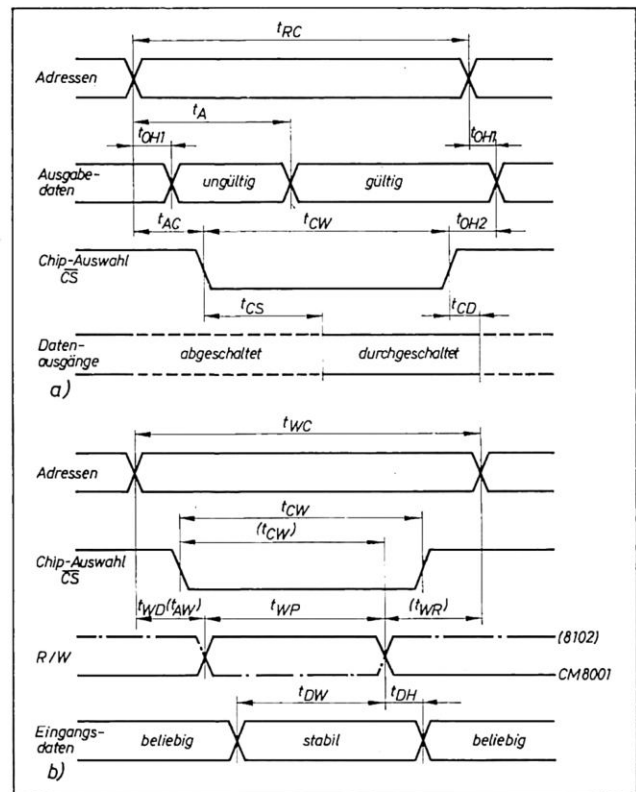


Bild 1.3.1.: Zeitdiagramme für statische Lese/Schreibspeicher. a) Lesezyklus; b) Schreibzyklus

1.3.3. Speicherschaltkreise

Moderne Speicherschaltkreise enthalten eine vollständige Dekodierung der Adresseneingänge und verfügen über einen oder mehrere Baustein-Auswahl-Eingänge (CE = chip enable, CS = chip select). Die Daten-Ausgangsstufen werden als Three-State- (TS), Open-Drain- (OD) oder Open-Collector-Stufen (OC) realisiert, die durch $\overline{CS} = 0$ (d. h. $\overline{CS} = 1$) in den inaktiven, hochohmigen Zustand gebracht werden können. Unter der Voraussetzung, daß jeweils nur die Ausgangsstufen eines Schaltkreises durch $\overline{CS} = 1$ aktiviert sind, können mehrere Speicherbausteine ausgangsseitig parallelgeschaltet werden (wired OR), damit werden Speicher größerer Kapazität geschaffen.

Der Energiebedarf von Speicherschaltkreisen ist im nicht-aktivierten Zustand merklich niedriger als im aktivierten, besonders ist das bei den dynamischen Speichern ausgeprägt. Da in der Regel die meisten Bausteine eines Speichers nicht aktiviert sind, ergeben sich daraus Schlußfolgerungen für die Dimensionierung der Stromversorgung.

Alle Eingänge von MOS-Speichern sind durch integrierte Schutzdioden gegen statische Aufladungen geschützt.

In der Tafel 1.3.2. sind die wichtigsten Kenndaten einiger wichtiger MOS-Speicherschaltkreise, die in der DDR oder international weit verbreitet sind, angegeben.

Tafel 1.3.2.: MOS-Speicherschaltkreise

	statische RAM CM 8001	8101-2	8102-A4	dynamische RAM U 253 D	8107-B4	ROM U 501 D	EPROM 1702-A	8708
Speicherkapazität in Bit	256 × 1	256 × 4	1K × 1	1K × 1	4K × 1	256 × 8	256 × 8	1K × 8
Technologie	p-MOS	n-MOS	n-MOS	p-MOS	n-MOS	MNOS	p-MOS	n-MOS
Versorgungsspannung in V	+5; -9	+5	+5	+19,5; +16	+12; +5; -5	+5; -9	+5; -9	+12; +5; -5
max. Verlustleistung in mW	685	350	350	400	960	800	800	750
TTL-kompatibel	ja	ja	ja	nein	ja	ja	ja	ja
Belastbarkeit (TTL-Last)	1	1	1	—	1	1	1	1
Zugriffszeit t_A bzw. t_{ACC} in ns	1 500	850	450	205	270	1 000	1 300	450
Zugriffszeit t_{CO} für CS in ns	$t_A - t_{AC}$ 400	650	230	185	250	$t_A - t_{CS}$ 300	$t_A - t_{CS}$ 900	120
Lesezyklus in ns	1 500	850	450	480	470	1 000	1 300	450
Schreibzyklus in ns	900	850	450	580	590	—	—	—
Schreibimpulsdauer in ns	400	630	300	50	50	—	—	—

1.3.3.3.1. Statische Lese/Schreibspeicher (statische RAM)

Die Steuerung der Funktion „zerstörungsfreies Lesen“ oder „Schreiben“ erfolgt über einen zusätzlichen Eingang R/W (Read/Write) oder WE (Write-Enable). Die Zuordnung der logischen Werte zu den Signalen und auch die Bezeichnungen der Kenngrößen sind bei verschiedenen Bauelementen und Bauelementherstellern uneinheitlich. So bedeutet beim CM 8001 R/W = 1 „Schreiben“, beim 8102 aber „Lesen“.

Für die Zusammenarbeit von CPU und Speicher sind bestimmte Kennwerte der Speicher von besonderem Interesse (die angegebenen Werte gelten für den CM 8001). Diese Werte werden vom Bild 1.3.1. verdeutlicht.

- Lesezyklus $t_{RC} > 1,5 \mu s$
- Schreibzyklus $t_{WC} > 0,9 \mu s$
- Zugriffszeit $t_A < 1,5 \mu s$
- Durchschaltverzögerung Adresse zu CS $t_{AC} < 1,1 \mu s$
- Verzögerung der Ausgänge gegen CS $t_{CS} < 0,4 \mu s$
- Abschaltverzögerung der Ausgänge gegen CS $t_{CD} < 0,4 \mu s$
- Schreibimpulsbreite $t_{WP} > 0,4 \mu s$
- Verzögerung des Schreibimpulses $t_{WD} > 0,4 \mu s$
- Schreibzeit $t_{DW} > 0,4 \mu s$
- Datenhaltezeit $t_{DH} > 0,1 \mu s$.

Das Blockschaltbild des CM 8001 und dessen Anschlußbild zeigt Bild 1.3.2.

Der Komfort dieser Speicherschaltkreise ist sehr unterschiedlich. Als Besonderheit existiert z. B. beim 8101-2 (Intel) ein zusätzlicher Steuereingang (OD = output disable), der eine Abschaltung der Three-State-Datenausgänge durch OD = 1 auch bei durch CS = 0 aktiviertem Baustein bewirkt. Damit ist beim Schreiben die Abschaltung der Ausgänge und damit eine Zusammenschaltung von Datenein- und Datenausgängen und deren direkte Verbindung mit dem bidirektionalen CPU-Datenbus ohne zusätzlichen Aufwand möglich. Die Entwicklung geht zu größeren Speicherkapazitäten, geringeren Verlustleistungen und niedrigeren Zugriffszeiten. Als Beispiel soll der statische RAM MK 4114 von Mostek genannt werden, der eine Speicherkapazität von $1 K \times 4$ bit, eine typische Zugriffszeit von 150 ns und einen typischen Leistungsbedarf von 80 mW hat.

1.3.3.2. Dynamische Lese/Schreibspeicher (dynamische RAM)

Dynamische Lese/Schreibspeicher ermöglichen bei gleicher Chipgröße gegenüber statischen Speichern etwa viermal

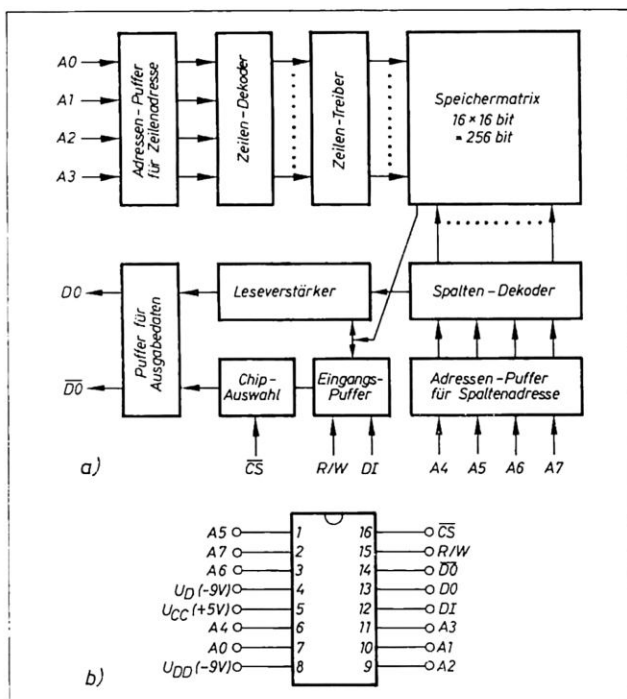


Bild 1.3.2.: Statischer Lese/Schreibspeicher CM 8001. a) Blockschaltbild; b) Anschlußbelegung

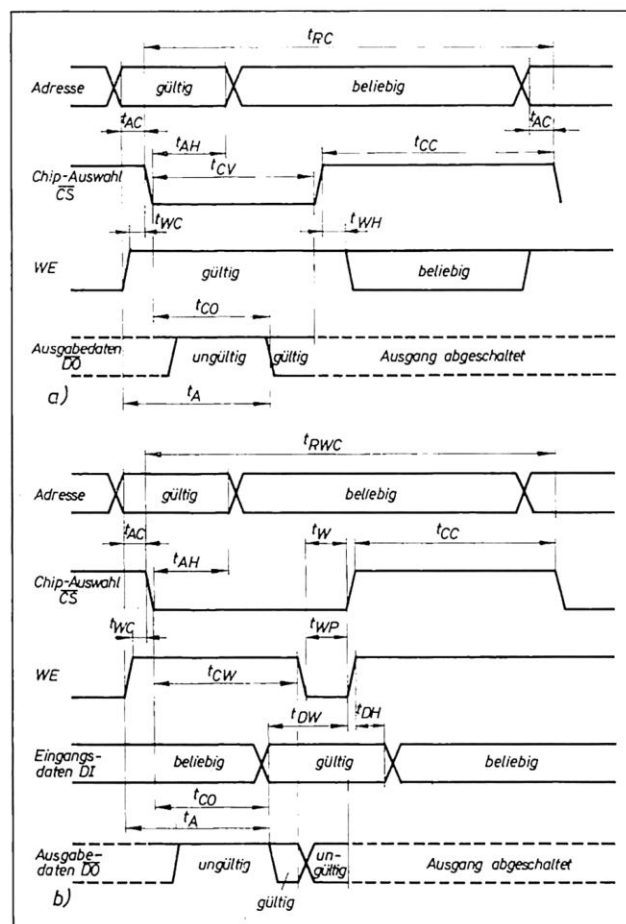


Bild 1.3.3.: Zeitdiagramme für dynamische Lese/Schreibspeicher. a) Lesezyklus; b) Lese/Schreibzyklus

größere Speicherkapazitäten. Als Nachteil ergibt sich jedoch die Notwendigkeit, die gespeicherte Information aufzufrischen (Refresh), was in Form eines Lese- oder Lese/Schreibzyklus für jede Zeile der Speichermatrix (Spalte beliebig!) nach maximal 2 ms erfolgen muß. Das erfordert zusätzlichen Aufwand für die Auffrisch-Logik und ergibt geringe Einschränkungen des Speicherzugriffs, wenn nicht durch die Funktion des realisierten Gerätes die notwendigen Speicherzugriffe ohnehin realisiert sind.

Außerdem ist bei dynamischen Speichern die Einhaltung der zeitlichen Bedingungen (s. Bild 1.3.3.) kritischer und erfordert eine sorgfältige Auslegung der Speichersteuerung.

Der Leistungsbedarf dynamischer Speicher im nichtaktivierten Zustand liegt in der Größenordnung von 1 % des Leistungsbedarfes des aktivierten Bausteines. Damit wird die mittlere Leistung im Betriebszustand vom Verhältnis t_{CV}/t_{RC} (s. Bild 1.3.3.) und damit von der Wiederholungsrate der Speicherzugriffe abhängig.

Die Ruheleistung unter Berücksichtigung der notwendigen Auffrischzyklen beträgt nur einige Prozent des Leistungsbedarfes des aktivierten Bausteines.

Auf Grund des Zusatzaufwandes für die Speicher- und Auffrischsteuerung werden dynamische Speicher vorwiegend zur Realisierung größerer Speicherkapazitäten eingesetzt.

Die Steuerung der internen dynamischen Abläufe erfolgt durch das Signal CS. Durch das Ansprechen des Speicherbausteines bei CS = 1 wird ein Zyklus ausgelöst: die durch die Adresseneingänge adressierte Speicherzelle wird zerstört und wieder aufgefrischt. Soll nicht die alte Information, sondern eine neue wieder eingeschrieben werden, so muß das durch einen Schreib-Steuerimpuls ($\overline{WE} = 0$) nach dem Lesen erfolgen, d. h., ein Schreibzyklus ist immer ein Lese/Schreibzyklus, unabhängig davon, ob die alte Information wirklich verwendet wird (modifizierter Lese/Schreibzyklus) oder nicht.

Im Bild 1.3.3. sind für den dynamischen RAM U 253 D des VEB Funkwerk Erfurt die Zeitdiagramme für einen Lese- und einen Lese/Schreibzyklus angegeben, Bild 1.3.4. zeigt die Blockschaltung und Anschlußbelegung.

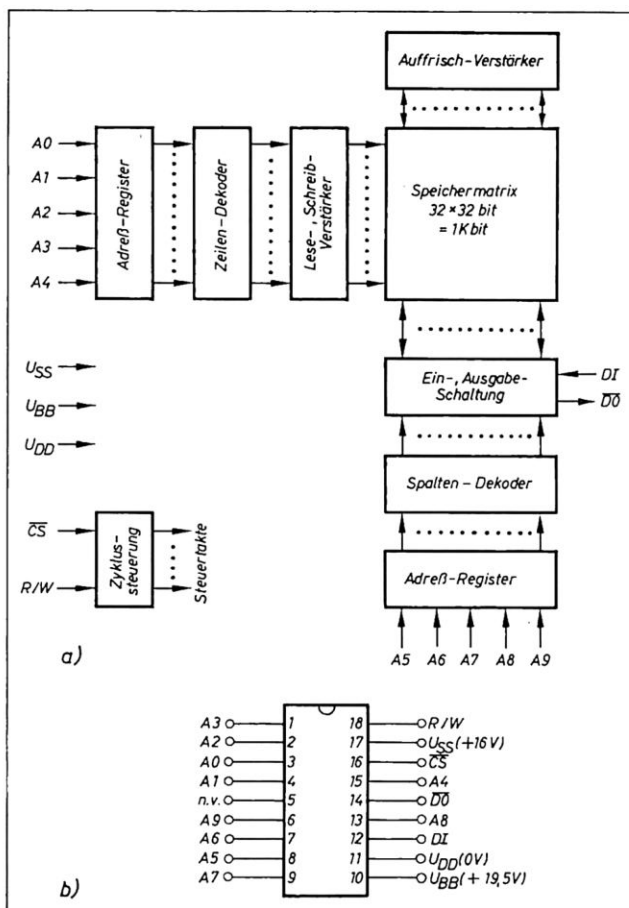


Bild 1.3.4. Dynamischer Les/Schreibspeicher U 253 D. a) Blockschaltbild; b) Anschlußbelegung

Sowohl beim Les- als auch beim Les/Schreibzyklus wird der Zyklus durch die Vorderflanke von CS ausgelöst, die Adressen müssen nur von t_{AC} (> 0 ns) vor Beginn bis t_{AH} (> 100 ns) nach Ende dieser Flanke gültig sein. CS bewirkt ebenfalls die Durchschaltung der Datenausgänge. Die am Ausgang anliegenden Daten sind jedoch erst t_{CO} (< 185 ns) nach dem Ende der Vorderflanke von CS bzw. t_A (< 205 ns) nach dem Anlegen der gültigen Adresse gültig. Beachtenswert ist, daß CS innerhalb eines bestimmten Bereiches $210 \text{ ns} < t_{CV} < 500 \text{ ns}$ liegen und daß dann mindestens für t_{CC} (> 230 ns) $CS = 0$ gelten muß.

Beim Les/Schreibzyklus muß der Schreibimpuls (Dauer $t_{WP} > 50$ ns) im Bereich $210 \text{ ns} < t_{CW} < 500$ ns nach der Vorderflanke von CS erscheinen. Die Eingangsdaten müssen von t_{DW} (> 105 ns) vor bis t_{DH} (> 10 ns) nach der Rückflanke des Schreibimpulses (bzw. der Rückflanke von CS, wenn diese eher eintrifft) gültig sein. Von Beginn des Schreibimpulses an sind die Ausgabedaten ungültig, im Gegensatz dazu bleiben beim 8107 (Intel) die Ausgabedaten auch während des Schreibimpulses und danach gültig. Beim 8107 hat der CE-Eingang die Steuerfunktion für den Zyklus, und $\overline{CS}$ dient nur der Bausteinauswahl.

Der U 253 D ist ein p-MOS-Schaltkreis. Daraus folgen die höheren Betriebsspannungen und die nicht TTL-kompatiblen Pegel: $U_{low} = U_{DD} \pm 1 \text{ V}$, $U_{high} = U_{SS} \pm 1 \text{ V}$.

Zur Anpassung an TTL- oder TTL-kompatible Schaltkreise sind die Schaltkreise D 491 (Treiber) und D 122 (Leseverstärker) vorgesehen.

In neuerer Zeit wurden dynamische Les/Schreibspeicher entwickelt, bei denen Zeilen- und Spaltenadresse der angesprochenen Zelle in der Speichermatrix zeitmultiplex übertragen werden. Dadurch ist es möglich, mit geringerer Zahl von Anschlüssen und damit kleineren Gehäusen auszukommen.

1.3.3.3. Festwertspeicher ROM und PROM

Da Halbleiter-Lese/Schreibspeicher beim Abschalten der Betriebsspannung ihre Information verlieren, müssen Programmteile, die unbedingt funktionsnotwendig sind (z. B. Urlader) oder sofort ohne Ladeoperationen verfügbar sein

sollen (z. B. Monitorprogramme bei Entwicklungssystemen), gegen Verlust geschützt werden. Dazu bestehen zwei prinzipielle Möglichkeiten:

- Verwendung von Halbleiter-Lese/Schreibspeichern, die einen so geringen Ruhestrombedarf haben, daß sie ihre Versorgungsenergie bei abgeschaltetem Netz aus einer kleinen Zusatzbatterie erhalten können (z. B. CMOS)
- Verwendung von Festwertspeichern.

Festwertspeicher werden vom Hersteller nach Kundenwunsch programmiert (maskenprogrammierte ROMs). Bei geringen Stückzahlen, insbesondere für kleine Geräteserien und Versuchszwecke sind die dabei anfallenden Maskenkosten zu hoch, und man verwendet zu den jeweiligen ROMs pin-kompatible EPROMs oder bipolare PROMs, die durch das Wegbrennen dafür vorgesehener Verbindungen auf dem Chip vom Anwender einmalig programmiert werden können.

Bezüglich des Lesens unterscheiden sich ROMs und PROMs nicht von statischen RAMs. Die Programmierung von PROMs erfordert jedoch merklichen Aufwand. Sie erfolgt mit speziellen PROM-Programmiergeräten, die oft durch Mikrorechner gesteuert werden und z. T. durch Auswechseln besonderer Anpaßeinheiten (personality cards) die bei verschiedenen PROM- und EPROM-Typen z. T. sehr unterschiedlichen Programmierungsabläufe realisieren können. Bei Entwicklungssystemen ist die PROM-Programmierung sehr komfortabel, da die Ein- und Ausgabemöglichkeiten und das Softwarepaket des Entwicklungssystems benutzt werden können.

1.3.4. Interface-Schaltkreise

Die Zusammenarbeit zwischen Rechner und Datenverarbeitungs- oder Prozeßperipherie erfordert je nach Anwendungsfall sehr unterschiedliche und z. T. recht aufwendige Anpaßeinrichtungen. Standardisierte Schnittstellen (Standard-Interfaces) erlauben die Senkung dieses Aufwandes, indem sehr verschiedene Geräte über das gleiche Interface anschließbar sind. Die leistungsmäßige, zeitliche und elektrische Anpassung zwischen Rechner und Peripherie erfolgt in der Rechentechnik und Prozeßrechentechnik durch Kanal- und Anschluß-Steuereinheiten, die an Komplexität mit der ZVE vergleichbar sind.

Eine analoge Entwicklung ist in der Mikrorechentechnik festzustellen, wenn auch in für diese Technik spezifischen Formen. Die dazu entwickelten LSI-Interface-Schaltkreise können durch die Programmierung ihrer Funktionsweise mit wenig Zusatzaufwand die meisten Standard-Interfaces realisieren. An der Schnittstelle zwischen CPU und Interface-Schaltkreis liegt in der Regel eine Anpassung an die speziellen Bedingungen der jeweiligen CPU vor, für die der Schaltkreis vorgesehen ist. Gegenwärtig haben zwei Interface-Schaltkreisarten besondere Verbreitung gefunden:

- Bausteine für serielle Ein- und Ausgabe von Daten. Die Übertragung von Daten zwischen CPU und Schaltkreis erfolgt parallel im 8-Bit-Format, sie wird durch Ein- und Ausgabebefehle der CPU gesteuert. Peripherieseitig erfolgt die Übertragung seriell zu Fernschreibern oder Bildschirmgeräten. Der Baustein führt die Parallel-Serien- und Serien-Parallel-Wandlung durch, erzeugt Start- und Stopp-Bits und Paritätsbits.
- Bausteine für parallele Ein- und Ausgabe von Daten. Die CPU-seitige Übertragung erfolgt wie vorher, die peripherieseitige Übertragung jedoch parallel. Dabei sind durch die Programmierung zahlreiche Modifikationen der peripherieseitigen Datenformate, der Übertragungsrichtung für die einzelnen Anschlüsse sowie deren Benutzung für Steuerzwecke möglich.

Die grundlegende Funktionsweise und Leistungsfähigkeit solcher Peripherie-Schaltkreise soll an den Beispielen 8251 und Z 80-PIO erläutert werden.

Wird fortgesetzt

1.3.4.1. Programmable Communication Interface 8251

Bild 1.3.5. zeigt die Blockschaltung des programmierbaren Interface-Bausteines 8251 (Intel).

Der Datenbus D7 bis D0, die Steuersignale Reset, CLK = $\emptyset$, $\overline{WR}$, $\overline{RD}$ sind mit den entsprechenden Anschlüssen der CPU 8080 verbunden. Die CPU führt Ein- oder Ausgabeoperationen wie mit anderen E/A-Geräten durch. Da die Daten sowohl Sende- bzw. Empfangsdaten als auch Steuer- bzw. Statusinformationen sein können, muß eine Auswahl durch C/D getroffen werden (C/D = 1 Steuer- und Statusinformation, C/D = 0 Daten). Meist wird Bit 0 der Geräteadresse zur Auswahl von C/D verwendet, die Dekodierung der restlichen sieben Adreß-Bits dient der Erzeugung von $\overline{CS}$ = 0 und damit der Bausteinauswahl. Es werden also je zwei Geräteadressen in Ein- und Ausgaberrichtung durch einen Schaltkreis 8251 belegt.

Die Ausgänge der Modemsteuerung können zur Steuerung der Peripherie, die Eingänge zur Abfrage des Zustandes des peripheren Gerätes über das Lesen eines Zustandswortes (Status) verwendet werden.

Die Arbeitsweise der Peripherieschnittstelle ist programmierbar. Diese Programmierung erfolgt durch Ausgabe von Steuerworten von der CPU an den 8251. Nach dem Anfangsrücksetzen mittels Reset oder internem Rücksetzen durch ein entsprechendes Kommando muß zuerst die Betriebsart (Mode) übertragen werden. In dem Betriebsart-Steuerwort (1 Byte) sind kodiert:

- Zeichenlänge (5 bis 8 bit)
- Festlegungen bezüglich Paritätsprüfung
- Festlegung der Arbeitsweise (synchron oder asynchron)
- bei asynchroner Arbeitsweise Zahl der Stopp-Bits (1, 1½, 2) und die Taktfrequenz der seriellen Übertragung (Baud-Rate) als Bruchteil der über RXC (Empfangstakt) oder TXC (Sendetakt) vorgegebenen Taktperiode
- bei synchroner Arbeitsweise die Nutzungsart des Syn-Det-Anschlusses als Eingang oder Ausgang und die Zahl der Synchronzeichen (1, 2).

Bei synchroner Arbeitsweise müssen danach ein oder zwei Synchronworte übertragen werden. Erst nachdem ein weiteres Steuerwort, das Kommandowort von der CPU an den 8251, übertragen wurde, kann die eigentliche Datenübertragung über das serielle Interface erfolgen (s. Bild 1.3.6.).

Das Kommandowort beinhaltet Festlegungen über

- Sende- und Empfangsfreigabe
- internes Rücksetzen
- Rücksetzen des Fehler-Flip-Flops
- Freigabe des Suchens nach Synchronzeichen
- Setzen der Ausgänge DTR oder RTS
- Schalten des seriellen Datenausganges TXD auf Low-Pegel.

1.3.4.2. Parallel Input/Output Z 80-PIO

Bild 1.3.7. zeigt die Prinzipschaltung des Interface-Bausteins Z 80-PIO (Zilog).

Die Verbindung zwischen Z 80-CPU und Z 80-PIO erfolgt durch direktes Zusammenschalten über den bidirektionalen Datenbus und über Steuersignale. B/A-Select (B/A = 1 Kanal B, B/A = 0 Kanal A) und C/D (C/D = 1 Steuerung, C/D = 0 Daten) werden meist mit den beiden niederwertigsten Adreß-Bits verbunden. Der Schaltkreis Z 80-PIO belegt also vier Geräteadressen, Steuerworte dürfen jedoch nur in Richtung zum PIO übertragen werden. Auf der Peripherieseite stehen zwei 8-Bit-Kanäle A und B (bidirektional, Three-State-Ausgänge) mit je zwei Quittungssignalen (Ready,

TECHNIK DER MIKRORECHNER

8

Strobe) zur Verfügung. Jeder der beiden Kanäle kann durch zwei bis fünf Steuerworte von der CPU programmiert werden. Es können vier Betriebsarten festgelegt werden:

- Byte-Ausgabe
- Byte-Eingabe
- bidirektionale Byte-Ein- und Ausgabe (nur Kanal A)
- Bit-Ein- und Bit-Ausgabe (ohne Quittungssignale).

Die Arbeitsweise des PIO ist auf die Benutzung einer speziellen UB-Betriebsart der Z 80-CPU orientiert. Das Signal Strobe wird von der Peripherie als Quittung für das beendete Übernehmen von Ausgabedaten bzw. für die Gültigkeit

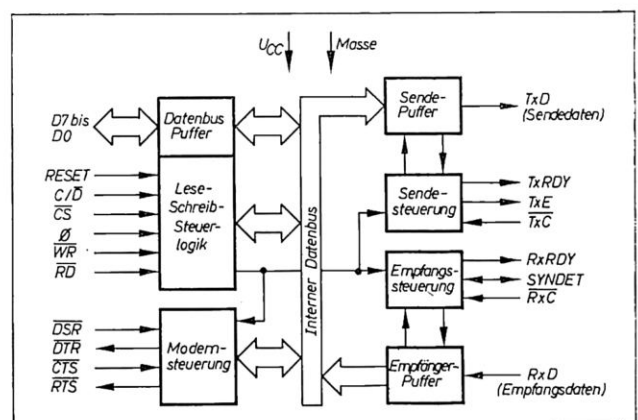


Bild 1.3.5. (oben):
Blockschaltbild
des Interface-
Bausteins 8251

Bild 1.3.6. (rechts):
Typischer
Datenblock
beim 8251

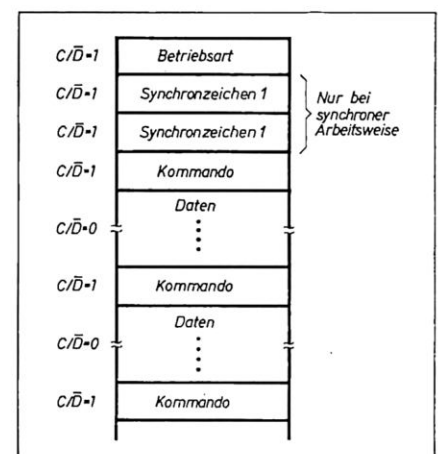
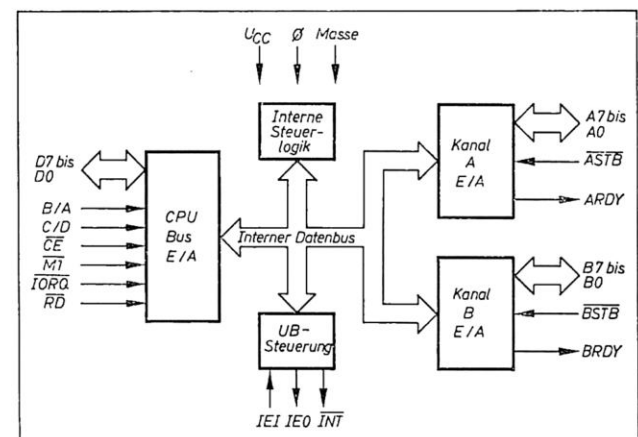


Bild 1.3.7. (unten):
Blockschaltbild
des Interface-
Bausteins Z 80-PIO



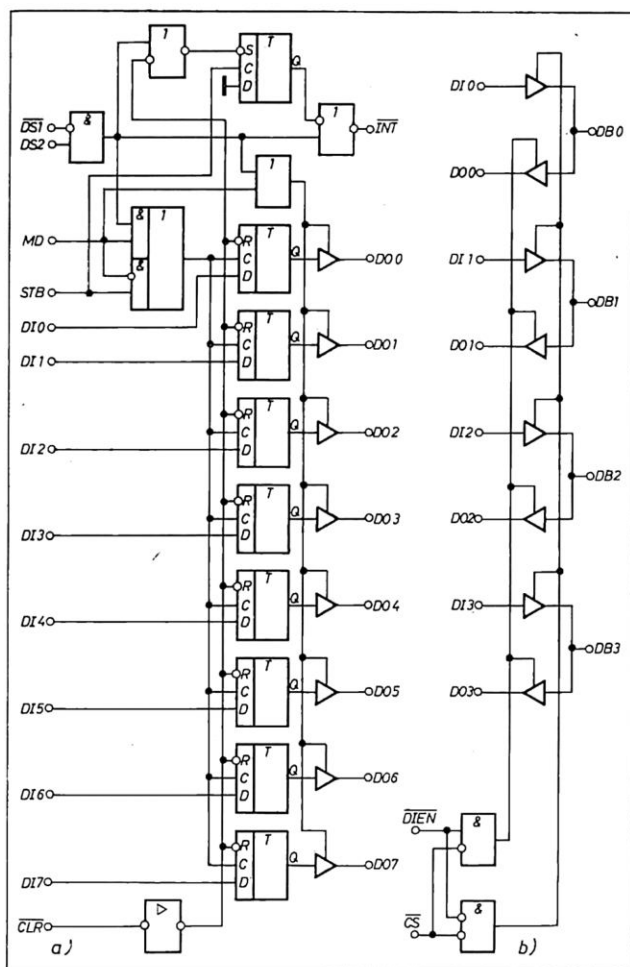


Bild 1.3.8.: Blockschaltbilder von Anpassungsbausteinen. a) 8212; b) 8216

der angelegten Eingabedaten erzeugt und löst eine Unterbrechungsanforderung ($\overline{INT} = 0$) aus. Durch die damit aktivierten Unterbrechungsprogramme muß dann eine weitere Ausgabe bzw. das Lesen der im PIO abgelegten Daten durch die CPU erfolgen. Das Signal Ready teilt der Peripherie mit, daß im PIO Ausgabedaten zur Übergabe bzw. daß bereits Eingabedaten für die CPU bereitstehen, so daß keine weiteren Daten von der Peripherie gesendet werden dürfen.

Die Betriebsart Bit-Ein- und Bit-Ausgabe ist besonders für Prozeßsteuerungen geeignet, dabei wird ohne Quittungssignale gearbeitet. Durch entsprechende Steuerworte können die einzelnen Anschlüsse der Kanäle A und B als Ein- oder Ausgänge programmiert werden, weiterhin können 0- oder 1-Signale an vorgebbaren Eingängen zur Erzeugung von Unterbrechungsanforderungen dienen.

Nach dem Einschalten der Betriebsspannung und dem automatischen Anfangsrücksetzen muß zuerst die Programmierung erfolgen. Es sind folgende Steuerworte möglich:

- UB-Vektor (wird bei eintretender UB an die CPU gesendet)
- Festlegung der Betriebsart
- UB-Steuerwort
- Festlegung der Anschlüsse als Ein- oder Ausgänge bei der Betriebsart Bit-Ein- und Ausgabe
- UB-Maske (Festlegung der Eingänge, die bei Betriebsart Bit-Ein- und Ausgabe eine Unterbrechung auslösen können).

Erst danach kann die Nutzung des Bausteins für die eigentliche Datenübertragung erfolgen.

1.3.5. Anpassungsbausteine

Die Belastbarkeit der Ausgänge von MOS-CPU's reicht nicht aus, um die restlichen Komponenten in größeren Mikrorechnersystemen anzusteuern, beim U 808 ist z. B. nur eine Low-Power-TTL-Last zulässig. Von einigen Bauelementherstellern wurden deshalb spezielle Anpassungsbausteine in Schottky-TTL-Technologie entwickelt, die sehr geringe Eingangsströme

aufweisen und mit höher belastbaren Three-State-Ausgangsstufen ausgerüstet sind. Als Beispiel sollen die Schaltkreise 8212 und 8216 (Intel) dienen, deren Innenschaltungen Bild 1.3.8. zeigt.

Charakteristische Werte

Eingangsstrom bei U_{low}	$I_{in} < 0,25 \text{ mA}$
Ausgangspegel	$U_{high} > 3,65 \text{ V}$
Ausgangsbelastbarkeit bei U_{low}	$I_{out} > 15 \text{ mA (DO)}$ $I_{out} > 30 \text{ mA (DB)}$

Der 8212 ist ein 8-Bit-Register (Latch) mit ausgangsseitigen Three-State-Puffern. Die Arbeitsweise wird durch MD bestimmt: Bei $MD = 0$ wirkt STB (Strobe) als statisches Taktsignal für die internen D-Flip-Flops, die Durchschaltung der Ausgänge erfolgt, wenn $\overline{DS1} \cdot \overline{DS2} = 1$ ist. Bei $MD = 1$ wirkt $\overline{DS1} \cdot \overline{DS2}$ als statisches Taktsignal, die Ausgänge sind ständig durchgeschaltet.

Eine zusätzliche Einrichtung dient der Erzeugung von Bedienforderungen in der Zeit vom Auftreten eines Strobe-Signals (STB) bis zum Ende der Bedienung durch Ansprechen des Bausteins durch $\overline{DS1} \cdot \overline{DS2}$. Der 8216 ist ein bidirektionaler 4-Bit-Bustreiber, der durch $\overline{CS} = 0$ aktiviert und dessen Übertragungsrichtung durch DIEN bestimmt wird ($\overline{DIEN} = 0$ von DI nach DB, $\overline{DIEN} = 1$ von DB nach DO).

1.3.6. Beispiel eines Schaltkreissortimentes

Um eine Vorstellung von dem Umfang eines kompletten Schaltkreissortimentes zu geben, ist in Tafel 1.3.3. das Sortiment des Mikrorechnersystems 8080 zusammengestellt. Allein aus dieser Zusammenstellung ist ersichtlich, daß der Mikroprozessor (also die CPU) allein noch nichts nutzt, für seine Funktion ist noch eine umfangreiche Peripherie erforderlich, die sich den verschiedenen Einsatzbedingungen anpassen lassen muß.

Tafel 1.3.3.: Vollständiges Bausteinsortiment eines Mikrorechners (System 8080 A)

8-Bit-CPU	8080 A
Taktgenerator für CPU	8224
Bus-Steuer-Baustein für CPU	8228
EPROM (256 × 8 bit)	8702 A ¹⁾
EPROM (1K × 8 bit)	8708 ²⁾
EPROM (2K × 8 bit)	8716 ²⁾
maskenprogrammierbarer ROM (256 × 8 bit)	8302 ¹⁾
maskenprogrammierbarer ROM (1K × 8 bit)	8308 ²⁾
maskenprogrammierbarer ROM (2K × 8 bit)	8316 ²⁾
statischer RAM mit getrennten Ein- und Ausgängen (256 × 4 bit)	8101-2
statischer RAM mit gemeinsamen Ein- und Ausgängen (256 × 4 bit)	8111-2
statischer RAM (1K × 1 bit)	8102 A
statischer RAM mit gemeinsamen Ein- und Ausgängen (1K × 4 bit)	2114
statischer CMOS-RAM (256 × 4 bit)	5101
dynamischer RAM (4K × 1 bit)	8107 B
dynamischer RAM im DIL-16-Gehäuse (4K × 1 bit)	2104
dynamischer RAM im DIL-16-Gehäuse (16K × 1 bit)	2116
Refresh-Baustein für dynamische RAMs	8222
Register (8 bit)	8212
bidirektionaler Bustreiber (4 bit)	8216/8226
1-aus-8-Dekoder	8205
Prioritätssteuerung für Interrupt	8214
programmierbarer Baustein für serielle Datenein- und Datenausgabe	8251
programmierbarer Baustein für parallele Datenein- und Datenausgabe	8255
programmierbarer Zeitgeber	8253
programmierbare DMA-Steuerung	8257
programmierbare UB-Steuerung	8259

¹⁾ untereinander pinkompatibel; ²⁾ untereinander pinkompatibel;
³⁾ untereinander pinkompatibel

Literatur

- [1.3.1.] Schlechter, J.; Tschelbiew, A.: Mikrorechner – eine Übersicht. radio fernsehen elektronik 24 (1975) H. 8, S. 264–268
- [1.3.2.] Gößler, R.; Schwerte, J.: Auf dem Wege zur Mikroprozessor-Praxis. Elektronik 25 (1976) H. 3, S. 74–86
- [1.3.3.] Höhne, M.: Der Mikroprozessor U 808 D. radio fernsehen elektronik 26 (1977) H. 5, S. 145–150; H. 6, S. 187–198
- [1.3.4.] Shima, M.; Faggin, F.; Ungermann, R.: Z 80 chip set heralds third microprocessor generation. Electronics (1976) 19. August, S. 89–93
- [1.3.5.] Knobloch, J.: Der Speicherschaltkreis U 253. Vortrag, gehalten auf dem 8. Halbleiterbauelementesymposium vom 11.–13. 10. 77 in Frankfurt (Oder)

1.4. Mikrorechnerkonfigurationen

Eine der Schaltungsentwicklung im allgemeinen zugrundeliegende Forderung ist die nach einem genügend breiten Einsatzspektrum der Elemente. Diese Bedingung wird von Mikroprozessoren erfüllt (s. Abschnitt 1.1.). Nun ist jedoch das Prozessorelement allein noch nicht in der Lage, ihm gestellte Aufgaben zu erfüllen. Zum Aufbau eines autonom arbeitenden Mikrorechners ist zunächst grundsätzlich eine Stromversorgungseinheit notwendig. Das Vorhandensein dieser Komponente ist jedoch allgemein eine Voraussetzung für den Betrieb elektronischer Baugruppen und in keiner Weise spezifisch für Mikrorechner.

Weitere wichtige Komponenten zum Aufbau eines Mikrorechners sind

- Speichermodul(n) zur Aufnahme von Programmen und Daten
- E/A-Steuereinheit(en) für die Kommunikation des Rechners mit der Umwelt
- Taktgenerator
- Bedienkonsole.

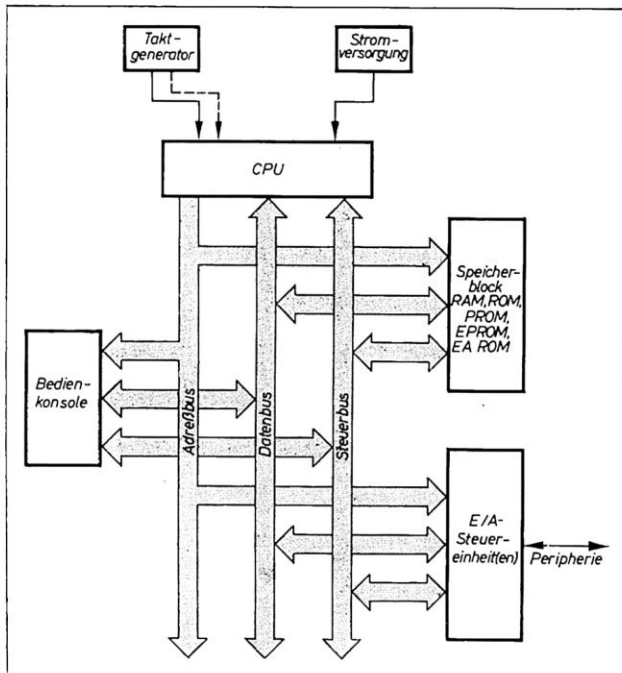


Bild 1.4.1.: Minimalconfiguration eines Mikrorechners

Damit stellt die im Bild 1.4.1. gezeigte Anordnung die minimale Ausstattung eines autonom arbeitenden Mikrorechners entsprechend der in Abschnitt 1.1. gegebenen Begriffserläuterung dar.

Einige der auf dem Weltmarkt angebotenen Prozessorschaltkreise enthalten auf einem Chip neben der CPU bereits einen RAM-Speicher (typ. 64 Byte), einen ROM-Speicher (typ. 1 K Byte) sowie E/A-Steuerschaltungen und eine Logik zur Taktableitung, so daß dafür extern nur noch die Schwingungserzeugung (Quarz, RC-Glied) vorzunehmen ist.

Da damit die benötigten Funktionsgruppen eines Mikrorechners vorhanden sind, kann die für diese Schaltkreise mitunter bereits gebrauchte Bezeichnung „Ein-Chip-Rechner“ als gerechtfertigt angesehen werden. Das Ein-Komponenten-System Intel 8748 verfügt z. B. über die bekannten, durch einen internen Datenbus verbundenen Funktionsgruppen wie ALU, Registerblock, Befehlsregister/-dekoder, Akkumulator, Programmzähler und zusätzlich sowohl über einen RAM- als auch einen ROM-Bereich.

Im 64-Byte-RAM-Bereich befinden sich acht wahlfrei adressierbare 8-Bit-Register sowie ein 8-Ebenen-Stapelspeicher. Der verbleibende Speicherraum kann entweder zur Einrichtung eines zusätzlichen Registerblocks bzw. vollständig als Schreib-Lese-Speicher für die vom System zu verarbeitenden Daten genutzt werden. Als Festwertspeicher hat der Schaltkreis 8748 einen 1-K-Byte-EPROM. Auf Grund dieser Flexibi-

lität sowie seiner E/A-Möglichkeiten läßt sich der Schaltkreis sehr gut während der Phase des Austestens einer Aufgabenrealisierung einsetzen, bevor auf einen Mikroprozessor mit PROM übergegangen wird.

Nach [1.4.2.] ist gegenwärtig eine Tendenz des Wachstums der Länge von Anwenderprogrammen zu verzeichnen. Danach erstellen z. B. in Großbritannien nur noch etwa 15% der Anwender Mikrorechnerprogramme, die kürzer als 1 K Byte sind; typische Programmlängen liegen gegenwärtig im Bereich von 0,5...4 K Byte. Damit wären zur Zeit die Ein-Chip-Rechner mit ihren (1...2)-K-Byte-ROMs bei einigen Aufgaben überfordert bzw. Realisierungen nur durch Zusatzschaltkreise möglich.

Unter diesem Gesichtspunkt ist eine umfangreichere Ausbaustufe bei den als Ein-Karten-Rechnern (engl.: Single Board Computer, SBC) bezeichneten Anordnungen zu finden. Hier sind auf einer Leiterkarte (z. B. „DoppeleuropafORMAT“ 233 mm x 160 mm) CPU, Speicherbausteine, E/A-Steuereinheit(en), Taktgenerator und in einigen Fällen Bedienelemente vereinigt, so daß nach Anschluß an eine entsprechende Stromversorgungseinheit ein autonomer Betrieb möglich wird.

Steht die in den vorangegangenen Ausführungen als „Mikrorechner“ eingeführte Konfiguration schließlich in einem oder mehreren Gehäusen samt seiner dem Anwendungsfall entsprechenden Peripherie sowie der zugehörigen Software (Steuerprogramm, Betriebshilfen wie z. B. residenter Assembler, Debugger u. a.) zur Verfügung, so wird dafür häufig der Begriff „Mikrorechnersystem“ gebraucht. Dabei wird in vielen Fällen auch die Nutzer-Software in den Software-Begriff des Mikrorechnersystems einbezogen.

Der eingangs dieses Abschnittes aufgestellten Forderung nach einem breiten Einsatzspektrum tragen nun die angebotenen unterschiedlichen Mikrorechnerkonfigurationen durch qualitative und vor allem quantitative Strukturunterschiede Rechnung. Auf Grund der im allgemeinen gewährleisteten Modularität im Strukturaufbau ist eine Anpassung der jeweiligen Hardware-Konfiguration an unterschiedliche Aufgabenstellungen relativ leicht möglich. Dazu werden von den verschiedenen Herstellern im Rahmen ihrer Bausteinfamilien (s. Abschnitt 1.3.) z. B. Moduln zur Erweiterung der Speicherkapazität (RAM, (P)ROM), programmierbare Bausteine zur Organisation der gewünschten E/A-Operationen mit der für die jeweilige Aufgabe notwendigen Peripherie, programmierbare Zählerbausteine für Echtzeitsteuerungen usw. angeboten. Die Palette möglicher Konfigurationen ist extrem breit, und diese Artikelserie bietet nicht genug Raum für den Versuch einer detaillierten Darstellung unterschiedlichster Realisierungsvarianten. Es sei vielmehr auf zwei in gewisser Hinsicht als Grenzen anzusehende Konfigurationen hingewiesen, auf die sogenannten Kits und die Gruppe der vollständig ausgebauten Entwicklungssysteme.

Eine Minimalconfiguration umfaßt die zur Funktionsfähigkeit als autonomer Mikrorechner unbedingt notwendigen Komponenten (s. Bild 1.4.1.). Mit dem Begriff Kit wird nun ein Baukasten bezeichnet, der zunächst den selbständigen Aufbau einer derartigen Minimalconfiguration ermöglicht. Er enthält einen Satz von Moduln, deren Zusammenstellung unter dem Gesichtspunkt minimaler Anschaffungskosten konzipiert ist. Diese Moduln für eine Basisvariante (Standardausstattung) werden im allgemeinen in Einzelteilen, ohne Gehäuse und ohne Stromversorgung angeboten. Der Modulsatz kann eine Grundplatine mit Steckplätzen für eine CPU-Karte, für Speicherkarte(n) und eine Karte zur E/A-Steuerung umfassen. Mitunter wird auch nur eine Leiterkarte mit einer Vielzahl von Steckplätzen für die in einem Plastebeutel mitgelieferten Schaltkreise angeboten (Anlehnung an SBC-Konzeption). Zur Kommunikation mit dieser Basisvariante gehört im Minimalfall eine Hexadezimal-Tastatur sowie eine mehrstellige Hexadezimal-Anzeige.

Das Bild 1.4.2. zeigt als Beispiel einen von der Firma Zilog angebotenen Bausatz. Die im Bild gestrichelten Umrandungen deuten mögliche Erweiterungen an.

Aus dieser Basisvariante können prinzipiell durch Zusatzmoduln je nach Anwenderbedarf leistungsfähigere Konfigurationen entwickelt werden. Von einer Verwendung voll ausgebaute Kits als sog. Entwicklungsplätze (für Hardware-

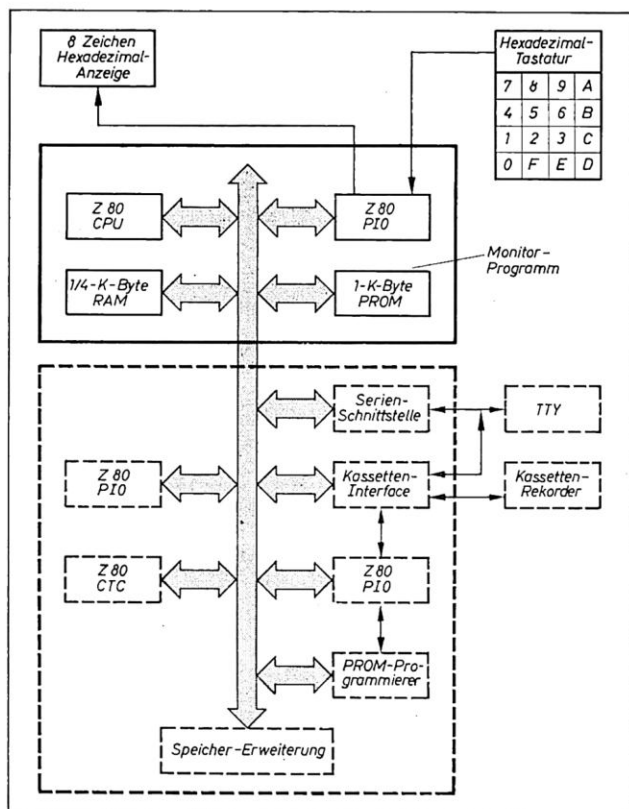


Bild 1.4.2.: Kit-Variante der Firma Zilog

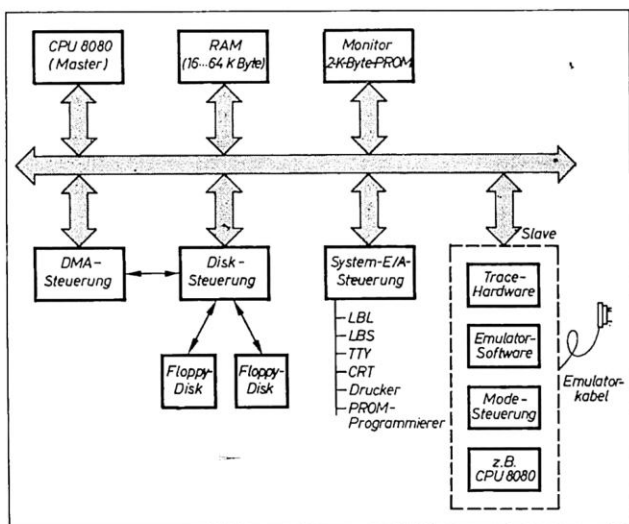


Bild 1.4.3.: Entwicklungssystem Intellec MDS

und Softwaretests bei zu realisierenden Aufgabenstellungen) wird im allgemeinen jedoch abgeraten, weil die vom Hersteller mitgelieferte Software und dabei speziell die Prüfprogramme (Debugger) für kompliziertere Aufgabenstellungen nicht genügen. Dagegen eignet sich die Kit-Konzeption jedoch sehr gut für Ausbildungs- und Trainingszwecke.

Für den Einsatz als sogenannte Entwicklungssysteme wurden von den Schaltkreisherstellern mit ihren einzelnen Mikroprozessortypen spezielle Systemkonfigurationen entwickelt. Diese dürften nach Umfang und Leistungsfähigkeit als obere Grenze dessen anzusehen sein, was mit einem bzw. bei Emulationsabläufen auch bereits mit zwei Mikroprozessoren zu verwirklichen ist. Bild 1.4.3. zeigt die Blockdarstellung des von der Firma Intel angebotenen Systems Intellec MDS (Microcomputer Development System).

Bei dieser Anlage handelt es sich um ein modular auf- und abrüstbares System, das auf Grund seiner Konzeption prinzipiell für Untersuchungen von Aufgabenrealisierungen mit unterschiedlichen μ P-Typen (der Herstellerfirma!) genutzt werden kann. Besonders ausgebaut ist es jedoch für Untersuchungen von Anwendungen der CPU 8080.

Das Hauptgerät des MDS umfaßt in einem Gehäuse eine Stromversorgungseinheit, den Prozessormodul, den 16-K-Byte-RAM, den 2-K-Byte-ROM und eine Bedienkonsole.

Diese Grundausführung verfügt über Anschlüsse für

- ein Konversationsterminal (110 Baud)
- einen Fernschreiber (TTY) (6 Zeichen/s)
- ein Datensichtgerät (110...9 600 Baud)
- einen Lochbandleser (200 Zeichen/s)
- einen Lochbandstanzer (75 Zeichen/s)
- einen Schnelldrucker (165 Zeichen/s)
- ein PROM-Programmiergerät.

Als Software gehören ein System-Monitor einschließlich Debug-Programm sowie ein residenter Assembler und Editor zum System. Weiterhin wird ein Floppy-Disk-Betriebssystem angeboten, das weitere Unterprogramme enthält. Für eine Programmierung in PL/M, einer höheren Programmiersprache, wird ein PL/M-Compiler angeboten, dessen Einsatz allerdings einen Ausbau der 16-K-Byte-RAM-Kapazität des Grundmodells verlangt.

Sowohl Intellec MDS als auch andere bekannte leistungsfähige Entwicklungssysteme bieten darüber hinaus die Möglichkeit einer sogenannten In-Circuit-Emulation. Unter Emulation wird im allgemeinen die programmäßige Nachbildung eines Rechners X auf einer anderen Anlage Y derart verstanden, daß die für den Rechner X erstellten Programme auf der fremden Anlage Y konfliktlos abgearbeitet werden können. Das System Y verhält sich also nach außen hin wie der Rechner X und ist wie dieser zu programmieren.

Mit dem Auf- und Ausbau leistungsfähiger Entwicklungssysteme wurde nun der Begriff In-Circuit-Emulator (ICE) geprägt. Vergeben wird dieser Begriff gegenwärtig für die von derartigen Systemen gebotene Möglichkeit der Nachbildung bestimmter Hardware-Details aus einem Anwendersystem durch das Entwicklungssystem selbst.

Realisiert wird diese als wichtiges Leistungsmerkmal einzuschätzende Betriebsweise durch eine entsprechende Synthese von speziellen Programmen und Hardware-Details. Im Bild 1.4.3. ist die dazu notwendige Hardware im als Slave gekennzeichneten Teil des Blockschaltbildes dargestellt.

Für den Nutzer von Intellec MDS bedeutet das, daß er z. B. die CPU seines Hardwareentwurfs mit Hilfe des MDS realisiert. Dazu stellt er eine Verbindung zwischen dem Steckplatz der CPU in seiner Schaltung und dem MDS mit dem Emulator-Kabel her und hat damit für das Austesten seiner Lösung die Software-Entwicklungshilfen des MDS zur Verfügung. So ermöglicht z. B. die Trace-Hardware im Slave-Teil des MDS in einem bestimmten Umfang und an vom Nutzer wählbaren Punkten eine „Protokollierung“ des Programmablaufs durch die Abspeicherung von während der Programmabarbeitung aufgerufenen Adressen.

Zusammenfassend sei festgehalten, daß derartig ausgebaute Systeme zu einem wertvollen Hilfsmittel und u. U. sogar zur Voraussetzung für ein optimales Herangehen bei Hardware- und Software-Entwicklungen werden.

Wird fortgesetzt

Literatur

- [1.4.1.] Höhne, M.: Der Mikroprozessor U 808 D. radio fernsehen elektronik 26 (1977) H. 5, S. 145-150; H. 6, S. 187 und 188; 197 und 198
- [1.4.2.] Koch, G. R.: Stand und Trends der Programmierung von Mikroprozessoren. Elektronik 26 (1977) H. 1, S. 63-71
- [1.4.3.] Labudda, H.-J.: Mikroprozessor - Kehrseite der Medaille. Elektronik 26 (1977) H. 1, S. 55-57
- [1.4.4.] Meiling, W.: Einige Tendenzen in der Entwicklung und Anwendung von Mikroprozessoren und Mikrorechnern. Nachrichtentechnik - Elektronik 27 (1977) H. 1, S. 7-10
- [1.4.5.] Tschelebiev, A.; Schlechter, J.: Mikrorechner - eine Übersicht. radio fernsehen elektronik 24 (1975) H. 8, S. 264-268

1.5. Multimikroprozessor- und Multimikrorechnersysteme

Die beiden in der Überschrift genannten Begriffe tauchen in letzter Zeit häufiger in der Fachliteratur auf, und Fragen der Realisierung derartiger Systeme sind Gegenstand internationaler Konferenzen. Das jüngste Ergebnis dieser Entwicklung ist ein von der Firma Intel auf der Anfang 1977 in den USA durchgeführte International Solid State Circuit Conference ISSCC 77 vorgestelltes Zweiprozessorsystem auf einem Chip. Das in n-MOS-Technik gefertigte Bauelement enthält auf diesem einen Chip etwa 22 000 Transistoren. Als Vergleich dazu wird der im Abschnitt 1.4. erwähnte Ein-Chip-Mikrorechner 8048 der gleichen Firma mit etwa 18 000 Transistoren realisiert.

Bevor nun in diesem Abschnitt auf die Notwendigkeit zur Schaffung solcher Konfigurationen eingegangen wird, soll zunächst eine Begriffserläuterung erfolgen.

Unter einem Multimikroprozessorsystem wird im allgemeinen ein Rechner verstanden, der zwei oder mehr Mikroprozessoren enthält, die unter einer gemeinsamen Steuerung arbeiten. Der Begriff Mikroprozessor wird dabei in erweitertem Sinne verwendet, d. h., der Mikroprozessor kann bereits über einen eigenen Speicherbereich verfügen. Er kann jedoch noch nicht als Mikrorechner betrachtet werden, weil er nicht als autonom arbeitsfähige Einheit die Funktionen eines Rechners voll erfüllen kann.

Zusammenfassend lassen sich folgende wesentliche Merkmale eines Multimikroprozessorsystems angeben:

- Das System enthält zwei oder mehr Verarbeitungseinheiten.
- Es existiert ein gemeinsamer Hauptspeicher; die Form der Zusammenarbeit mit diesem Speicher kann in der Zugriffsmöglichkeit aller μP des Systems auf den gesamten Speicherbereich bis zur Reservierung bestimmter Speicherbereiche für einzelne (oder jeden) μP variieren.
- Das E/A-System ist funktionell aufteilbar, d. h., es ist von jedem μP des Multimikroprozessorsystems erreichbar.
- Das gesamte System arbeitet unter der Steuerung eines einheitlichen Betriebssystems.
- Für die Funktionsweise des Systems ist auf allen Ebenen eine ausgeprägte Wechselwirkung sowohl zwischen den einzelnen Softwareabschnitten als auch zwischen den Hardwarekomponenten charakteristisch.

Als Multimikrorechnersystem bezeichnet man einen Verband von zwei oder mehr Mikrorechnern, also autonom funktionsfähigen Anordnungen, mit jeweils selbständiger Steuerung. Nach Auflösung eines derartigen Verbandes wäre jede Einheit in der Lage, ihr zugewiesene Aufgaben allein zu lösen. Damit lassen sich die folgenden, die Unterschiede zu einem Multimikroprozessorsystem kennzeichnenden wesentlichen Merkmale angeben:

- Ein Multimikrorechnersystem enthält zwei oder mehr Mikrorechner.
- Es existiert im allgemeinen kein gemeinsamer Hauptspeicher.
- Die Verbindung zwischen den Mikrorechnern kann über deren E/A-Systeme hergestellt werden.

Von Multimikrorechnersystemen sind grundsätzlich zwei Vorteile zu erwarten:

- Leistungssteigerung durch Einsatz mehrerer Verarbeitungseinheiten
- höhere Zuverlässigkeit und Verfügbarkeit durch redundanten Aufbau der Systeme.

TECHNIK DER MIKRORECHNER

9

Der erste Vorteil resultiert aus der noch mehr oder weniger begrenzten Leistungsfähigkeit eines einzelnen Mikroprozessors. Im Ergebnis einer möglichen Arbeitsteilung innerhalb eines Verbandes mehrerer Mikroprozessoren durch parallele Bearbeitung entsprechender Teilaufgaben ist prinzipiell eine Leistungssteigerung bei der Realisierung bestimmter Aufgaben zu erzielen [1.5.1.].

Zum zweiten Gesichtspunkt ist zu bemerken, daß mit LSI-Schaltkreisen auf Grund der hohen Zuverlässigkeit dieser Bausteine sowie der reduzierten Anzahl notwendiger externer Verbindungen Systeme mit hoher Funktionssicherheit realisiert werden können. Darüber hinaus bestehen noch während der Entwurfsphase von Multimikroprozessor- und Multimikrorechnersystemen Möglichkeiten der Einleitung besonderer Maßnahmen zur Erhöhung der Sicherheit:

- Sicherung einer hohen Verfügbarkeit durch redundanten Aufbau des Systems
- Einsatz spezieller Prüfprozessoren, z. B. für die Zeitüberwachung bestimmter Operationen, für die Paritäts- und Kodeprüfung usw.
- Informationsduplizierung in einen „sicheren“ Speicherbereich, d. h. auf ein Speichermedium, das bei einem eventuellen Spannungsausfall die ihm übergebene Information nicht verliert (Floppy Disk, Magnetband, Kernspeicher).

Beim Auftreten bzw. nach Erkennen eines Fehlers ist eine differenzierte Reaktion der Systeme möglich. Im allgemeinen wird zunächst ein als defekt erkannter Modul automatisch abgetrennt. Hat der aufgetretene Fehler jedoch eine Gefährdung der Funktionsfähigkeit des gesamten Systems zur Folge, so werden alle Module zur „sicheren Seite hin“ abgeschaltet (engl.: fail safe system), d. h., das System wird auf eine solche Weise stillgelegt, daß der Defekt zu keiner Gefährdung der vom System beeinflussten „Umwelt“ führt. Bedeutet der Ausfall des Moduls nur eine Einschränkung der Leistungsfähigkeit des Systems, so wird nach dem Außerbetriebsetzen der defekten Einheit die Aufgabenbearbeitung fortgesetzt (engl.: fail soft system).

Zusammenfassend kann damit der Aufbau von Multimikroprozessor- und Multimikrorechnersystemen folgende Eigenschaften der Systeme gewährleisten:

- gleichmäßige Belastung der Mikroprozessoren
- mögliche Reduzierung des Hardware-Aufwandes durch eine der Aufgabenstellung entsprechend angepaßte Systemorganisation
- einheitliche Fehlerüberwachung
- Möglichkeit der Bearbeitungsfortsetzung nach Ausfall eines Elementes durch Systemumstrukturierung
- vom System selbständig ausführbare Wiedereinschaltung eines reparierten Elementes in die Systemarbeit
- eine große Flexibilität und Wartungsfreundlichkeit des Systems durch einen einheitlichen modularen Aufbau.

Einen weiteren wichtigen Gesichtspunkt, der allerdings im konkreten Einsatzfall immer in Verbindung mit der zu lösenden Aufgabenstellung zu sehen ist, stellt die für das jeweilige System gewählte Form der Arbeitsorganisation dar. Auf Grund entsprechender Aufgabenzuordnungen und Wechselwirkungen zwischen einzelnen Modulen werden für bestimmte Strukturen in der Literatur spezielle Begriffe verwendet. Im folgenden sollen fünf der gebräuchlichsten kurz erläutert

werden, sie sind jedoch nicht im Sinne einer strengen Klassifizierung aufzufassen.

● Mehrfachsystem

Mit diesem Begriff werden zur allgemeinen Charakterisierung Systeme bezeichnet, die mehrere, untereinander als Teile eines Gesamtsystems zusammenarbeitende Mikroprozessoren oder Mikrorechner enthalten. Für die Realisierung der übertragenen Aufgabe ist die richtige Funktion jeder Einheit notwendig.

Entsprechend ihrer Betriebsart ist eine nochmalige Unterteilung der Systeme möglich in

- Systeme mit fester Aufgabenzuweisung an die einzelnen Systemkomponenten
- Systeme mit dynamischer Aufgabenzuweisung an die einzelnen Systemkomponenten.

Die folgenden Begriffe bringen gegenüber der vorstehenden allgemeinen Bezeichnung stärker eine bestimmte Spezifik des jeweiligen Systems zum Ausdruck.

● Redundanzsystem

Um eine hohe Zuverlässigkeit zu erreichen, enthält das System zwei oder mehrere gleichartige Mikroprozessoren oder Mikrorechner, die die gleichen Programme mit der gleichen Datenmenge ausführen. Die Ergebnisse der Operationen der entsprechenden Funktionseinheiten werden miteinander verglichen, und bei Übereinstimmung gelangen sie auf den Ausgang des Systems.

● Pipeline-System

Die zu verarbeitenden Daten (Zwischenergebnisse) „fließen“ von einem Teilsystem zum anderen. Die einzelnen Einheiten wiederholen ihre (spezifischen) Operationen fortlaufend, so daß sich eine überlappte Arbeitsweise ergibt (Fließbandverfahren).

● Parallelsystem

Mehrere Mikroprozessoren oder Mikrorechner führen gleichzeitig (parallel) ihre Operationen mit den Elementen einer jedem Teilsystem zugeordneten Teilmenge der Gesamteingabemenge des Systems aus. Von diesen Teilsystemen werden dann jeweils auch nur Teilmengen der Ausgabemenge des Systems bereitgestellt. Charakteristisch ist eine sehr geringe Zusammenarbeit der Mikroprozessoren bzw. Mikrorechner.

● Netzsystem

Die grundsätzlich unabhängig voneinander arbeitenden Mikrorechner kommunizieren miteinander über ein Nachrichtennetz. Diese lose Verbindung beschränkt sich auf einen Austausch bestimmter Informationen.

In Multimikroprozessorsystemen wird die Arbeit der einzelnen Prozessoren von einem gemeinsamen Betriebssystem gesteuert. Das Betriebssystem realisiert die Aufgabenverteilung nach den vorstehend genannten Prinzipien und verwaltet die dazu entsprechend notwendigen Verbindungen zwischen den Systemkomponenten. Die Arbeit des Betriebssystems selbst kann nach drei Prinzipien organisiert sein:

- Master-Slave-Betriebssystem: Das Supervisor-Programm läuft immer auf einem ausgewählten Mikroprozessor.
- Verteiltes Betriebssystem: Die Supervisor-Aufgaben sind in den eigenen Speichern der einzelnen Mikroprozessoren abgelegt und werden getrennt ausgeführt.
- Symmetrische Behandlung der Mikroprozessoren: Die identisch aufgebauten Mikroprozessoren führen den Teil des Supervisor-Programms aus, der für die aktuell zu bearbeitende Aufgabe benötigt wird.

Die bereits existierenden Multimikroprozessor- und Multimikrorechnersysteme weisen eine Vielzahl verschiedener Hardware-Ausführungen für die Kopplung auf. Im folgenden sollen fünf Lösungen kurz vorgestellt werden.

Direkte ZVE-ZVE-Verbindung

Der Leistungsumfang der Kopplungsschaltung (u. U. Spezialschaltkreis) variiert von der Gewährleistung einfacher Signalaustauschmöglichkeiten (z. B. für die Synchronisation der Arbeit verbundener ZVE) bis zur vollständigen Interfaceschaltung zwischen den ZVE (s. Bild 1.5.1.).

Zeichenerklärung für die Bilder 1.5.1. bis 1.5.5.

ZVE _i	Zentrale Verarbeitungseinheit ($1 \leq i \leq n$)
SP _{1,j} , SP _{1,k}	} Speicher(modul) ($1 \leq i \leq n, 1 \leq j \leq p, 1 \leq k \leq q$)
SP _j	
E/A _{1,1} , E/A _{1,m}	} Ein- und Ausgabeeinheit ($1 \leq i \leq n, 1 \leq l \leq r, 1 \leq m \leq s$)
E/A _l	
SB ₁	Systembus ($1 \leq i \leq n$)
BSE	Bussteuereinheit
KIF	Kommunikations-Interface

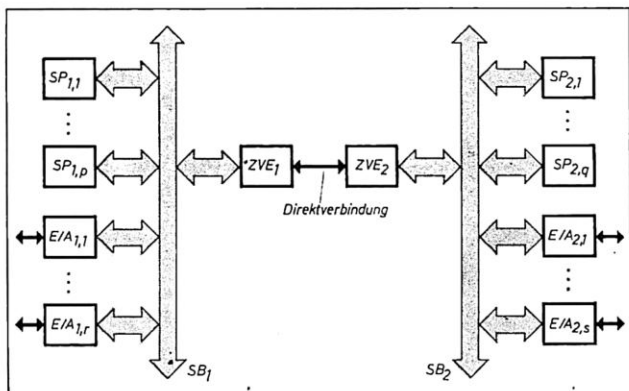


Bild 1.5.1.: Beispiel einer direkten ZVE-ZVE-Verbindung

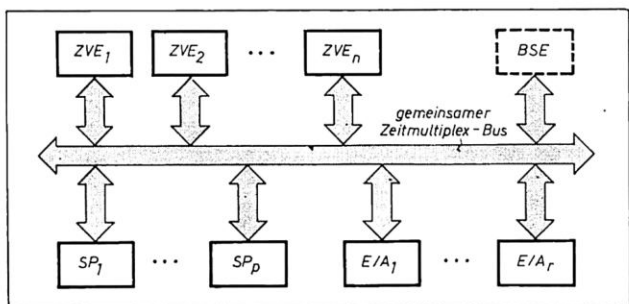


Bild 1.5.2.: Beispiel eines gemeinsamen Zeitmultiplex-Busses

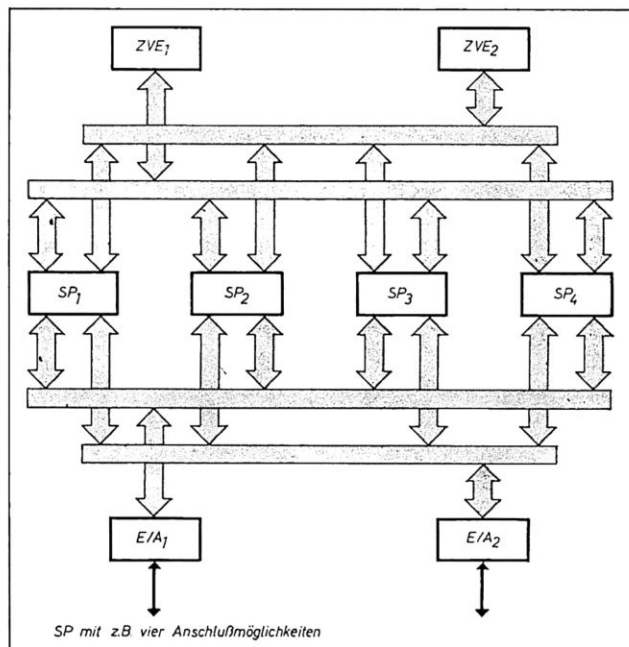


Bild 1.5.3.: Beispiel eines Multiport-Multibus-Systems

Gemeinsamer Zeitmultiplex-Bus

Für die Verbindung zwischen den Funktionseinheiten steht ein gemeinsamer Datenweg (Bus) zur Verfügung. Es darf jedoch zu jedem Zeitpunkt immer nur ein Modul als aktiver Sender den Bus mit Informationen belegen. Die Steuerung der Arbeit auf bzw. mit dem Bus wird entweder von einer der

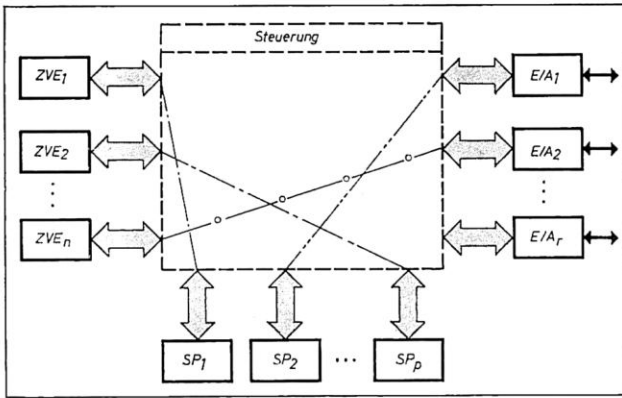


Bild 1.5.4.: Beispiel einer Crossbar-Schaltmatrix

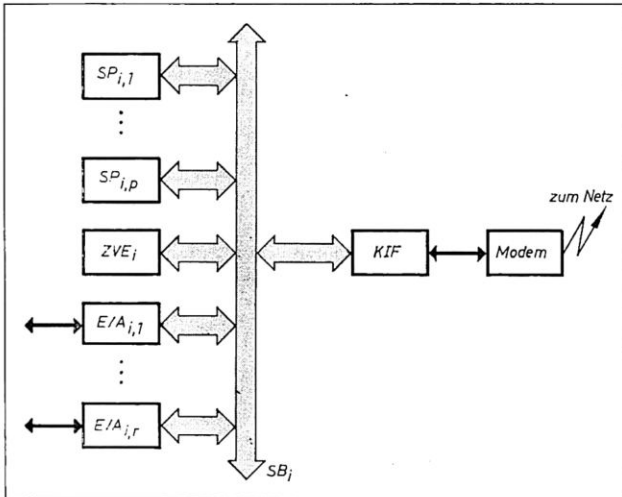


Bild 1.5.5.: Beispiel einer Kopplung über ein Nachrichtenübertragungsnetz

angeschlossenen ZVE oder von einer selbständigen, speziellen Bussteuereinheit übernommen.

Diese Lösung zeichnet sich durch ihre Flexibilität aus, ein Defekt des Busses hat jedoch den Systemausfall zur Folge (s. Bild 1.5.2.).

Multipoint-Multibus-System

Die Verbindung der Komponenten untereinander wird über das Speichersystem, das mehrere Anschlüsse besitzt, realisiert. Die Eigenbussysteme der einzelnen ZVE bzw. E/A-Mo-

dulen werden an die verschiedenen Anschlüsse der Speichermodulen geschaltet. Prinzipiell sind somit innerhalb dieses Systems auch mehrere Datenübertragungen zum gleichen Zeitpunkt möglich (in verschiedenen Komponentenpaaren). Das System ist relativ starr, ein Fehler in einem Verbindungsweg führt unter Umständen nur zur Leistungsminderung bei Funktionsfähigkeit des Gesamtsystems (s. Bild 1.5.3.).

Crossbar-Schaltmatrix

Die Funktionseinheiten werden durch ein Schaltsystem verbunden, das nach dem Kreuzschienenverteilerprinzip arbeitet. Die Steuerung der Schaltmatrix wird entweder von einer ZVE oder von einer speziellen Steuereinheit übernommen. Von den bisher vorgestellten Varianten bietet diese Anordnung die potentiell höchste Leistungsfähigkeit und Verfügbarkeit ihrer Komponenten. Als Nachteil sind jedoch unbedingt die ebenfalls hohen Hardware-Kosten anzuführen.

Bei einem modularen Aufbau der Schaltmatrix ist die Flexibilität der Anordnung gewährleistet. Defekte in Verbindungselementen der Schaltmatrix führen – bis zu einer von der jeweiligen Matrixgestalt abhängigen Zahl und Lage zueinander – nur zur Leistungsminderung bei Funktionsfähigkeit des Gesamtsystems (s. Bild 1.5.4.).

Nachrichtenübertragungsnetz

Die einzelnen Mikrorechner besitzen Anschlußeinheiten (Nachrichtenübertragungsinterface und Modem) zu einem Übertragungsnetz. Die Redundanz des Netzes gewährleistet eine hohe Zuverlässigkeit für die Herstellung und den Betrieb notwendiger Verbindungen zwischen Systemkomponenten. Die Leistungsfähigkeit und Flexibilität des Verbindungsnetzes werden von der Übertragungskapazität bzw. den Gestaltungsvorschriften des Nachrichtenübertragungsnetzes bestimmt (s. Bild 1.5.5.).

Literatur

- [1.5.1.] Widdoes, L. C.: The MINERVA Multi-Microprocessor. IEEE Conference Proceedings 3. Symposium „Computer Architecture“ (1976) S. 34–39
- [1.5.2.] Searle, B. C.; Freberg, D. E.: Microprocessor Applications in Multiple Processor Systems. Computer 8 (1975) H. 10, S. 22–30
- [1.5.3.] Sastry, K. V.; Kain, R. Y.: On the Performance of Certain Multiprocessor Computer Organisations. IEEE Transactions on Computers C-24 (1975) H. 11, S. 1066–1074
- [1.5.4.] Ravindraw, V. K.; Thomas, T.: Characterisation of Multiple Microprocessor Networks. IEEE Proceedings Compcon (1973) S. 133 bis 137
- [1.5.5.] Spang, H. A.: Distributed Computer Systems for Control. IFAC/IFIP Symposium (SOCOCO '76) Tallin (1976), S. 47–66
- [1.5.6.] Arnold, R. G.; Page, E. W.: A Hierarchical Restructurable Multi-Microprocessor Architecture. IEEE Conference Proceedings 3. Symposium „Computer Architecture“ (1976) S. 40–45
- [1.5.7.] Enslow, P. H.: Multiprocessors and Parallel Processing. New York: John Wiley & Sons, 1974

2. Hardware

2.1. Mikroprozessor – ZVE eines Mikrorechners

2.1.1. Allgemeiner Aufbau

Der Mikroprozessor stellt das Kernstück der zentralen Verarbeitungseinheit eines Mikrorechners dar. Der Mikroprozessor ist die LSI-Komponente, die unter Programmsteuerung eine Vielzahl von logischen und arithmetischen Operationen sequentiell ausführt. Er läßt sich entsprechend dem Bild 2.1.1. prinzipiell in vier Grundfunktionseinheiten aufteilen:

Steuereinheit interne Speichereinheit
Recheneinheit interne E/A-Einheit.

Die Steuereinheit umfaßt im wesentlichen das Befehlsregister, den Befehlsdekoder und die gesamte Zeitablaufsteuerung. Sie realisiert in Abhängigkeit vom eingegebenen Befehl und der Belegung der Steuer- und Takteingänge die interne Steuerschleife und/oder die intern mikroprogrammgesteuerten Abläufe des Mikroprozessors und die Ausgabe von Steuerinformationen zu den anderen Schaltkreisen.

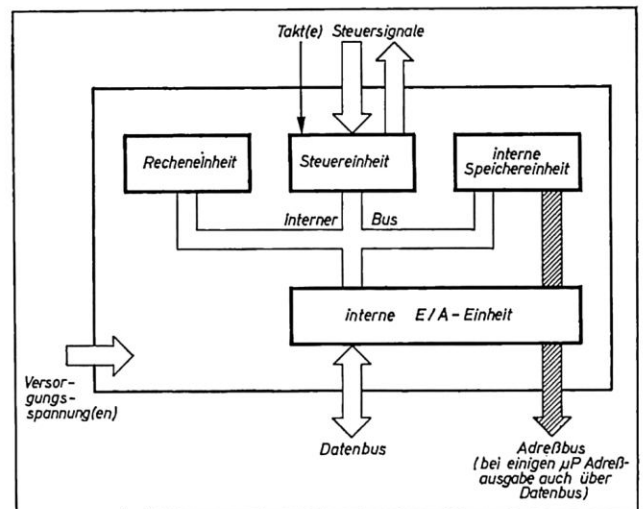


Bild 2.1.1.: Funktionseinheiten des Mikroprozessors

Die Recheneinheit besteht aus der Arithmetik- und Logik-Einheit (ALU), den Kennzeichen- bzw. Markierungs-Flip-Flops (Flags) und den Rechenregistern. Es können zusätzliche Einheiten (Dezimalkorrektureinheit beim 8080) integriert sein.

Die interne Speichereinheit umfaßt im allgemeinen den Programmzähler und die für den Programmierer zugänglichen Zwischenregister (scratch pad register). Sie kann ebenfalls einen in der Ebenenzahl begrenzten Kellerspeicher (Stack) enthalten (U 808 D), der über einen Markierer SP adressiert wird. Viele Mikroprozessoren enthalten nur einen Stack-Pointer, der Stack wird extern im RAM angelegt.

Die interne E/A-Einheit umfaßt die entsprechenden Treiber- und Fangregister (Buffer, Latch), um den Datenbus bidirektional und zusätzliche Busse (z. B. Adreßbus) unidirektional durchzuschalten.

Der interne Verkehr zwischen diesen vier Funktionseinheiten wird über ein internes Bussystem, das ebenfalls von der Steuereinheit gesteuert wird, realisiert.

Heute sind bereits Mikroprozessoren bekannt, die diese internen Funktionseinheiten in weiter ausgebauter Form enthalten. Die Entwicklung geht allgemein in die Richtung, einen kompletten Mikrorechner mit programmierter Ein- und Ausgabe und entsprechender interner Speicherkapazität (PROM, RAM) auf einem Chip zu integrieren. Dieser Trend kann anhand der Schaltkreisentwicklungslinie vom 8008 bis zum 8748 (Intel) verfolgt werden.

In den Bildern 2.1.2. bis 2.1.4. sind die Blockschaltbilder der in diesem Abschnitt untersuchten Mikroprozessoren U 808 D, 8080 (Intel) und Z 80 (Zilog) als Vertreter typischer Familien gezeigt. Dabei sollen insbesondere die Funktionseinheiten nach Tafel 2.1.1. und ihre Bedeutung erläutert werden.

Wird fortgesetzt

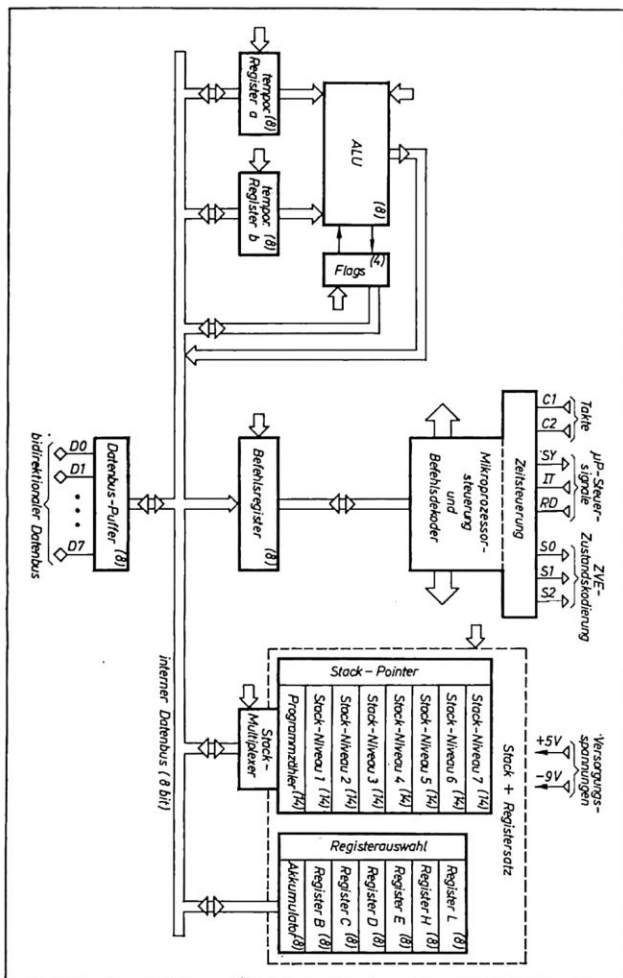


Bild 2.1.2.: Blockschaltung des U 808 D

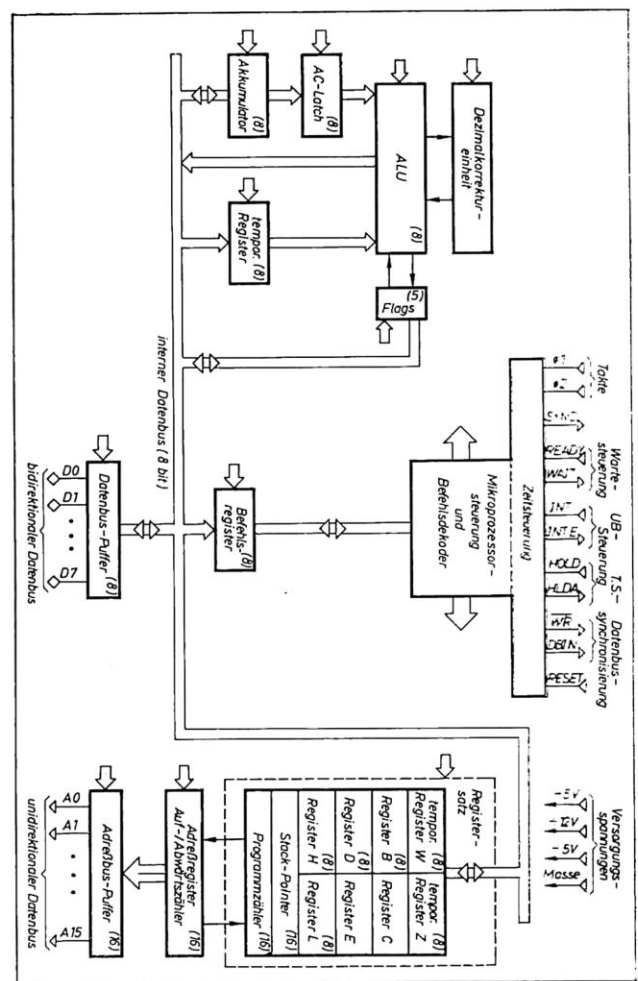


Bild 2.1.3.: Blockschaltung des 8080 (rechts unten Adreßbus statt Datenbus)

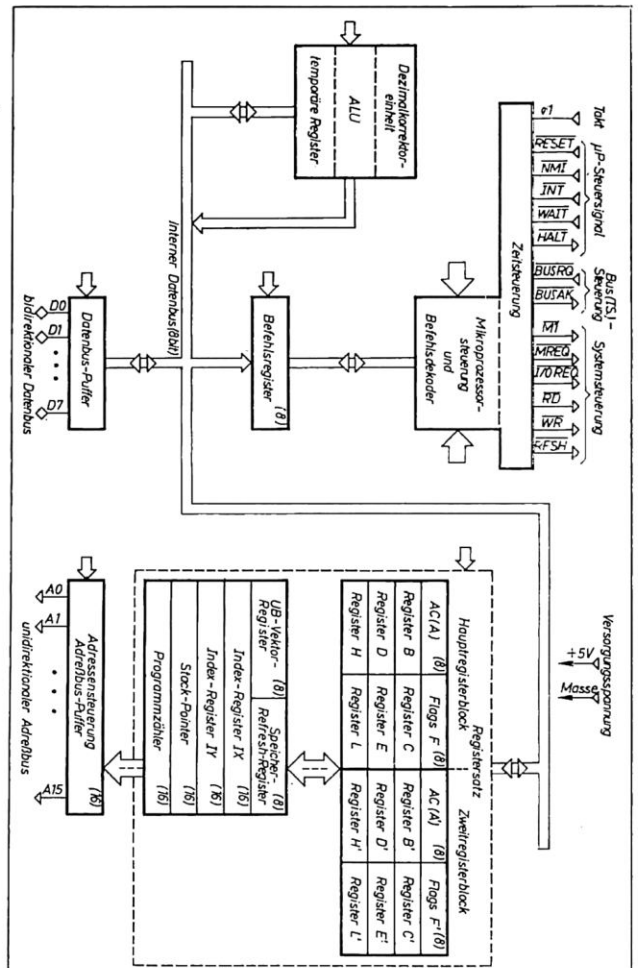


Bild 2.1.4.: Blockschaltung des Z 80-CPU

Dipl.-Ing. JURGEN SCHLECHTER und
Dr.-Ing. STEFAN LINDNER

Der Ausstattungsgrad eines Mikroprozessors mit internen Registern bestimmt seine Leistungsfähigkeit mit. Während ein minimaler Registersatz für die Funktion des Mikroprozessors unerlässlich ist, ermöglicht ein umfangreicherer eine wesentlich effektivere Programmierung und bringt Zeitvorteile, da intern vorhandene Speicherkapazität die Anzahl der RAM-Zugriffe reduziert. Die Operationsgeschwindigkeit des Mikroprozessors ist also unter anderem auch eine Funktion der Registeranzahl.

2.1.2. Ausgewählte Steuerungskomplexe bei Mikroprozessoren und deren Nutzung

In diesem Abschnitt werden solche Funktionen untersucht, die zum besseren Verständnis als Steuerungskomplexe von der Gesamtarbeitsweise des Mikroprozessors abgespalten werden können. In diesen Steuerungskomplexen äußert sich für den Nutzer der Zusammenhang zwischen intern realisierter Ablaufsteuerung des Mikroprozessors und externer Beschaltung der Mikroprozessoranschlüsse.

Die untersuchten Komplexe sollen wieder an typischen Beispielen (U 808 D, 8080, Z 80 CPU) verdeutlicht werden.

2.1.2.1. Taktansteuerung des Mikroprozessors

Für die zeitliche Synchronisation mikroprozessorinterner, aber auch externer Komponenten ist bei den Mikroprozessoren

Tafel 2.1.1.: Funktionseinheiten eines Mikrorechners

Funktionseinheit	Erläuterung
Akkumulator	wichtiges Register, das eng mit dem Rechenwerk verbunden ist, das bei Ein-Adress-Befehlen vor der Operation einen Operanden und bei den meisten arithmetischen und logischen Befehlen das Ergebnis enthält
ALU	Teil des Mikroprozessors, in dem arithmetische, logische und Verschiebeoperationen ausgeführt werden
Programnzähler	Register, das die Adresse der Befehlsinformation, die als nächstes vom Speicher zu holen ist, enthält
Befehlsregister	Register, das den aktuellen Maschinenbefehl (Befehlscode) während der Befehlsabarbeitung zwischenspeichert
Stack (Stapelspeicher, Kellerspeicher, push down memory)	Speicher aus einer Folge von Registern oder Speicherplätzen, bei dem Informationen nur an derselben Stelle entnommen werden können, an der sie hinzugefügt wurden. Das häufig angewandte Entnahmeprinzip wird deshalb auch LIFO (last in – first out; das zuletzt eingespeicherte wird zuerst ausgelesen) genannt
Stack-Pointer	Adressverweis auf den zugänglichen Speicherplatz des Stacks, auf den geschrieben oder von dem gelesen werden soll
Indexregister	Dient zur Adressmodifikation, z. B. bei der Indexadressierung, bei der die effektive Adresse als Summe des Inhaltes eines Indexregisters und des Inhaltes des Adressfeldes des Befehls gebildet wird
Allgemeine Register (scratch pad register, Zwischenspeicher)	Register, die einzeln oder verbunden zur Abspeicherung von Zwischenergebnissen verwendet werden
Flag (bit) (Markierungsbit, Bedingungsbit)	wird von Hardware oder Software gesetzt, um das Eintreten eines bestimmten Ereignisses festzuhalten, so daß es bei Bedarf später abgefragt werden kann. [Speziell im μP werden Flags entsprechend bestimmter, sich aus dem Ergebnis einer logisch-arithmetischen Operation ergebender Bedingungen (z. B. Ergebnis-Null, Parität gerade, Übertrag usw.) festgelegt. Entscheidungsoperationen sind dann vom Stand der Flags abhängig]
Temporäre Register (Pufferregister)	Register, die zur Abspeicherung von internen Informationen dienen; nicht zugänglich für den Programmierer

TECHNIK DER MIKRORECHNER

10

ren und Mikrorechnern eine Taktversorgung allgemein notwendig. Während das große Spektrum der zur Zeit international angebotenen Mikroprozessoren eine externe Takterzeugung voraussetzt, gibt es bereits Mikroprozessoren mit integrierter Taktableitung, wobei allerdings das frequenzbestimmende Element (Quarz, RC-Netzwerk) extern angeschlossen wird.

Der Aufwand für die Erstellung einer externen Taktversorgung wird dadurch bestimmt, ob für den Mikroprozessor ein Einphasen- oder ein Mehrphasentakt (häufig Zweiphasentakt; nichtüberlappend) notwendig ist, und ob die Takte mit TTL-Pegel oder in einem speziellen Pegel zur Verfügung gestellt werden müssen.

Generell ergibt sich die Notwendigkeit, das vom Mikroprozessor-Hersteller geforderte Zeitschema (mit entsprechenden Toleranzen), das durch die dynamische Arbeitsweise des größten Teiles der Mikroprozessoren bedingt ist, einzuhalten (s. Bild 2.1.5.). Eine Möglichkeit des Einzeltaktbetriebes gibt es nur bei Mikroprozessoren mit internen statischen Speicherelementen.

Häufig werden integrierte Schaltkreise zur Taktansteuerung des entsprechenden Mikroprozessors hergestellt (z. B. 8024 – Taktgenerator für 8080). Die Takterzeugung kann prinzipiell in die Funktionseinheiten nach Bild 2.1.6. unterteilt werden.

2.1.2.2. Daten-Ein- und Ausgabeoperationen ohne und mit Fertig/Warte-Steuerung

Die Daten-Ein- und Ausgabeoperation stellt den Oberbegriff sowohl für die Zuweisung (bzw. Zuordnung) der Informationen auf dem Datenbus als auch für ihre zeitliche Synchronisation mit externen Elementen dar.

Unter der Zuordnung der Informationen wird insbesondere bei multivalenter Nutzung des Datenbusses die Entscheidung verstanden, ob es sich um den Befehlscode, Adressinformationen, E/A-Daten, den UB-Vektor usw. handelt.

Die Zuordnung wird durch entsprechende Zustands- (Status-) oder Steuerinformationen angegeben.

Die zeitliche Synchronisation der Ein-/Ausgabeoperationen realisiert durch entsprechende Steuersignale die Auswahl des Zeitintervalls, in dem die Daten auf dem Datenbus aktuell (valid) sind.

Die Fertig/Warte-Steuerung (Ready/Wait-Steuerung) ermöglicht zusätzlich eine Synchronisation des Mikroprozessors mit Elementen, die eine langsamere als die vom Mikroprozessor geforderte Arbeitsgeschwindigkeit bezüglich der Informationsaustauschoperationen besitzen.

Das Prinzip beruht bei den meisten Mikroprozessoren auf der Einführung von Wartezuständen TW. Damit ergeben sich z. B. beim Mikroprozessor 8080 die im Bild 2.1.7. dargestellten vereinfachten Zeitdiagramme für die Ein-/Ausgabe.

2.1.2.3. Unterbrechungsbearbeitung

Die Unterbrechungssteuerung ermöglicht die externe Unterbrechung des internen Programmablaufes im Mikroprozessor. Im folgenden soll ausschließlich die Komponente der Unterbrechungssteuerung behandelt werden, die im Mikroprozessor realisiert ist.

Bei den Mikroprozessoren werden verschiedene Möglichkeiten der Unterbrechungsmarkierung angeboten, die je nach den Erfordernissen der Unterbrechungsverarbeitungsmethode mehr oder weniger durch Zusatzhardware modifiziert werden können. Folgende grundsätzliche Möglichkeiten werden dabei einzeln oder kombiniert genutzt:

● Vektorielle UB-Markierung

Unter vektorieller UB-Markierung ist zu verstehen, daß

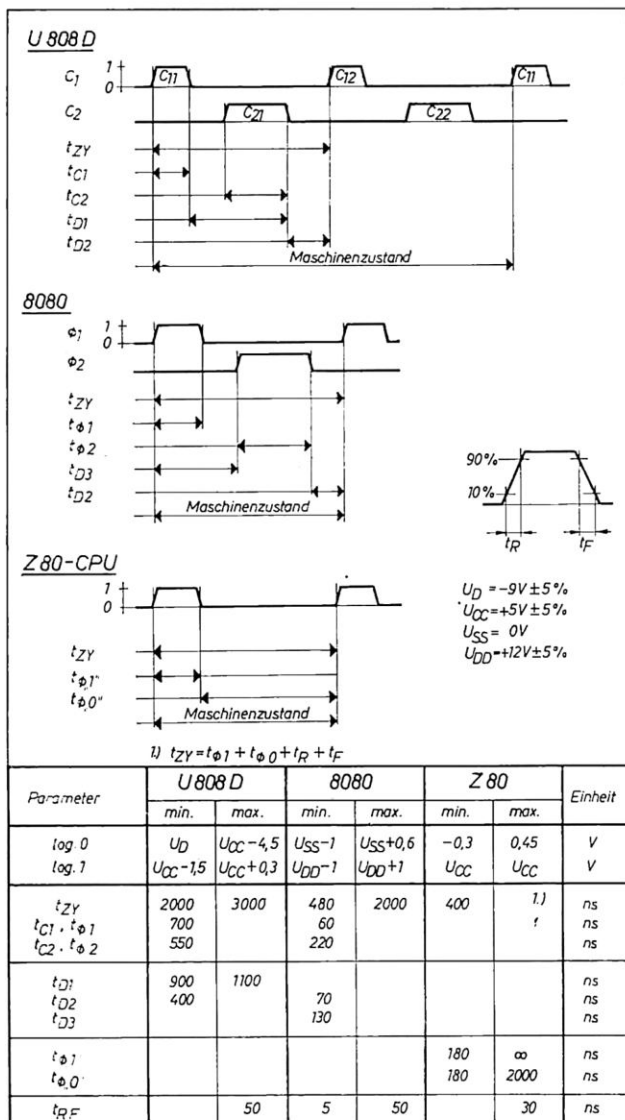


Bild 2.1.5.: Zeitbedingungen für die Taktversorgung

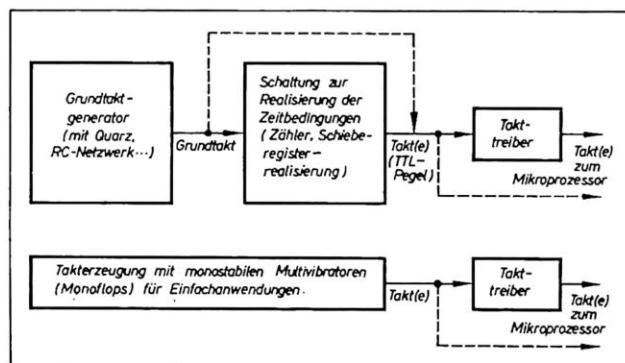


Bild 2.1.6.: Takterzeugung

neben der Ankündigung der Unterbrechung über einen speziellen Steuereingang befehlsmäßig eine Ab sprung-adresse (auch Ab sprung-vektor genannt) angegeben wird. Je nach Prozessortyp erfolgt dies im Befehlscode oder als Direktoperand nach einem entsprechenden Befehl. Beim U 808 D und auch beim 8080 ist diese sogenannte RESTART-Adresse im Befehlscode enthalten. Damit kann nach Annahme der Unterbrechung die Identifizierung der UB-Quelle durch Eingabe eines Ein-Byte-Befehls über den Datenbus erfolgen. Dieser RESTART-Befehl erlaubt beim 8080, maximal acht UB-Ab sprung-routinen zu je 8 Byte Länge zu initiieren.

● Maskierte Unterbrechung

Diese Methode gestattet die programmierte Maskierung von Unterbrechungsanmeldungen. Dabei wird bei

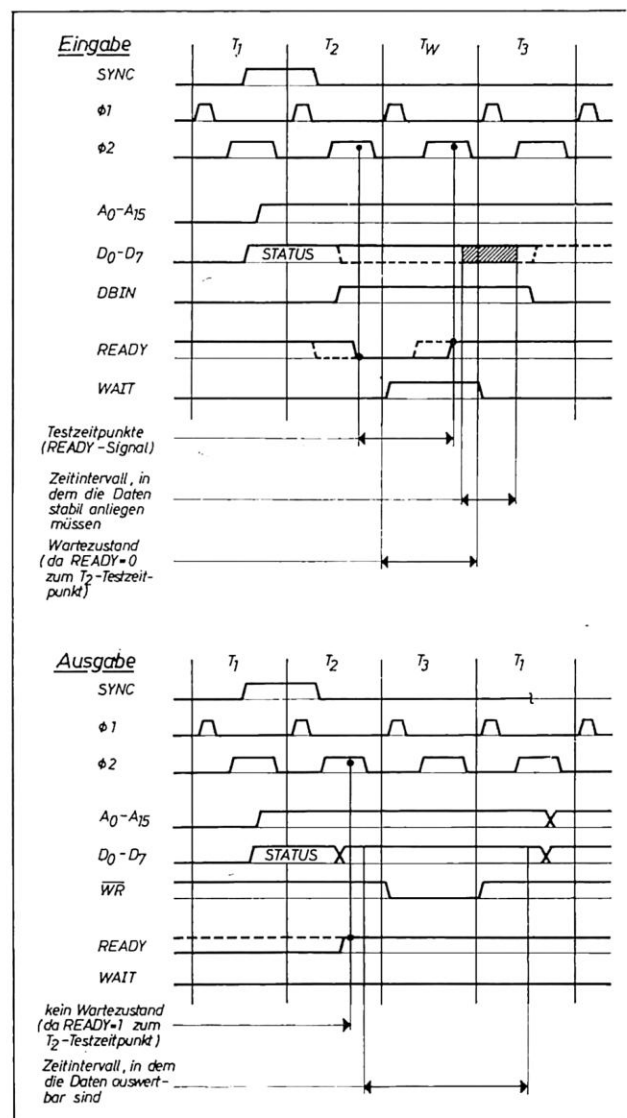


Bild 2.1.7.: Vereinfachte Zeitdiagramme für die Ein- und Ausgabe von Datenbusinformationen. Der erste Pfeil bei „Testzeitpunkte (READY-Signal)“ muß von der anderen Seite angezeichnet werden.

mehreren UB-Anmeldesignalen abgesichert, daß nur eine determiniert ausweisbare Unterbrechung akzeptiert wird.

● Unterbrechung mit Prioritätensteuerung

Diese Methode basiert auf einer mikroprozessorintern realisierten Prioritätensteuerlogik. Da das eine direkte Zuführung der UB-Anmeldungen zum Mikroprozessor erfordert, ergibt sich bei der technologisch bedingten Begrenzung der Pinanzahl je Schaltkreis eine Einschränkung der UB-Kanäle (meist nur 4). Diese Methode wird selten genutzt.

Die untersuchten Mikroprozessoren weisen folgende Unterbrechungsmöglichkeit auf:

U 808 D

Ein Unterbrechungsanmeldeeingang IT, nicht maskierbar

8080

Ein Unterbrechungsanmeldeeingang INT, maskierbar durch das INTE-Flip-Flop, das durch die Befehle EI (Unterbrechungserlaubnis) und DI (Unterbrechungsblockierung) gesetzt bzw. rückgestellt wird (mit vektorieller UB-Adressierbarkeit).

Z 80-CPU

Zwei Unterbrechungsanmeldeeingänge

$\overline{INT}$ maskierbar über die DI-, EI-Befehle, für drei Betriebsarten (Moden) programmierbare UB-Bearbeitung:

Mode 0: 8080-Betriebsart und zusätzliches Einfügen von zwei Warte-Zuständen

Mode 1: RESTART zur Adresse 0038 H (H: Hexadezimalangabe)

Mode 2: indizierter Unterprogrammaufruf

NMI nichtmaskierbar, führt zum unbedingten RESTART zur Adresse 0066H (UB-Service route höchster Priorität).

2.1.2.4. Schaltung in den tristabilen Zustand (TS-Schaltung)

Bezugnehmend auf die im Abschnitt 1.2.4.3. angeführten schaltungstechnischen Grundlagen soll hier die Anwendung dieser Schaltungsmöglichkeit bei Mikroprozessoren behandelt werden.

Durch das Schalten von TS-Ausgängen in den hochohmigen Zustand werden diese im Prinzip vom Bus, an den sie angeschlossen sind, „abgetrennt“. Diese Möglichkeit kann in einfacher Weise zur Realisierung eines multiplexen Betriebes von mehreren Elementen an einem Bus genutzt werden (Bild 2.1.8.)

Die meisten Mikroprozessoren weisen die Möglichkeit auf, ihren Daten- und – wenn vorhanden – ihren Adreßbus unter Steuerungspriorität des Mikroprozessors in den hochohmigen Zustand zu schalten. Damit „trennt“ sich der Mikroprozessor im Prinzip von diesen Bussen, so daß ein anderes Element diese Busleitungen mit Informationen belegen kann. Dieser interne Zustand des Mikroprozessors wird allgemein mit HOLD-Zustand bezeichnet. Oft wird die Synchronisation dieses Zustands durch ein Signalpaar (Anforderung des HOLD-Zustandes; Anerkennung der HOLD-Anforderung durch den Mikroprozessor) realisiert.

Diese Möglichkeit wird insbesondere bei Mehrprozessorbetrieb und bei DMA-Betrieb genutzt. Eine externe Zuweisungssteuerung muß dabei die Zuschaltbedingung realisieren.

Zu beachten ist, daß oftmals die Steuersignale des Mikroprozessors nicht in den hochohmigen Zustand geschaltet werden. Dies muß dann, wenn es erforderlich ist, durch eine Zusatzbeschaltung realisiert werden.

Der Bussteuer-Baustein 8228 für den Mikroprozessor 8080 schaltet z. B. nach Anforderung neben dem Datenbus die Steuersignale zur Synchronisation des Datenbusses (INTA, MEMR, MEMW, I/O R, I/O W) ebenfalls in den hochohmigen Zustand.

Werden Adreß- und Datenbusse durch Treiberschaltkreise beschaltet, so muß die Schaltung in den tristabilen Zustand auf diese übertragen werden.

Literatur

- [2.1.1.] Höhne, M.: Der Mikroprozessor U 808 D. radio fernsehen elektronik 26 (1977) H. 5, S. 145–150; H. 6, S. 187, 188, 197, 198

2.2. Speichereinheit eines Mikrorechners

2.2.1. Organisation des Arbeitsspeichers

In diesem Abschnitt sollen der Aufbau und die Arbeitsweise der gesamten Speichereinheit eines Mikrorechners beschrieben werden. Bild 2.2.1. zeigt das Blockschaltbild des Arbeitsspeichers. Er wird über drei Busse, den bidirektionalen Datenbus, den Adreßbus und den Steuerbus, an den Mikroprozessor (CPU) angeschlossen.

Da Halbleiter-Schreib-Lesespeicher gegenwärtig nur nach dem flüchtigen Speicherprinzip arbeiten können, besteht der Arbeitsspeicher eines Mikrorechners im allgemeinen aus zwei Teilen, dem Festwertspeicher (ROM) und dem Schreib-Lesespeicher (RAM). Als Festwertspeicher können maskenprogrammierte ROM, PROM oder EPROM eingesetzt werden. Als Schreib-Lesespeicher stehen statische oder dynamische RAMs zur Verfügung. Die Kapazität der ROM und RAM wird durch das geplante Einsatzgebiet des Mikrorechners bestimmt. So werden hinreichend ausgetestete Programme für unveränderliche Aufgaben zweckmäßigerweise in für große Stückzahlen billigen maskenprogrammierten ROM untergebracht werden. Dagegen sollte ein Mikrorechner, der für Test- oder Entwicklungszwecke in einem Labor eingesetzt wird, über eine möglichst große Schreib-

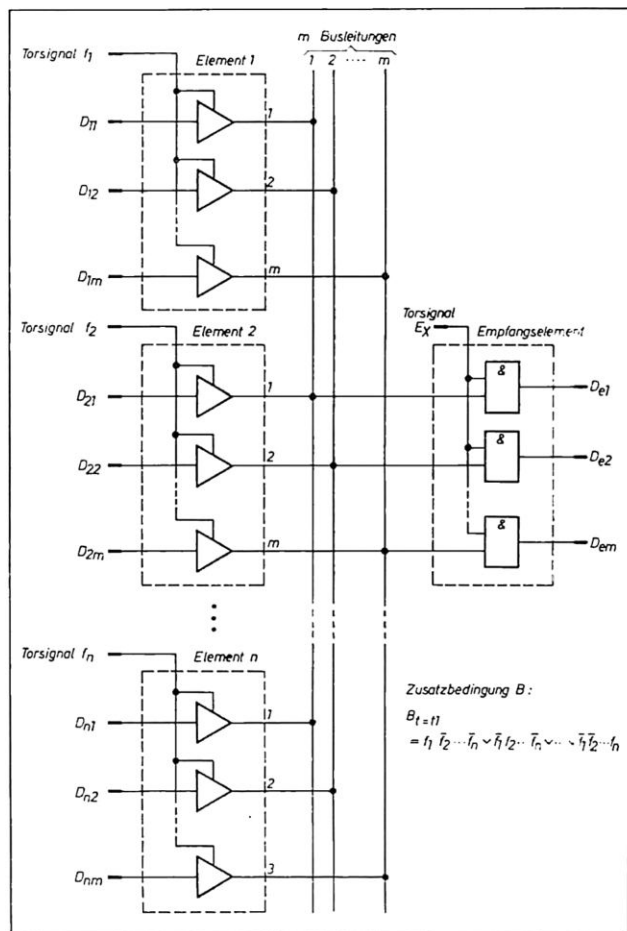


Bild 2.1.8.: Busbeschaltung durch Elemente mit tristabilen Ausgangsstufen

- [2.1.2.] Meiling, W.: Einige Tendenzen in der Entwicklung und Anwendung von Mikroprozessoren und Mikrorechnern. Nachrichtentechnik - Elektronik 27 (1977) H. 1, S. 7–10
 [2.1.3.] Tschelebiev, A.; Schlechter, J.: Mikrorechner – eine Übersicht. radio fernsehen elektronik 24 (1975) H. 8, S. 264–268
 [2.1.4.] Microprocessors – a special issue. Electronics (1976) April 15, S. 74–128
 [2.1.5.] Schoeffler, J.-D.: Microprocessor architecture. IEEE Transactions IECI-22 (1975) H. 3, S. 256–276

Lesespeicherkapazität verfügen und mit einem kleinen lösch- und programmierbaren Festwertspeicher (EPROM) ausgerüstet sein. Oftmals genügt ein aus wenigen Bytes bestehendes Start- und Ladeprogramm zur Arbeitsaufnahme mit dem Mikrorechner. Für diesen Zweck kann man den teuren EPROM durch eine einfache, diskrete Diodenmatrix völlig ausreichend ersetzen. Benötigt man einen schnellen Spei-

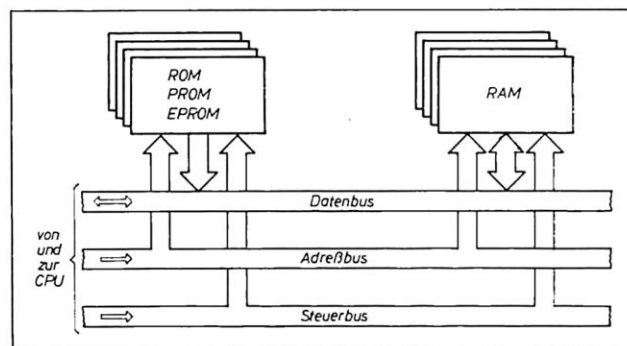


Bild 2.2.1.: Blockschaltbild des Arbeitsspeichers

cher, so muß man eventuell auf bipolare PROM und RAM zurückgreifen.

Grundsätzlich sind beim logischen Entwurf des Arbeitsspeichers vier Parameter zu berücksichtigen:

- Speicherkapazität
- Aufrufbreite
- Zugriffszeit
- Zykluszeit.

Die Speicherkapazität gibt die Anzahl der adressierbaren Speicherplätze der ROM bzw. RAM in (Speicher-)Worten an. Multipliziert man diese Anzahl mit der Länge eines Speicherwortes, so erhält man die Speicherkapazität in Bit (gleichbedeutend mit der Anzahl der Speicherzellen). Beispielsweise bedeutet die Angabe $16\text{ K} \times 8\text{ bit}$, daß der Speicher eine adressierbare Kapazität von 16 K Worten ($\text{K} = 2^{10}$) besitzt, wobei ein Speicherwort 8 bit lang ist. Die maxi-

male Speicherkapazität beträgt $m = 2^n$ Worte, wobei n die Anzahl der Adreßbits der CPU ist.

Die Aufrufbreite des Speichers gibt die Anzahl der in einem Speicherzyklus (Zykluszeit) parallel auslesbaren oder einschreibbaren Bits an. Üblicherweise gilt in der Mikroprozessortechnik zur Vereinfachung des Steueraufwandes die Beziehung: Aufrufbreite $\triangleq$ Länge eines Speicherwortes $\triangleq$ Verarbeitungsbreite der CPU $\triangleq$ Länge eines Datenwortes. Bei der Zugriffszeit eines Halbleiterspeichers ist zwischen Adreßzugriffszeit und Freigabezugriffszeit (Chip-Enable-Zugriffszeit oder Chip-Select-Zugriffszeit) zu unterscheiden. Die Adreßzugriffszeit ist die Verzögerungszeit vom Anlegen der Adresse bis zum Erhalt der gültigen Ausgangsdaten. Die Freigabezugriffszeit ist die Verzögerungszeit vom Anlegen des Freigabesignals (Chip Enable CE bzw. Chip Select CS), das zur Chipauswahl bei Speichererweiterung verwendet wird, bis zum Erscheinen der gültigen Ausgangsdaten.

Die Zykluszeit gibt die Zeit eines vollständigen Lese- oder Schreibzyklus an. Nach Ablauf dieser Zeit darf durch Änderung der Adresse ein neuer Speicherzyklus eingeleitet werden.

Diese drei Zeiten sind beim Entwurf eines Mikrorechners sehr sorgfältig mit dem Taktschema der CPU abzustimmen.

2.2.2. Festwertspeicher

Für einen Mikrorechner soll ein Festwertspeicher mit einer Kapazität von $1\text{ K} \times 8\text{ bit}$ entworfen werden, für den die maskenprogrammierten ROM U 501 D verwendet werden sollen. In der Entwicklungsphase wird der pin-kompatible MOS-EPROM 1702 eingesetzt (s. Abschnitt 1.3.), der eine Kapazität von 2048 bit bei einer Organisation von $256 \times 8\text{ bit}$ hat. Zur Realisierung der Aufgabe sind also vier Schaltkreise erforderlich. Bild 2.2.2. zeigt das Funktionsschaltbild.

Die Adreßleitungen der acht niederwertigsten Adreßbits werden an die Adreßeingänge aller Schaltkreise gelegt. Damit kann jeweils eines der 256 Worte eines Schaltkreises adressiert werden. Mit Anlegen der gewünschten Adresse wird in allen vier Schaltkreisen der Lesevorgang eingeleitet. Damit aber nur ein Datenwort des 1-K-Speichers auf den Datenbus gelangt, sind die Datenausgänge der Schaltkreise durch Puffer verriegelt, die mit Hilfe der Chip-Select-Eingänge ($\overline{\text{CS}}$) gesteuert werden. Diese Eingänge haben dabei die zwei Aufgaben zu erfüllen, den gewünschten Schaltkreis auszuwählen und die Ausgangspuffer der nichtausgewählten Schaltkreise in den hochohmigen Zustand (Three State) zu versetzen, so daß ein „Wired-OR“ aller Datenausgänge auf den Datenleitungen D0 bis D7 möglich ist. Die Steuersignale $\overline{\text{CS0}}$ bis $\overline{\text{CS3}}$ werden dabei durch Dekodierung der höherwertigen Adreßbits gewonnen. Auf diese Weise entsteht eine ein- und ausgangseitig TTL-kompatible Speichereinheit (Belastung mit einer TTL-Last), die mit einer Adreßzugriffszeit von maximal $1,3\text{ }\mu\text{s}$ beim 1702 A und maximal $1\text{ }\mu\text{s}$ beim U 501 D arbeitet. Die Verlustleistung eines Schaltkreises liegt bei etwa 700 mW. Durch Taktung der Betriebsspannung U_{GG} läßt sich der Leistungsverbrauch herabsetzen.

Wird fortgesetzt

◀ Bild 2.2.3.: $(1\text{ K} \times 8\text{ bit})$ -RAM mit Bausteinen CM 8001

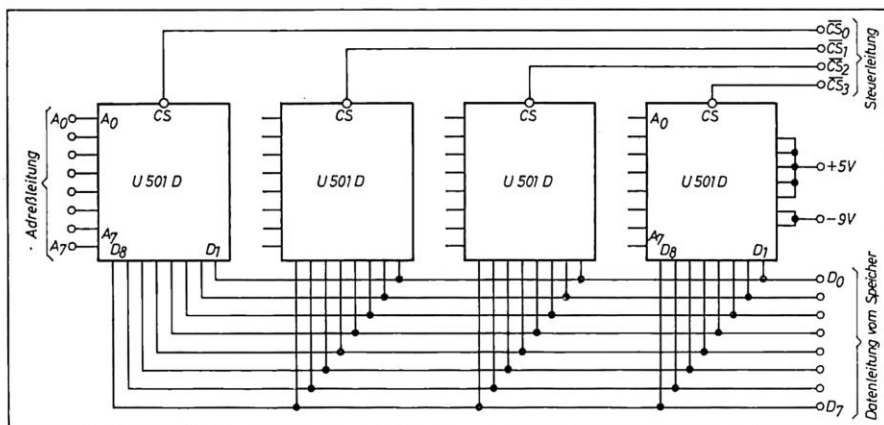
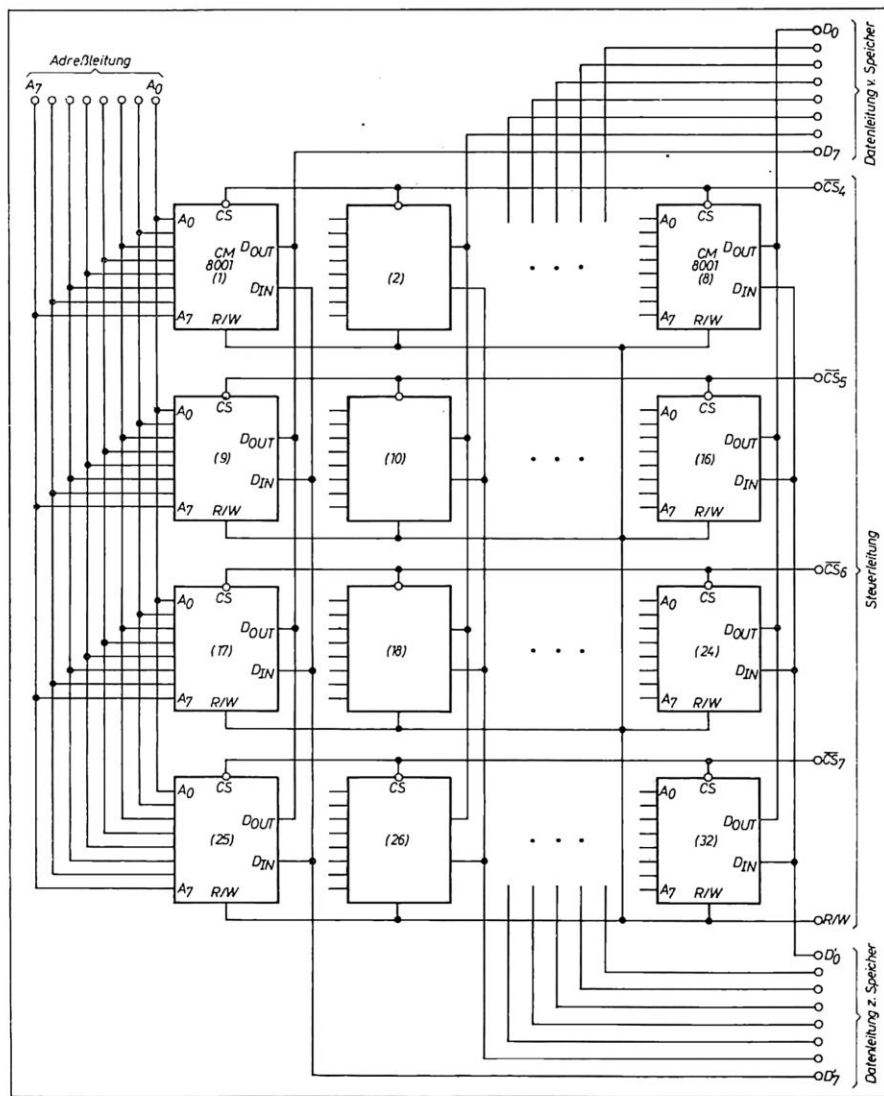


Bild 2.2.2.: Festwertspeicher $1\text{ K} \times 8\text{ bit}$



Dr.-Ing. STEFAN LINDNER

2.2.3. Schreib-Lesespeicher

2.2.3.1. Speicher mit statischen Bausteinen

Als Beispiel soll ein statischer Schreib-Leseverstärker mit einer Kapazität von $1\text{ K} \times 8\text{ bit}$ betrachtet werden. Als Schaltkreise werden der 256-Bit-Speicher CM 8001 und der 1024-Bit-Speicher Intel 8102 verwendet. Der Baustein CM 8001 hat eine Organisation von $256 \times 1\text{ bit}$. Zur Realisierung der Aufgabe werden also 32 Bausteine benötigt. Dagegen werden bei Verwendung des Intel 8102 mit einer Organisation von $1024 \times 1\text{ bit}$ nur acht Bausteine gebraucht. Bild 2.2.3. zeigt die Schaltung bei Verwendung des CM 8001. Zur Realisierung der Wortbreite von 8 bit sind jeweils acht Schaltkreise parallelzuschalten. Vier solcher Gruppen ergeben die Kapazität von 1 K Worten.

Zur Adressierung einer der 256 Speicherzellen eines Schaltkreises werden die acht niederwertigsten Adreßbits des Adreßbusses benötigt. Mit den Steuereingängen $\overline{\text{CS4}}$ bis $\overline{\text{CS7}}$ wird jeweils eine der vier Wortgruppen ausgewählt. Die Bildung dieser Signale erfolgt wiederum aus den höherwertigen Adreßbits. Da es sich um einen Schreib-Lese-Speicher handelt, wird je ein 8 bit breiter Datenbus für Lesedaten (D0 bis D7) und für Schreibdaten (D0' bis D7') benötigt. Die Three-State-Datenausgänge erlauben in Verbindung mit dem Chip-Select-Signal die Wired-OR-Verknüpfung der entsprechenden Schaltkreisausgänge auf dem Datenbus D0 bis D7 .

Mit dem Steuersignal R/W (Read/Write) wird zwischen Lese- oder Schreibzyklus gewählt, deren zeitliche Abläufe im Abschnitt 1.3. beschrieben wurden. Der so aufgebaute Speicher ist ein- und ausgangsseitig TTL-kompatibel (Belastung mit einer TTL-Last). Die Adreßzugriffszeit beim Lesen beträgt maximal $1,5\text{ }\mu\text{s}$, der Schreibzyklus dauert minimal $0,9\text{ }\mu\text{s}$. Die Betriebsverlustleistung eines Schaltkreises liegt bei $1,5\text{ mW/bit}$. Im Ruhezustand sinkt die Verlustleistung auf etwa $0,12\text{ mW/bit}$ ab.

Bei Verwendung des 1-K-Bit-Speichers 8102 (Bild 2.2.4.) sinkt der Bausteinanfang auf ein Viertel gegenüber der ersten Variante ab. Zur Adressierung werden jetzt zehn Adreßleitungen A0 bis A9 benötigt. Dadurch vereinfacht sich

die Bildung der Steuersignale. Nur bei Speichererweiterung muß das Auswahlsignal $\overline{\text{CE}}$ (Chip Enable, Chip-Freigabe) aus den höherwertigen Adreßbits gebildet werden. Ansonsten wird $\overline{\text{CE}} = \text{L}$ gewählt. Der Speicher ist TTL-kompatibel (Belastung mit einer TTL-Last).

Bei Verwendung des Baustein 8102 A-4 erhält man maximale Lesezugriffszeiten von 450 ns und Schreibzykluszeiten von minimal 450 ns . Die typische Verlustleistung liegt bei nur 150 mW für einen Schaltkreis. Außerdem wird nur eine einzige $+5\text{-V}$ -Versorgungsspannung benötigt.

2.2.3.2. Speicher mit dynamischen Bausteinen

Dynamische Speicherschaltkreise sind durch größere Speicherkapazität, höhere Arbeitsgeschwindigkeit und sehr niedrige Ruheverlustleistung gegenüber den statischen Speichern gekennzeichnet. Diese Vorteile werden durch den Nachteil der Notwendigkeit des periodischen Auffrischens (Refreshing) der gespeicherten Information erkauft. Mit dynamischen Schaltkreisen werden insbesondere größere Speichereinheiten aufgebaut.

Als Beispiel soll ein $(1\text{ K} \times 8\text{ Bit})$ -Schreib-Lesespeicher mit den Schaltkreisen U 253 betrachtet werden.

Bild 2.2.5. zeigt die Schaltung der Speichereinheit. Entsprechend der Wortbreite von 8 bit werden acht U 253 parallelgeschaltet. Der Speicheraufbau entspricht damit dem des statischen $(1\text{ K} \times 8\text{ bit})$ -Speichers nach Bild 2.2.4. Der Datenausgang $\overline{\text{DOUT}}$ des U 253 ist ein Open-Drain-Ausgang, der eine Parallelschaltung der Datenausgänge bei Speichererweiterung gestattet. Je Datenausgang ist damit ein Pull-up-Widerstand ($100\text{ }\Omega \dots 1\text{ k}\Omega$) gegen U_{DD} notwendig. Um eine TTL-kompatible Speichereinheit zu erhalten, sind Zusatzschaltkreise notwendig. Für alle Eingänge sind Treiberbausteine (z. B. 75 361 A) vorzusehen, die die notwendigen MOS-Pegel (Spannungshub $\approx 16\text{ V}$) liefern. Dabei ist gegebenenfalls die negierende Wirkung der Treiber zu berücksichtigen. Da die Datenausgänge einen Ausgangs-H-Pegel von nur etwa 50 mV liefern, sind Leseverstärker (z. B. D 122) nachzuschalten.

Zur Auffrischung der dynamischen Speicherschaltkreise ist eine Auffrischsteuereinheit notwendig. Für den U 253 wird gefordert, daß alle 32 Zeilen der Speichermatrix (Adreßbits A0 bis A4) mindestens einmal in 2 ms gelesen werden. Das Prinzipschaltbild der Auffrischsteuereinheit zeigt Bild 2.2.6. Solange die CPU keinen Speicherzyklus benötigt ($\text{MEMR} \vee \text{MEMW} \vee \overline{\text{CS}} = \text{H}$), läuft der Auffrischzyklus ungestört ab. Die Zeilenadresse wird von einem fünfstelligen Dualzähler (0 bis 31) geliefert, der über den Adreßmultiplexer an die Adreßleitungen A0 bis A4 der Speichereinheit angeschlossen ist. Die Belegung der Spaltenadreßbits A5 bis A9 ist für den Auffrischvorgang uninteressant. Der

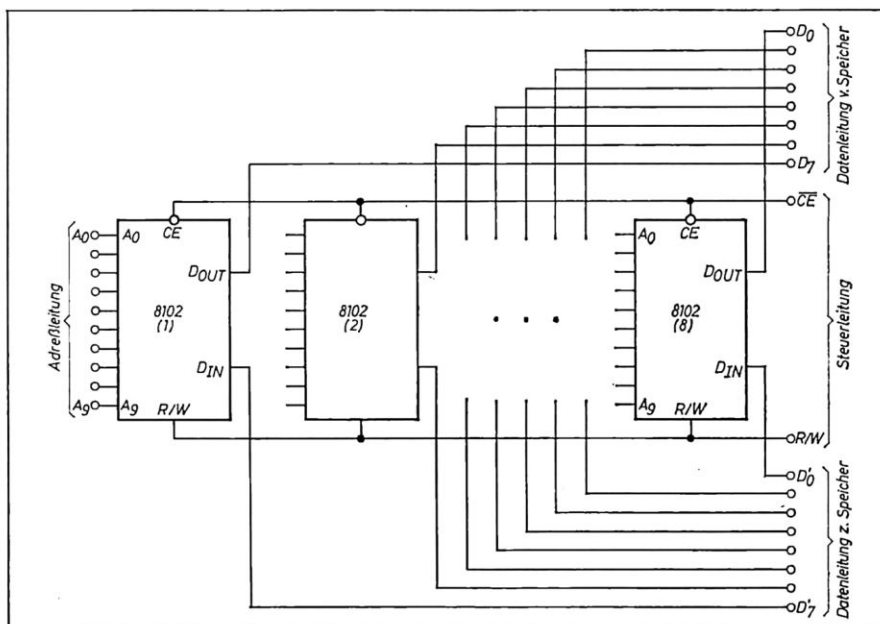


Bild 2.2.4.: $(1\text{ K} \times 8\text{ Bit})$ -RAM mit Bausteinen 8102

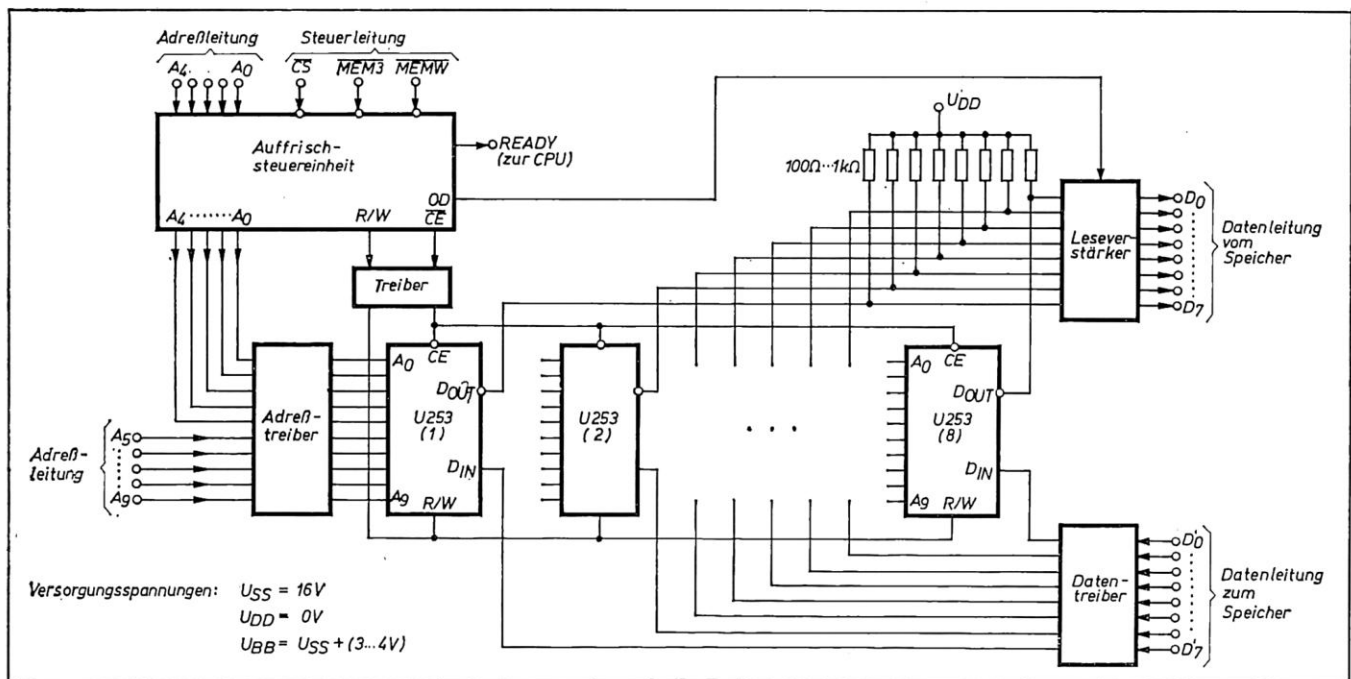


Bild 2.2.5.: (1 k × 8 Bit-RAM (dynamisch) mit Bausteinen U 253 U (lies MEMR statt MEM3)

Zähler wird von der Taktzentrale ($f = 16 \text{ kHz}$) weitergestellt. Im gleichen Rhythmus liefert die Steuerlogik auf der $\overline{CE}$ -Leitung Impulse, die mit $R/W = H$ alle $62,5 \mu s$ einen Lesezyklus auslösen. Mit dem Ausgang OD (Output Disable) werden die Leseverstärkerausgänge blockiert, so daß die LeseDaten beim Auffrischen nicht auf den Datenbus der CPU gelangen können. Mit dem Signal $\overline{CS} = L$, das aus den höherwertigen Adreßbits dekodiert wird, wählt die CPU die Speichereinheiten aus.

Ein Lesezyklus wird durch $\overline{MEMR} = L$, ein Schreibzyklus durch $\overline{MEMW} = L$ aktiviert. Läuft bei CPU-Speicheranforderung $\overline{MEMR} \cdot \overline{MEMW} \vee \overline{CS} = L$ gerade ein Auffrischzyklus, wird die CPU durch $READY = L$ in den Wartezustand versetzt. Nach Beendigung des Auffrischzyklus wird der Zähler gestoppt, und der Adreßmultiplexer wird auf die Adreßleitungen der CPU geschaltet. Danach wird mit den Steuer-

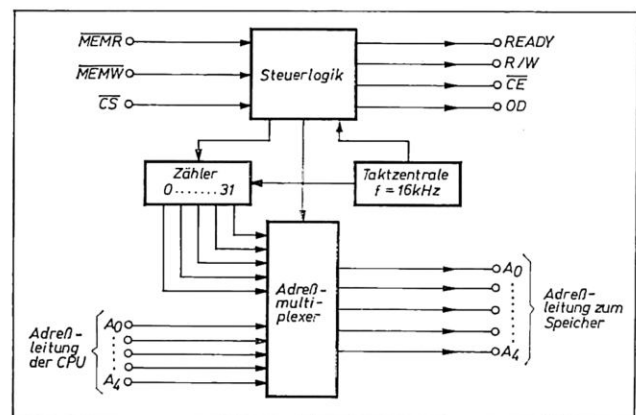
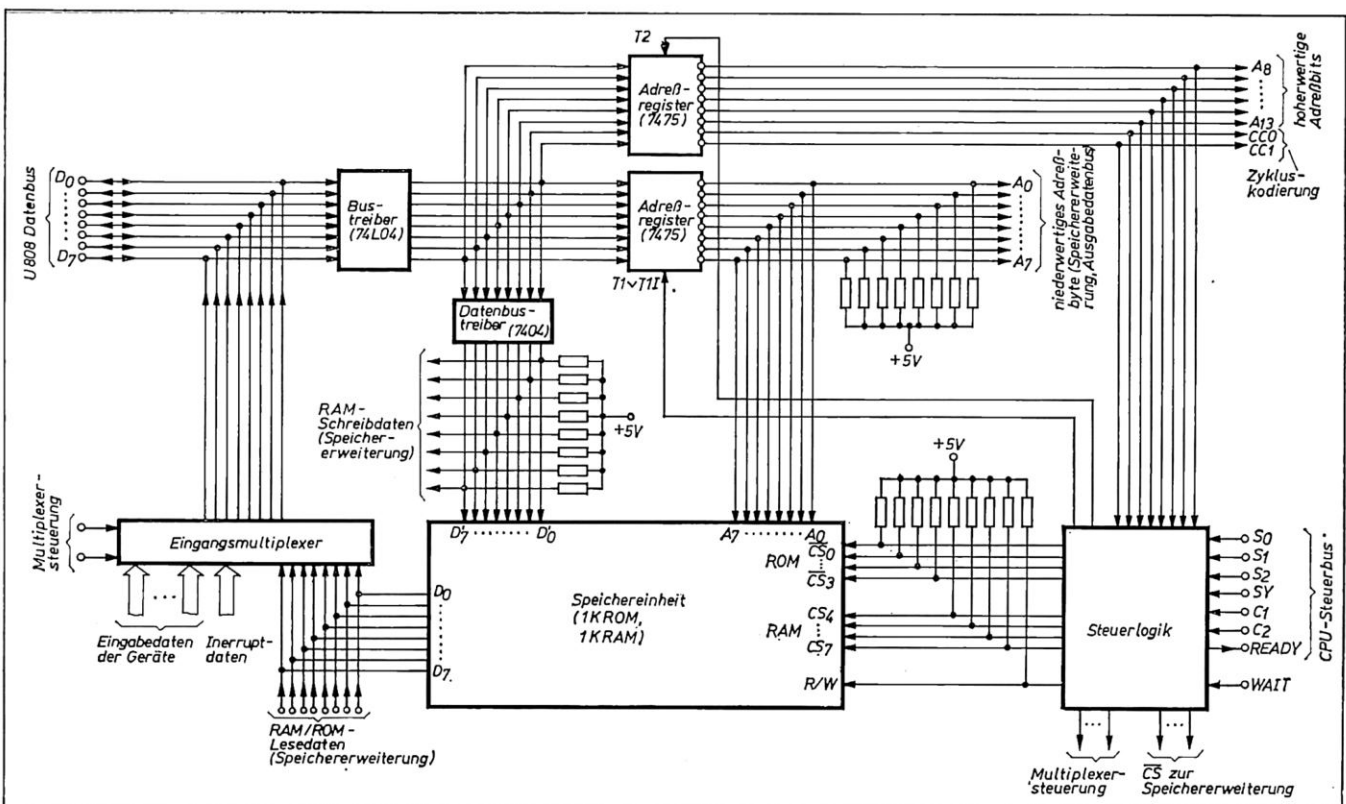


Bild 2.2.6.: Prinzipschaltbild der Auffrischsteuereinheit

Bild 2.2.7.: Anschluß der Speichereinheit an den U 808 D (lies $\overline{CS}_4$ bis $\overline{CS}_7$ statt $\overline{CS}_4$ bis $\overline{CS}_7$)



signalen $\overline{CE}$ und R/W der gewünschte Lese- oder Schreibzyklus ausgeführt.

Bei einem Schreibzyklus (Belegung der Datenleitung zum Speicher) bleiben über das Steuersignal OD die Leseverstärkerausgänge gesperrt. Dadurch können die Datenleitungen D0 bis D7 und D0' bis D7' direkt zu einem bidirektionalen Datenbus zusammengeschaltet werden. Nach Beendigung dieses Speicherzyklus wird das Auffrischen mit dem erreichten Zählerstand fortgesetzt.

Bei der schaltungstechnischen Realisierung der Auffrischsteuereinheit sind die Taktschemata der CPU und der Speicherschaltkreise zu berücksichtigen. Die so aufgebaute Speichereinheit ist voll TTL-kompatibel. Ein Lesezyklus dauert mindestens 480 ns, ein Schreibzyklus mindestens 580 ns. Die Verlustleistung eines Bausteins bei minimaler Zykluszeit beträgt maximal 400 mW. Die Ruheverlustleistung liegt unter 2 mW. Die notwendigen Zusatzschaltkreise machen den Einsatz erst bei größeren Speichern ökonomisch interessant.

2.2.4. Zusammenarbeit des Speichers mit der CPU

An die CPU eines Mikrorechners soll eine Speichereinheit angeschlossen werden, die aus einem ($1\text{ K} \times 8\text{ bit}$)-ROM-Bereich nach Bild 2.2.2. und einem ($1\text{ K} \times 8\text{ bit}$)-RAM-Bereich nach Bild 2.2.3. besteht. Dem Festwertspeicherbereich werden die Adressen 0 bis 1023 und dem Schreib-Lesespeicherbereich die Adressen 1024 bis 2047 zugeordnet.

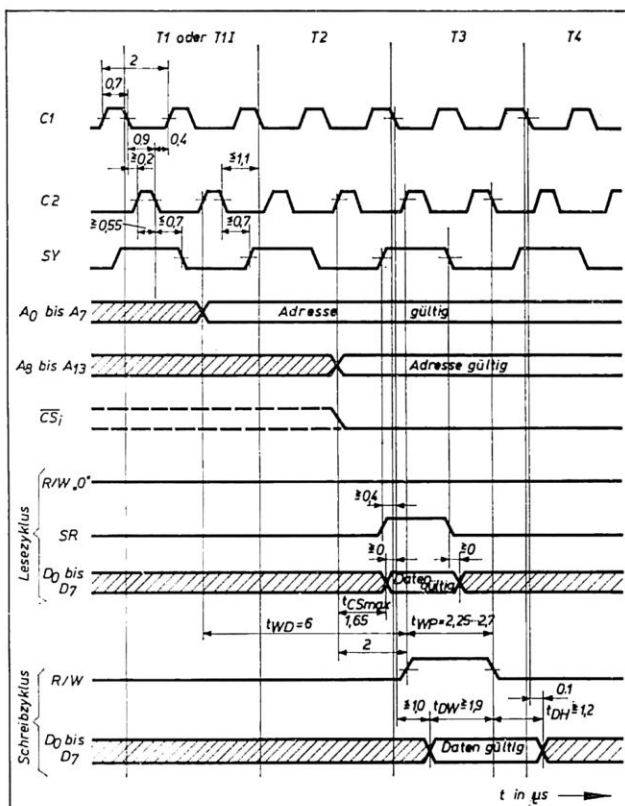
Eine Speichererweiterung soll möglich sein.

Im folgenden werden die Probleme des Speicheranschlusses an das CPU-Bussystem, der Adressierung des Speichers, der Bildung der notwendigen Steuersignale und des zeitlichen Zusammenspiels zwischen CPU und Speichereinheit für den Anschluß des Speichers an den U 808 D, den 8080 A und die Z 80-CPU betrachtet.

2.2.4.1. Anschluß der Speichereinheit an die CPU U 808 D

Zum Anschluß der Speichereinheit muß die CPU einen Adreßbus, einen bidirektionalen Datenbus und einen Steuerbus bereitstellen. Da der U 808 nur einen bidirektionalen Bus D0 bis D7 besitzt, über den sowohl die achtstelligen Datenwörter als auch die vierzehnstelligen Adressen übertragen werden müssen, ist zum Aufbau des Bussystems eine umfangreiche Zusatzlogik erforderlich (Bild 2.2.7.). Der Datenbus des U 808 (Belastung mit $1/4$ TTL-Last möglich) wird

Bild 2.2.8.: Taktschema für Speicheroperationen beim U 808 D



über einen Bustreiber (z. B. 74L04) aufgefächert. Über einen Datenbustreiber (z. B. 7404 oder K 155 LH 1) werden die Schreibdatenleitungen des RAM-Bereiches belegt. Die Lese-daten vom RAM bzw. vom ROM gelangen über einen Eingangsmultiplexer, über den auch der Eingabedaten- und der Interruptdatenbus geführt wird, auf den bidirektionalen Datenbus des U 808. Da die 14stellige Adresse byteweise übertragen wird, sind Adreßregister (z. B. Bausteine 7475) zur Zwischenspeicherung vorzusehen. Das niederwertige Adreßbyte wird direkt an die Adreßeingänge A0 bis A7 des Speichers geschaltet. Das höherwertige Adreßbyte, das aus den sechs Adreßbits A8 bis A13 und zwei Bits CC0, CC1 zur CPU-Zykluskodierung besteht, gelangt zu einer externen Steuerlogik, die u. a. die Steuersignale $\overline{CS0}$ bis $\overline{CS7}$ und R/W bildet.

Obwohl die Speichereinheit TTL-kompatibel ist, sind alle Eingangsleitungen mit einem Pull-up-Widerstand ($1\text{ k}\Omega$) gegen $+5\text{ V}$ zu beschalten, da für diese Speicherschaltkreise ein Eingangs-H-Pegel $\geq U_{CC} - 2\text{ V}$ ($U_{CC} = 5\text{ V} \pm 5\%$) gefordert wird, TTL-Schaltkreise aber nur einen Ausgangs-H-Pegel von $2,4\text{ V}$ garantieren. Entsprechend dem CPU-Taktschema und Befehlsabarbeitungszyklus (s. Abschn. 2.1.) läuft eine Speicheroperation wie folgt ab (das Taktschema für minimale CPU-Zykluszeit zeigt dazu Bild 2.2.8.):

● Adressenbereitstellung

Im Zustand T1 bzw. T11 werden die niederwertigen acht Adreßbits auf den Datenbus ausgesendet. Die Steuerlogik bildet dazu das Steuersignal

$$(T1 \vee T11) \overline{SY} C2 \quad (2.2.1.)$$

mit

$$T1 = \overline{S0} S1 \overline{S2} \quad (2.2.2.)$$

und

$$T11 = \overline{S0} S1 S2 \quad (2.2.3.)$$

das mit $\overline{SY} C2$ verzögert ist und das das Einschreiben in das Adreßregister für das niederwertige Byte bewirkt.

Im Zustand T2 werden die höherwertigen Adreßbits und die Zykluskodierung CC0, CC1 auf den Datenbus ausgesendet. Mit dem Steuersignal

$$T2 \overline{SY} C2 \quad (2.2.4.)$$

mit

$$T2 = \overline{S0} \overline{S1} S2 \quad (2.2.5.)$$

wird das Einschreiben in das Adreßregister für das höherwertige Adreßbyte veranlaßt.

Damit liegen die niederwertigen Adreßbits A0 bis A7 an der Speichereinheit an, und die höherwertigen Adreßbits können von der Steuerlogik dekodiert werden. Die Adreßvoreinstellzeit vor dem Schreibimpuls t_{WD} beträgt damit $\approx 6\text{ }\mu\text{s}$.

● Bildung der Chip-Select-Signale

Die Chip-Auswahlsignale $\overline{CS0}$ bis $\overline{CS7}$ werden aus den Adreßbits A8 bis A13 nach folgenden logischen Gleichungen dekodiert:

$$\overline{CS0} = \overline{A13} A12 A11 A10 A9 A8 \quad (2.2.6.)$$

$$\overline{CS1} = A13 A12 A11 A10 A9 A8 \quad (2.2.7.)$$

$$\overline{CS2} = \overline{A13} A12 A11 A10 A9 A8 \quad (2.2.8.)$$

$$\overline{CS3} = A13 A12 A11 A10 A9 A8 \quad (2.2.9.)$$

$$\overline{CS4} = \overline{A13} A12 A11 A10 A9 A8 \quad (2.2.10.)$$

$$\overline{CS5} = A13 A12 A11 A10 A9 A8 \quad (2.2.11.)$$

$$\overline{CS6} = \overline{A13} A12 A11 A10 A9 A8 \quad (2.2.12.)$$

$$\overline{CS7} = A13 A12 A11 A10 A9 A8 \quad (2.2.13.)$$

Die Auswahlsignale bei Speichererweiterung sind entsprechend zu bilden. Damit ist das gewünschte Speicherwort adressiert. Das Chip-Select-Signal erscheint dabei $\approx 2\text{ }\mu\text{s}$ vor dem R/W-Signal.

● Bildung des R/W-Signals

Das R/W-Signal entscheidet darüber, ob ein Lese- oder Schreibzyklus ausgeführt wird. R/W = L bedeutet Lesen, R/W = H bedeutet Schreiben bei den Bausteinen CM 8001.

Das R/W-Signal wird in einem JK-Flip-Flop (D 172) gespeichert ($R/W = Q$), das sich normalerweise im Ruhezustand $Q = L$ befindet. Es wird über den statischen Setzeingang $\bar{S}$ mit

$$\bar{S} = \overline{CC1 \ CC0 \ T3 \ SY \ C2} \quad (2.2.14.)$$

$$T3 = S0 \ \bar{S1} \ \bar{S2} \quad (2.2.15.)$$

gesetzt. Daraus folgt $Q = R/W = H$, wenn die Zykuskodierung $CC1 \ CC0 = H$ ist und damit einen Schreibzyklus anzeigt. Die Bildung dieses Signals erfolgt im Zustand T3 (Holen von Befehlen oder Daten oder Aussenden von Daten) mit dem Zeitsignal $SY \ C2$. Das Rücksetzen des Flip-Flops erfolgt mit $J = L$, $K = H$ durch den Takt

$$C = \bar{SY} \ C2 \quad (2.2.16.)$$

auf der H/L-Flanke. Der Schreibimpuls hat damit eine Dauer t_{WP} von mindestens $2,55 \mu s$. Bei einem Lesezyklus bleibt $R/W = L$ konstant.

● Datenbereitstellung beim Schreibzyklus

Die Aussendung des Schreibdatenwortes erfolgt im Zustand T3 max. $1 \mu s$ nach der H/L-Flanke des ersten Impulses von C1. Damit werden eine Datenvoreinstellzeit $t_{DV} \geq 1,9 \mu s$ und eine Datenhaltezeit $t_{DH} \geq 1,2 \mu s$ erreicht.

● Datenübernahme beim Lesezyklus

Der Lesezyklus wird durch $\bar{CS}_i = 0$ ausgelöst. Nach der Zugriffszeit $t_{CS} \leq 0,4 \mu s$ (beim CM 8001) bzw. $t_{CO} \leq 0,9 \mu s$ (beim 1702) stehen die Daten auf den Datenausgangsleitungen D0 bis D7 des Speichers zur Verfügung. Über den Eingangsmultiplexer werden die Lesedaten auf den CPU-Datenbus geschaltet, so daß sie im Zustand T3 in die CPU übernommen werden können. Aus dem Taktschema ergibt sich eine maximale Chip-Select-Zugriffszeit $t_{CS} (t_{CO})$ von $\approx 1,65 \mu s$ (abzüglich der Zeit zur Dekodierung der $\bar{CS}_i$).

● Eingangsmultiplexer

Der Eingangsmultiplexer ist durchzuschalten, wenn Speicherdaten, Eingabedaten oder Interruptdaten in die CPU eingegeben werden sollen. Der CPU-Datenbus wird dann im Eingabemodus betrieben, indem die Ausgangsstufen in den hochohmigen Zustand geschaltet werden (s. auch Bild 1.2.28.). Stehen keine Multiplexer mit tristabilen Ausgängen zur Verfügung, kann man eine Schaltung nach Bild 2.2.9. verwenden. Mit einem Baustein D 154 je Datenleitung D_i läßt sich ein 4-auf-1-Multiplexer realisieren, der vier verschiedene Datensignale auswählen kann. Durch einfache Kaskadenschaltung läßt sich die Anzahl der Eingänge weiter erhöhen.

Mit den Steuersignalen SS1 bis SS4 wird das gewünschte Datenwort ausgewählt. Sie werden wie folgt gebildet:

$$SS1 = \bar{T1}' \ \bar{CC0} \ T3A \quad (2.2.17.)$$

$$SS2 = T1' \ \bar{CC0} \ T3A \quad (2.2.18.)$$

$$SS3 = \bar{A13} \ \bar{A12} \ \bar{A11} \ \bar{A10} \ \bar{A9} \ \bar{CC1} \ \bar{CC0} \ T3A \quad (2.2.19.)$$

$$SS4 = \bar{A13} \ \bar{A12} \ \bar{A11} \ \bar{A10} \ \bar{A9} \ \bar{CC1} \ \bar{CC0} \ T3A \quad (2.2.20.)$$

$$T3A = SY \ Q_D \quad (2.2.21.)$$

$$D = H \quad (2.2.22.)$$

$$C = \bar{T2} \ C2 \ \bar{SY} \quad (2.2.23.)$$

$$\bar{R} = \bar{T3} \ \bar{SY} \quad (2.2.24.)$$

Die Datensignale D_i vom Speicher werden mit SS1 durchgeschaltet, wenn ein Lesezyklus PCI oder PCR ($CC0 = L$) vorliegt und nicht der UB-Erkennungszustand T11 durchlaufen wurde. Falls ein UB-Signal erkannt und damit T11 durchlaufen wurde, werden mit SS2 die Interruptdaten I_i auf den CPU-Bus geschaltet. Die Signale $T11'$ bis $\bar{T11}'$ werden einem Flip-Flop entnommen, das in jedem Zyklus mit dem Taktsignal nach Gl. (2.2.1.) durch $T11 = \bar{S0} \ S1 \ S2$ gesetzt bzw. rückgesetzt ($\bar{S0} \ S1 \ S2 = L$) wird. Die Steuersignale SS3 und SS4 werden im PCC-Zyklus ($\bar{CC1} \ \bar{CC0} = H$) bei einer Eingabeoperation ($\bar{A13} \ \bar{A12} = H$) gebildet. Mit A9 bis A11 werden dabei die Eingabetore kodiert (im Beispiel E-Tor 0 und E-Tor 1). Alle Steuersignale werden durch das Zeitsignal T3A so in das CPU-Taktschema (s. Bild 2.2.8.) eingeordnet, daß die Daten zur Übernahme in die CPU zum vorgesehenen

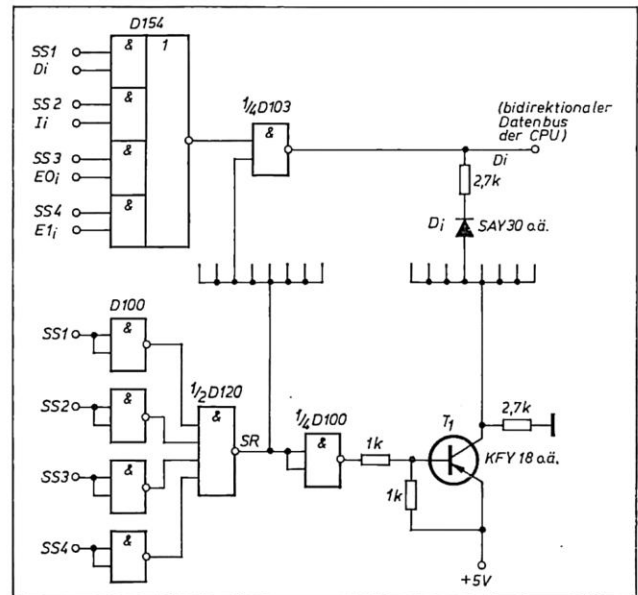


Bild 2.2.9.: Eingangsmultiplexer (für eine Binärstelle)

Zeitpunkt bereitstehen. T3A wird aus der UND-Verknüpfung von SY und dem Ausgang Q_D eines D-Flip-Flops gebildet, das im Zustand T2 gesetzt und im Zustand T3 zurückgesetzt wird.

Der gesamte Multiplexer besteht dann aus acht D 154 mit jeweils einem nachgeschalteten D-103-Gatter. Die ausgangsseitige Entriegelung des Multiplexers erfolgt mit $SR = H$, sobald eines der Steuersignale SS1 bis SS4 auftritt, wobei

$$SR = SS1 \vee SS2 \vee SS3 \vee SS4 \quad (2.2.25.)$$

Neben der Öffnung des D-103-Gatters erfolgt mit $SR = H$ die Anschaltung eines Pull-up-Widerstandes von $2,7 k\Omega$ an jede Datenleitung des bidirektionalen Datenbusses, indem der Transistor T_1 durchgeschaltet wird. Auf diese Weise wird eine Eingangsspannung $U_{eH} \geq U_{CC} - 1,5 V$ für den U 808 erreicht. Mit $SR = L$ wird der Multiplexer gesperrt, und die Pull-up-Widerstände werden abgeschaltet, damit sie den Datenbus bei Datenausgabe nicht zusätzlich belasten. Die Dioden D_i dienen der gegenseitigen Entkopplung der Datenleitungen des Datenbusses.

● WAIT

Aus Bild 2.2.8. lassen sich die wichtigsten Zeitforderungen für die Speichereinheit bei minimaler CPU-Zykluszeit ablesen:

Chip-Select-Zugriffszeit	$t_{CS}(t_{CO}) \leq 1,65 \mu s$
Adreßvoreinstellzeit vor Schreibimpuls	$t_{WD}(t_{AW}) \leq 6 \mu s$
Chip-Select-Voreinstellzeit vor Schreibimpuls	$t_{CW} \leq 4,55 \mu s$
Schreibimpulsbreite	$t_{WP} \leq 2,55 \mu s$
Datenvoreinstellzeit	$t_{DW} \leq 1,9 \mu s$
Datenhaltezeit	$t_{DH} \leq 1,2 \mu s$

Dabei wurden die Verzögerungszeiten, die durch Treiber, Register, Dekodierschaltungen u. a. auftreten, vernachlässigt.

Die bekannten bipolaren und MOS-Speicherschaltkreise genügen diesen Zeitforderungen. Soll ein langsamerer Speicher angeschlossen werden, muß die Speichereinheit den WAIT-Eingang der CPU-Steuerlogik aktivieren, die ihrerseits den U 808 mit $READY = L$ in den Wartezustand T3W zwingt, solange das WAIT-Signal vorliegt. Zur Bildung von WAIT bzw. READY muß das CPU-Taktdiagramm berücksichtigt werden.

Wird fortgesetzt

Dr.-Ing. STEFAN LINDNER,
Dr.-Ing. RUDOLF KÜBART und
Dipl.-Ing. JÜRGEN SCHLECHTER

2.2.4.2. Anschluß der Speichereinheit an die CPU 8080 A

Durch die Trennung von Daten- und Adreßbus beim 8080 A ergibt sich eine wesentliche Vereinfachung im Datenaustausch mit der Speichereinheit und der Peripherie. Mit Hilfe angepaßter Puffer- und Treiberbausteine mit tristabilen Ausgängen (8218, 8216) läßt sich der Aufwand weiter reduzieren. Bild 2.2.10. zeigt den Anschluß der Speichereinheit an die CPU 8080 A.

Die Adreßleitungen A0 bis A15 werden über Adreßtreiber (8212, 8216) verstärkt. Die niederwertigen Adreßbits A0 bis A7 werden der Speichereinheit zugeführt. Die höherwertigen Adreßbits A8 bis A15 dienen analog den Gleichungen (2.2.6.) bis (2.2.13.) der Chip-Select-Dekodierung. Die Schreibdaten werden über einen Ausgangstreiber (8212) dem Speicher vom CPU-Datenbus zugeführt. Die Speicherlesedaten werden gemeinsam mit den Eingabedaten der peripheren Geräte und den Interruptdaten einem Eingangsmultiplexer zugeführt. Die Ankopplung an den bidirektionalen CPU-Datenbus erfolgt über den Three-State-Puffer 8212, der mit dem CPU-Steuersignal DBIN am Auswahlgang DS2 geöffnet wird, wenn die CPU zur Datenübernahme bereit ist. Auf Grund des minimalen Ausgangs-H-Pegels von 3,65 V der Bausteine 8212 bzw. 8216 können die Pull-up-Widerstände nach Bild 2.2.8. entfallen.

Die externe Steuerlogik hat die Aufgabe, aus den CPU-Status- und -Steuersignalen (s. Abschnitt 2.1.) die Speicher-, E/A- und UB-Steuersignale sowie die Multiplexerauswahlsignale zu bilden:

Lese-Schreibsignal $R/W = \overline{WR} \text{ OUT}$ (2.2.26.)

Multiplexerauswahl
für Speicherdaten $SS1 = \text{MEMR}$ (2.2.27.)

TECHNIK DER MIKRORECHNER

12

für Interruptdaten $SS2 = \text{INTA}$ (2.2.28.)

für Eingabedaten $SSi = \text{INP} \cdot (\text{Geräteadresse } i)$ (2.2.29.)

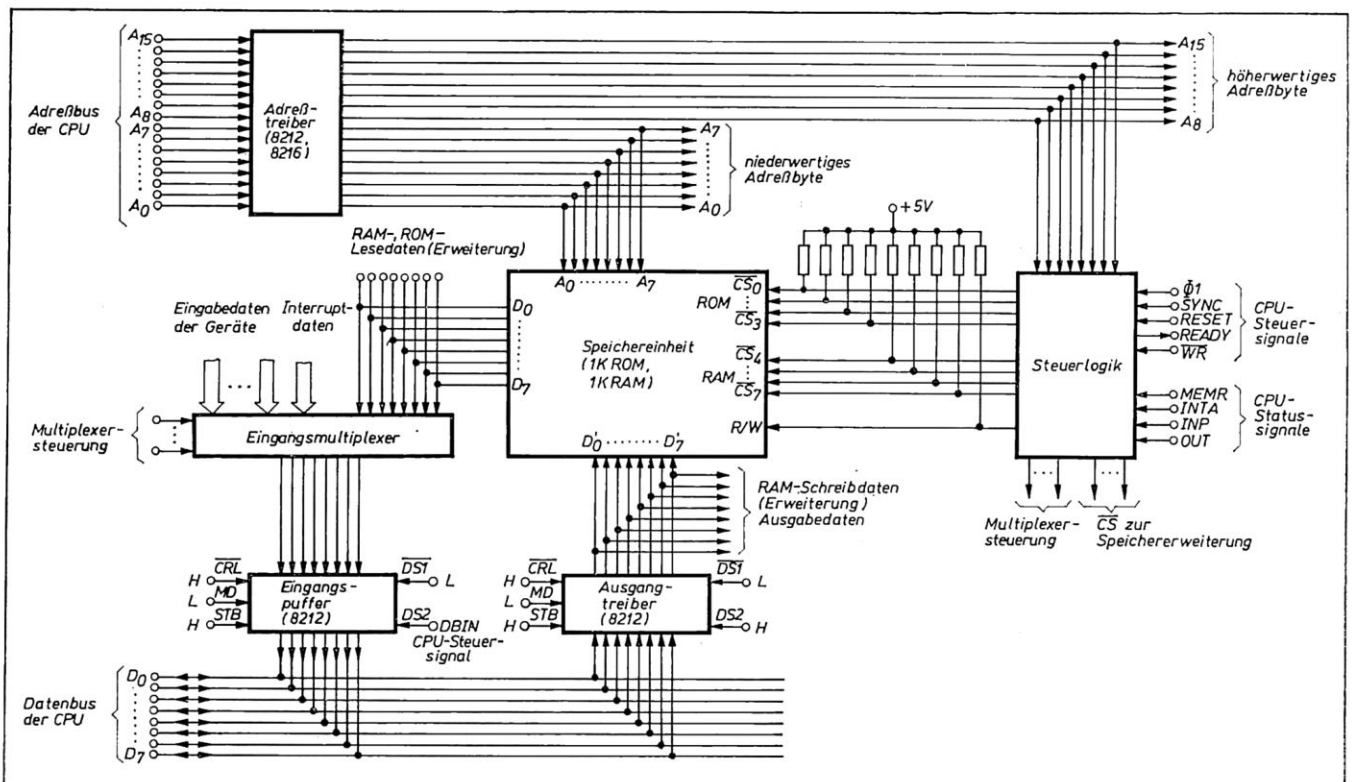
Der dem Multiplexer nachgeschaltete Eingangspuffer sorgt durch die Steuerung mit DBIN für die richtige zeitliche Bereitstellung der einzugebenden Daten.

Bild 2.2.11. zeigt das Taktschema für eine Speicheroperation. Daraus lassen sich die wichtigsten Zeitforderungen für die Speichereinheit bei einer CPU-Zykluszeit von 500 ns ablesen:

Adreßzugriffszeit	$t_{\text{ACC}}(t_A)$	$\leq$	600 ns
Chip-Select-Zugriffszeit	$t_{\text{CS}}(t_{C0})$	$\leq$	550 ns
Adreßvoreinstellzeit vor Schreibimpuls	$t_{\text{WD}}(t_{\text{AW}})$	$\leq$	650 ns
Chip-Select-Voreinstellzeit vor Schreibimpuls	t_{CW}	$\leq$	1 100 ns
Schreibimpulsbreite	t_{WP}	$\geq$	500 ns
Datenvoreinstellzeit	t_{DW}	$\geq$	620 ns
Datenhaltezeit	t_{DH}	$\geq$	100 ns

Die angenommene Speichereinheit mit den Bausteinen 1702 A und CM 8001 erfüllt die Zeitforderungen bezüglich der Zugriffszeit nicht. Aus diesem Grunde müssen zwischen den CPU-Zuständen T2 und T3 mindestens zwei Wartezyklen TW ($\geq 2 \times 500 \text{ ns}$) eingefügt werden, um die maximale Adreßzugriffszeit von $1,5 \mu\text{s}$ des Bausteins CM 8001 zu garantieren. Das wird erreicht, indem die READY-Leitung in T2 auf L und zu Beginn des zweiten Wartezyklus wieder auf H gelegt wird. Nach Beendigung des zweiten Wartezyklus wird mit dem Zustand T3 fortgefahren. Unter Berücksichtigung der Voreinstell- und Haltezeit für das READY-Signal (Bild 2.2.11.) ergibt sich eine mögliche Schaltung nach Bild 2.2.12. Es handelt sich um einen Automaten mit drei Zuständen, der durch $\overline{\text{RESET}} = \text{L}$ bei Spannungseinschalten

Bild 2.2.10.: Anschluß der Speichereinheit an den 8080 A



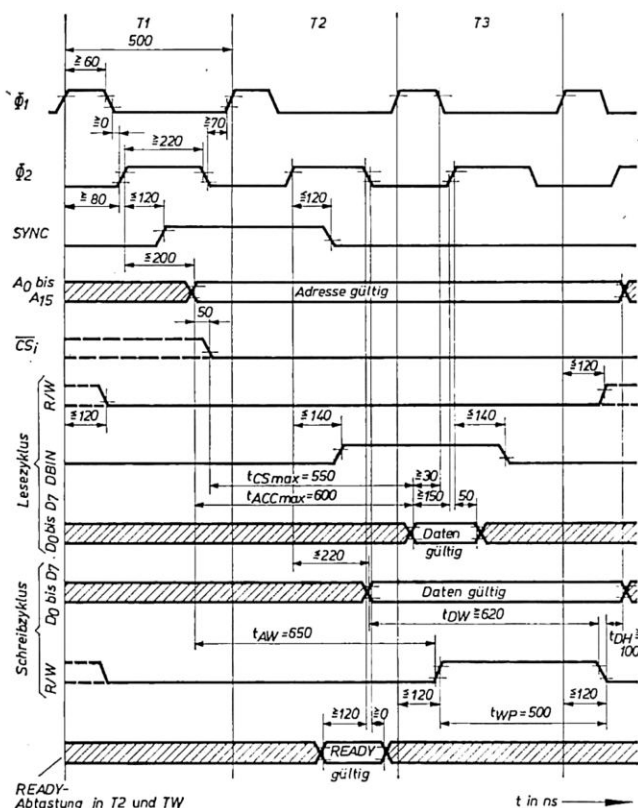


Bild 2.2.11.:
Taktchema für
Speicheroperationen
beim 8080 A.
Errechnung der
Speicherzeitenanforderungen
unter
Worst-case-Bedingungen, Zeit für
CS-Dekodierung
mit 50 ns angenommen

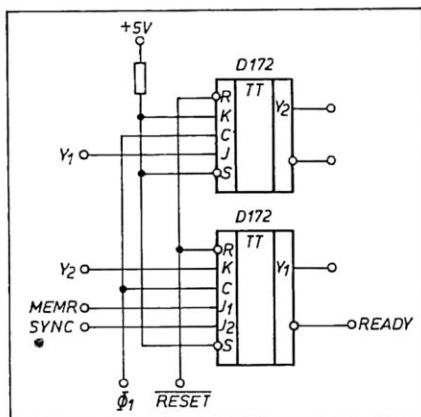


Bild 2.2.12.:
READY-Steuerung

in den Ausgangszustand ($Y_1 = L$, $Y_2 = L$) gezwungen wird. Bei einem Speicherlesezyklus (CPU-Statussignal MEMR = H) wird die Schaltung während T2 (SYNC = H) mit dem Takt Φ_1 in den Zustand $Y_1 = H$, $Y_2 = L$ gebracht, gleichzeitig wird READY = L erzeugt. Im nächsten Taktzyklus (erster Wartezustand) schaltet der Automat in den Zustand $Y_1 = H$, $Y_2 = H$. READY bleibt L. Zu Beginn des zweiten Wartezustandes wird mit Φ_1 der Ausgangszustand $Y_1 = L$, $Y_2 = L$ erreicht, so daß zum Abtastzeitpunkt der READY-Leitung READY = H erkannt wird. Im anschließenden Zustand T3 liegen die Lesedaten vom Speicher an der CPU an.

Soll auf die Warteschaltung verzichtet werden, muß man schnellere Speicher verwenden oder die CPU-Zykluszeit entsprechend den Möglichkeiten verlängern.

2.2.4.3. Anschluß der Speichereinheit an die Z 80-CPU

Eine weitere Vereinfachung ergibt sich beim Anschluß der Speichereinheit an die Z 80-CPU. Der Anschluß der Adreß- und Datenleitungen sowie die Chip-Select-Dekodierung erfolgt wie beim 8080 A entsprechend Bild 2.2.10. Die Vereinfachung ergibt sich für die externe Steuerlogik auf Grund der bereitgestellten CPU-Steuersignale des Z 80 (s. Abschnitt 2.1.). Daraus folgt:

Lesen-Schreibsignal R/W = MREQ WR (2.2.30.)

Multiplexerauswahl

für Speicherdaten SS1 = MREQ RD (2.2.31.)

für Interruptdaten SS2 = IOREQ M1 (2.2.32.)

für Eingabedaten SSi = IOREQ RD
(Geräteadresse i) (2.2.33.)

Die Freigabe des Eingangspuffers 8212 erfolgt mit

DS2 = RD $\vee$ SS2 (2.2.34.)

Im Taktschema der CPU besteht ein Unterschied zwischen einem Befehlsaufrufzyklus (M1) und einem Speicherlesezyklus für Daten. Bei einem M1-Zyklus müssen die Lese-daten bereits auf der Vorderflanke von T3 der CPU zur Verfügung stehen, bei einem Datenlesezyklus erst auf der Rückflanke von T3. Daraus ergeben sich beim M1-Zyklus die höchsten Zeitforderungen für die Speichereinheit. Aus Bild 2.2.13. läßt sich bei minimaler CPU-Zykluszeit von 400 ns (ohne Berücksichtigung der Verzögerungszeiten der Puffer, Treiber usw.) folgendes ablesen:

M1-Zyklus
Adreßzugriffszeit $t_{ACC}(t_A) \leq 590$ ns
Chip-Select-Zugriffszeit $t_{CS}(t_{C0}) \leq 540$ ns

Dementsprechend lassen sich die Zeitforderungen für einen Datenlesezyklus und einen Schreibzyklus ermitteln:

Datenlesezyklus
Adreßzugriffszeit $t_{ACC}(t_A) \leq 780$ ns
Chip-Select-Zugriffszeit $t_{CS}(t_{C0}) \leq 730$ ns
Schreibzyklus
Adreßvoreinstellzeit vor Schreibimpuls $t_{WD}(t_{AW}) \leq 440$ ns
Chip-Select-Voreinstellzeit vor Schreibimpuls $t_{CW} \leq 790$ ns
Schreibimpulsbreite $t_{WP} \leq 360$ ns
Datenvoreinstellzeit $t_{DW} \leq 580$ ns
Datenhaltezeit $t_{DH} \leq 150$ ns

Will man mit minimaler CPU-Zykluszeit arbeiten und stehen keine schnelleren Speicher zur Verfügung, so sind wieder entsprechend dem Abschnitt 2.2.4.2. WAIT-Zyklen zwischen

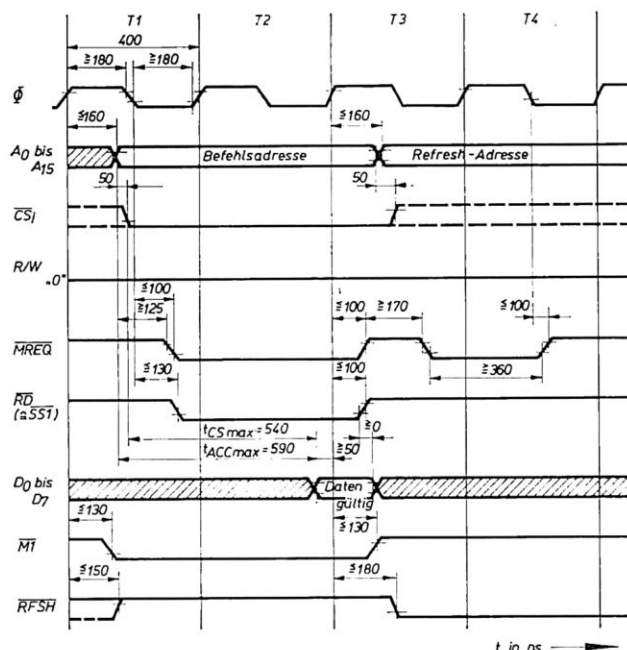


Bild 2.2.13.: M1-Zyklus für Z 80-CPU

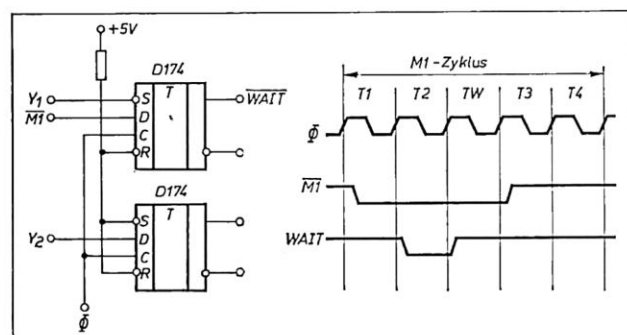


Bild 2.2.14.: Warteschaltung für M1-Zyklen beim Z 80 (statt WAIT lies WAIT)

T2 und T3 einzufügen. Unter Umständen ist nur für den M1-Zyklus ein Wartezustand notwendig. Eine solche Schaltung zeigt Bild 2.2.14. nach [2.2.1.]. Die Z 80 CPU bietet einen weiteren Vorteil beim Anschluß dynamischer Speicher. Während eines M1-Zyklus (s. Bild 2.2.13.) wird in den Taktzuständen T3 und T4 ein Refresh-Signal $\overline{RFSH} = L$ erzeugt. Das bedeutet, daß die niederwertigen 7 bit des Adreßbusses eine Refresh-Adresse für dynamische Speicher führen

und das laufende MREQ-Signal zur Einleitung eines Refresh-Zyklus für alle angeschlossenen dynamischen Speicher benutzt werden kann.

Literatur

[2.2.1.] Blomeyer-Bartenstein, H.-P.: Anwendung von Standard-Speicherbausteinen bei Mikrocomputersystemen. Elektronik 27 (1977) H. 3, S. 75-82

2.3. Ein- und Ausgabe-Einheit

2.3.1. Ein- und Ausgabe-Prinzipien

Die Ein- und Ausgabe besitzt bei Mikrorechnern ebenso wie bei Rechnern anderer Leistungsklassen eine außerordentliche Bedeutung. Erst durch die Realisierung der Ein- und Ausgabefunktionen kann ein Rechner bezüglich der Kommunikation mit der Umwelt seinen Aufgaben gerecht werden. Gerade bei Mikrorechnern, die verstärkt in vielen kommunikationsintensiven Prozessen eingesetzt werden, stehen Fragen der effektiven Gestaltung der Ein- und Ausgabe im Mittelpunkt. Der bisher betriebene Aufwand führte einerseits zur Entwicklung optimaler Methoden der Realisierung der Ein- und Ausgabe-Funktionen und andererseits auch zur Konzipierung universeller programmierbarer hochintelligenter E/A-Schaltkreise.

Die Programmierbarkeit entstand aus der Notwendigkeit der Massenanfertigung. Durch die in die Hände des Anwenders gegebene Programmierungsmöglichkeit kann damit ein einmal hardwaremäßig festgelegter Mikrorechner noch bezüglich der E/A-Interface-Gestaltung modifiziert werden.

Die Anwendungsgebiete der Ein- und Ausgabe-IS bestehen sowohl in der Interface-Gestaltung für Peripheriegeräte, wie z. B. die beim Mikrorechner typischen Datensichtgeräte mit oder ohne Tastatur, Fernschreiber, Lochbandleser, Lochbandstanzer, Floppy-Disk, Magnetbandkassette, PROM-Programmierungseinrichtung, Thermodrucker usw., als auch in der Realisierung von Standard-Interface der Automatisierungstechnik (z. B. SI 1.2) und in der Gestaltung von Spezial-Interface für den direkten Anschluß von Prozeßelementen.

Die Ein- und Ausgabe kann prinzipiell in die Einleitungs- und die Ausführungsphase eingeteilt werden. Die Aufgaben der Einleitungsphase bestehen in einer Identifizierung bzw. Auswahl der entsprechenden E/A-Komponente, während die Ausführungsphase den Algorithmus zur Durchführung der Informationsübertragung zwischen Rechner und entsprechenden E/A-Elementen realisiert. Beide Phasen bilden in ihrer Realisierung eine Einheit, können jedoch oftmals getrennt betrachtet werden. Tafel 2.3.1. gibt dazu eine Übersicht.

2.3.1.1. E/A-Einleitungsphase

Die E/A-Einleitungsphase, die auch mit Signalquellenerkennung bezeichnet werden kann, wird prinzipiell in die Abfragemethode und die Unterbrechungsmethode unterteilt. Beide Methoden unterscheiden sich durch die Auslösungsmöglichkeit des Bearbeitungswunsches und seiner Quittierung (Bild 2.3.1.).

Bei der Abfragemethode werden die Prozeß- oder Peripherieelemente vom Mikroprozessor abgefragt. Ein Bearbeitungswunsch wird frühestens dann akzeptiert, wenn der Abfragezeitpunkt erreicht ist. Häufig wird deshalb zuerst das Gesamtspektrum abgefragt und anschließend entsprechend einer Wichtung bearbeitet. Der Vorteil dieses Verfahrens besteht darin, daß innerhalb eines bestimmten Zeitintervalls jedes Element einmal abgefragt wird, ein Nachteil jedoch ist das möglicherweise relativ lange Zeitintervall zwischen Bearbeitungswunschanmeldung und -anerkennung. Die Unterbrechungsmethode basiert auf der Möglichkeit, im Mikrorechner laufende Programme durch Unterbrechungsanmeldesignale zu unterbrechen und spezielle Bearbeitungsprogramme der UB-Quellen (UB-Serviceroutinen) zu starten. Im Prinzip wird zwischen der sequentiellen und verschachtelten Unterbrechungsverarbeitung unterschieden.

Die sequentielle Unterbrechungsverarbeitung realisiert dabei eine sequentielle Abarbeitung der Unterbrechungsanmeldungen in der Reihenfolge ihrer Prioritäten. Sobald die UB-Serviceroutine mit der höchsten Priorität läuft, werden

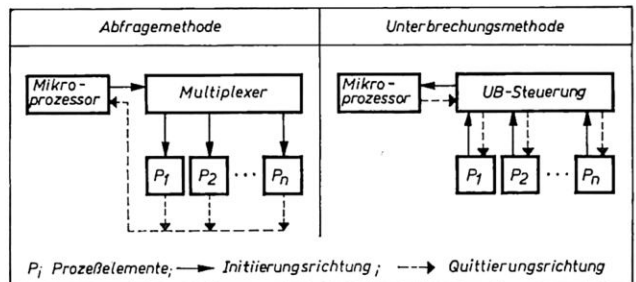
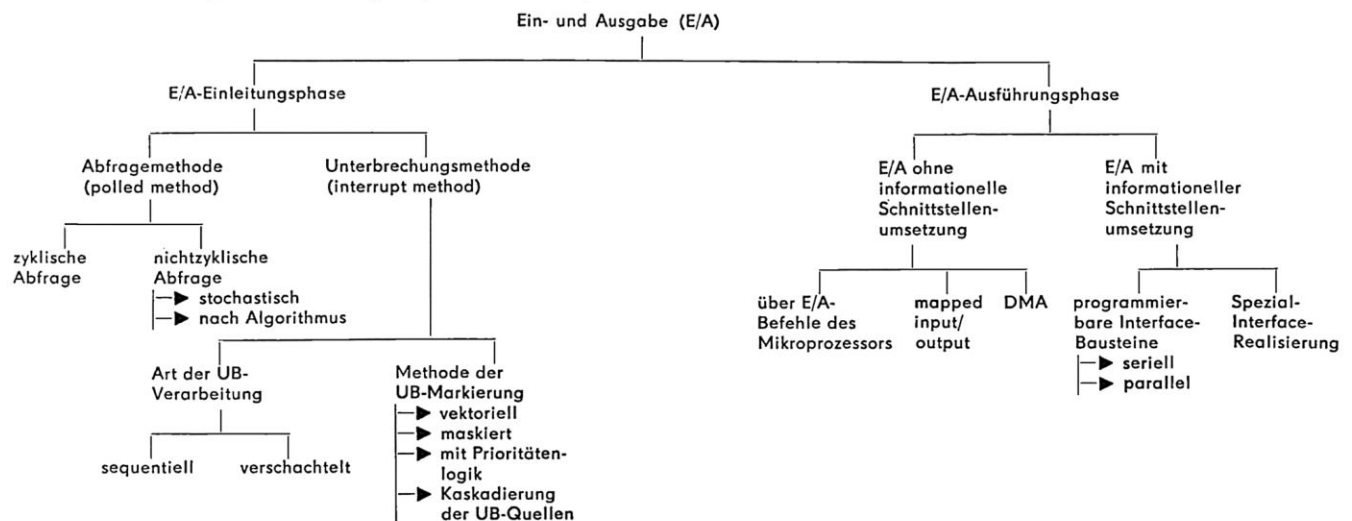


Bild 2.3.1.: Abfrage und Unterbrechungsmethode

Tafel 2.3.1. Unterteilung der Ein- und Ausgabe (bei Mikrorechnern)



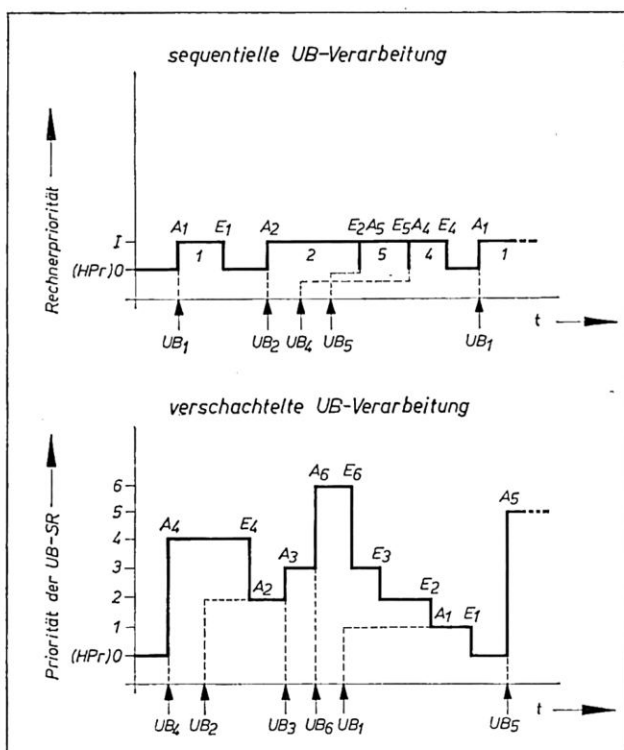


Bild 2.3.2.: Sequentielle und verschachtelte Unterbrechungsverarbeitung. HPr = Hauptprogramm; UB-SR = UB-Serviceroutine; UB_i = Unterbrechung der Priorität i; A_i, E_i = Anfang bzw. Ende der UB-SR_i. Es gilt: Priorität (UB-SR_i) < Priorität (UB-SR_{i+1})

alle anderen Unterbrechungen hardware- oder softwaremäßig maskiert. Das gilt auch, wenn inzwischen eine Unterbrechungsanmeldung höherer Priorität eingetroffen ist. Ein eventuell vorhandenes Hauptprogramm dient zur Bearbeitung von zeitunkritischen Aufgaben.

Bei der verschachtelten Unterbrechungsverarbeitung (nesting interrupt), die die kürzeste Reaktionszeit besitzt, kann eine UB-Serviceroutine jederzeit von einer UB-Serviceroutine mit höherer Wertigkeit unterbrochen werden, während Unterbrechungen mit niedriger Wertigkeit warten müssen. Da sich dabei mehrere UB-Serviceroutinen gegenseitig unterbrechen können, ist zum Abspeichern der verschiedenen Rechnerzustandsinformationen ein Stapelspeicher (Stack) notwendig. Das Hauptprogramm läuft im Hintergrund in einer Endlosschleife oder wird nach Bearbeitungsteilen in den HALT-Zustand gebracht. Die verschachtelte Unterbrechungsverarbeitung wird immer dann angewendet, wenn die Teilprozesse stark unterschiedliche Prioritäten haben und die Reaktionszeiten klein sein müssen.

Die Mikroprozessoren ermöglichen mit oder ohne Zusatzbeschaltung durch entsprechende Hardware eine von beiden oder beide Unterbrechungsverarbeitungsarten. Bild 2.3.2. verdeutlicht die prinzipiellen Zusammenhänge.

Eine gewisse Zwischenstellung zwischen der Abfrage- und der Unterbrechungsmethode nimmt die rotierende Prioritätszuordnung ein. Hier wird versucht, die Vorteile beider Methoden, also die garantierte Abfrage innerhalb bestimmter, softwaremäßig festgelegter Zeitintervalle und die möglichst kurze Reaktionszeit auf UB-Anforderungen zu vereinen. Das erfolgt nach dem Prinzip, daß nach jeder erfolgten Anerkennung einer Unterbrechungsanmeldung oder nach einem fest vorgegebenen Zeitraster die Prioritäten jeweils um eine Ebene erhöht werden. Die höchste Priorität wird dann allerdings zur niedrigsten (Rotation im geschlossenen Ring). Dieses Verfahren wird, um die Zeitvorteile zu nutzen, hardwaremäßig unterstützt und findet gerade bei Prozessen mit ausgeprägtem Echtzeitverhalten Anwendung.

Eine weitere Möglichkeit, das Verfahren der prioritätsbestimmenden Kaskadierung (daisy chain priority), findet verstärkt bei hochintegrierten Schaltkreisen Anwendung. Allgemein wird durch eine durchgeschleifte Blockier- bzw. Befähigungsleitung (je nach konkretem Arbeitsprinzip) die Priorität eines Bausteins oder Elementes durch seine Eingliederung in eine Kettenschaltung bestimmt. Dies ist eine

einfache Realisierungsvariante der Prioritätensteuerung ohne zusätzlichen Hardware-Aufwand. Über entsprechende softwaremäßige Maskierung kann zusätzlich der Bearbeitungswunsch einer prioritätsmäßig höheren UB-Serviceroutine während der Abarbeitung einer laufenden Routine zurückgestellt werden.

2.3.1.2. E/A-Ausführungsphase

Die E/A-Ausführungsphase beinhaltet den konkreten Algorithmus der Informationsübertragung sowie dessen zeitliche Synchronisation. Man unterscheidet zwischen einer Ein- und Ausgabe mit und ohne informationelle Schnittstellenumsetzung. Man versteht darunter die Umsetzung der Mikroprozessor-Schnittstelle in die spezielle Schnittstelle der Peripheriegeräte oder der Prozeßsteuerung hinsichtlich logischer und zeitlicher Zusammenhänge. Probleme der elektrischen (z. B. Pegelanpassung, AD- und DA-Wandlung) und der konstruktiven Anpassung wurden in der Einteilung nach Tafel 2.3.1. nicht berücksichtigt, sie sollen noch gesondert behandelt werden.

Die informationelle Schnittstellenumsetzung bietet wesentliche Grundlagen für eine komplexe Anwendung und ist vor allem dort notwendig, wo ein gesicherter Datentransfer, oftmals nach dem Forderungs- und Quittungsprinzip (Handshaking-Prinzip), notwendig ist. Die programmierbaren E/A-Schaltkreise gestatten generell die Möglichkeit einer informationellen Schnittstellenumsetzung. Es muß aber auch bemerkt werden, daß dies Zeitverluste bringt. Bei zeitkritischen Prozeßsteuerungen sollten die Möglichkeiten der Ein- und Ausgabe ohne Schnittstellenumsetzung genutzt werden.

Eine Möglichkeit wäre die Ein- und Ausgabe-Methode mit in vorgegebenen Bereichen fortlaufender Adressierung der E/A-Elemente (mapped input/output). Deshalb werden häufig speicherbezogene Befehle, die diese Möglichkeit bieten, genutzt. Die Unterscheidung, ob es sich um ein Lesen bzw. Schreiben von E/A- oder Speicher-Informationen handelt, erfolgt durch Kennzeichnung. Im einfachsten Fall wird dazu ein Bit der Speicheradresse genutzt. Das schränkt zwar die Speicheradressierfähigkeit auf die Hälfte ein, ist aber bei den meisten Mikroprozessoren eine günstige Lösungsvariante. Mapped Input/Output gestattet bei angepaßten schnellen Ein- und Ausgabe-Elementen und softwaremäßig optimal festgelegten E/A-Algorithmen relativ hohe Datenaustauschraten.

Eine zweite Möglichkeit ist der direkte Speicherzugriff (DMA), der auch eine Form der Ein- und Ausgabe ist. Hier wird unter der Verwaltungspriorität des Mikroprozessors der direkte Zugriff zum Arbeitsspeicher (hier RAM) realisiert. Der Informationsaustausch erfolgt ohne Nutzung des Mikroprozessors, der sich dann bei Unibus-Systemen im tristabilen Zustand bezüglich seiner Busanschlüsse befindet, direkt mit dem Arbeitsspeicher. Eine entsprechende Steuerung, die ebenfalls mit hochintegrierten Schaltkreisen (s. Abschnitt 2.3.3.2.) realisiert werden kann, ermöglicht die Synchronisation des Datenaustausches, der dann optimal blockweise erfolgt.

Wird fortgesetzt

SOEBEN ERSCIENEN

A. Jügel

Mikroprozessorsysteme

204 Seiten, 89 Bilder, 30 Tafeln, Pappband 20,00 Mark
Bestellnummer 552 6460

Das vorliegende Buch beschreibt Entwurf und Anwendung von Mikrorechnern. In den einzelnen Abschnitten werden Mikroprozessoren, Halbleiterspeicher (RAM, ROM) sowie Eingabe- und Ausgabe-Verfahren unter dem Gesichtspunkt ihres Zusammenwirkens in einem Mikroprozessorsystem ausführlich behandelt. Aufbau und Arbeitsweise werden soweit erklärt, wie sie für den Anwender wichtig sind. Auf schaltungstechnische Details und technologische Probleme des Schaltkreisentwurfs und der Schaltungsherstellung wird nicht eingegangen. Neben Grundbegriffen der Programmierung werden einige Programmbeispiele und die Befehlsliste eines Mikroprozessors erläutert. Das Buch soll dem Anwender als Entscheidungsgrundlage für die Einsatzmöglichkeiten des Mikrorechners, für die Wahl des Mikroprozessortyps und für die Einsatzvorbereitung eines Mikrorechners dienen.

VEB VERLAG TECHNIK BERLIN

2.3.2. Programmierbare E/A-Schaltkreise

Die im Abschnitt 1.3. vorgestellten komplexen MOS-E/A-Schaltkreise sind durch ihre Programmierbarkeit sehr universell anwendbar. Die Realisierung von E/A-Schnittstellen mit derartigen Bausteinen ist deshalb als Einheit von Hard- und Software zu betrachten. Die Programmierung, d. h. die detaillierte Festlegung des Ablaufs der Ein- oder Ausgabe, erfolgt durch Übertragung von Steuerworten von der CPU zum E/A-Schaltkreis, wo die entsprechende Steuerinformation in Flip-Flop-Speicherzellen so lange erhalten bleibt, bis die neue Steuerinformation eingeschrieben wird. Die Programme zur Programmierung der E/A-Schaltkreise und die Routinen zur Durchführung der Ein- bzw. Ausgabe sind Bestandteil der Mikrorechnerprogramme und meist im ROM-Teil des Speichers abgelegt.

Neben dieser Art von Schaltkreisen, die in ihrer grundsätzlichen Arbeitsweise (seriell oder parallel) festgelegt sind, wurden in neuerer Zeit auch Bausteine entwickelt, die selbst eine CPU und ROMs zur vom Nutzer programmierbaren Durchführung der Ein- oder Ausgabe enthalten, wodurch der eigentliche Mikrorechner entlastet und extreme Flexibilität in der Gestaltung der E/A-Schnittstellen erreicht wird.

2.3.2.1. Serielle Ein- und Ausgabe

Blattschreiber werden als E/A-Geräte für Mikrorechner oft eingesetzt, weil sie relativ billig sind und sowohl als Kommunikationsmittel als auch zur Erzeugung schriftlicher Protokolle verwendet werden können. Insbesondere bei der Programmentwicklung kann man kaum auf Protokolle verzichten, und als Alternativlösung stehen nur wesentlich aufwendigere Konfigurationen wie Bildschirmgerät plus Seriendrucker zur Verfügung.

Der Nachteil dieser Fernschreiber besteht in ihrer niedrigen Geschwindigkeit und in der Verwendung des Fünf-Kanal-Kodes nach CCITT, der Umschaltzeichen (Zahlen/Sonderzeichen oder Buchstaben) benötigt, um den gesamten Zeichenvorrat darzustellen. Die Übertragung erfolgt seriell mit einer Schrittgeschwindigkeit von 50 Baud, die Übertragungszeit für ein Zeichen beträgt 150 ms (s. Bild 2.3.3.). Daraus ergeben sich folgende Festlegungen z. B. für die Programmierung eines E/A-Schaltkreises 8251:

- asynchroner Betrieb, Teilerfaktor für den internen Teiler 64 gewählt, um bei der Erzeugung des Empfangs- und Sendetaktes aus dem Systemtakt Teilerstufen zu sparen ($D1 = 1$, $D0 = 1$)
- Zeichenlänge 5 Bit ($D3 = 0$, $D2 = 0$)
- keine Bildung eines Paritätsbits ($D4 = 0$)
- ungerade Parität ($D5 = 0$), ohne Bedeutung
- $1\frac{1}{2}$ Stopp-Bits ($D7 = 1$, $D6 = 0$).

Diese Festlegungen sind in dem Betriebsart-Steuerwort D7 bis $D0 = 10000011$ (83H; H = hexadezimal) enthalten, das an die Steuereinheit des 8251 (Adressierung durch $\overline{CS} = 0$,

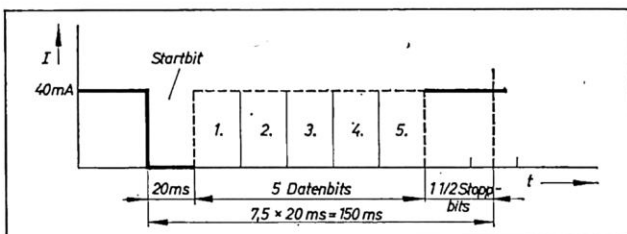


Bild 2.3.3.: Zeitlicher Ablauf bei der Übertragung eines Fernschreiber-Zeichens

TECHNIK DER MIKRORECHNER

13

$\overline{C/D} = 1$) übertragen wird. Danach erfolgt die Aussendung des Kommandowortes, ebenfalls an die Steuereinheit (s. a. Bild 1.3.6.).

Im vorliegenden Beispiel wird nur ein Teil der Information benötigt:

- Sendefreigabe ($D0 = 1$)
- Empfangsfreigabe ($D2 = 1$)
- im Ruhezustand, d. h., wenn kein Zeichen ausgesendet wird, führt der TXD-Ausgang entsprechend Ruhestrom 1-Potential ($D3 = 0$)
- kein Rücksetzen in Empfangsbereitschaft für eine erneute Betriebsart-Übertragung ($D6 = 0$).

Die restlichen Bits des Kommandowortes haben hier keine Bedeutung und werden willkürlich festgelegt: $D1 = 1$, $D4 = 0$, $D5 = 1$, $D7 = 0$. So ergibt sich das Kommandowort D7 bis $D0 = 00100111$ (27H).

Nach dem Anfangsrücksetzen des gesamten Mikrorechners erfolgt damit die Initialisierung, d. h. das in Betriebsbereitschaft Setzen des 8251 durch folgendes Programm (Adresse des Steuerkanals OFF H, des Datenkanals OFE H):

MVI A, 83H; Laden des AC der 8080-CPU mit dem Betriebsart-Steuerwort 83H

OUT OFFH; Ausgabe des AC-Inhalts nach der Steuereinheit des 8251.

MVI A, 27H; Laden des AC mit dem Kommandowort 27H
OUT OFFH; Ausgabe des AC-Inhalts nach der Steuereinheit des 8251.

Damit ist der 8251 programmiert, von nun ab erfolgt nach einer Ausgabe des AC-Inhalts nach dem Datenkanal durch den Befehl OUT OFEH die serielle Ausgabe über den TxD-Ausgang zum Fernschreiber.

Bei einem Befehl IN OFEH wird das vorher über den RxD-Eingang seriell eingelesene Fernschreiberzeichen in den AC der CPU übertragen. Voraussetzung für eine ordnungsgemäße Funktion ist jedoch, daß vor der Ausgabe der Sendepuffer frei ist bzw. bei Eingabe das Fernschreiberzeichen bereits vollständig in den Empfangspuffer eingelesen wurde.

Diese Information erhält die CPU aus dem Status des 8251, der durch den Befehl IN OFFH, also Lesen über den Steuerkanal, in den AC übertragen wird und dann für eine Auswertung verfügbar ist.

Im Statuswort haben die beiden niederwertigsten Bits die Bedeutung:

$D0 = 1$: Sender ist mit der vorherigen seriellen Ausgabe fertig und bereit, ein neues Zeichen aufzunehmen und zu übertragen

$D1 = 1$: Empfänger hat ein Fernschreiberzeichen vollständig empfangen.

Somit ergeben sich folgende Routinen für die Ein- und Ausgabe, die über einen CALL-Befehl (CALL EING bzw. CALL AUSG) von einem Programm aufgerufen werden können.

- | | |
|----------------|---|
| EING: IN OFFH; | Einlesen der Statusinformation nach dem AC |
| ANI 02H; | Ausblenden des Bit 1 der Statusinformation |
| JZ EING; | Rücksprung zum Anfang, wenn das Zeichen noch nicht zur Verfügung steht, d. h. die serielle Eingabe noch nicht beendet ist (Polling) |
| IN OFEH; | Eingabe des Fernschreiberzeichens nach dem AC |

RET; Rücksprung in das Programm, das diese Routine über CALL EING aufgerufen hatte.

Bei der Ausgabe wird der AC zum Auswerten der Statusinformation benötigt, deshalb sei hier vorausgesetzt, daß das auszugebende Zeichen im C-Register der CPU bereitgestellt ist.

AUSG: IN OFFH; Einlesen des Status nach dem AC
 ANI 01H; Ausblenden des Bit 0 der Statusinformation
 JZ AUSG; Rücksprung zum Anfang, solange der Sendepuffer nicht frei ist (Polling)
 MOV A, C; Transport des auszugebenden Zeichens vom C-Register zum AC
 OUT OFEH; Ausgabe des Zeichens vom AC zum Sendepuffer des 8251
 RET; Rücksprung in das Programm, das diese Routine über CALL AUSG aufgerufen hatte.

Bei der Bereitstellung auszugebender Daten und der Auswertung eingegebener Daten ist zu beachten, daß das niederwertigste Datenbit als erstes übertragen wird. Im Bild 2.3.4. ist das Prinzipschaltbild eines Fernschreiberanschlusses dargestellt, der logische 1-Pegel ist dabei einem Strom von 40 mA im Fernschreiberkreis zugeordnet.

2.3.2.2. Parallele Ein- und Ausgabe-Schnittstelle

Der Einsatz des Z80-PIO zur Realisierung von parallelen Interfaces soll den Schaltungsaufwand weitgehend reduzieren. Komplizierte Interface-Bedingungen erfordern jedoch zwei Kanäle: je einen für die Steuerinformation (Quittungssignale, Kommando- und Statusinformationen) und für die zu übertragenden Daten. Damit kann mit einem PIO-Baustein nur ein Eingabe- oder ein Ausgabekanal verwirklicht werden, wenn nicht durch zusätzlichen Schaltungsaufwand Unterstützung gegeben wird. Das widerspricht aber der eigentlichen Zielstellung beim Einsatz des PIO.

Ein Eingabe- und ein Ausgabekanal lassen sich mit einem PIO nur dann direkt realisieren, wenn sehr einfache Interface-Bedingungen vorliegen, die sich auf ein Steuersignalkonzept je Kanal (realisiert durch die PIO-Signale STB und RDY) beschränken. Dabei muß dann die Auslösung eines Ausgabevorganges von der CPU, die Auslösung einer Eingabe aber vom Eingabegerät erfolgen.

Beim Anschluß von Peripheriegeräten der EDV liegen solche einfachen Bedingungen nicht vor. Am Beispiel des ESEG-Standardanschlusses gemäß daro 1000 (SIF 1000) soll die Realisierung mit Z80-PIO gezeigt werden. Dieses Interface benutzt zwei Handshake-Signale RUF und END, maximal drei Kommando- und maximal drei Statussignale je Kanal. Die Kommandoinformation gibt detaillierte Anweisungen für die E/A-Operation, die Statusinformation gibt der CPU Auskunft über den Zustand des peripheren Gerätes.

Kommando- und Statussignale werden bei den verschiedenen Geräten unterschiedlich bzw. nur teilweise verwendet. Die Auslösung des Datentransportes geht von der CPU aus und erfolgt durch Aussendung von RUF-E bzw. RUF-A, und der Vollzug wird vom peripheren Gerät durch END-E bzw. END-A quittiert (Bild 2.3.5.).

Im folgenden soll eine Eingabeschnittstelle beschrieben werden, die den Anschluß eines Lochband- und Lochkartenlesers daro 1210-0330 ermöglicht. Bei dieser Schnittstelle werden die Signale als negierte KME-3-Pegel übertragen. Es werden alle drei Kommandosignale KOM-E1, KOM-E2, KOM-E3 sowie das Statussignal STA-E3 benutzt. Sie haben folgende Bedeutung:

KOM-E1: 1 $\triangleq$ Lesen
 0 $\triangleq$ Transport
 KOM-E2: 1 $\triangleq$ alle Lochkombinationen werden verarbeitet
 0 $\triangleq$ Kombinationen 0000 0000 und 1111 1111 werden überlesen
 KOM-E3: 1 $\triangleq$ Rückwärtstransport
 0 $\triangleq$ Vorwärtstransport
 STA-E3: 1 $\triangleq$ Bandende, Bandriß, wird ohne END-E übertragen
 0 $\triangleq$ Normalbetrieb.

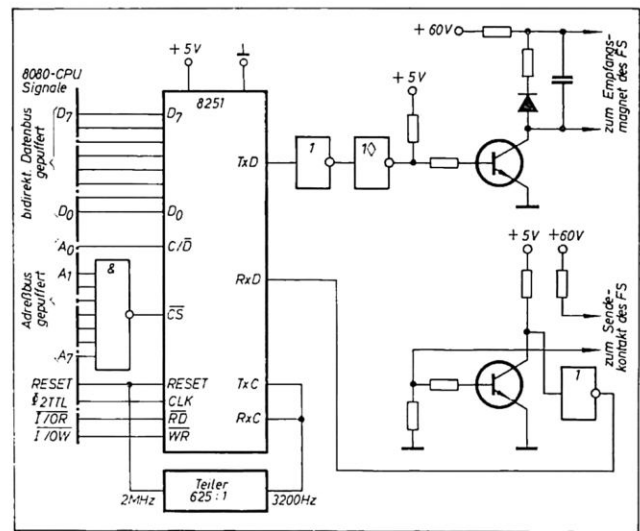


Bild 2.3.4.: Schaltbild des Fernschreiberanschlusses

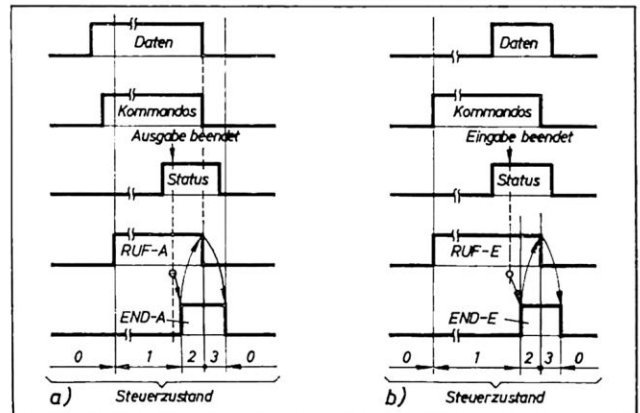


Bild 2.3.5.: Zeitabläufe des SIF 1000. a) Ausgabe; b) Eingabe

Zur Übertragung der Steuerinformation soll Kanal B, zur Übertragung der Daten Kanal A, beide in der Betriebsart Bit-Ein-Ausgabe, verwendet werden (vgl. Abschnitt 1.3.4.2.). Bei Kanal B seien B7 = RUF-E, B6 = KOM-E3, B5 = KOM-E2, B4 = KOM-E1, B3 = STA-E1, B2 = STA-E2, B1 = STA-E3, B0 = END-E.

B7 bis B4 müssen in Ausgaberrichtung, B3 bis B0 in Eingaberrichtung betrieben werden.

Die Kommandos können gleichzeitig mit RUF-E eingeschaltet werden. Danach muß so lange gewartet werden, bis die einzugebenden Daten bereitgestellt sind, was durch END-E = 1 signalisiert wird.

Es bestehen zwei grundsätzliche Möglichkeiten:

- Die CPU liest so lange Kanal B und wertet B0 aus, bis B0 = 1 wird, danach erfolgt die Dateneingabe über Kanal A.
- B0 von Kanal B wird in die Lage versetzt, eine Unterbrechung auszulösen, und die dadurch aufgerufene Routine führt die Eingabe der Daten über Kanal A durch.

Das Problem besteht jedoch darin, daß bei Fehlern im Leser oder bei Bandende/Bandriß kein Signal END-E erzeugt wird. Deshalb muß eine Zeitüberwachung vorgenommen werden, die bei Überschreiten einer bestimmten Zeit ein Fehlermaßnahmeprogramm einleitet. Bei Verwendung der Variante mit Auslösung einer Unterbrechung durch END-E ist die Zeitüberwachung durch den Einsatz eines Zählers/Zeitgeberbausteins Z80-CTC (s. Abschnitt 2.3.3.1.) zweckmäßig.

Hier soll eine leichter überschaubare Lösung skizziert werden, die ohne Unterbrechungsbetrieb auskommt.

Es sei CS = A7 A6 A5 A4 A3 A2, B/A = A0, C/D = A1, wodurch folgende Geräteadressen zugeordnet werden:

OFCH = Datenkanal A, OFDH = Datenkanal B
 OFEH = Steuerkanal A, OFFH = Steuerkanal B.

Die Programmierung des PIO wird durch das folgende Programm durchgeführt.

```
LD A, OCFH:   Laden des AC mit 1100 1111, Betriebsart-
               steuerwort Bit-Ein-Ausgabe für Kanal A
OUT OFEH, A:   Ausgabe des AC nach Steuerkanal A
LD A, OFFH:   Laden des AC mit 1111 1111, alle An-
               schlüsse des Kanals A sind Eingänge
OUT OFEH, A:   Ausgabe des AC nach Steuerkanal A
LD A, OCFH:   Laden des AC mit 1100 1111, Betriebsart-
               steuerwort Bit-Ein-Ausgabe für Kanal B
OUT OFFH, A:   Ausgabe des AC nach Steuerkanal B
LD A, OFH:    Laden des AC mit 0000 1111, Anschlüsse
               B7 bis B4 Ausgänge, B3 bis B0 Eingänge
OUT OFFH, A:   Ausgabe des AC nach Steuerkanal B
LD A, 03H:    Laden des AC mit Steuerwort für Unterbre-
               chungssperre
OUT OFEH:     Ausgabe nach Steuerkanal A
OUT OFFH:     Ausgabe nach Steuerkanal B.
```

Die Routine für eine Eingabe (Vorwärtslesen aller Zeichen, KOM-E1 = KOM-E2 = 1, KOM-E3 = 0) ohne Zeitüberwachung lautet:

```
EIN:   LD A, OEOH:   Laden des AC mit RUF-E = 1,
                   KOM-E1 = KOM-E2 = 1, KOM-
                   E3 = 0
      OUT OFDH, A:   Ausgabe des AC nach Kanal B
MARK 1: IN A, OFDH:  Eingabe von Kanal B
      AND 01H:       Ausblenden von B0
      JP Z, MARK 1:  Sprung nach MARK 1, wenn B0
                   = END-E = 0 ist
      IN A, OFCH:    Eingabe der Daten nach dem AC
      LD C, A:        Transport AC nach C-Register
      LD A, OOH:      Laden des AC mit 0
      OUT OFDH, A:   Aussenden des AC nach Kanal
                   B, damit RUF-E = KOM-E1,
                   KOM-E2, KOM-E3 = 0
MARK 2: IN A, OFDH:  Eingabe von Kanal B
      AND 01H:       Ausblenden von B0
      JP NZ, MARK 2: Sprung nach MARK 2, wenn
                   END-E noch nicht 0
      LD A, C:        Transport C-Register nach AC
      RET:            Rücksprung in das Programm,
                   das die Routine EIN gerufen
                   hatte.
```

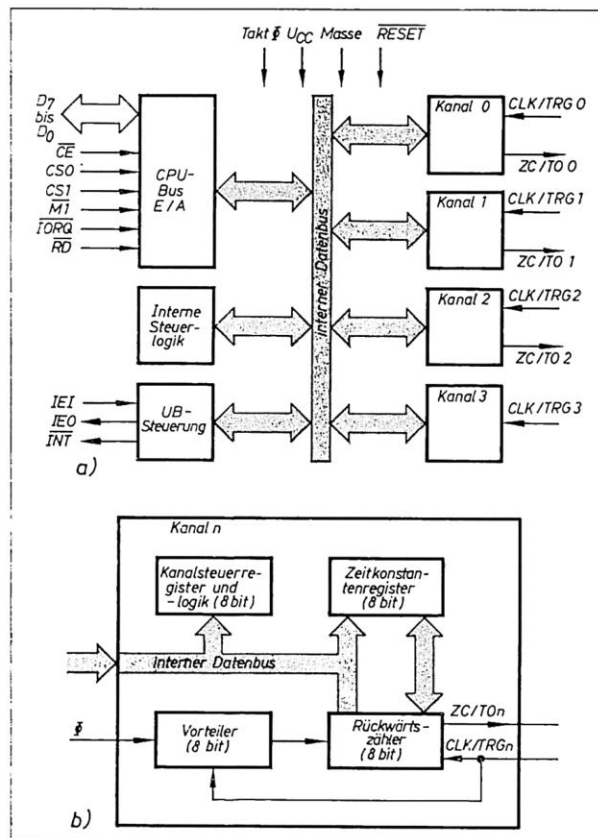


Bild 2.3.6.: Blockschaubild des Z80-CTC. a) gesamt; b) ein Kanal

Die Berücksichtigung einer Zeitüberwachung und das Auslesen des STA-E3 sind durch Programmiererweiterung leicht möglich.

Die Schaltung enthält keine Schwierigkeiten, die Verbindung zwischen PIO und Lochbandleser erfordert lediglich negierende TTL-KME3-Pegelwandler, und die Ausgänge des Kanals B (B7 bis B4) sind über 390-Ω-Widerstände nach Masse zu verbinden (Pull-down-Widerstände).

2.3.3. Sonderbausteine

2.3.3.1. Zähler-Zeitgeber-Baustein

In vielen Anwendungsfällen ist die Einhaltung bestimmter Zeitforderungen notwendig. So muß z. B. beim Ausbleiben von Quittungssignalen bei der Ein- oder Ausgabe nach Ablauf einer Wartezeit eine Unterbrechung der Warteschleife erfolgen, um den Mikrorechner nicht auf unbestimmte Zeit zu blockieren. Bei Prozeßsteuerungen müssen oft in bestimmten Zeitabständen Eingabekanäle abgefragt werden, oder es müssen über Ausgabekanäle Informationen ausgesendet werden. Derartige Zeitbedingungen können zwar von der CPU durch Abarbeitung spezieller Zeitprogramme realisiert werden, jedoch steht während dieser Zeit die CPU für die eigentlichen Nutzerprogramme nicht zur Verfügung, und außerdem ist die Einhaltung exakter Zeitbedingungen schwer möglich oder unmöglich, da die Nutzerprogramm-Laufzeit berücksichtigt werden muß, oft aber diese Laufzeit von Eingabewerten abhängt und damit variabel und nicht vorher bekannt ist.

Deshalb wurden spezielle programmierbare Zähler-Zeitgeber-Bausteine (Timer) entwickelt, die von der CPU durch Steuerworte programmiert werden, dann unabhängig von der CPU entsprechend einem angeschalteten Takt zählen, bis der vorgegebene Endwert erreicht und damit die vorgegebene Zeit vergangen ist. Das Erreichen des Endwertes löst eine Unterbrechung aus. Meist sind diese Bausteine auch als Zähler programmierbar. Anstelle des Taktes werden die zu zählenden Impulse angelegt, der aktuelle Zählerstand kann jederzeit durch die CPU über Eingabebefehl gelesen werden, bei Erreichen eines vorgegebenen Endwertes wird ebenfalls eine Unterbrechung ausgelöst.

Als Beispiel soll hier der Zähler-Zeitgeber-Baustein Z80-CTC betrachtet werden, dessen Steuersignale und Arbeitsweise an die der Z80-CPU und des Z80-PIO angepaßt sind, sein Blockschaubild zeigt Bild 2.3.6a.

Die Übertragung zwischen CPU und CTC erfolgt über den bidirektionalen Datenbus, die Bausteinwahl durch $\overline{CE} = 0$. CS1 und CS0 bestimmen den verwendeten Kanal (duale Kodierung der Kanalnummer). Meist wird CS0 = A0 und CS1 = A1 verwendet und CE aus der Dekodierung der sechs höherwertigen Geräte-Adressbits erzeugt, d. h., es werden je vier Geräteadressen in Ausgabe- und Eingaberichtung belegt. Die Übertragungsvorgänge werden durch die von der CPU ausgesendeten Signale M1 = Befehlschleisyklus, IORQ = Ein/Ausgabezugriff und RD = Lesen gesteuert.

Jeder Kanal enthält ein 8-Bit-Zeitkonstantenregister, ein 8-Bit-Kanalsteuerregister, einen 8-Bit-Rückwärtszähler sowie einen 8-Bit-Vorteiler für den Systemtakt (CPU-Takt) Φ (s. Bild 2.3.6.b). Die Programmierung der CTC erfolgt durch Ausgabe von Steuerworten von der CPU an den jeweiligen Kanal. Durch das niederwertigste Bit des Steuerwortes wird bestimmt, ob es sich um den UB-Vektor (D0 = 0) oder das Kanalsteuerwort (D0 = 1) handelt, aus Bit D2 des Kanalsteuerwortes geht hervor, ob anschließend die Zeitkonstante übertragen wird.

Durch das Kanalsteuerwort werden definiert:

- UB-Freigabe des Kanals (D7 = 1)
- Betriebsart (Zeitgeber: D6 = 0, Zähler: D6 = 1)
- Vorteilerverhältnis (16: D5 = 0, 256: D5 = 1)
- Triggerflanken des CLK/TRG-Signals (negative Flanke: D4 = 0, positive Flanke: D4 = 1)
- Triggerzeitpunkt (bestimmt durch Maschinenzklus: D3 = 0, durch vorgegebene Flanke am CLK/TRG-Eingang: D3 = 1)
- Ankündigung, daß als nächstes Steuerwort die Zeitkonstante übertragen wird (D2 = 1)

- Rücksetzen des Zählers (Weiterzählen: $D1 = 0$, Anhalten des Zählers bis zur Eingabe einer Zeitkonstante: $D1 = 1$).

Die Festlegungen durch Bit 5, 4, 3 sind nur in der Betriebsart „Zeitgeber“ von Bedeutung.

Bei Annahme einer vom CTC ausgesendeten UB-Forderung ($\overline{INT} = 0$) werden von der CPU $I/O\overline{RQ} = 1$ und $M1 = 1$ ausgesendet. Damit wird veranlaßt, daß der CTC den UB-Vektor auf den Datenbus schaltet. Dieser Vektor muß vorher als spezielles Steuerwort ($D0 = 0$) zum Kanal 0 übertragen worden sein. Der CTC modifiziert diesen UB-Vektor, indem er die Kodierung des unterbrechenden Kanals in $D2, D1$ einsetzt und so für jeden der Kanäle einen unterschiedlichen UB-Vektor erzeugt.

In der UB-Betriebsart 2 der Z80-CPU sind die Anfangsadressen der UB-Routinen in einer Tabelle im Speicher festgelegt. Die jeweilige Tabellenadresse bildet die CPU aus ihrem I-Register (höherwertiges Byte) und dem vom CTC gelieferten UB-Vektor (niederwertiges Byte). Die Anschlüsse IEI und IEO dienen wie auch beim PIO der hardwaremäßigen Prioritätsfestlegung mehrerer CTC oder PIO nach dem Daisy-Chain-Prinzip. Innerhalb des CTC hat Kanal 0 die höchste, Kanal 3 die niedrigste Priorität.

In der Betriebsart „Zeitgeber“ wird der Rückwärtszähler automatisch mit dem Wert des Zeitkonstantenregisters geladen und je nach Kanalsteuerwort mit 1/16 oder 1/256 der System-Taktfrequenz heruntergezählt, dabei kann der Verteiler durch die positive oder negative Flanke des Signals CLK/TRG getriggert werden.

In der Betriebsart „Zähler“ wird der Rückwärtszähler von jedem Impuls am CLK-TRG-Eingang um 1 erniedrigt. Hat der Rückwärtszähler den Wert Null erreicht, werden das Signal $ZC/TO = 1$ und eine Unterbrechungsforderung ($\overline{INT} = 0$) erzeugt, wenn im Kanalsteuerwort das UB-Freigabe-Bit gesetzt war. Danach wird der Rückwärtszähler automatisch mit dem Wert des Zeitkonstantenregisters nachgeladen.

In der Betriebsart „Zeitgeber“ ist demzufolge der Zeitraum zwischen zwei aufeinanderfolgenden Unterbrechungen gleich $(\text{Taktperiode}) \times (16 \text{ oder } 256) \times (\text{Inhalt des Zeitkonstantensteuerregisters})$.

In der Betriebsart „Zähler“ wird eine Unterbrechungsforderung erzeugt, wenn die Zahl der Impulse am CLK/TRG-Eingang gleich dem Inhalt des Zeitkonstantenregisters geworden ist. Unabhängig davon kann der Inhalt der Rückwärts-

zähler des CTC jederzeit von der CPU durch Eingabebefehl gelesen werden.

2.3.3.2. DMA-Bausteine

Die Übertragung größerer Informationseinheiten (Datenblöcke) zwischen Arbeitsspeicher eines Mikrorechners und schnellen E/A-Geräten (z. B. Plattenspeicher) durch normale Ein- und Ausgabeoperationen, die von der CPU programmgemäß abgearbeitet werden, ist relativ langsam. Für den Transport eines Bytes muß jeweils eine Programmschleife durchlaufen werden, dafür wird oft mehr Zeit als für den eigentlichen Datentransport benötigt. Dabei ist der Algorithmus sehr einfach, da der Transport aufeinanderfolgender Bytes von oder zu aufeinanderfolgenden Speicherplätzen erfolgt. Für derartige Fälle hat sich deshalb eine spezielle Arbeitsweise durchgesetzt, der DMA-Betrieb (engl.: direct memory access = direkter Speicherzugriff). Hierbei wird einer speziellen DMA-Steuereinheit von der CPU die Anfangsadresse des zu ladenden bzw. auszugebenden Speicherbereiches sowie die Zahl der zu übertragenden Bytes (Blocklänge) mitgeteilt. Sobald das schnelle E/A-Gerät sende- oder empfangsbereit ist, übernimmt die DMA-Steuereinheit die Kontrolle über Daten-, Adreß- und Steuerbus, während die CPU abgeschaltet wird und in den Wartezustand übergeht (HOLDA bei 8080-CPU, BUSAK bei Z80-CPU). Die DMA-Steuereinheit bildet die aufeinanderfolgenden Speicheradressen und Steuersignale für die Übertragung und beendet ihre Tätigkeit, wenn die vorgegebene Zahl von Bytes übertragen wurde. Danach übernimmt die CPU wieder die Steuerung.

Der gegenwärtige Entwicklungsstand erlaubt die Herstellung derartiger DMA-Steuereinheiten auf einem Chip. Als Beispiel soll hier die „Programmable DMA Controller“ 8257 von Intel betrachtet werden, sein Blockschaltbild zeigt Bild 2.3.7.

Der Baustein ist für vier DMA-Kanäle ausgelegt, jedes der vier anschließbaren Geräte kann eine DMA-Forderung ($DRQ = \text{DMA-Request}$) stellen und erhält über die DMA-Bestätigung ($DACK = \text{DMA-Acknowledge}$) mitgeteilt, daß seine Forderung akzeptiert wurde. Liegen gleichzeitig mehrere Forderungen vor, so wird durch Prioritätsentscheidung festgelegt, welche Forderung angenommen wird. Es sind zwei Arbeitsweisen möglich:

- feste Priorität (Kanal 0 höchste, Kanal 3 niedrigste)
- rotierende Priorität (zyklische Vertauschung gegenüber fester Priorität, wobei der vorher bediente Kanal die niedrigste Priorität erhält).

Für jeden Kanal sind ein 16-Bit-Adreßregister und ein 16-Bit-Byte-Zählregister (von diesem stehen aber nur 14 bits für die Zählung zur Verfügung, während die zwei höchstwertigen Bits für Betriebsartfestlegungen verwendet werden) integriert. Nach jedem DMA-Zyklus, d. h. nach der Übertragung eines Bytes im DMA-Betrieb, wird der Inhalt des Adreßregisters um 1 erhöht, damit wird die nächste Speicheradresse gebildet und der Byte-Zähler um 1 erniedrigt. Wenn der Byte-Zähler auf 0 steht, wird noch eine Übertragung durchgeführt, und durch $TC = 1$ wird mitgeteilt, daß die Blockübertragung abgeschlossen ist.

Die beiden höchstwertigen Bits des Byte-Zählers bestimmen die grundlegende Betriebsart:

- DMA-Schreibzyklus (Übertragung vom Eingabegerät zum Speicher, erzeugte Steuersignale $I/\overline{OR}$ und $\overline{MEMW}$)
- DMA-Lesezyklus (Übertragung vom Speicher zum Ausgabegerät, erzeugte Steuersignale $I/\overline{OW}$ und $\overline{MEMR}$)
- DMA-Prüfzyklus (keine echte Übertragung, es werden keine Steuersignale erzeugt, dient zur Steuerung interner Abläufe im E/A-Gerät).

Die Programmierung, d. h. das Laden der 16-Bit-Register und des 8-Bit-Betriebsartregisters, erfolgt durch Ausgabe von der CPU zum 8257. Ebenso ist das Lesen der 16-Bit-Register und eines 8-Bit-Statusregisters durch Eingabe zur CPU möglich. Bei den 16-Bit-Registern müssen die Zugriffe unbedingt paarweise erfolgen, da zuerst das niederwertige und danach das höherwertige Byte übertragen werden.

Die Adressierung der jeweiligen Register erfolgt über $A0, A1, A2, A3$ bis $\overline{CS} = 0$, mit $I/\overline{OR} = 1$ werden die Register gelesen, bei $I/\overline{OW} = 1$ geladen.

Wird fortgesetzt

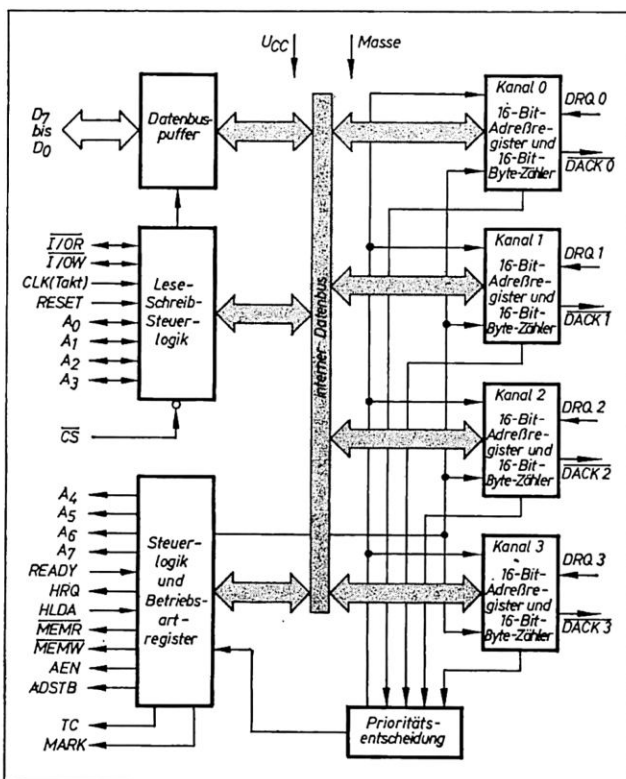
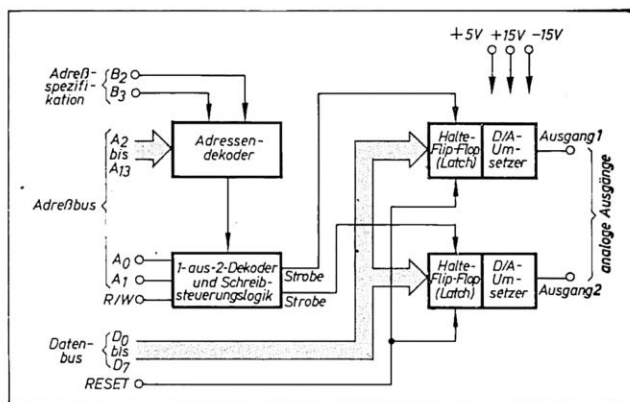


Bild 2.3.7.: Blockschaltbild des DMA-Bausteins 8257



Literatur

- [2.3.1.] Baumann, W.; Kurtze, P.; Naumann, G.: Standardinterfaces der elektronischen Meßtechnik. Reihe Automatisierungstechnik Bd. 165. Berlin: VEB Verlag Technik 1974
- [2.3.2.] Hammer, D.: Interruptverarbeitung bei Prozeßsteuerungen. Elektronik 26 (1976) H. 5, S. 69-74

2.4. Mikrorechner-Konzepte

Bezugnehmend auf den Abschnitt 1.4. sollen im folgenden Teil einige Mikrorechner bezüglich ihres strukturellen und schaltungstechnischen Aufbaus untersucht werden. Prinzipiell ergeben sich zur Zeit allgemein zwei Anwendungslinien bei der Nutzung von Mikrorechnern.

Eine Anwendungslinie ist die Nutzung von „Standard“-Mikrorechnern mit entsprechend ausgerüsteten Anschlußsteuereinheiten, die sowohl Standardinterfaces (z. B. SIF daro 1000, SI 1.2, SI/V24) als auch den direkten Anschluß von Peripheriegeräten (z. B. Display, Floppy disk) realisieren. Diese Nutzungsvariante belastet den Anwender nur minimal mit Fragen der Hardware der Mikrorechner, da er im Prinzip nur die hardwaremäßige Interface-Anpassung realisieren muß. Der Rechner ist mit seiner modularen erweiterungsfähigen Hardware bereits vom Hersteller vorgegeben.

Gerade bei Systemen, die mit den in der Labor- und Meßgerätetechnik sowie den in der Automatisierungstechnik üblichen Interfaces ausgerüstet sind, ist nur eine entsprechende, auf den konkreten Einsatzfall zugeschnittene Software-Realisierung notwendig.

Diese Möglichkeit der Nutzung eines Rechners (z. B. des K 1510 von Robotron, der ein Bestandteil des Mikrorechner-systems MRS K 1510 ist), besteht dort, wo Spezialanwendungen mit geringen Stückzahlen vorhanden sind, im Musterbau oder dort, wo vorerst Einsatzmöglichkeiten untersucht werden.

Obwohl bei vielen Mikrorechnern, insbesondere bei den modularen und erweiterungsfähigen Baukastensystemen, keine scharfe Grenze zwischen universellen und speziell auf den Prozeß zugeschnittenen Mikrorechnern gezogen werden kann, erweist sich die zweite Möglichkeit als effektiv.

Hier bieten die sogenannten Einbaumikrorechner, die in abgerüsteter Form oftmals ohne oder mit nur geringen Kommunikationsmöglichkeiten direkt in die zu steuernde Anordnung eingebaut sind, günstige ökonomische Parameter.

Allgemein kann festgestellt werden, daß sich eine nichtgenutzte Redundanz im Mikrorechner bereits bei Stückzahlen über 100 als unrentabel erweist.

Diese total abgerüsteten, spezifisch auf ein bestimmtes Steuerungsproblem zugeschnittenen, oftmals fast an kundenspezifische Entwicklungen (Zündzeitpunkteinstellung,

Tafel 2.3.2.: Ausgewählte Kennwerte des MP 10

Anzahl der analogen Ausgänge	2
Ausgangsspannungsbereich	± 10 V
Ausgangs impedanz	1 Ω
Ausgangs-Entscheidungszeit (Wandlerzeit)	25 μ s
niedrigste Auflösung (1 LSB)	78,1 mV
Durchsatzgenauigkeit	$\pm 0,4$ „FSR“)
Temperaturkoeffizient der Genauigkeit	$\pm 0,008$ FSR) K
Arbeitstemperaturbereich	0...70 °C
Lagertemperaturbereich	-55...85 °C

1) FSR $\triangleq$ voller Wertebereich (= 20 V)

Bild 2.3.9. (links): Blockschaltbild des D-A-Umsetzers MP 10

- [2.3.3.] Meiling, W.: Einige Tendenzen in der Entwicklung und Anwendung von Mikroprozessoren und Mikrorechnern. Nachrichtentechnik - Elektronik 27 (1977) H. 1, S. 7-10
- [2.3.4.] Wakerly, I. F.: Microprocessor input/output architecture. Computer (1977) February, S. 26-33
- [2.3.5.] Schlechter, J.: Mikroprozessoren und Prozeßanpassung - Aspekte des effektiven Einsatzes von Mikroprozessoren. Vortrag auf der 15. Halbleitertagung, Gera 1979

Waschmaschinensteuerung usw.) heranreichenden Lösungen werden als an Entwicklungssystemen getestete Einheiten erstellt.

In den folgenden Abschnitten sollen einige dieser Möglichkeiten behandelt werden.

2.4.1. Mikrorechner K 1510

Der Mikrorechner K 1510, der die gerätetechnische Hauptkomponente des Mikrorechnersystems K 1510 darstellt, ist ein Erzeugnis des VEB Robotron-Elektronik Zella-Mehlis. Dieser Mikrorechner, der aus einem Sortiment von Funktionseinheiten auf EGS-Steckeinheiten (Leiterkartenformat 135 mm $\times$ 170 mm, 90poliger direkter Steckverbinder) besteht, gestattet einen modularen Aufbau und eine optimale Anpassung an den jeweiligen Einsatzfall (Bild 2.4.1.). Er kann sowohl für Standardanwendungen bei mittleren Anforderungen an die Arbeitsgeschwindigkeit und die Speicherkapazität Verwendung finden, kann aber auch, ausgerüstet mit der entsprechenden Peripherie, als einfaches Entwicklungssystem genutzt werden.

Die wesentlichen Leistungsmerkmale dieses Mikrorechners werden durch den Mikroprozessor U 808 D bestimmt.

- Die ZVE (Bild 2.4.2.), die insgesamt auf zwei Steckeinheiten realisiert ist, enthält neben dem U 808 D weitere Funktionsblöcke, wie z. B. den quarzstabilisierten 480-kHz-Taktgenerator, Anpaßschaltungen, Auffangregister zur Speicherung der zeitmultiplex vom U 808 D ausgegebenen Information, die gesamte TTL-ZVE-Steuerung, eine Interrupt-Steuerung sowie Schaltungen zur Adreßdekodierung der Speicher bzw. E-A-Adressen und die Adreßerweiterung zur Verdopplung der Zahl der E-A-Adressen.

- Der ZVE-Zusatz (eine Steckeinheit) bringt eine Erweiterung bezüglich der Möglichkeiten der ZVE-Interruptorganisation.

Durch den zusätzlichen, auf dieser Steckeinheit realisierten Kellerspeicher mit einer Speicherkapazität von 32 Byte werden bei Unterbrechung der Inhalt des Befehlszählers, des Akkumulators, eines Universalregisters und die Bedingungsbits in den Keller gerettet. Vom Programm können anschließend andere Universalregister in den Operativspeicher (RAM) übertragen werden.

Nach Beendigung der Interrupt-Operation erfolgt die Auskellerung in umgekehrter Reihenfolge.

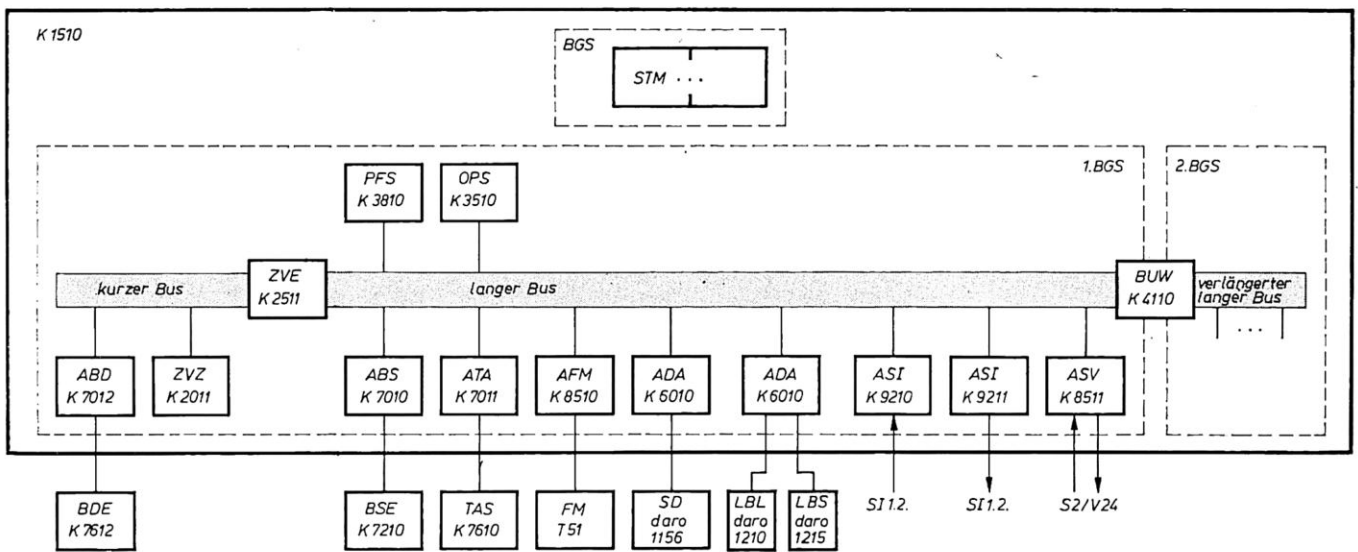


Bild 2.4.1.:
Struktur des K 1510

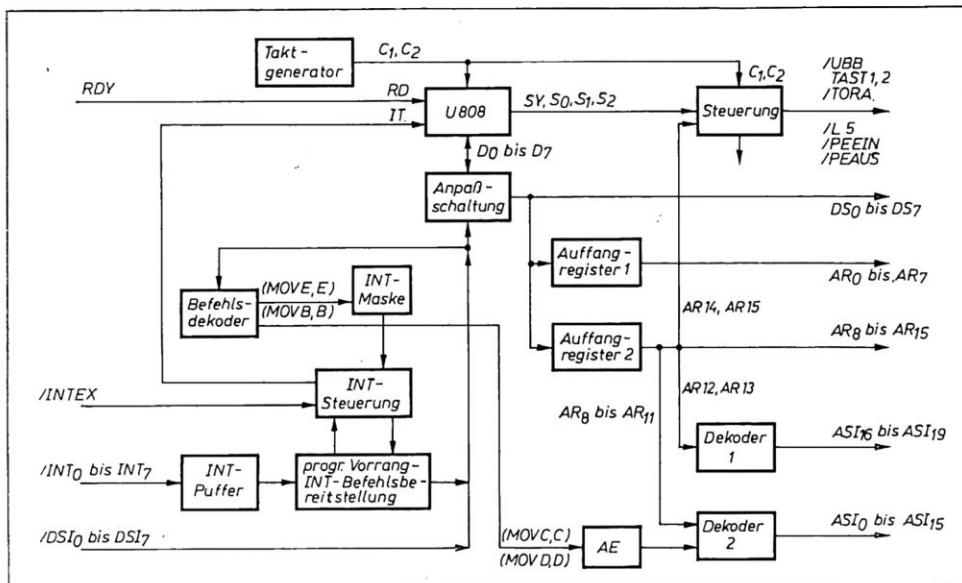


Bild 2.4.2.:
Struktur der ZVE des K 1510

Erläuterungen zum Bild 2.4.1.

ZVE	Zentrale Verarbeitungseinheit
ZVZ	ZVE-Zusatz
PFS	Programmierbarer Festwertspeicher
OPS	Operativspeicher (hier RAM)
ABD	Anschlußsteuerung Bedieneinheit
BDE	Bedieneinheit
ABS	Anschlußsteuerung Bildschirmeinheit
BSE	Bildschirmeinheit
ATA	Anschlußsteuerung für alphanumerische Tastatur
TAS	alphanumerische Tastatur
AFM	Anschlußsteuerung Fernschreibmaschine
FSM	Fernschreibmaschine
ASI K 9210	Anschlußsteuerung SI 1.2 Eingang
ASI K 9211	Anschlußsteuerung SI 1.2 Ausgang
ASV	Anschlußsteuerung mit Interface S2/V24
ADA	Anschlußsteuerung SIF daro 1000
SD	Seriendrucker (hier: daro 1156 Typ ASC 2)
LBL	Lochbandleser (hier: daro 1210)
LBS	Lochbandstanzer (hier: daro 1215)
BUW	Busweiterleitung (für große Konfigurationen)
BGS	Baugruppeneinsatz
STM	Stromversorgungsmodul

Erläuterungen zum Bild 2.4.2.

DS0 bis DS7	Auswärtsdaten ungepuffert für Schreibspeicher
AR0 bis AR7	Auswärtsdaten gepuffert für Anschlußsteuerungen (AS) und Adressen für Speicher
AR8 bis AR15	Adressen und Zyklusnummer kodiert
/DS10 bis /DS17	Einwärtsdaten von AS und Speicher (negiert)
/ASI0 bis /ASI15	Dekodierte Adreßsignale (1 aus 16) für AS und Speicher (negiert)
ASI16 bis ASI19	Adreßgruppensignale (1 aus 4) für AS und Speicher
/PEAUS	Übernahmesignal für die Auswärtsdaten in der AS (negiert)
/PEEIN	Übernahmesignal für die AS-Daten nach dem Einwärtsdatenbus (negiert)
L, S	Lese- und Schreibsignal für den Speicher
TAST 1	Steuersignal für Anschlußsteuerungen
TAST 2	Tastsignal für die Speicherbetriebsspannung
RDY	Haltsignal, führt zum Wartezustand
/INTX	Interrupt-Steuersignal (negiert)
/INT0 bis /INT7	Interrupt-Signale (negiert)
/UBB	Betriebsbereitschaft (negiert), Löschesignal nach dem Einschalten
/TORA	Adreß-Torungssignal für PFS (negiert)

Der Halbleiterspeicher des K 1510 kann bis zu einer maximalen Kapazität von 16 Byte, bedingt durch die Adressierfähigkeit des U 808 D, je nach Erfordernis aus ROM- und RAM-Steckeinheiten zusammengesetzt werden, statt einfacher ROM können auch PROM oder REPRON verwendet werden.

- Die Festwertspeichereinheit enthält auf einer Leiterkarte 24polige Fassungen, so daß maximal 16 ROM mit der Organisationsform 256×8 bit aufgesteckt werden können. Die Maximalkapazität der ROM-Steckeinheit beträgt demnach 4 K Byte.

- Der Operativspeicher enthält je Steckeinheit 0,75 Byte RAM-Kapazität (24 Schaltkreise CM 8001).
- Außerdem kann der K 1510 mit entsprechenden Anschlußsteuerungen für periphere Geräte und Standard-Interface-Realisierungen ausgerüstet werden. Dazu zählen jeweils die Anschlußsteuerungen für die speziell für den K 1510 neuentwickelten Anschlußgeräte – Bedieneinheit K 7612 – Bildschirmeinheit K 7210 (31 cm Bildschirm, 8 Zeilen mit je 32 Zeichen, Zeichenvorrat: 64 alphanumerische und Sonderzeichen)

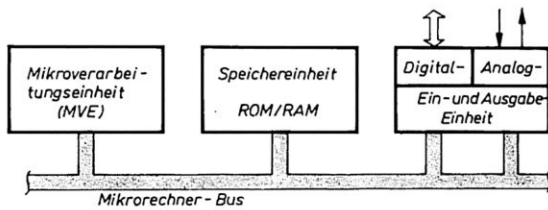


Bild 2.4.3.: Struktur eines Einbau-Mikrorechners (für Prozeßsteuerung)

- alphanumerische Tastatur K 7610 (alphanumerische Zeichen und programmierbare Funktionstasten)
- die Anschlußsteuereinheiten für die Mikrorechner-Standardperipherie wie
 - Fernschreibmaschine T 51 (Anschlußbild S 1, 5-Bit-Kode, 50 Baud, wahlweiser Zweidraht- oder Vierdrahtbetrieb)
 - SIF daro 1000 Peripherie mit KME-3-Pegel (Lochbandstanzer daro 1215, Lochbandleser daro 1210, oder dem Seriendrucker daro 1156)
- und die Anschlußsteuerungen für die Umsetzung in Standardschnittstellen wie
 - die Schnittstelle S 2 (V 24 entsprechend der CCITT-Empfehlung) zur Punkt-zu-Punkt-Verbindung im ISO-7-Bit-Kode mit einer wählbaren Übertragungsrate von 200, 600, 1 200 Baud. (Modemanschluß zur Fernübertragung ist möglich)
 - das SI 1.2, das im wesentlichen für den Anschluß von Labor- und Meßgeräten, die mit diesem parallelen TTL-Interface ausgerüstet sind, gedacht ist.

Im folgenden soll das Standard-Interface SI 1.2 näher untersucht werden, da der Nutzer eines K 1510 gerade mit den beiden Anschlußsteuereinheiten ASI K 9210 (Eingabe) und ASI K 9211 (Ausgabe) ein TTL-Interface zur Verfügung gestellt bekommt, mit dem große Informationsmengen parallel verarbeitet werden können.

Im Normalfall sind diese ASI-Einheiten für eine Kopplung mit der Peripherie nach den logischen und elektrischen Bedingungen des Standard-Interface SI 1.2 (nach TGL 29 248) vorgesehen, sie lassen sich aber auch für eine Digitalwert-ein- und -ausgabe ohne diese Interface-Bedingungen einsetzen.

Das SI 1.2 ist ein TTL-Standard-Interface für den Kettenverkehr von Funktionseinheiten zur Gewinnung, Aufbereitung, Verarbeitung, Ausgabe und Speicherung von Informationen, das sich auf die Festlegung der informationellen Wechselbeziehungen beschränkt. Die logischen Bedingungen geben eine Aussage über die zwischen zusammengeschalteten Funktionseinheiten auszutauschenden Signale und ihre Wirkung. Dabei sind im SI 1.2 folgende Signalgruppen festgelegt:

- Steuersignale (Befehls- und Meldesignale)
- Informationssignale
- Programmsignale.

Werden beim K-1510-Einsatz die ASI-Einheiten (Tafel 2.4.1.) genutzt, ist folgendes zu beachten:

Je nach Erfordernis des Nutzers können das Interrupt-System und das RDY-Signal (Ready) in das Ein- und Ausgabeprogramm einbezogen werden, was durch Brücken auf den entsprechenden Steckeinheiten festgelegt wird.

Im allgemeinen entscheidet man sich jedoch entweder für die Arbeitsweise mit Interrupt oder mit dem RDY-Signal. Die Nutzung der Interrupt-Möglichkeit, wobei für jede ASI das Interrupt markierbar ist, gestattet einen zeitoptimalen Informationsaustausch.

Bei der Verwendung der Synchronisationsmöglichkeit mit dem RDY-Signal geht das Programm in den Wartezustand,

- wenn an die ASI K 9210 ein Eingabebefehl gegeben wird, ohne daß eine Eingabemeldung vom Gerät vorliegt
- wenn an die ASI K 9211 ein Ausgabebefehl gegeben wird, obwohl von dieser noch Ausgabemeldung anliegt.

Bei einer Nutzung der ASI-Steuerungen ohne SI 1.2, so wird vom Hersteller angegeben [2.4.3.], ist folgendes zu beachten:

Tafel 2.4.1.: Kennwerte der ASI Einheiten K 9210, K 9211

	ASI K 9210	ASI K 9211
Kanäle je Steckeinheit	1 Eingabekanal gepuffert (auf 2 Kanäle aufteilbar)	1 Ausgabekanal gepuffert
Übertragungsbreite (zur Peripherie)	max. 56 Eingabeinformationen (können parallel vom Empfangspuffer aufgenommen und vom Programm byteweise nacheinander abgefragt werden)	max. 48 Ausgabeinformationen (können vom Programm byteweise in den Ausgabepuffer gebracht und parallel ausgegeben werden)
Übertragungsentfernung	max. 20 m	max. 20 m
Übertragungsgeschwindigkeit		
8 Bit	5 kHz	5 kHz
24 Bit	1,9 kHz	1,9 kHz
48 Bit	1,37 kHz	1,37 kHz
56 Bit	1,2 kHz	
Steuersignale:		
Meldesignale von der ASI	M1 Bereitschaft M0 z. B. als Löschesignal M4 zur Eingabeforderung oder Fehlermeldung	M2 Ausgabeanmeldung M0 z. B. als Löschesignal M4 Fehlermeldesignal o. ä.
Befehlssignale von Gerät	B2 Eingabeanmeldung B4 z. B. Fehleranmeldung oder Eingabemeldung bei 2 Kanälen	B1 Ausgabequittierung B4 in Verbindung mit B1 z. B. zur Fehleranmeldung
Belastung der Eingangsleitungen in Lasteinheiten	Daten 1 B2 2 B4 2	B1 2 B4 2
Belastbarkeit der Ausgangsleitungen durch das Gerät in Lasteinheiten	M1 30 M0 30 M4 30	10 M2 30 M0 10 M4 10
Pegel für Informations- und Steuerleitungen	TTL	TTL
Signalbewertung Informationsleitungen	H	H
Steuerleitungen	M1 L M0 L M4 L B2 H-L B4 H-L	M2 L M2 L M4 L B1 H-L B4 H-L

Die ASI K 9210 ist befehlsmäßig auf diese Betriebsart einzustellen. Der Inhalt der Eingabeleitungen wird dann automatisch etwa aller $2 \mu s$ in den Empfangspuffer überschrieben.

Während der Abfrage des Eingangspuffers vom Rechner bleibt die Datenübernahme bis nach Abfrage des siebenten Bytes oder der Ausgabe des Befehls OUT 16 H mit A = 00100100 (Steuerung des Eingabepuffers in den Grundzustand) gesperrt.

Da im Gegensatz zur Betriebsweise nach SI 1.2 die Datenübernahme im Schaltzustand der Eingabeleitungen erfolgen kann, ist eine senderseitige Anpassung der Informationsleitungen zweckmäßigerweise vom Anwender vorzunehmen.

Bei der ASI K 9211 können jederzeit beliebige Bytes auf den Ausgabeleitungen softwaremäßig verändert werden.

2.4.2. Entwurf eines Einbaumikrorechners

In diesem Abschnitt soll ein Einbau-Mikrorechner, der für Prozeßsteuerungsaufgaben vorgesehen ist, in Bezug auf die Hardware-Entwicklungsschritte betrachtet werden. Aufbauend auf den Abschnitten 1.3., 2.1., 2.2. und 2.3. soll die Entwicklung an Hand des vom Bauelementeaufwand her überschaubaren Systems 8080 erfolgen. Es wird im folgenden nicht mehr auf die Schaltkreise und Baugruppen, sondern auf ihr prinzipielles Zusammenwirken eingegangen. Der Mikrorechner wird nach dem im Bild 2.4.3. gezeigten Strukturprinzip realisiert.

Wird fortgesetzt

2.4.2.1. Mikroverarbeitungseinheit (MVE)

Die wesentlichste Aufgabe dieser Einheit besteht darin, neben den gesamten Verarbeitungsaufgaben die Steuerung des Mikrorechners zu übernehmen, d. h., unter anderem das Bussystem mit den entsprechenden Signalen zu versorgen und Bussignale auszuwerten.

Die MVE enthält als wesentliche Komponente den Mikroprozessor und dort, wo dieser nicht die gesamte Systemsteuerung übernehmen kann, weitere LSI-, MSI-, SSI-Schaltkreise.

Der Mikroprozessor 8080 benötigt zwei weitere Schaltkreise, den Taktreiber 8224 und die Systemsteuerung 8228, um die für den Mikrorechner notwendigen Busanschlüsse zu realisieren.

Wird der Adreßbus des Mikroprozessors mit mehr als einer TTL-Lasteinheit durch Speicher- oder E-A-Schaltkreise belastet, so ist eine Abpufferung über zwei Schaltkreise 8212, die als Treiber geschaltet werden, notwendig (s. Bild 2.4.4.).

einem beliebigen Zeitpunkt asynchron eintreffen) mit langsamen peripheren Geräten.

Es erfolgt die Einführung von Wartezuständen entsprechend den externen Zugriffszeiten.

2. Einfügen genau eines Wartezustandes T_W bei jedem Maschinenzklus (günstig bei solchen Anwendungen, wenn z. B. die PROM-Zugriffszeit einen Wartezustand erfordert, aber keine extra Wartesteuerung aufgebaut werden soll).

● Tristate-Schaltung

Bei der Schaltung des Busses in den tristabilen Zustand ist zu beachten, daß sowohl der Adreß- und Datenbus als auch wesentliche Steuersignale einbezogen werden.

Im Bild 2.4.5.c wird eine Schaltungsvariante vorgestellt. Das Systembus-Enable-Signal (BUSEN) kann bei Brückung 1 die Tristate-Schaltung bei H-Belegung hervorrufen. Brücke 2 garantiert eine ständige Busbefähigung.

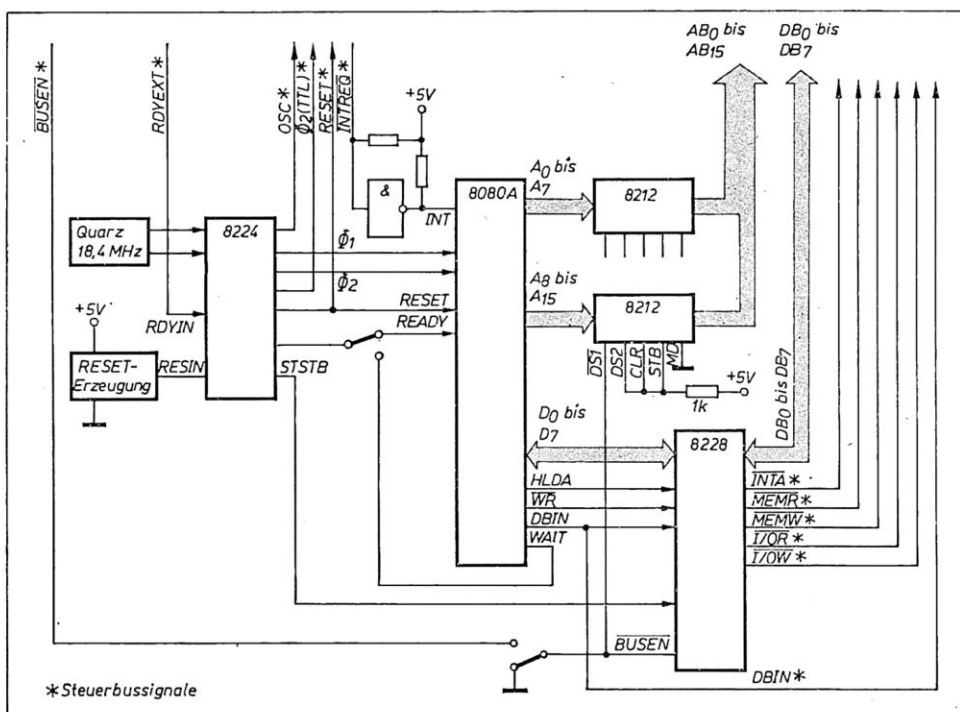


Bild 2.4.4.: Prinzipschaltbild der MVE

Beim Aufbau sind spezielle Schaltungsbesonderheiten zu beachten:

● RESET-Erzeugung

Die RESET-Erzeugung (Bild 2.4.5.a) ist für zwei Aufgaben zu konzipieren:

1. Anfangsrückstellen (abgeleitet vom Anstieg der Versorgungsspannung +5V wird das RESIN-Signal gebildet)
2. Rückstellen bei Betätigung des RESET-Tasters

● READY-Steuerung

Für die Realisierung der READY-Steuerung ergeben sich zwei Möglichkeiten (die durch entsprechende Brückung ausgewählt werden können) (Bild 2.4.5.b).

1. READY-Synchronisation über das RDYIN-Signal des 8224 (das RDYEXT – externes READY – kann zu

● Interrupt-Nutzung (Minimalauslegung ohne zusätzliche Interrupt-Steuerung)

Wird die im Bild 2.4.5.d gezeigte Beschaltung vorgenommen (INTA-Ausgang des 8228 über 1 kΩ an +12 V), so wird bei einer Interrupt-Auslösung (INTREQ = low) automatisch ein RESTART 7 veranlaßt. Der entsprechende Initialisierungsbefehl für eine UB-Routine muß dann in Adresse 0038 H festgelegt sein.

Diese Interrupt-Möglichkeit in einer Ebene stellt eine effektive Lösung für Einfach-Varianten dar.

2.4.2.2. Bussystem

Das Bussystem ist ein System von informationsübertragenden Leitungen, die schaltungstechnisch oder auch nur logisch zusammengefaßt die verschiedenen Komponenten

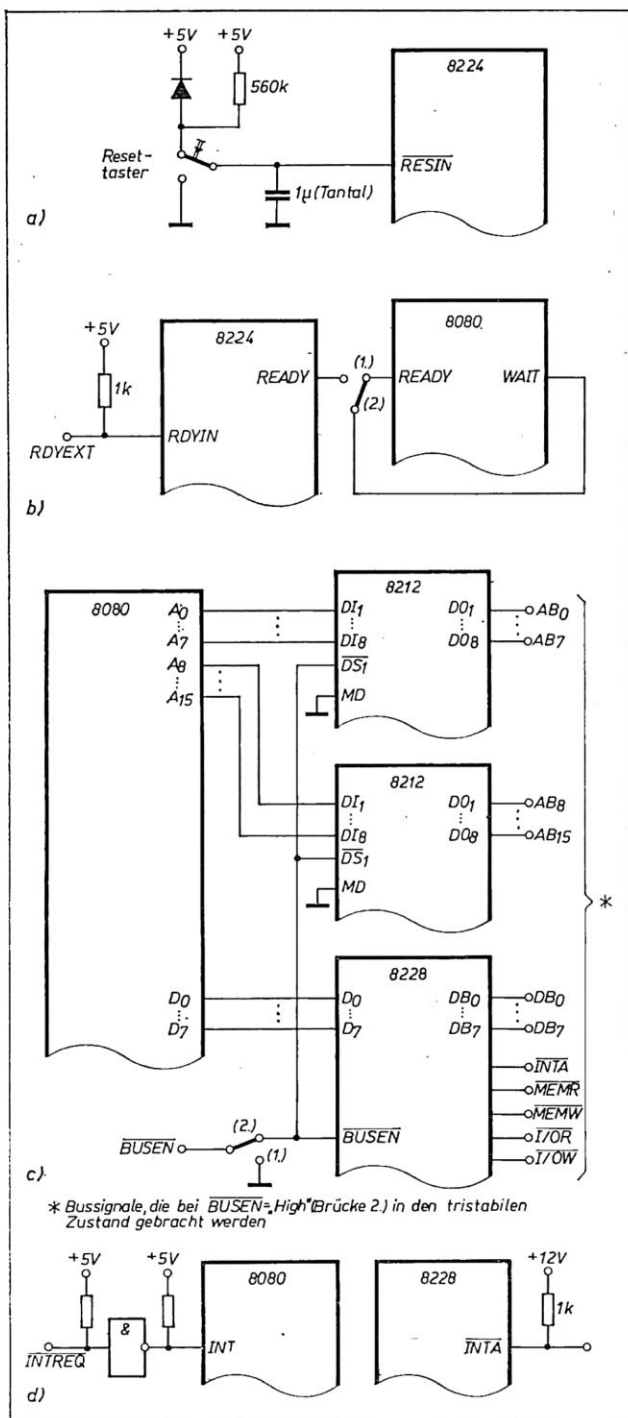


Bild 2.4.5.: Schaltungsbesonderheiten der MVE

eines Rechners so verbinden, daß dieser alle Funktionen erfüllen kann.

Die Gestaltung des Bussystems übt einen beträchtlichen Einfluß auf die Leistungsparameter des gesamten Mikrorechners aus. Man muß deshalb bereits bei der Konzipierung des Mikrorechners Kennwerte und Arbeitsprinzipien des Bussystems einbeziehen.

Besonderen Schwerpunkt bildet dabei die Garantierung der Modularität und Erweiterungsfähigkeit des gesamten Rechners.

Die Unibus-(Einheitsbus-)Struktur bietet bei Mikrorechnern optimale Möglichkeiten, dieses Prinzip durchzusetzen.

Der Unibus ist dabei ein zentraler Bus, der alle Baugruppen des Mikrorechners verbindet und garantiert, daß an jeder Stelle des Busses das Gesamtspektrum der Signale verfügbar ist. Durch die einheitliche Steckerbelegung wird es möglich, eine Rückverdrahtungsleiterplatte einzusetzen.

International realisieren Hersteller von Mikrorechnern oft ihr eigenes Bussystem, es wird aber im Zuge des Dranges zur Normung angestrebt, ein genormtes Bussystem (z. B. den

S-100-Bus [2.4.5.]) durchzusetzen. Damit steht dann natürlich sofort die Frage der Verbindungselemente (z. B. erfordert der S-100-Bus einen 100poligen Steckverbinder) im Vordergrund.

In dem hier vorgestellten konkreten Entwicklungsfall ergibt sich das im Bild 2.4.6. gezeigte Bussystem.

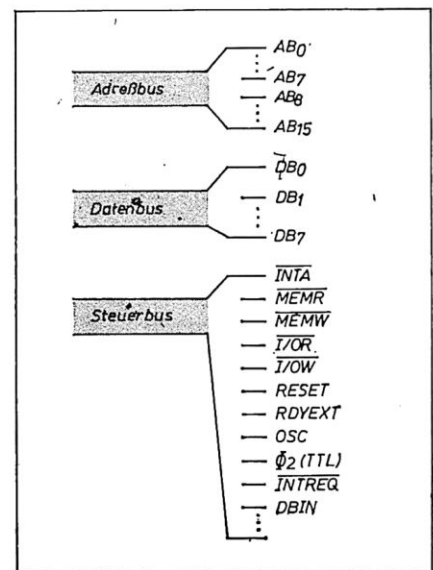


Bild 2.4.6.: Bussystem

Der 16-Bit-Adreßbus ist unidirektional, d. h., die Adreßinformationen werden ausgehend von der Lasteinheitenerhöhung (zweimal 8212) an alle Einheiten gesendet.

Der 8-Bit-Datenbus ist bidirektional. Damit wird es ermöglicht, daß eine Einheit auf die gleiche Leitung senden und von ihr empfangen kann, wenn sie eine entsprechende Eingangs- und Ausgangsstufe aufweist (s. a. Abschnitt 1.2.3.3.).

Die Steuersignale MEMR, MEMW, I/OR, I/OW, INTA, RDYEXT, Ø2 (TTL) und RESET sind allgemein ein Minimum an Steuersignalen. Sie sind unbedingt für die Datenbussynchronisation mit den angeschlossenen Einheiten notwendig. Zu bemerken ist, daß die Lese-Schreib-Steuersignale negiert geliefert werden.

Das ist notwendig, da die auslösende Signalkomponente, d. h. der Signalpegel, der eine Funktion bewirken soll, L-Pegel aufweisen muß, da ein eingangsseitig unbeschalteter TTL-Schaltkreis diesen Eingang wie mit H beschaltet interpretiert.

Weiterhin sind folgende Steuersignale auf dem Bus vorhanden: INTREQ, BUSEN, DBIN, OSC.

2.4.2.3. Speichereinheit

Die Speichereinheit, umfassend bereits im Abschnitt 2.2. erläutert, teilt sich bei einem Mikrorechner zweckmäßig in einen Anteil ROM-Kapazität und einen RAM-Kapazität (Bild 2.4.7.).

Üblich ist bei Nutzersystemen (nicht bei Entwicklungssystemen, die einen größeren RAM-Bereich besitzen) ein Verhältnis von der RAM- zur Festwertspeicher-Kapazität von 1:4 bis 1:6.

Die Speicheradressierung eines gesamten Systems hängt im wesentlichen von der Adressierfähigkeit des Mikroprozessors ab. Es ist jedoch bei der Speicher-Block- bzw. Chip-Auswahl darauf zu achten, daß immer eine vollständige Dekodierung des ausgenutzten Adreßraumes erfolgt (auch im Interesse einer fortlaufenden Adressierung und eines minimalen Aufwandes bei der Dekodierung). Dabei ergibt sich in Abhängigkeit der genutzten Adreßbits die Zuordnung nach Bild 2.4.8.

Allgemein wird die ROM-Einheit bei den Nutzeranwendungen nach der Adresse 0000H angeordnet, da der Rechner mit dem Anfangsrückstellen von dieser Adresse an sein Programm bearbeitet. Bei Entwicklungssystemen soll der Test von Nutzerprogrammen auch in diesem niedrigen Adreßbereich möglich sein. Deshalb wird hier nur in der Initialisie-

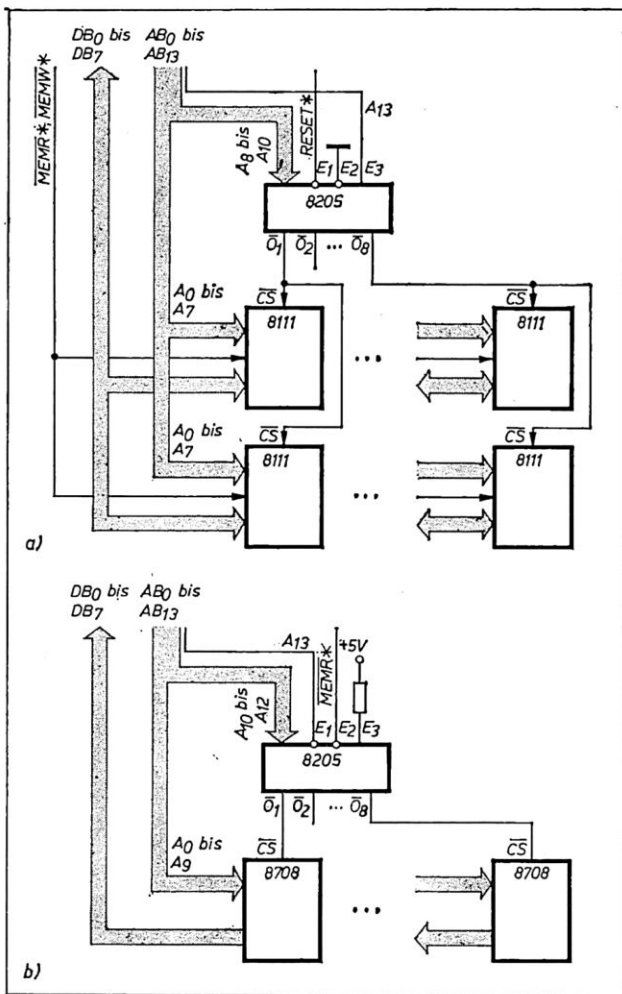


Bild 2.4.7.: Speichereinheit, a) 2-K-Byte-RAM, ab 8 K adressiert; b) 8-K-Byte-RePROM

rungsphase des Systems (zeitweilig) ein PROM mit dem Urlader (Bootstrap-Lader) verwendet. Für die eigentliche Nutzung steht in diesem Adreßbereich RAM-Kapazität zur Verfügung.

2.4.2.4. Digital- und Analog-E-A-Einheit (für Prozeßsteuerungen)

Die Digital- und Analog-E-A-Einheit nach Bild 2.4.9. unterteilt sich in einen Komplex für digitale und einen für ana-

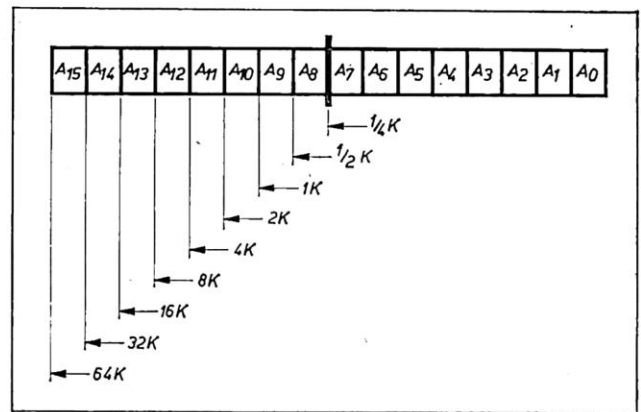


Bild 2.4.8.: Adressierfähigkeit

loge Ein- und Ausgabe. Die Digital-E-A-Einheit, die im wesentlichen aus dem im Abschnitt 2.3.2.1. beschriebenen Schaltkreis 8251 und dem Parallel-Interface-Schaltkreis 8255, der ähnlich der PIO aufgebaut ist, besteht, garantiert die Ansteuerung von seriellen und parallelen E-A-Prozessen und Geräten, die mit digitalen Formaten arbeiten.

Im folgenden sollen die Anschlußmöglichkeiten an den Bus erläutert werden. Neben den Datenbussynchronisationssignalen $I/O\overline{W}$ und I/OR ist das RESET-Signal zu beschalten, damit einmal sofort nach Anschalten der Versorgungsspannung, aber auch nach Havarien ein definierter Anfangszustand der Interface-Schaltkreise hergestellt werden kann. Der Schaltkreis 8255 muß mit zwei Adreßsignalen zur Auswahl der internen Ports (A, B, C, Steuerport) versorgt werden. Hier sind zweckmäßigerweise die niedrigsten Adreßbits A_0 und A_1 zu verwenden. Die weiteren Adreßbits stehen dann für eine Schaltkreisauswahl (über den $\overline{CS}$ -Eingang) zur Verfügung. Bei Minimalkonfigurationen (max. sechs Schaltkreise) können diese direkt genutzt werden, im anderen Fall muß je nach dem genutzten Adreßraum dieser vollständig dekodiert werden. Es ist zweckmäßig, eine Redundanz für mögliche Erweiterungen vorzusehen.

Bei der Nutzung von programmierbaren Schaltkreisen ist bereits von Anfang an zu beachten, daß eine entsprechende PROM-Kapazität zur Verfügung gestellt werden muß. Die Analog-E-A-Einheit, die als mit zum Mikrorechner zugehörig betrachtet werden soll, ist mit den im Abschnitt 2.3.3.4. erwähnten D-A- und A-D-Umsetzern MP10 und MP20 (Burr Brown) für einfache Anwendungsfälle aufgebaut.

Sie wurden ausgewählt, da sie ohne zusätzlichen Hardware-Aufwand direkt an den Mikrorechnerbus angeschaltet werden können. Diese Schaltkreise werden in ihrer Betriebs-

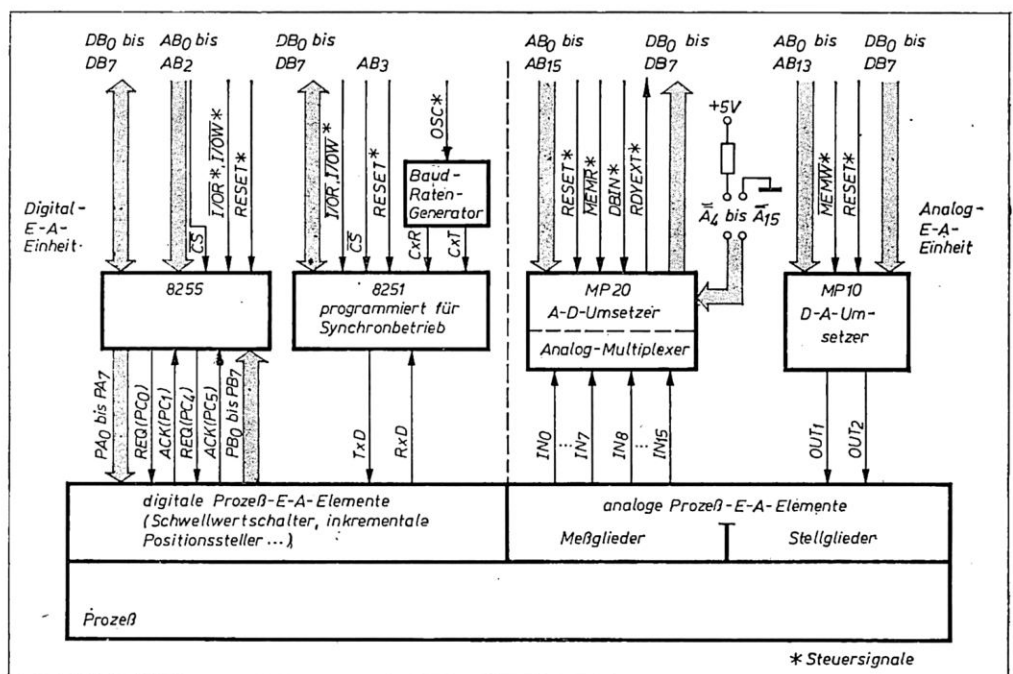


Bild 2.4.9.: Digital- und Analog-E-A-Einheit

Einführung in die GRUPPENINTEGRATION

Von Ing. Adi J. K h a m b a t a

Aus dem Amerikanischen
Deutsche Bearbeitung:
Dr.-Ing. Klaus Schade.

1. Auflage 1971.
208 Seiten, 75 Bilder,
30 Tafeln, Kunstleder,
18,00 M, Ausland 24,00 M
Bestell-Nr. 551 727 3

Von Gruppenintegration, auch Großintegration genannt, spricht man ab einem Integrationsgrad von 100 Elementarzellen (entspricht etwa 1 000 Bauelementen) je Chip. Die Einführung der Gruppenintegration ergibt folgende Vorteile: Weitere Steigerung der Zuverlässigkeit und Arbeitsgeschwindigkeit, Verringerung der Herstellungs- und Systemkosten, vorteilhaftere und zuverlässigere Systemkonzeptionen. Allerdings wirft die Gruppenintegration auch eine Reihe von Fragen auf, die besonders den Systementwurf, d. h. das Verhältnis vom Kunden zum Hersteller, betreffen.

Das Buch ist die erste geschlossene Darstellung dieser Thematik in deutscher Sprache. Es ist eine Einführung und deshalb auch für diejenigen Leser verständlich, die nicht über spezielle Kenntnisse der Halbleitertechnik verfügen.

Durch jede Buchhandlung erhältlich



VEB VERLAG TECHNIK BERLIN

phase wie Speicher angesprochen. Bei einer Ausgabe an den MP10 muß lediglich softwaremäßig die Wandlerzeit von 25 μ s beachtet werden, d. h., erst nach minimal 25 μ s darf an das gleiche Port die nächste Digitalinformation übergeben werden.

Die Synchronisation des A-D-Umsetzers MP20, der eine maximale Wandlerzeit von 40 μ s je Kanal aufweist, erfolgt mit den RDYEXT-Signal (externes READY).

Dieser Schaltkreis, der 16 einfache oder acht Differenz-Analogeingänge über einen internen Analogmultiplexer adressiert (Adreßbits A_0 bis A_3), ist an dem gesamten 16-Bit-Adreßbus angeschlossen.

Die zusätzlich an diesem 80poligen Hybridschaltkreis vorhandenen Adreßeingänge $\bar{A}_4$ bis $\bar{A}_{15}$ werden für eine wahl-

weise Blockzuordnung dieses A-D-Umsetzers jeweils fest mit L- oder H-Potential beschaltet.

Wird fortgesetzt

Literatur

- [2.4.1.] Dawidczak, S.; Weise, K. D.: Mikrorechnersystem robotron K 1510. messen steuern regeln 20 (1977) H. 12, S. 671-675
- [2.4.2.] Mikrorechner K 1510. Betriebsdokumentation. VEB Robotron Zentrum für Forschung und Technik, 1977
- [2.4.3.] Baumann, W.; Kurtze, P.; Naumann, G.: Standard-Interfaces der elektronischen Meßtechnik. Reihe Automatisierungstechnik, Band 165. Berlin: VEB Verlag Technik, 1974
- [2.4.4.] Köhler, V.: Strukturelle Merkmale von Mini- und Mikrorechnern. radio fernsehen elektronik 26 (1977) H. 7, S. 217 und 218
- [2.4.5.] Pol, B.: Der S-100-Bus - ein de-facto-Standard auf dem Mikrocomputermarkt. Elektronik 27 (1978) H. 2, S. 47-51

Dr.-Ing. WOLFGANG HERRMANN

3. Programmierungstechnik

Die Programmierungstechnik von Mikrorechnern beinhaltet zwei große Komplexe:

- Rechnerarchitektur: Eigenschaften und Spezifika der Mikrorechner vom Standpunkt des Programmierers
- Technologie der Programmierung: Prozeß der Erarbeitung von Programmen für den Mikrorechner mit den dafür typischen Methoden und Verfahren einschließlich der Rationalisierungs- und Automatisierungsmittel und der Organisationsprinzipien.

Einleitend sollen einige Schritte auf dem Weg vom Problem zum Programm herausgegriffen werden, wobei mehr informativ als streng formal vorgegangen wird.

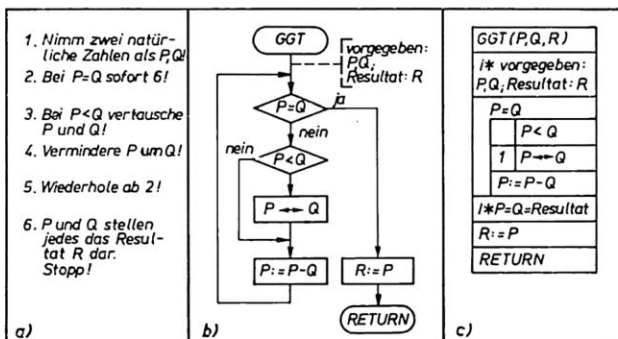
3.1. Vom Problem zum Programm

3.1.1. Algorithmen

Der Einsatz eines Mikrorechners für ein Problem der Datenverarbeitung oder Prozeßsteuerung setzt voraus, daß ein Lösungsverfahren bereits vorliegt oder unter mehreren bekannten Verfahren eines ausgewählt worden ist. Dieses Verfahren ist dann so detailliert und so präzise darzustellen, daß es die Eigenschaften eines Algorithmus erlangt [3.1.]. Ein solcher Algorithmus ist eine exakte Vorschrift

- für die Ausführung der detailliert aufgezeigten Schritte (Operationen) in einer von vornherein festliegenden (statischen) oder im Ablaufgeschehen eindeutig entscheidbaren (dynamischen) Reihenfolge (Merkmal der Determiniertheit)
- für die Erzielung einer Lösung (Resultat) nach einer endlichen Anzahl ausgeführter Operationen (Merkmal der Terminiertheit oder Endlichkeit), und zwar für jeden vom Problem umfaßten konkreten Fall, d. h. für jeden zulässigen Satz von Werten der Problemgrößen (Operanden, Argumente) (Merkmal der Universalität).

Das klassische Illustrationsbeispiel, auf das im folgenden Abschnitt mehrfach Bezug genommen wird, ist der Euklidische Algorithmus für die Ermittlung des Größten Gemeinsamen Teilers GGT(P, Q) zu jedem Paar natürlicher Zahlen P, Q als Operanden (s. Bild 3.1.1.a). Das Phänomen einer solchen Vorschrift besteht darin, daß sie für jedermann und jedes Ding auch ohne das mindeste Verständnis für den Sinn und Zweck des Resultats GGT nachvollziehbar ist, wenn nur jeder Einzelschritt ausgeführt und die Ablauffolge gewahrt bzw. entschieden werden kann. Übrigens wurde im Bild 3.1.1.a von der üblichen Vereinbarung Gebrauch ge-



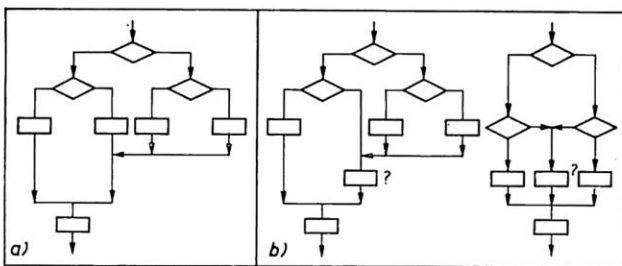


Bild 3.1.3.: Darstellungen von Verzweigungen. a) Korrekte Einmündung; b) Gegenbeispiele

Der Schlüssel zur Herausarbeitung von Grundstrukturen im Verlauf der Flußlinie liegt in der differenzierten Handhabung der Entscheidungsoperation. Folgende zwei Fälle sind möglich:

- Beide Verzweigungen münden derart in einen gemeinsamen Nachfolger, daß sie mit Verzweigungen anderer Entscheidungen höchstens ein „blankes“ Stück Flußlinie dieser Einmündung gemeinsam haben, oder sie können formal in eine solche Darstellung umgezeichnet werden (s. Bild 3.1.3.a; Gegenbeispiele s. Bild 3.1.3.b).
- Wenigstens eine der Verzweigungen ist Bestandteil einer mit Sinnbildern besetzten Schleife in der Flußlinie, und diese Schleife schneidet bei gleichem Drehsinn aller Schleifen (im Uhrzeigersinn) bzw. bei gleicher Einmündungsrichtung der Rückführung (von links) keine andere Schleife (s. Bild 3.1.4.; Gegenbeispiele s. Bild 3.1.4.b).

Vermerkt sei noch, daß in der EDV der herkömmliche Ablaufplan durch einen Datenflußplan nach dem gleichen Standard ergänzt wird; gerade dieser Apparat scheint, wenigstens z. Z., in der Mikrorechner-technik weitgehend entbehrlich zu sein.

3.1.3. Struktogramme

Die grafische Dokumentierung eines Algorithmus unter Verwendung von Sinnbildern, die die prinzipiell unterschiedlichen Verknüpfungstypen von Operationen im Ablaufgeschehen hervorheben, heißt Struktogramm.

Jeder Algorithmus ist als Struktogramm darstellbar, und zwar unter Verwendung von Sinnbildern (Strukturblöcken) für nur drei Grundstrukturen (Sequenz, Fallauswahl, Zyklus), die je genau einen Eingang und einen Ausgang haben und die beliebig hierarchisch schachtelbar sind, so daß jegliche explizite Übergänge (Sprünge) vermieden werden können. Die Anordnung der Strukturblöcke im Struktogramm erfolgt so, daß zu einem gegebenen Strukturblock

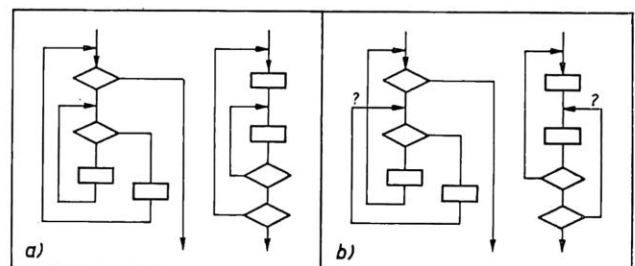
- in vertikaler Richtung Vorgänger und Nachfolger stehen (entsprechend direkt ober- und unterhalb)
- in horizontaler Richtung über- und untergeordnete Strukturblöcke (von links nach rechts) derart angegeben sind, daß der betreffende Strukturblock vollständig von seinem unmittelbar übergeordneten umfaßt wird und seinerseits den unmittelbar nachgeordneten vollständig umfaßt.

Dieses generelle Schema bestimmt auch den Ablauf vollständig. Die Arten und Unterarten der Strukturblöcke sind im Bild 3.1.5. zusammengestellt und durch äquivalente herkömmliche Ablaufpläne beschrieben (mit den A, A1, ...; B, B1, ... in der Bedeutung von Anweisungen und Bedingungen).

Erläuterungen zum Bild 3.1.5.

a) Die Sequenz ist eine Folge von Strukturblöcken, die vertikal gereiht sind, gegebenenfalls durchsetzt mit Bemerkungen. Die Glieder der Reihung besitzen den gleichen Verschachtelungsgrad und sind nicht etwa verschiedene Fälle (wie im Teilbild b) oder mit Zyklus-Austrittsbedingungen durchsetzt (wie im Teilbild c). Ein komplettes Struktogramm ist immer eine Sequenz mit einer „Identifizierung“ am Eingang (a2) und einer „spezifizierten Terminierung“ am Ausgang bzw. Ende (a3).

b) Die Fallauswahl ist eine Vorschrift für die Auswahl genau einer von n Substrukturen, genannt „Fälle“, bzw. eines „sonstigen“ Falls bei Nichtzutreffen aller dieser n Fälle. Die Fälle selbst sind durch ihre „Fallnummer“ identifiziert und



3.1.4.: Darstellung von Schleifen. a) Korrekter Schleifenschluß; b) Gegenbeispiele

daher in beliebiger vertikaler Reihenfolge angebbbar. Der Einzelfall ist dem ganzen Strukturblock untergeordnet und hat daher einen um Eins höheren Verschachtelungsgrad.

Die Auswahlvorschrift im Kopf des Strukturblocks kann als Liste von n Bedingungen notiert sein, die intern als von 1 an durchnummeriert angesehen werden (b1). Die Internummer derjenigen Bedingung, die sich bei einer Durchmusterung von links an als erste erfüllt erweist, bestimmt die Fallnummer. Bei Nichterfüllung aller n Bedingungen wird die Fallnummer mit 0 festgelegt und der „0-Fall“ gewählt. Die Wahl der Nummer eines etwa fehlenden Falles (darunter insbesondere des 0-Falles) bewirkt den sofortigen Ausgang aus dem ganzen Strukturblock.

Die Alternative ist eine Fallauswahl mit $n = 1$. Sie heißt vollständig, wenn der 0- und der 1-Fall angegeben sind (b2) und unvollständig bei Angabe nur eines von beiden Fällen (b3, b4).

Die Fallauswahl nach (b5) kann als Kürzel in Form einer Entscheidungstabelle notiert werden. Der Erfüllungsstand aller n Bedingungen wird mit den angegebenen δ -Regeln (das sind 0-1-Serien mit Leerstellen bei $\delta = 0 \vee 1$, d. h. bei Gleichgültigkeit einer Bedingung) in einem Durchmusterungsprozeß von oben nach unten verglichen. Die erste sich dabei ergebende Übereinstimmung wählt den zugehörigen Fall aus (Beispiel s. Bild 3.1.7.a). Dem 0-Fall entspricht hier ein letzter Fall mit einer leeren δ -Serie.

c) Der Zyklus ist eine Vorschrift, eine (im allgemeinen alternierende) Folge von Substrukturen A_i und Bedingungen B_i so weit abzuarbeiten und so lange zyklisch zu wiederholen, bis erstmals eine der Bedingungen erfüllt ist und den Austritt aus der Folge bewirkt (Bild 3.1.5. c1). Da bei mehr als einer Bedingung diese Bedingungen außerdem intern als von Eins an durchnummeriert angesehen werden, ist an den Austritt zwangslos noch eine Fallauswahl anschließbar, die (wenn vorhanden) als in den Strukturblock des Zyklus integriert gilt. Die Nummer der wirksamen Austrittsbedingung bestimmt einfach die Fallnummer, und der gewählte Fall fungiert als individueller Appendix des betreffenden Austritts. Danach erfolgt der endgültige Ausgang aus dem ganzen Strukturblock.

In morphologischer Hinsicht stellt der Hauptteil des Zyklus einen Kamm mit den B_i als Zinken dar, die nicht (wie die A_i) vom Kammrücken durch Senkrechtrich abgesetzt sind.

Die Formulierung der B_i als Austrittsbedingungen erscheint wegen der anschließbaren Fallauswahl zweckmäßig. Es sollen jedoch die Entsprechungen notationeller Varianten nach Bild 3.1.6. gelten.

Bei nur einer Austrittsbedingung liegen die drei Fälle des inwendig, kopfseitig und endseitig gesteuerten Zyklus vor (Bild 3.1.5. c2 bis 3.1.5. c4), wobei c3 auch Abweisschleife und c4 Nichtabweisschleife heißt. Die in c5 beschriebenen Schleifen können als Kürzel in Form sogenannter Laufanweisungen notiert werden, wobei die Schleifenart c3/4 weiterhin erkennbar sein muß.

Ein Algorithmus, der entsprechend der eingangs angedeuteten theoretischen Möglichkeit auch wirklich unter durchgängiger Verwendung der drei Grundstrukturen aufgebaut ist, soll durchgängig oder regulär strukturiert heißen. Wie das Beispiel im Bild 3.1.7.a für die Ermittlung des Minimums M unter drei Größen X, Y, Z zeigt, kann eine regulär strukturierte Lösung oft in mehreren sehr unterschiedlichen Varianten gefunden werden (was typisch für die Programmierungstechnik ist): der Fallauswahl mit einer Liste aus zwei Bedingungen und der Entscheidungstabelle mit drei Bedin-

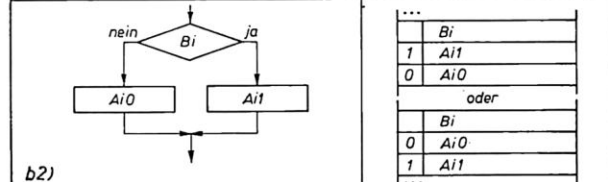
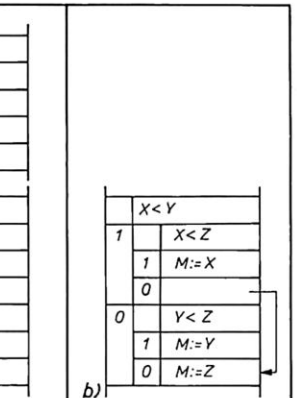
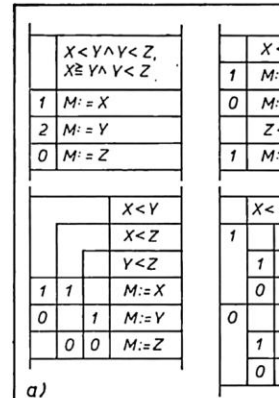
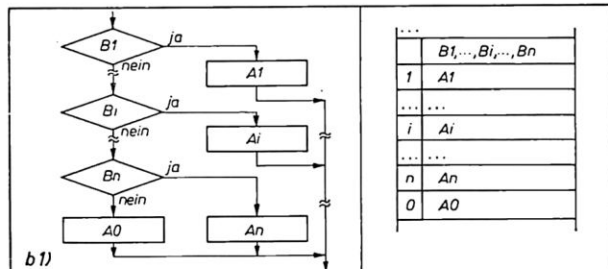
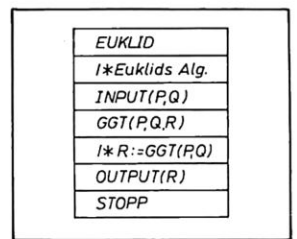
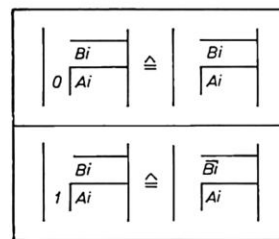
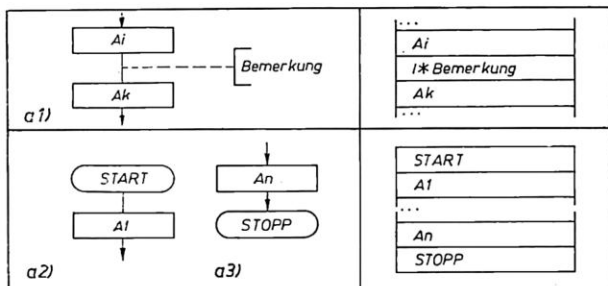


Bild 3.1.6.: Notationelle Varianten im Hauptteil des Zyklus

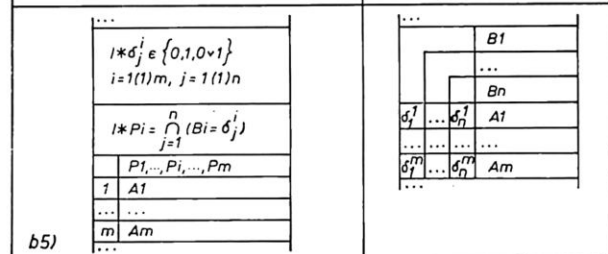
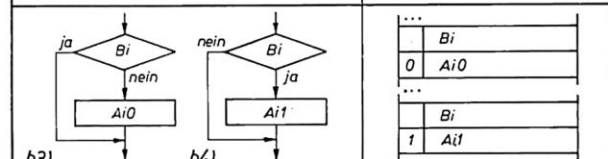
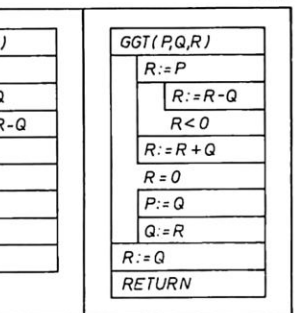
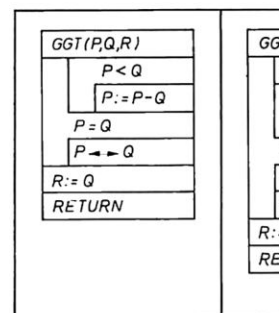


Bild 3.1.6.: Notationelle Varianten im Hauptteil des Zyklus

Bild 3.1.7.: Beispiel der Strukturierung von $M := \min(X, Y, Z)$. a) Regelmäßig strukturiert (4 Varianten); b) irregulär strukturiert (Verwendung eines Überganges)

Bild 3.1.8.: Struktogramm EUKLID für ein Hauptprogramm

Bild 3.1.9.: Varianten des Struktogramms GGT

gungen steht die wohl beste Lösung in Form einer Sequenz aus einer vollständigen und einer unvollständigen Alternative und die gewiß aufwendigste in Form verschachtelter Alternativen gegenüber. Die letztgenannte Version birgt außerdem (wegen des zweimaligen Auftretens des Falles der gleichen Zuweisung $M := Z$) die Gefahr in sich, daß der Programmierer hier „spontan rationalisiert“ und die reguläre Struktur unter Verwendung eines „gezielten Übergangs“ durchbricht (Bild 3.1.7.b).

Ganz allgemein kündigt ein irreguläres Struktogramm entweder vom Unvermögen des Programmierers oder von der Tücke und Sprödigkeit des Problems in bezug auf eine durchgängige Anwendung der Struktogramm-Methode.

Abschließend sei mit den definierten Grundstrukturen das Struktogramm des Euklidischen Algorithmus für den GGT angegeben (Bild 3.1.1.c): Es ist eine Sequenz aus Identifikator, Abweisschleife (mit $P = Q$ als Austritts- und Abweisbedingung), Ergebniszweisung und unterprogrammspezifischem Terminator. Die einzige Substruktur der Schleife ist eine Sequenz aus einer unvollständigen Alternative (mit einem Wertetausch als 1-Fall, Schachtelungstiefe 2) und einer Zuweisung.

Das funktionelle Zusammenwirken dieses Struktogramms mit einem logisch übergeordneten namens „EUKLID“ (Vorlage für ein sog. Hauptprogramm), das „GGT“ mit Operanden versorgt und dessen Resultat weiterverarbeitet, zeigt

Bild 3.1.5.: Darstellung der Grundstrukturen als Strukturblöcke. a) Sequenz; b) Fallauswahl; c) Zyklus (im Teilbild b5 lies Bj statt Bi)

a)				b)			c)
LOC	OBJ-CODE			NAME	OPERATOREN	OPERANDEN; BEMERKUNGEN	
1000				P	CSEG		GGT: PROCEDURE (P,Q) BYTE; DECLARE (P,Q,R) BYTE; DO WHILE P < Q; IF P < Q THEN DO; R = P; P = Q; Q = R; END; P = P-Q; END; RETURN P; END;
1000	C 2			Q	EQU	E	
1001	BC			R	EQU	A	
1002	68	10	10		ORG	1000H	
1005	40	0A	10	GGT:	MOV	A,P ; A:= P	
1008	D4				CMP	Q ; P>=< Q?	
1009	E0				JZ	AUS ; BET P=Q	
100A	C2				JNC	DIF ; BET P>Q	
100B	94				MOV	P,Q ; P:= QALT	
100C	DO			DIF:	MOV	Q,A ; Q:= PALT	
100D	44	00	10		SUB	A,P ; A:= P-Q	
1010	C2				MOV	P,A ; P:= A	
1011	07			AUS:	JMP	GGT	
					MOV	R,P ; R:= GGT(P,Q)	
					RET		

Bild 3.1.10.: Kodierung in einer Programmiersprache. a) Maschinensprache; b) Assemblersprache (beide maschinenorientiert, Art U 808 D); c) PL/M-Version (problemorientiert)

Bild 3.1.8. Bild 3.1.9. stellt weitere Varianten zu Bild 3.1.1.c vor, die sämtlich mit „EUKLID“ zusammenarbeiten können. Sie sind teils erheblich komplizierter als der Fall nach Bild 3.1.1.c und sämtlich effektiver. Die letzte dieser Varianten erscheint zwar sehr voluminös, ist aber in Wirklichkeit Ablaufvorlage für die Aufstellung des optimalen Mikrorechnerprogramms.

3.1.4. Programme und Programmiersprachen

Die Gesamtheit der strukturellen und inhaltlichen Komponenten eines Algorithmus wird schließlich mit Hilfe eines Programmiersprache (PS) genannten genormten Notationssystems in einer „maschinenlesbaren“ Form aufgezeichnet. Das entstandene Produkt heißt im engeren Sinne Programm und stellt als solches ein Kommunikationsobjekt dar (nicht nur für den angestrebten Austausch Mensch – Maschine, sondern auch für den Austausch Mensch – Mensch, evtl. sogar Maschine – Maschine).

Die Wahl des Niveaus der PS beeinflusst die potentiellen Austauschmöglichkeiten außerordentlich. Es sind vier verschiedene Niveaus bei PS unterscheidbar:

- niedrigstes Niveau: Maschinensprachen (MS)
- angehobenes Niveau: Assemblersprachen (AS)
- angehobenes AS-Niveau: Makro-Assemblersprachen (Makro-AS)
- höchstes Niveau: Verfahrens- (VOPS), problem- (POPS) oder prozeßorientierte PS.

Ansteigendes Niveau bedeutet einen fortschreitenden Verzicht auf das explizite Aufzeigen von Details, also einen Abstraktionsvorgang. Das wird an den Beispielen nach Bild 3.1.10., die Programmfassungen des GGT nach Bild 3.1.1.b oder 3.1.1.c darstellen, deutlich illustriert.

Programmieren im engeren Sinne ist demnach

- das Umsetzen (Kodieren) der vorbereiteten grafischen oder allgemein nicht-maschinenlesbaren Dokumentationsform eines Algorithmus (herkömmlicher Ablaufplan, Struktogramm) in eine PS
- die ad-hoc-Niederschrift des Algorithmus in einer PS.

Maschinensprachen erzwingen die Kodierung des Algorithmus mit primitivsten Mitteln als Quasi-Maschinenprogramm. Dabei handelt es sich um die vorausschauende Projizierung von Maschinenkode in einer noch „maschinen-externen“ Darstellungsform [z. B. als Hexaziffernfolgen, wie in der mit OBJ-CODE (Objektkode) überschriebenen Spalte von Bild 3.1.10.a] auf die tatsächlich in Frage kommenden Speicherplätze (Spalte Location LOC; Zuordnung: erstes Hexaziffernpaar kommt auf den Platz LOC, das folgende auf LOC + 1 usw.). Zum Einlesen dieses Kodes dient ein einfachstes Hilfsprogramm, meist eine bestimmte Monitor-Routine.

Speziell dem Mikrorechneranwender muß hier angekündigt werden, daß er bei einem bestimmten Anfangsstand (z. B.

seiner Ausrüstung in Form eines einfachen Trainingsgerätes) kaum umhin kommen wird, sich mit der MS-Programmierung vertraut zu machen.

Gehobene maschinenorientierte PS (MOPS) verwenden in ihrer Grundform als sog. Assemblersprachen symbolische bzw. mnemonische Bezeichnungen für die in einem Programm auftretenden Größen und Operationen (s. Bild 3.1.10.b).

Der MOPS-Kode wird durch ein Hilfsprogramm (Assembler) eingegeben und im wesentlichen mit Hilfe geeigneter Tabellen in die MS umkodiert. Die zeilenweise Anordnung wurde im Bild 3.1.10. gerade so getroffen, daß Bild 3.1.10.a als automatisch erzeugtes MS-Äquivalent der Assemblersprachen-Fassung 3.1.10.b verstanden werden kann. Die Bedeutung von MOPS liegt darin, daß dank der verwendeten Symbolik und der übrigen Organisationsmittel die in MOPS abgefaßten Programme einen hinreichenden „selbst-dokumentierenden“ Wert besitzen.

Höhere PS (POPS u. a.) verwenden eine formalisierte „Kunstsprache“ mit genormten Zeichen eines Alphabets (darunter mathematische Formelzeichen), Grammatikregeln (Syntax) und Bedeutungsfestlegungen (Semantik) für die Aufstellung von rechnerexternen Programmfassungen (Bild 3.1.10.c). Diese „Quellprogramme“ werden mit Hilfe komplizierter Hilfsprogramme (Compiler, Interpreter) eingelesen und in die MS übersetzt. Generierte MS-Elemente werden entweder zunächst zum vollständigen Maschinenprogramm aufgelistet (Kompilation des Quellprogramms) oder unmittelbar ausgeführt (Interpretation). Wie ein Vergleich der Bilder 3.1.10.c und 3.1.1.c zeigt, besteht ein (beabsichtigter) hoher Ähnlichkeitsgrad zwischen einem Struktogramm und dem daraus gewonnenen Programm in der POPS.

3.1.5. Fallbeispiel einer Problemaufbereitung

Eine bekannte Aussage der Hersteller von Mikroprozessoren lautet, daß der Mikroprozessor ein universeller, aber insofern neutraler Baustein sei, als er seine Anwendungseigenschaften erst durch die Programmierung erlange.

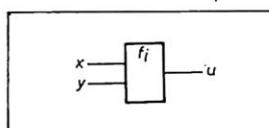


Bild 3.1.11.: Universelles Schaltelement

Der Mikroprozessor soll für die Realisierung beliebiger elementarer Schaltfunktionen $u = f_i(x, y)$, $0 \leq i \leq 15$ programmiert werden. Mit anderen Worten, es ist ein universelles Schaltelement nach Bild 3.1.11. mit

$$f_0 \triangleq 0, f_1 \triangleq \text{NOR}, \dots, f_7 \triangleq \text{NAND}, \dots, f_{15} \triangleq 1 \quad (1)$$

zu modellieren.

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

Dabei beschränkt man sich auf die im Bild 3.1.12. links angeführten wirklichen Schaltfunktionen zweier Argumente mit den ihnen zugeordneten, logisch äquivalenten Schaltzeichen.

In einem Mikroprozessor ist standardmäßig eine Verarbeitungsbreite (in Bit angegeben) realisiert, es existieren rechnerinterne Vorzugsplätze A, B, C, D, E, ..., die Werte der genannten Verarbeitungsbreite halten können und dem Programmierer ausdrücklich zugänglich sind. Weiterhin sind folgende logische Operationen mit den spezifizierten Operanden parallel über allen Bitstellen der Verarbeitungsbreite ausführbar:

$$\left. \begin{aligned} A &:= \bar{A} \\ A &:= A \omega r \\ A &:= A \omega c \\ (:= \text{bedeutet Zuweisung}) \\ \omega &\in \{\wedge, \oplus, \vee\} \\ r &\in \{B, C, D, E, \dots\} \\ c &= \text{beliebige Konstante der angegebenen Verarbeitungsbreite} \end{aligned} \right\} \quad (3.1.2.)$$

Wählen wir einen Mikroprozessor mit der Verarbeitungsbreite von 2 Bit, um beide Eingangssignale in der „gepackten“ und natürlich kodierten Form (y, x) zuführen zu können. Weiter gelten dann die Darstellungen

$$\left. \begin{aligned} A &\triangleq (A_1, A_0), B \triangleq (B_1, B_0), \\ C &\triangleq (C_1, C_0), \\ D &\triangleq (D_1, D_0), E \triangleq (E_1, E_0), \dots \\ c &\in \{(0,0), (0,1), (1,0), (1,1)\} \end{aligned} \right\} \quad (3.1.3.)$$

Wegen der häufigen Verwendung negierter Argumente und der unvermeidbaren Ausgabebreite von 2 Bit für das Resultat soll ein umfassenderes Modell für Gl. (3.1.1.) realisiert werden, das im Bild 3.1.13. dargestellt ist. Die Schaltfunktionen haben die Form

$$A := f_i(A, B, C) \quad (3.1.4.)$$

mit der garantierten Anfangsbelegung der Argumente

$$\left. \begin{aligned} A &\triangleq (A_1, A_0) = (0, x) \\ B &\triangleq (B_1, B_0) = (0, y) \\ C &\triangleq (C_1, C_0) = (1, \bar{y}) \end{aligned} \right\} \quad (3.1.5.)$$

wobei das Resultat als

$$A \triangleq (A_1, A_0) = (v, u) \quad (3.1.6.)$$

anfallen soll. Effektive Lösungen ausgewählter Varianten realisierender Schaltelemente sind im Bild 3.1.12. rechts als Sequenzen dargestellt. Neben jeder Sequenz ist eine „Protokollspalte“ eingerichtet, die den Prozeß der Resultatbildung in A aus der dort vorhandenen Anfangsbelegung (0, x) widerspiegelt. Die Auswechslung der realisierenden Sequenz für ein f_i durch die Sequenz für ein f_j , $i \neq j$, ist dann der Ausführung einer anderen elementaren Schaltfunktion gleichbedeutend.

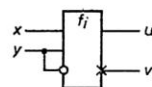


Bild 3.1.13.: Erweitertes Modell für ein universelles Schaltelement nach Bild 3.1.11.

Das funktionelle Zusammenwirken der f_i -Sequenzen mit einem logisch übergeordneten Ablauf, dargestellt durch das Struktogramm namens SCHALTFU, zeigt Bild 3.1.14. SCHALTFU übernimmt die Eingabe von (y, x), die Herstellung der Anfangsbelegung (5) unter Benutzung einer „Bitwechselloperation“

$$(A_1, A_0) := (A_0, A_1) \quad (3.1.7.)$$

aktiviert die vorgegebene f_i -Sequenz und gibt das Resultat (v, u) aus. Die Aktivierung der f_i -Sequenz ist in diesem Fall dadurch realisiert, daß die betreffende Sequenz für den in SCHALTFU auftretenden Strukturblock mit dem Text $A := F(A, B, C)$ substituiert wird.

Bild 3.1.15. zeigt für einen Halbaddierer drei realisierende Sequenzen, die sämtlich mit SCHALTFU nach der Substitutionsmethode zusammenarbeiten können. Bemerkenswert ist, daß der Operationsreichtum (3.1.2.) und (3.1.7.) das ge-

Bild 3.1.12.: Modellierung elementarer Schaltfunktionen zweier Argumente

Schaltfunktion $u = f_i(x, y)$					äquivalente Symbole		$A := f(A, B, C), A = (0, x), B = (0, y), C = (1, \bar{y})$				
Art	i	$\begin{matrix} x & 1100 \\ y & 1010 \\ u & \dots \end{matrix}$	Benennung	Symbol	Variante 1	Variante 2	Variante 1	$\frac{(A_1, A_0)}{(0, x)}$	Variante 2	$\frac{(A_1, A_0)}{(0, x)}$	
symmetrisch bezüglich x, y	1	0001	NOR Pierce	$x \downarrow y$			$A := A \vee B$ $A := \bar{A}$	$(0, x \vee y)$ $(1, \bar{x} \vee \bar{y})$	$A := \bar{A}$ $A := A \wedge C$	$(1, \bar{x})$ $(1, \bar{x} \bar{y})$	
	6	0110	EXCL-ODER Antivalenz	$x \oplus y$			$A := A \oplus B$	$(0, x \oplus y)$			
	7	0111	NAND Sheffer	$x \uparrow y$			$A := \bar{A}$ $A := A \vee C$	$(1, \bar{x})$ $(1, \bar{x} \vee \bar{y})$	$A := A \wedge B$ $A := \bar{A}$	$(0, xy)$ $(1, \bar{x} \bar{y})$	
	8	1000	UND, AND Konjunktion	xy					$A := A \wedge B$	$(0, xy)$	
	9	1001	Äquivalenz	$x \sim y$					$A := A \oplus B$ $A := \bar{A}$	$(0, x \oplus y)$ $(1, \bar{x} \oplus \bar{y})$	
	14	1110	ODER, OR Disjunktion	$x \vee y$			$A := A \vee B$	$(0, x \vee y)$			
asymmetrisch bezüglich x, y	2	0010	Inhibition	$\bar{x}y$	$y \rightarrow x$			$A := A \vee C$ $A := A$	$(1, x \vee \bar{y})$ $(0, x \vee \bar{y})$	$A := \bar{A}$ $A := A \wedge B$	$(1, \bar{x})$ $(0, \bar{x}y)$
	4	0100		$x\bar{y}$	$x \rightarrow y$					$A := A \wedge C$	$(0, x\bar{y})$
	11	1011	Implikation	$x \leq y$ $\bar{x} \vee y$	$x \rightarrow y$			$A := \bar{A}$ $A := A \vee B$	$(1, \bar{x})$ $(1, \bar{x} \vee y)$	$A := A \wedge C$ $A := \bar{A}$	$(0, x\bar{y})$ $(1, \bar{x} \bar{y})$
	13	1101		$y \leq x$ $x \vee \bar{y}$	$y \rightarrow x$			$A := A \vee C$	$(1, x \vee \bar{y})$		

Lfd. Nr.	Flag: U 808 D: SZ 8080: SZ HP/V C Z 80: SZ HP/V NC	Befehlsformat 1. Byte 2. Byte 3. Byte	U-808-D-Befehl	8080-Befehl	Z-80-Befehl	U 808 D Operationskodes, hexadezimal 0 1 2 3 4 5 6 7	7 6 5 4 3 2 1 0 op, Binär q ₂ q ₁ q ₀ h ₁ h ₀
Laden, Byte:							8080 Z 80 Operationskodes, hexadezimal 0 1 2 3 4 5 6 7
1	r1:=r	op	MOV r1, r	MOV r1, r	LD r1, r	r1	r1
2	r1:=(HL)	op	MOV r1, M	MOV r1, M	LD r1, (HL)	B	40 41 42 43 44 45 46 47
3	(HL):=r	op	MOV M, r	MOV M, r	LD (HL), r	C	48 49 4A 4B 4C 4D 4E 4F
						D	50 51 52 53 54 55 56 57
						E	58 59 5A 5B 5C 5D 5E 5F
						H	60 61 62 63 64 65 66 67
						L	68 69 6A 6B 6C 6D 6E 6F
						M, (HL)	70 71 72 73 74 75 76 77
4	r:=n	op n	MVI r, n	MVI r, n	LD r, n	A	78 79 7A 7B 7C 7D 7E 7F
5	(HL):=n	op n	MVI M, n	MVI M, n	LD (HL), n		06 0E 16 1E 26 2E 3E
						r1	3E
Zählen, Byte:							r1
6	r+=1	op	INR r	INR r	INC r	F:	B C D E H L M A
7	(HL)+:=1	op	INR M	INR M	INC (HL)		04 0C 14 1C 24 2C (HL) 3C
8	r-=1	op	DCR r	DCR r	DEC r		34
9	(HL)-:=1	op	DCR M	DCR M	DEC (HL)		05 0D 15 1D 25 2D 3D
							3F
Arithmetik u. Vergleich, Byte:							r1
10	r	op	ADD r	ADD r	ADD r	F:	B C D E H L M A
11	A+=	op	ADD M	ADD M	ADD (HL)		80 81 82 83 84 85 (HL) 87
12	A+=	op n	ADI n	ADI n	ADD n		86
13	A+=	op	ADC r	ADC r	ADC r		C6
14	A+=	op	ADC M	ADC M	ADC (HL)		88 89 8A 8B 8C 8D 8F
15	A+=	op n	ACI n	ACI n	ADC n		8E
16	A+=	op	SUB r	SUB r	SUB r		CE
17	A+=	op	SUB M	SUB M	SUB (HL)		90 91 92 93 94 95 97
18	A+=	op n	SUI n	SUI n	SUB n		96
19	A+=	op	SBB r	SBB r	SBC r		D6
20	A+=	op	SBB M	SBB M	SBC (HL)		98 99 9A 9B 9C 9D 9F
21	A+=	op n	SBI n	SBI n	SBC n		9E
22	A+=	op	CMP r	CMP r	CP r		DE
23	A+=	op	CMP M	CMP M	CP (HL)		B8 B9 BA BB BC BD BF
24	A+=	op n	CPI n	CPI n	CP n		BE
							FE
Logik, Byte:							r1
25	SZ HP 00	op	ANA r	ANA r	AND r	F:	B C D E H L M A
26	SZ HP 00	op	ANA M	ANA M	AND M		A0 A1 A2 A3 A4 A5 (HL) A7
27	SZ HP 00	op n	ANI n	ANI n	AND n		A6
28	SZ OP 00	op	XRA r	XRA r	XOR r		E6
29	SZ OP 00	op	XRA M	XRA M	XOR (HL)		A8 A9 AA AB AC AD AF
30	SZ OP 00	op n	XRI n	XRI n	XOR n		AE
31	SZ OP 00	op	ORA r	ORA r	OR r		EE
32	SZ OP 00	op	ORA M	ORA M	OR (HL)		B0 B1 B2 B3 B4 B5 B7
33	SZ OP 00	op n	ORI n	ORI n	OR n		B6
							F6

Verschieben, links/rechts									
34	CY:=A ₇ , A:=(A ₆ ...A ₀ , A ₇)	0 0C	RLC	RLCA	02	00 000 010	07	00 000 111	
35	CY:=A ₆ , A:=(A ₅ ...A ₀ , CY)	0 0C	RAL	RLA	12	00 010 010	17	00 010 111	
36	A:=(A ₀ , A ₇ ...A ₁); CY:=A ₀	0 0C	RRC	RRCA	0A	00 001 010	0F	00 001 111	
37	A:=(CY, A ₇ ...A ₁); CY:=A ₀	0 0C	RAR	RRA	1A	00 011 010	1F	00 011 111	
Sprung, unbedingte									
38	PC:=adr	op	ladr	hadr	JP adr	01 x 100	C3	11 000 011	
Aufruf, unbedingte (8·q = hhl)									
39	INSTACK (PC); PC:=8·q	op			RST q	00 q 101	q:	0 1 2 3 4 5 6 7	
40	INSTACK (PC); PC:=adr	op	ladr	hadr	CALL adr	01 x 110	CD	11 001 101	
Rückkehr, unbedingte									
41	OUTSTACK (PC)	op			RET	00 x 111	C9	11 001 001	
Bedingte Operationen:									
42	Wenn end erfüllt, dann wie	op	ladr	hadr	JP end, adr	01 end 000	end:	NZ Z NC C PO PE P M	
43	unbedingte Operation,	op	ladr	hadr	CALL end, adr	01 end 010	C2	CA D2 DA E2 EA F2 FA	11 end 010
44	sonst nur PC:=3, 3 bzw. 1	op			RET end	00 end 011	C4	CC D4 DC E4 EC F4 FC	11 end 100
Ein/Ausgabe: (n = qqQ)									
45	A:=Kanal (n)	op	n		IN n	01 00 q ₀ 1	DB		11 011 011
46	Kanal (n):=A	op	n		OUT n	01 q ₁ 0 ₀ 1	D3		11 010 011
	(2. Byte nur bei 8080, Z 80)								
Halt:									
47	PC:=1; Stopp der ZVE	op			HLT	00 000 000	76	01 110 110	

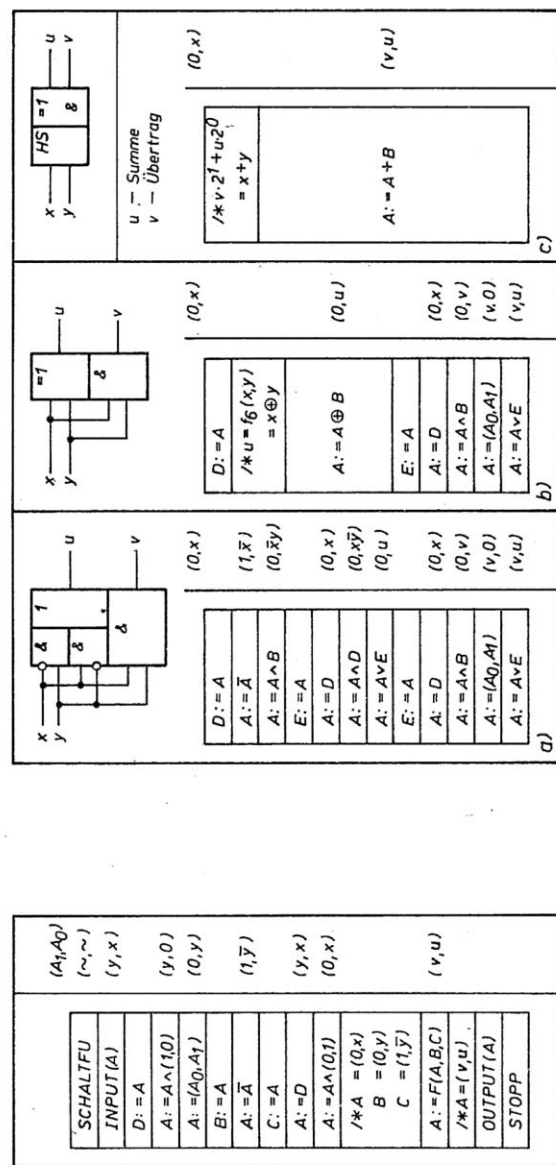


Bild 3.1.14.: Strukturprogramm für ein Hauptprogramm zur Schaltfunktion $(v,u) := F(x,y)$

naue Nachempfinden des Detaillierungsgrades der Schaltungen in den Bildern 3.1.15.a und 3.1.15.b zuläßt. Für die Variante nach Bild 3.1.15.c muß allerdings eine weitere installierte Operation des Mikroprozessors herangezogen werden, die x und y arithmetisch addiert, wobei dann die Identität gilt:

$$v \cdot 2^1 + u \cdot 2^0 = x + y \quad (8)$$

Selbstverständlich ist die Stelle v des Halbadder-Resultates im Unterschied zum Modell im Bild 3.1.13. ein „gewolltes“ Resultatbit.

Literatur

- [3.1.] Trachtenbrot, B. A.: Wieso können Automaten rechnen? Berlin: Deutscher Verlag für Wissenschaften (DVW) 1965
- [3.2.] DDR-Standard: Informationsverarbeitung; Datenfluß- und Programmablaufpläne; Sinnbilder. TGL 22 451, Juni 1975

3.2. Mikrorechner-Architektur

Mit dem Begriff der Architektur charakterisiert man die Eigenschaften eines informations- oder datenverarbeitenden Systems (hier also eines Mikroprozessors, Mikroprozessorsystems oder eines Mikrorechners) vom Standpunkt des Programmierers:

- allgemeine Struktur aus zugänglichen Elementen und ausführbaren Operationen über bzw. mit zulässigen Werten
- funktionelles Gesamtverhalten in bezug auf die Programmabarbeitung.

Damit stellt dieses Teilgebiet der Programmierungstechnik alle die Sachangaben und Voraussetzungen zur Verfügung, die für eine präzise Beschreibung der Maschinensprache erforderlich sind. Außerhalb der Architektur bleiben Organisation des Informations- und Steuerflusses, logische Schaltungsgrundlagen und physische Realisierung einschließlich technologischer Aspekte des Detail- und Gesamtaufbaues.

3.2.1. Programmierbare Prozessor- bzw. Rechenelemente

Die Eigenschaft der Programmierbarkeit eines Elementes ist seiner Zugänglichkeit gleichzusetzen. Das drückt sich in der Möglichkeit aus, die vom betreffenden Element potentiell aufnehmbare Informationsmenge im Idealfall gezielt folgenden Aktionsarten unterwerfen zu können:

- Laden = Setzen = Vorgeben eines Anfangsstandes bzw. -wertes
- Manipulieren = Verändern des Iststandes bzw. -wertes
- Lesen = Merken = Abspeichern eines Zwischen- oder des Endstandes bzw. -wertes
- Analysieren = Auswerten des Standes oder Wertes ohne vorheriges explizites Auslesen.

In Gegenüberstellung zum Hardware-Einteilungsprinzip nach Grundfunktionseinheiten (s. 2.1.1. und Tafel 2.1.1.) sind damit vom Standpunkt der Architektur folgende Einheiten von Interesse:

- Recheneinheit: Akkumulator A Flag- bzw. Statusregister F
- interner Speicher: allgemeine Register B, C, D, E, H, L; Befehlszähler PC, evtl. interner Kellerspeicher Stack oder Kellerzeiger SP für externen Kellerspeicher, evtl. Indexregister u. ä. m.
- externer Speicher: Kapazitätsangabe in K Byte bzw. entsprechender Adressierungsumfang; Einrichtbarkeit eines Stacks; Grundwortlänge
- E-A-Kanäle: Kanalkapazität; Kanalnummern bzw. -adressen.

Aus den Blockschaltungen der schon früher untersuchten Mikroprozessoren läßt sich damit eine auf der Basis dieser Prozessoren begründete Architektur wie im Bild 3.2.1. darge-

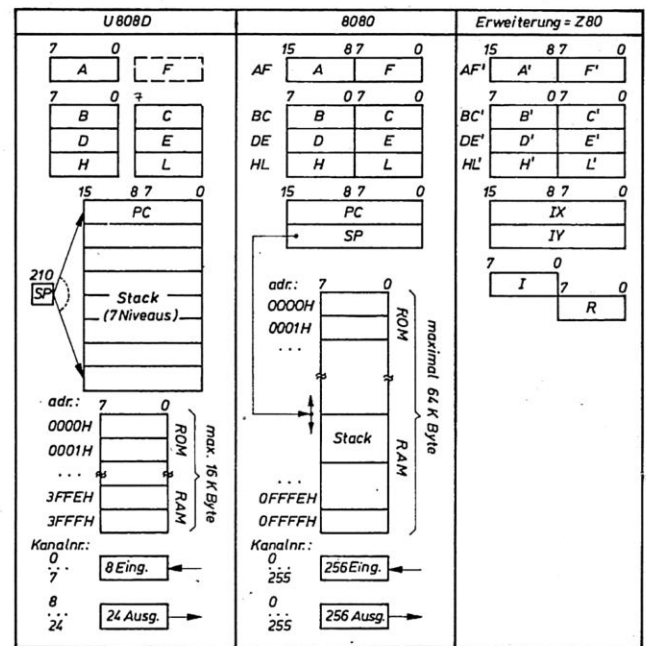


Bild 3.2.1.: Architekturkonfigurationen

A Akkumulator; F Flag, Statusregister; $F_{RHK} = CSSZP00$; $F_{RHS} = SZ0H0P1C$; $F_{ZPH} = SZ0H0P VNC$; B, C, D, E, H, L allgemeine Register; PC Befehlszähler; Stack Kellerspeicher; SP Kellerzeiger; IX, IY Indexregister; I Unterbrechungsregister; R Speicherrefreshregister

stellt resümieren. In diesem Bild spiegeln die über oder unter den die Elemente symbolisierenden Kästchen angebrachten Numerierungen die Indizes von Bitpositionen wider (ansteigend von rechts nach links). Sie kennzeichnen z. B. alle kurzen Kästchen durch acht Bitpositionen als Byte-Register. Neben den Speicherworten und E-A-Kanälen sind die maximalen Adressierungsumfänge (Hexa- bzw. Dezimalzahlen) angegeben.

Bei einer Architektur wie die des U808D ist der Befehlszähler nur 14stellig, das korrespondiert mit dem angegebenen Speicheradressierungsumfang:

$$PC_{\max}^{(808)} = 2^{14} - 1 = 16383 = 3FFFH \text{ (s. Tafel 3.2.1.)}$$

Der Stack gestattet eine maximal siebenfache sukzessive Kellierung des Befehlszählers bei Aufrufoperationen (Operationstyp 39,40 aus Tafel 3.2.2.). Lediglich in einem kompletten Mikrorechner auf der Basis von U808D kann das Flag gewöhnlich als Byte in spezieller Form (s. Bild 3.2.1.) gelesen werden. Die insgesamt 32 E-A-Kanäle sind in die beiden Funktionsgruppen für Eingabe (Kanalnummern 0 bis 7) und Ausgabe (8 bis 31) getrennt.

Wird fortgesetzt

Kleines Polytechnisches Wörterbuch RUSSISCH-DEUTSCH

Von Dipl. phil. Horst
G ö r n e r in Zusammen-
arbeit mit sowjetischen
Autoren

1. Auflage 1975. 372 Seiten,
Kunstleder, 12,00 M,
Bestell-Nr. 551 785 5

Mit etwa 25 000 Fachbegriffen bietet dieses Wörterbuch den Minimalwortschatz aller technischen Gebiete (vor allem Elektrotechnik/Elektronik, Automatisierungstechnik, Maschinenbau einschließlich Fertigungstechnik, Anwendungsgebiete der chemischen Technologie, Bauwesen) und der mit ihnen verbundenen Disziplinen Chemie, Physik und Mathematik.

Für alle, die die russische Fachsprache erlernen wollen bzw. sich zeitweilig oder ständig aus beruflichen Gründen mit ihr befassen.

Durch jede Buchhandlung erhältlich

VEB VERLAG TECHNIK BERLIN



Dr.-Ing. WOLFGANG HERRMANN

Eine Architektur auf der Basis von 8080 zeigt die Zusammenschließbarkeit „benachbarter“ Byteregister zu Doppelbyte-Registern (kurz: Doppel- bzw. langen Registern). Der 16stellige Befehlszähler korrespondiert mit dem Speicheradressierungsumfang:

$$PC_{\max}^{(8080)} = 2^{16} - 1 = 65535 = 0FFFFH \text{ (s. Tafel 3.2.1.)}$$

Tafel 3.2.1.: Übersicht über Kodierung von Byte-, Doppelbyte- und Adresswerten

Bitindex	7 6 5 4 3 2 1 0	5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
Binärkode	bbbbbb	bbbbbbbbb
Oktalkode	q ₂ q ₁ q ₀	q ₃ q ₂ q ₁ q ₀
Hexadezimalkode	h ₁ h ₀	h ₃ h ₂ h ₁ h ₀
Byte	n	n ₁ n ₀
Doppelbyte		nn
Adresse 8080, Z 80		adr
Adresse U 808 D		adr
Beschränkungen:	0 ≤ q ₂ ≤ 3 0 ≤ n ≤ 255	q ₃ = 0, 1 0 ≤ nn ≤ 65535 = 2 ¹⁶ - 1 0 ≤ adr ≤ 16383 = 2 ¹⁴ - 1 0 ≤ hadr ≤ 63 = 2 ⁶ - 1 n ₁ = hnn = nn ÷ 2 ⁸ n ₀ = lnn = n mod 2 ⁸
~ nur für U 808 D:		
Zerlegungen:	h ₁ = n ÷ 2 ⁴ h ₀ = n mod 2 ⁴	

Der Stack kann an jeder beliebigen RAM-Stelle eingerichtet werden, und der 16stellige Stackpointer verweist auf seine pulsierende Front. Das Flag ist vom Architekturstandpunkt ein Byteregister (mit drei konstant belegten Bitpositionen, s. Bild 3.2.1.). Potentiell sind je 256 E- und A-Kanäle verfügbar.

Durch die Erweiterung der 8080-Architektur infolge Doppelung des Arbeitsregistersatzes und Hinzufügen zweier Indexregister IX, IY sowie weiterer zwei Byteregister I und R entsteht eine Architektur auf der Basis des Z 80.

Für spätere notationelle Zwecke soll das Zusammensetzen von Einheiten aus Elementen bzw. die Aufgliederung in solche Elemente so beschrieben werden, wie das für die Angabe von „Listen“-Operationen üblich ist, wobei zur Vermeidung von Fehlinterpretationen der Klammerung stets mindestens zwei Elemente erforderlich sind:

Byteregister aus oder in Bitstellen

$$\begin{aligned} A &= (A_7, \dots, A_1, A_0) \\ B &= (B_7, \dots, B_1, B_0) \\ F &= (F_7, F_6, F_5, F_4, F_3, F_2, F_1, F_0) \end{aligned}$$

Doppelregister aus oder in Bitstellen

$$\begin{aligned} BC &= (BC_{15}, \dots, BC_1, BC_0) \\ PC &= (PC_{15}, \dots, PC_1, PC_0) \end{aligned}$$

Doppelregister aus oder in Byte-Register

$$\begin{aligned} AF &= (A, F) \\ BC &= (B, C) \text{ usw.} \end{aligned}$$

kombiniert

$$AF = (A, (S, Z, 0, H, 0, P, 1, C)) \text{ usw.}$$

obere Hälfte des Doppelregisters (h ≡ higher)

$$\begin{aligned} hAF &= h(A, F) = A \\ hPC &= (PC_{15}, \dots, PC_9, PC_8) \text{ usw.} \end{aligned}$$

untere Hälfte des Doppelregisters (l ≡ lower)

$$\begin{aligned} lAF &= l(A, F) = F \\ lPC &= (PC_7, \dots, PC_1, PC_0) \end{aligned}$$

Folgende Symbole sollen verwendet werden, um ein beliebi-

ges Element aus bestimmten Elementemengen zu bezeichnen:

$$\begin{aligned} r, r &\in \{A, B, C, D, E, H, L\} \\ rr, rr &\in \{BC, DE, HL\} \end{aligned}$$

3.2.2. Elementare Werte und ihre Kodierung

Aus der Vielfalt denkbarer Werte sollen diejenigen als elementar bezeichnet werden, die als Operanden vorhandener Prozessoroperationen zulässig, d. h. verarbeitbar bzw. interpretierbar sind. Die Architekturübersicht nach Bild 3.2.1. läßt schlußfolgern, daß als elementare Werte solche mit folgendem Informationsgehalt in Frage kommen:

- 1 bit, z. B. in F-Positionen
- 8 bit = 1 Byte, z. B. in Byte-Registern, Speicherzellen, als Kanalnummern
- 16 bit = 1 Doppelbyte, z. B. in Doppelregistern
- 16 bit = adr, d. h. in der speziellen Bedeutung von Adressen, z. B. im Befehlszähler (bei U 808 allerdings nur 14 bit), Stackpointer sowie in Registern mit der Sonderfunktion von Adressenspendern.

Bezüglich arbeitender Programme können solche Werte als unveränderlich (Konstanten) – z. B. und vor allen Dingen im ROM – sowie als zeitweilige Belegung von Variablen, insbesondere also der benannten Architekturelemente, auftreten. Sie sind rechnerintern stets als Binärfolgen dargestellt (= kodiert). Vom Standpunkt des Programmierers ist das in den meisten Fällen zu detailliert oder zu aufwendig gegenüber gewohnteren „rechnerexternen“ Darstellungsarten, z. B. als Dezimalzahlen.

Für die rationelle Umformung von Binärkodes in Ziffernkodes des Oktal-, Dezimal- und Hexadezimalsystems sind geeignete Hilfsmittel zu benutzen. Tafel 3.2.3. zeigt links die als Ziffern dienenden Grundzeichen dieser Systeme; rechts ist eine Äquivalenztabelle angegeben, die die in einer Zeile aufgeführten Darstellungen als Codes äquivalenter Werte ausweist. Zur Vermeidung von Verwechslungen ist in der für den Programmierer bestimmten rechnerexternen Darstellung jeder Kode mit einer Kennung zur Kenntlichmachung des gewählten Systems zu terminieren:

$$\begin{aligned} \text{Kennung B als Terminierung eines Binärkodes} & \quad b \dots bbB \\ \text{Kennung Q als Terminierung eines Oktalkodes} & \quad q \dots qqQ \\ \text{Kennung H als Terminierung eines Hexadezimalkodes} & \quad h \dots hhH \end{aligned}$$

Anmerkung:

Bei H-Kodes muß der führenden Ziffer, wenn diese ein Buchstabe ist, eine Null vorangestellt werden, um in der Programmierungstechnik Verwechslungen mit Bezeichnungen von Größen zu vermeiden.

Beispiel:

$$\begin{aligned} 11B &= 3Q = 3 = 3H \\ 11Q &= 1001B = 9 = 9H \\ 11 &= 1011B = 13Q = 0BH \end{aligned}$$

Die Äquivalenztabelle 3.2.3. hat als Grundlage die Kodierung der ganzen Zahlen ohne Vorzeichen in den Positionssystemen mit den Basen 2, 8, 10 und 16, so daß sie auf zwei Wegen erweitert werden kann. Zunächst ist wegen der Darstellung der Basen $8 = 2^3$ und $16 = 2^4$ wechselseitiges Umrechnen der B-, Q- und H-Kodes direkt oder über den Binärkode möglich. Für die 8-Bit-Darstellung eines Bytewertes findet man z. B.

$$\begin{aligned} n &\equiv bbbbbbB \\ &= b_7 \cdot 2^7 + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 \\ &\quad + b_1 \cdot 2 + b_0 \end{aligned}$$

$$\begin{aligned}
&= (b_7 \cdot 2 + b_6) \cdot 2^6 + (b_5 \cdot 2^2 + b_4 \cdot 2 + b_3) \cdot 2^3 \\
&\quad + (b_2 \cdot 2^2 + b_1 \cdot 2 + b_0) \\
&= q_2 \cdot 8^2 + q_1 \cdot 8 + q_0 \equiv \text{qqqQ} \quad (\text{mit } 0 \leq q_2 \leq 3, \\
&\quad 0 \leq q_1, q_0 \leq 7); \\
&= (b_7 \cdot 2^3 + b_6 \cdot 2^2 + b_5 \cdot 2 + b_4) \cdot 2^1 + (b_3 \cdot 2^3 + b_2 \cdot 2^2 \\
&\quad + b_1 \cdot 2 + b_0) \\
&= h_1 \cdot 16 + h_0 \equiv \text{hhH} \quad (\text{mit } 0 \leq h_1, h_2 \leq F)
\end{aligned}$$

Für Kode-Transformationen $B \rightarrow Q$ und $B \rightarrow H$ muß also der B-Kode von rechts her in Gruppen zu drei bzw. vier b-Zif-

q_0	0	(1)	7
$q_2 q_1$	00	...	7
	(1)	ddd	
	37	248	255
a)			
h_0	0	(1)	F
h_1	0	...	15
	(1)	ddd	
	F	240	255
b)			
$h_2 h_1 h_0$	000	(100H)	F00
h_3	0	...	3 840
	(1)	dddd	
	F	61 440	65 280
d)			
q_0	0	(1)	7
$q_2 q_1$	00	...	07H = 7
	(1)		
	07	38H = 56	3FH = 63
	10	40H = 64	47H = 71
(1)	17	78H = 120	7FH = 127
	20	80H = 128	87H = 191
(1)	27	0B8H = 184	0BFH = 135
	30	0C0H = 192	0C7H = 199
(1)	37	0F8H = 248	0FFH = 255
c)			

Bild 3.2.2.: Verschlüsselungstabellen. a) $\text{qqqQ} \leftrightarrow \text{ddd}$; b) $\text{hhH} \leftrightarrow \text{ddd}$; c) kombiniert $\text{qqqQ} \leftrightarrow (\text{hhH} = \text{ddd})$; d) $\text{hh00H} \leftrightarrow \text{dddd}$

Tafel 3.2.3.: Ziffernkodes des Binär-, Oktal-, Dezimal- und Hexadezimalsystems

Ziffernzeichen	Kode-Äquivalente von Zahlen			
b q d h	binär	oktal	dezimal	hexadezimal
0 0 0 0	0B	0Q	0	0H
1 1 1 1	1B	1Q	1	1H
2 2 2 2	10B	2Q	2	2H
3 3 3 3	11B	3Q	3	3H
4 4 4 4	100B	4Q	4	4H
5 5 5 5	101B	5Q	5	5H
6 6 6 6	110B	6Q	6	6H
7 7 7 7	111B	7Q	7	7H
8 8	1000B	10Q	8	8H
9 9	1001B	11Q	9	9H
A	1010B	12Q	10	0AH
B	1011B	13Q	11	0BH
C	1100B	14Q	12	0CH
D	1101B	15Q	13	0DH
E	1110B	16Q	14	0EH
F	1111B	17Q	15	0FH

fern eingeteilt und nach Tafel 3.2.3. gruppenweise durch q- bzw. h-Ziffern ersetzt werden; umgekehrt wird bei $Q \rightarrow B$ und $H \rightarrow B$ ziffernweise durch b-Gruppen ersetzt. Für Transformationen $Q \leftrightarrow H$ schiebt man den B-Kode als Zwischenschritt ein, d. h. $Q \rightarrow B \rightarrow H$ bzw. $H \rightarrow B \rightarrow Q$.

Umrechnungen aus dem bzw. in das Dezimalsystem sind mit Hilfstabellen vorzunehmen, die unabdingbares Arbeitsmittel des elementaren Programmierens sind. Je eine aufzustellende Tabelle nach Bild 3.2.2.a, b und c löst die Aufgabe $\text{qqqQ} \leftrightarrow \text{ddd}$ und $\text{hhH} \leftrightarrow \text{ddd}$ mit $0 \leq n = \text{ddd} \leq 255$, d. h. für den Zahlenbereich der ganzen Zahlen ohne Vorzeichen mit einem Byte Informationsgehalt. Bild 3.2.2.d ermöglicht, daß ein Speicheradressierungsraum von 64 K in Schritten von 100H = 256 überstrichen wird. Durch kombiniertes Anwenden beider Hexatabellen rechnet man nun leicht Adressen um:

Beispiel:

$$\begin{aligned}
10A0H &= 1000H + 0A0H = 4096 + 160 = 4256 \\
1000 &= 768 + 232 = 0300H + 0E8H = 3E8H \text{ und weiter:} \\
&= 11\ 1110\ 1000B = 1111101000B \\
&= 1\ 111\ 101\ 000B = 1750Q
\end{aligned}$$

Die eigentliche Zweckbestimmung von Werten in der Eigenschaft von Operanden ist für eine Reihe von Operationen (z. B. viele Ladebefehle) belanglos; wesentlich ist nur der repräsentierte Informationsgehalt im Umfang von Bit, Byte oder Doppelbyte. Andere Operationen interpretieren dagegen den Operanden als Exemplar einer bestimmten Wertart, also arithmetisch, logisch oder als Zeichen:

● Werte n mit 8 bit = 1 Byte Informationsgehalt

- ganze Zahlen ohne Vorzeichen
Wertebereich $0 \leq n \leq 2^8 - 1 = 255 = 0FFH$
Interpretation durch arithmetische und Vergleichsoperationen
- Reihungen von Wahrheitswerten b
Struktur der Reihung $(b_7, \dots, b_i, \dots, b_0)$, $7 \geq i \geq 0$
Interpretation durch logische Operationen
- Kode eines alphanumerischen Zeichens
Interpretation durch E-A-Operationen

● Werte nn mit 16 bit = 1 Doppelbyte Informationsgehalt

- ganze Zahlen ohne Vorzeichen
Wertebereich $0 \leq nn \leq 2^{16} - 1 = 65535 = 0FFFFH$
Interpretation durch arithmetische Operationen
- Adressen adr
Wertebereich wie nn-Zahlen
bei U 808 D: $0 \leq \text{adr} \leq 2^{14} - 1 = 16383 = 3FFFFH$
Interpretation durch Operationen mit Zugriff zum Speicherplatz mit Adresse adr.

Für arithmetische und Adressenwerte ist eine Zerlegung in den höher- und niederwertigen Teil von besonderem Interesse. Mit den Operatoren $\div$ für ganzzahlige Division (d. h. ohne Restbildung und Aufrundung) und mod für Modulorechnung, die durch die Beziehung

$$a \bmod b = a - (a \div b) \cdot b$$

verknüpft sind, realisiert man die „Hälftenbildung“:

$$\text{hnn} \equiv n_1 = nn \div 2^8$$

$$\text{lnn} \equiv n_0 = nn \bmod 2^8$$

Die Tafel 3.2.1. gibt eine zusammenfassende Übersicht über die Kodierung von Byte-, Doppelbyte- und Adresswerten.

3.2.3. Befehlsschlüssel

Unter einem Maschinenbefehl wird eine Anweisung zur Ausführung einer hardwaremäßig installierten Prozessoroperation unter Benennung aller expliziten Operanden verstanden (implizite Operanden werden einem Verarbeitungsschritt ohne besondere Benennung zugeführt). Explizite Operanden können so vielgestaltig sein, daß der Informationsgehalt von Maschinenbefehlen sehr unterschiedlich ist und nicht mit einem einzigen Kodierungsschema (= Format) abzudecken geht. Es werden daher zunächst Ein-, Zwei- und Drei-Byte-Befehle unterschieden, deren erstes Byte die Operation spezifiziert, während es für weitere Bytes bestimmte generelle Festlegungen gibt (s. Bild 3.2.3.).

Bitindex in Byte:	7	0	7	0	7	0
1-Byte-Befehle:	op					
2-Byte-Befehle:	op	n				
3-Byte-Befehle:	op	lnn			hnn	
	op	lqdr			hadr	
a)						
2 explizite Operanden	.. r1	r2	Befehlsart 1	aus		
1 expliziter Operand	.. r	...	Befehlsart 6	Tafel		
1 expliziter Operand	 r		Befehlsart 10	3.2.2.		
kein expliziter Operand			Befehlsart 12			
b)						

Bild 3.2.3.: Befehlsformate. a) Anzahl und Bedeutung der Bytes im Befehl; b) Benennung von expliziten Operanden r im op-Byte

In der Tafel 3.2.2. war eine Zusammenstellung aller Befehle gegeben, die den betrachteten drei Mikroprozessoren gemeinsam sind. Die 47 Befehlsarten sind in Gruppen unterteilt und zunächst symbolisch beschrieben. In der eigentlichen Befehlsvergleichsliste sind die äquivalenten Befehle der drei Prozessoren mit symbolischer Bezeichnung der Ope-

ration und Nachstellung der expliziten Operanden zusammengestellt. Für das Operationsbyte (op) sind Hexa- und Binärkodes angegeben. Die Binärkodes der Befehlsarten Nr. 1, 6, 10 und 12 sind z. B. Repräsentanten der vier denkbaren Grundaufteilungen des op-Bytes nach Bild 3.2.3. Bild 3.2.4. zeigt ein Schema zur Aufstellung von Befehlsschlüsseln.

q ₈	0	1	2
q ₂ q ₁			
00 00	HLT	01 HLT	0L RLC
01 08	INR B	09 DCR B	0A RRC

Bild 3.2.4.: Schema zur Aufstellung von Befehlsschlüsseln (nach Oktalkodes)

Mit den Angaben aus Tafel 3.2.2. gewinnt man dann einen sehr übersichtlichen und vollständigen Befehlsschlüssel für den U 808 D sowie in entsprechenden weiteren Tabellen den Grundstock der Befehlsschlüssel für den 8080 und Z 80.

Tafel 3.2.4. zeigt, wie die Befehlsschlüssel des 8080 und des Z 80 komplettiert werden können.

Ein Vergleich der erhaltenen Befehlsschlüssel zeigt den eingeschlagenen Entwicklungsweg der untersuchten Prozessoren vom Standpunkt der Vervollkommenheit und Differenzierung ihrer operationellen Möglichkeiten.

3.3. Maschinensprache

3.3.1. Arbeitsprogramme

Für jeglichen Rechner stellt seine Maschinensprache (MS) die Gesamtheit der Mittel dar, die die Aufstellung von Arbeitsprogrammen (des Anwenders) und von Dienstprogrammen (des Systemprogrammierers) in einer Form ermöglicht, in der diese Programme im Speicher des Rechners einge-

schrieben stehen. Arbeitsprogramme (AP) machen dabei nur von einer Untermenge an MS-Mitteln Gebrauch, die das Programmniveau (PN) darstellt. Das sind die im Rechner strukturell fixierten (= hardwaremäßig implementierten) Standardroutinen für Basisoperationen und Standardprozeduren. Die fundamentale Eigenschaft eines Arbeitsprogramms besteht darin, daß es vom Prozessor bzw. Rechner ohne jegliche weitere notationellen Veränderungen verarbeitet werden kann. Es heißt, daß das Arbeitsprogramm direkt abarbeitbar ist.

3.3.1.1. Programmiveau des Mikroprozessors

Das Programmiveau eines Mikroprozessors umfaßt als Standardroutinen im genannten Sinne alle Maschinenbefehle. Auf einfacheren Mikroprozessoren sind nur Basisoperationen wie := (Ergibt-Operation, Laden), +, -, ∧, ⊕, ∨ u. ä. realisiert. Auf entwickelteren Mikroprozessoren verfügt der Anwender auch über Standardprozeduren in Form von Gruppenoperationen (z. B. LDIR bei Z 80) oder Standardfunktionen.

Faktisch kann jedoch gerade bei Mikroprozessoren eine größere Anzahl von Maschinenbefehlen den Rang von Standardprozeduren haben; ein solcher Befehl muß abarbeitungsmäßig nur so realisiert sein, daß er sich als eine Folge anderer, meist Basisoperationen, darstellt und seine Wirkung der summarischen Wirkung dieser Folge äquivalent ist. Zur besseren Verständigung und Illustration betrachtet man Maschinenbefehle allerdings kaum in ihrer rechnerinternen (d. h. im Speicher eingeschriebenen bzw. einschreibbaren) Darstellung als Binärfolgen. Man bezeichnet sie vielmehr symbolisch und gibt für diese rechnerexterne Form Verschlüsselungsvorschriften an, wie das im Abschnitt 3.2. für elementare Werte und Operationen durch Angabe der Kodierung bzw. des Befehlsschlüssels praktiziert wurde.

Tafel 3.2.4.: Zusätzlicher gemeinsamer Befehlssatz von 8080 und Z 80 (s. a. Tafel 3.2.2.)

Lfd. Nr.	Flag 8080: SZHP C Z 80: SZHP/V NC	Befehlsformat 1. Byte 2. Byte 3. Byte	Befehlsvergleichsliste 8080-Befehl Z-80-Befehl	8080, Z 80 Op.-Kodes, hex. 0 1 2 3	76543210 ~, binär q ₂ q ₁ q ₀ h ₁ h ₀	
Laden, Byte:						
48	(rr) := A	op	STAX hrr	LD (rr), A	rr: BC DE 02 12	00 rr 0 010
49	A := (rr)	op	LDAX hrr	LD A, (rr)	0A 1A	00 rr 1 010
50	(adr) := A	op	STA adr	LD (adr), A	32	00 110 010
51	A := (adr)	op	LDA adr	LD A, (adr)	3A	00 111 010
Laden, Doppelbyte:						
52	rr := nn	op	LXI hrr, nn	LD rr, nn	rr: BC DE HL SP, AF 01 11 21	00 rr 0 001
53	SP := nn	op	LXI SP, nn	LD SP, nn	31	00 110 001
54	[(adr + 1), (adr)] := HL	op	SHLD adr	LD (adr), HL	22	00 100 010
55	HL := [(adr + 1), (adr)]	op	SHLD adr	LD HL, (adr)	2A	00 101 010
56	SP := HL	op	SPHL	LD SP HL	F9	11 111 001
57	INSTACK(rr)	op	PUSH hrr	PUSH rr	C5 D5 E5	11 rr 0 101
58	INSTACK(AF)	op	PUSH PSW	PUSH AF	F5	11 110 101
59	OUTSTACK(rr)	op	POP hrr	POP rr	C1 D1 E1	11 rr 0 001
60	OUTSTACK(AF)	op	POP PSW	POP AF	F1	11 110 001
Austausch, Doppelbyte:						
61	[(SP + 1), (SP)] ↔ HL	op	XTHL	EX (SP), HL	E3	11 100 011
62	DE ↔ HL	op	XCHG	EX DE, HL	EB	11 101 011
Zählen, Doppelbyte:						
63	rr := 1	op	INX hrr	INC rr	rr: BC DE HL SP 03 13 23	00 rr 0 011
64	SP+ := 1	op	INX SP	INC SP	33	00 110 011
65	rr- := 1	op	DCX hrr	DEC rr	0B 1B 2B	00 rr 1 011
66	SP- := 1	op	DCX SP	DEC SP	3B	00 111 011
Arithmetik, Doppelbyte:						
67	HL+ := rr	! 0C	DAD hrr	ADD HL, rr	rr: BC DE HL SP 09 19 29	00 rr 1 001
68	HL+ := SP	! 0C	DAD SP	ADD HL, SP	39	00 111 001
Logik, Byte und Bit:						
69	A := BCD [A, FLAG (H, N)]	SZHP C	DAA	DAA	27	00 100 111
70	A := $\bar{A}$	1 1	CMA	CPL	2F	00 101 111
71	CY := 1	0 01	STC	SCF	37	00 110 111
72	CY := $\bar{CY}$	! 0C	CMC	CCF	3F	00 111 111
Sprung, unbed., indirekt:						
73	PC := HL	op	PCHL	IP (HL)	E9	11 101 001
Unterbrechungsmöglichkeit:						
74	Unterbrechung ignorieren	op	DI	DI	F3	11 110 011
75	Unterbrechung erlauben	op	EI	EI	FB	11 111 011
keine Operation:						
76	PC+ := 1, Fortsetzung	op	NOP	NOP	00	00 000 000

3.3.1.2. Ressourcenbelegung

Unter einer solchen Belegung soll die dauernde oder zeitweilige Inanspruchnahme von zugänglichen Architekturelementen (s. Bild 3.2.1.) durch das Arbeitsprogramm bei seiner direkten Abarbeitung verstanden werden. Ressourcen sind daher die Register (insbesondere die allgemeinen), der Speicher und die E-A-Kanäle.

Vor Abarbeitungsbeginn belegen das Arbeitsprogramm und ein Satz von Anfangswerten gewöhnlich nur Teile des Speichers. Speziell die Anfangsbelegung mit Werten B_w ist als eine Menge von Zuordnungen $adr:w$ zu verstehen, wobei adr eine Speicheradresse und w einen elementaren Wert verkörpert. Die Wirkung selbst nur eines einzigen ausgeführten Maschinenbefehls kann abstrakt als Zuordnung einer neuen Wertebelegung zur gerade vorliegenden aufgefaßt werden. Dabei werden weitere oder andere Ressourcen in Anspruch genommen und/oder wieder freigegeben.

Die Belegung B_{AP} des für die Aufnahme der Maschinenbefehle bef vorgesehenen Speicherteils stellt im Normalfall eine unveränderliche Menge von Zuordnungen $adr:bef$ dar. Bei der Kodierung des Arbeitsprogrammes aus einer externen Fassung muß natürlich das algorithmische Merkmal der Determiniertheit des nächsten Schritts im Ablaufgeschehen gewahrt bleiben, woraus sich das Problem des sog. Nachfolgebefehls ergibt.

Die Aufgaben und Schwierigkeiten des Programmierens auf dem Programmiveau der Maschinensprache bestehen demnach in folgendem:

- Aufstellen der Anfangsbelegungen B_w und B_{AP}
- Voraussicht der in der Phase der Arbeitsprogrammverarbeitung vor sich gehenden Ressourcenbewegung
- strenges Auseinanderhalten der Adressenmengen beider Belegungen.

3.3.1.3. Operandenangaben

In Maschinenbefehlen sind alle expliziten Operanden anzugeben (vgl. Abschnitt 3.2.3.), damit bei Befehlsabarbeitung unter den insgesamt vorhandenen Werten auf die allein in Frage kommenden zugegriffen werden kann.

Konstanten

Konstanten n , nn sind unveränderliche Werte. Keine im Rahmen einer Zuordnung $adr:n$ etwa festgelegte Adresse darf daher als Zieladresse einer Lade- bzw. Speicheroperation auftauchen. Beim Mikrorechner geht das dadurch zu garantieren, daß der die Konstanten umfassende Teil der Belegung B_w im ROM eingerichtet wird.

Ein anderer Weg ist die Integration einer Konstanten in den Maschinenbefehl bef selbst, wodurch sie faktisch Bestandteil von B_{AP} wird:

$$bef = (op, n) \text{ bzw. } bef = (op, lnn, hnn) \quad (3.3.1.)$$

In den Mikrorechner-Befehlslisten wird so verfahren, daß dem Operationsbyte die Konstante folgt. Eine Adresse, die als Operandenadresse der Konstanten angesehen werden könnte, wird nicht herausgebildet; die Konstante tritt als „unmittelbarer“ (engl. immediate) Operand (Literal) auf, so daß für die betreffende Operation Sofortzugriff besteht, gleichzeitig diese Konstante aber für alle anderen Operationen verborgen ist.

In externer Notation wird eine Konstante nach Abschnitt 3.2.2. mit der erforderlichen Kennung angegeben, z. B.

$$LD A, 0FFH; A := 255$$

Registeroperanden

Werte aus der Registerbelegung werden durch Angabe von Registeradressen benannt. Wegen der geringen Registeranzahl (≤ 8) sind das nur „Kurzadressen“, die mit dem eigentlichen Operationskode zusammen im Operations-Byte der Maschinenbefehle untergebracht werden können.

Extern sind jedoch allen Registern echte Bezeichnungen zugeordnet, die bei der Verschlüsselung durch eben diese Kurzadressen ersetzt werden:

	A	B	C	D	E	H	L
U 808 D	0	1	2	3	4	5	6
8080, Z 80	7	0	1	2	3	4	5

(3.3.2.)

Ein Doppelregister rr kann durch eine der beiden Kurzadressen seiner Einzelregister benannt werden, wobei im Operationskode festgelegt ist, ob es sich z. B. um die gerade oder die ungerade der beiden Registeradressen handelt. Man kann jedoch auch so tun, als stünden lediglich vier zweistellige Binäradressen zur Verfügung

	BC	DE	HL	AF/SP
oktal $\triangleq$ 3 Binärstellen	0,1	2,3	4,5	6,7
2 Binärstellen	00	01	10	11

(3.3.3.)

In externer Notation repräsentiert die ungeklammerte Registerbezeichnung genauso den vom Register gehaltenen Wert, wie die Bezeichnung irgendeiner Variablen für ihren aktuellen Wert steht, z. B.

$$LD A, B; A := B$$

$$INC BC; BC += 1, \text{ d. h., erhöhe Wert in BC um 1}$$

Speicheroperanden

Werte aus der Speicherbelegung B_w werden durch Angabe von Speicheradressen benannt, die in Abhängigkeit von der potentiell installierbaren Speicherkapazität gewöhnlich 12; 14 oder 16 bit lang sind. Entwurfsmäßig werden Mikroprozessor-Befehlslisten jedoch so ausgelegt, daß nicht in jedem Maschinenbefehl mit Speicherzugriff eine volle Adresse auftreten muß.

Direkte Adressierung

Ein Speicheroperand ist direkt adressiert, wenn im Maschinenbefehl seine wirkliche Adresse angegeben wird. Der Umstand, daß in einem B_w -Element $adr:w$ nicht die Adresse adr sondern der Wert w den Operanden darstellt, die Adresse also nur Mittel für den Zugriff ist, kann nur mit Hilfe eines sog. Inhaltsoperators korrekt ausgedrückt werden:

$$w = (adr) \quad (3.3.4.)$$

In externer Notation steht folglich z. B.

$$LD A, (adr) ; A := (adr)$$

Es sei angemerkt, daß in höheren Programmiersprachen die Verhältnisse anders liegen und ein aktueller Wert durch eine Variablenbezeichnung x repräsentiert wird; die Adresse von x kann dann nur mit Hilfe eines sog. Adressenbildungsoperators ausgedrückt werden, etwa $adr =)x($.

Wird fortgesetzt

KONDENSATOR-MIKROFONE

- ▶ Studio-Mikrofone SMS 70 u.a.
- ▶ Drahtlose Studio-Übertragungsanlagen
- ▶ Meßmikrofone 20 Hz · · · 100 kHz

Fordern Sie bitte unsere Prospekte an

RFT·VEB MIKROFONTECHNIK GEFELL

DDR · 6552 Gefell · Telefon 262 · Telex 58-88 43

Exporteur: ELEKTROTECHNIK EXPORT-IMPORT

Volkseigener Außenhandelsbetrieb der DDR

DDR · 102 Berlin, Alexanderplatz · Haus der Elektroindustrie

Dr.-Ing. WOLFGANG HERRMANN

Indirekte Adressierung

Ein Speicheroperand ist indirekt adressiert, wenn im Maschinenbefehl nur ein Doppelregister angegeben wird, das die wirkliche Adresse fertig enthält. Der vom Doppelregister repräsentierte Wert ist die Adresse des Wertes im B_w -Element $adr:w$, so daß nach Beziehung (3.3.4.)

$$w = (rr) \quad (3.3.5.)$$

gilt, in korrekter externer Notation muß z. B. stehen

$$LD (BC), A; (BC) := A$$

Damit ein solcher Befehl sinnvoll ausgeführt werden kann, muß im Doppelregister die fragliche Adresse bereitstehen, z. B. infolge vorheriger Abarbeitung eines Befehls

$$LD BC, nn; BC := nn$$

wobei die lange Konstante nn die eigentliche spätere Adresse adr ist.

In den betrachteten Befehlslisten wird bevorzugt HL als „Adressenspeicher“ zur indirekten Adressierung von Speicheroperanden benutzt. Statt des häufigen (HL) wird oft das Kürzel M benutzt, was Speicheroperand (memory) bedeuten soll:

$$M \equiv (HL) \quad (3.3.6.)$$

Für U 808 D ist das die einzige realisierte Zugriffsmöglichkeit zu Speicheroperanden, so daß man z. B. für eine einfache Operation $A := (adr)$ eine ganze Sequenz programmieren muß:

$$\left. \begin{array}{l} MVI H, n; H := n \text{ mit } n = hadr \\ MVI L, n; L := n \text{ mit } n = ladr \\ MOV A, M; A := (adr) \text{ mit } HL = adr \end{array} \right\} \quad (3.3.7.)$$

Beim U 808 D existiert kein Doppelregister HL als solches (daher auch zwei vorbereitende Ladeoperationen für Byte-register).

In Maschinenbefehlen erhält die Adressierungsart M bzw. (HL) den bei der Registeradressierung (3.3.2.) noch freigebliebenen achten Oktalkode zugeordnet, also $M \triangleq 7$ für den U 808 und $(HL) \triangleq 6$ für 8080, Z 80.

Indizierte Adressierung

Ein Speicheroperand ist indiziert adressiert, wenn im Maschinenbefehl nur eine sog. Verschiebung (engl. displacement) e als spezielle vorzeichenbehaftete Konstante im Zweierkomplementcode angegeben wird, wobei meist

$$-128 \leq e \leq 127 \quad (3.3.8.)$$

festgelegt ist und die wirkliche Adresse jedesmal bei Befehlsabarbeitung berechnet wird:

$$adr = xy + e$$

daher

$$w = (xy + e) \quad (3.3.9.)$$

Auch hier gilt, daß xy schon passend geladen sein muß.

Bei $e = 0$ wird die indizierte scheinbar zur indirekten Adressierung. Es wird jedoch auch dann das volle Adressenbildungsgesetz (3.3.9.) abgearbeitet, was im Maschinenbefehl auch die Angabe der Konstanten 00H für e erforderlich macht.

Für Zwecke der Operationsbeschreibung können die beiden Methoden unter dem Aspekt zusammengefaßt werden, daß sie sich auf die Angabe nicht direkt adressierter Speicheroperanden beziehen; mit dem im Abschnitt 3.2.1. eingeführten $xx, xx \in \{HL, xy\}$ notiert man

$$(xx), (xx) \in \{(HL), (xy + e)\} \quad (3.3.10.)$$

Implizite Operanden

Implizit werden Operanden genannt, die man im Befehl nicht ausdrücklich angibt, weil sie unverwechselbar sind und in die Befehlsabarbeitung automatisch einbezogen werden. Die Betrachtung rechnerexterner Befehlsdarstellungen mit

ihren symbolischen Operationsbezeichnungen, deren Anzahl oft weit geringer ist als die Anzahl unterschiedlicher Operationscodes, erzwingt jedoch oft auch die Angabe solcher Operanden: Nur dann kann die externe Befehlsform eindeutig verschlüsselt werden. Beispiele hierfür sind

$$LD r1, r; r1 := r \text{ mit } op = (11B, r1, r)$$

$$LD A, (rr); A := (rr) \text{ mit } op = (00B, rr, 0010B)$$

$$LD SP, HL; SP := HL \text{ mit } op = 11111001B$$

Dieselbe Operationsbezeichnung kann sogar mit unterschiedlich vielen Operanden stehen, wenn nur die eindeutige Kodierbarkeit gewährleistet ist:

$$ADD r; A += r \text{ mit } op = (10000B, r)$$

$$ADD HL, rr; HL += rr \text{ mit } op = (00B, rr, 1001B)$$

3.3.1.4. Nachfolgebefehl

Als Befehlsfolger war die Aufgabe bezeichnet worden, zu einem in Verarbeitung befindlichen B_{AP} -Element $adr:bef$ eindeutig das Nachfolgeelement $adr':bef'$ zu bestimmen. Der Befehlszähler PC entbindet die Mehrzahl aller Befehle von der Notwendigkeit, ihren Nachfolger explizit benennen zu müssen. Bei Beginn der Verarbeitung eines Elementes $adr:bef$ hat zu gelten:

$$PC = adr;$$

folglich

$$bef = (PC) \quad (3.3.11.)$$

Dabei adressiert adr bei Mehr-Byte-Befehlen nur das erste Byte.

Zu Beginn einer Programmabarbeitung wird ein adr_0 vorgegeben und mit Hilfe der Hard- oder Software die Bedienungsoption $PC := adr_0$ vorgenommen. Nach Rechnerstart wird dann gemäß der Bedingung (3.3.11.) die Abarbeitung eines ersten Maschinenbefehls bef_0 initiiert.

Der Code eines Maschinenbefehls bef belegt sukzessiv die mit $adr, \dots, adr + lng - 1$ adressierten Worte in B_{AP} (beim byteorientierten Rechner also Bytes). Dann bezeichnet $lng = lng(bef)$ die Befehlslänge (in Adressierungseinheiten), und der gegebene Befehl heißt ein lng -Byte-Befehl (z. B. $lng = 1, 2$ oder 3). Bestandteil fast jeder Befehlsabarbeitung ist nun ein im Hintergrund ablaufendes Weiterschalten des Befehlszählers:

$$PC += lng(bef) \quad (3.3.12.)$$

Falls bef' also der in der MS ausgedrückte textliche Nachfolger von bef ist, dann gilt für die Belegung B_{AP} natürlich auch $adr' = adr + lng(bef)$ und der weiterschaltete Befehlszähler [s. Bedingung (3.3.12.)] erfüllt die Generalvereinbarung (3.3.11.) bezüglich des Elementes $adr':bef'$, so daß die Arbeitsprogrammverarbeitung fortschreiten kann. Im Normalfall adressiert bef seinen Nachfolger also implizit und indirekt mit Hilfe des Befehlszählers.

Möglichkeiten der expliziten Angabe des Befehlsnachfolgers sind mit den Sprungbefehlen gegeben.

Direkte Adressierung durch Laden des Befehlszählers mit einer Adresse

Im Drei-Byte-Befehl bef ist auf zwei Byte die Adresse adr' des Befehlsnachfolgers angegeben (adr' verhält sich wie eine lange Konstante von der Wertart Adresse). Das Fazit der Abarbeitung dieses Sprungbefehls besteht in einer Neueinstellung des Befehlszählers durch Laden von adr' :

$$PC := adr' \quad (3.3.13.)$$

Indirekte Adressierung durch Umladen eines Doppelregisters xx in den Befehlszähler

Das Fazit der Abarbeitung eines solchen Registersprungbefehls ist ebenfalls die Neuaufstellung des Befehlszählers:

$$PC := xx; \text{ z. B. } PC := HL \quad (3.3.14.)$$

(Firmenunterlagen weisen den zugehörigen Maschinenbefehl meist als JP (HL) aus, was im Vergleich mit der sonst korrekten Verwendung des Inhaltsoperators unmotiviert ist

28 (1979) H. 3 radio fernsehen elektronik

START	NOP	CND
/* AP in MS Z80	/* keine Operation	/* q1 = cnd
PC := adr ₀	END	/* F = (S, Z, ..., P/V, N, CY)
op := (PC)	DECR	(q1+2) = ?
PC := 1	/* q1 = r < r, r = (q1)	0 Fi = Z
op = 00H	r := 1	1 Fi = CY
1 NOP	END	2 Fi = P
0 LISTE 1	DECM(xx)	3 Fi = S
	(xx) := 1	b := (q1 mod 2)
	END	END
LISTE 1	LDRN	RETBED
(q2, q1, q0) := op	/* (PC) = n	CND
/* Oktalziffern	r := (PC)	Fi = b
q2 = ?	PC := 1	1 OUTSTACK(PC)
0 q0 = ?	END	END
0 ...	LDMN(xx)	JPBED
4 ...	(xx) := (PC)	CND
5 q1 = 6	PC := 1	Fi = b
0 DECR	END	1 JPUNB
1 DECM(HL)	LDRR	0 PC := 2
6 q1 = 6	/* r1 = (q1), r2 = (q0)	END
0 LDRN	r1 := r	DISPLACEMENT
1 LDMN(HL)	END	e := (PC)
7 ...	LDRM(xx)	PC := 1
1 q1 = 6	r1 := (xx)	END
0 q0 = 6	END	IND(xy <)
0 LDRR	LDMR(xx)	op := (PC)
0 LDRM(HL)	(xx) := r	PC := 1
1 LDMR(HL)	END	(q2, q1, q0) := op
1 HALT	HALT	q2 = ?
2 ...	STOPP	0 ...
3 q0 = ?	END	1 DISPLACEMENT
0 RETBED	JPUNB	q1 := 6
1 ...	ladr := (PC)	q0 := 6
2 JPBED	PC := 1	0 1 LDRM(xy + e)
3 q1 = ?	ladr := (PC)	1 0 LDMR(xy + e)
0 JPUNB	PC := (ladr, laddr)	FEHLER
1 LISTE 2	END	2 ...
2 ...	RST	3 ...
3 ...	INSTACK(PC)	END
4 ...	PC := (00H, q1 * 2 ³)	LISTE 2
5 q1 = ?	END	...
0 ...		END
2 ...		LISTE 3
3 IND(>IX <)		...
4 ...		END
5 LISTE 3		
6 ...		
7 IND(>IY <)		
6 ...		
7 RST		
END		

Bild 3.3.2.: Modell der Abarbeitung (Interpretation) eines Programms

Erläuterungen zum Bild 3.3.2.

Der als Start des Arbeitsprogrammes zu bezeichnende Vorgang besteht in der Initiierung des Befehlszählers PC mit der Adresse adr_0 des ersten Bytes des in logischer Hinsicht ersten Befehls (s. Abschn. 3.3.1.4.). Alles weitere läuft innerhalb einer formal unendlichen Schleife ab, die deshalb (endseitig) ohne Austrittsbedingung ist. Jeder Schleifendurchlauf verarbeitet genau den einen Befehl vollständig, dessen erstes Byte vom Programmzähler am Schleifenkopf aktuell adressiert wird (s. 3.3.1.11.). Dieses Byte wird aus der Belegung B_{A1} ausgelesen und so behandelt (was später evtl. korrigiert werden muß), als sei es definitiv das Operationsbyte. Danach wird der Befehlszähler sofort um Eins weiterschaltet. Beim ausgelesenen Byte wird der Fall eines möglichen NOP (keine Operation) ausgesondert bzw. alternativ versucht, das Byte in Liste 1 des Befehlsschlüssels zu identifizieren.

Der dafür bestimmte Ablaufteil LISTE 1 modelliert einen Dekoder. Er faßt das op-Byte als Oktalkode auf, zerlegt es und nimmt mit den Oktalziffern eine Art dreidimensionale Fallauswahl vor. Ziffer q2 wählt im Befehlsschlüssel das zugehörige Viertel, in dem dann q1 bzw. q0 eine ganze Zeile bzw. Spalte mit acht Operationen oder bei sukzessiver Anwendung im Schnittpunkt von Zeile und Spalte eine Einzeloperation ermitteln (dieser Auswahlvorgang ist im Befehlsschlüssel nach Tafel 3.3.1. mit einem einzigen Blick über-

schaubar). Die erkannte Operation wird mit einer Aktivierung der zugehörigen Ausführungsroutine realisiert, z. B.

bei $(q2 = 0) \wedge (q0 = 5) \wedge (q1 \neq 6)$ Routine DECR

bei $(q2 = 0) \wedge (q0 = 5) \wedge (q1 = 6)$ Routine DECM(HL)

Beide Routinen realisieren DEC-Befehle: DECR erniedrigt den Inhalt desjenigen Registers r um Eins, dessen Kurzadresse durch die Oktalziffer q1 gegeben ist, DECM erniedrigt dagegen einen Zähler in der Wertebelegung B_w . Die dazu erforderliche Adresse wird vom Doppelregister HL gespendet (da die Operation im Rahmen von Liste 1 identifiziert werden konnte). Die Bezeichnung von HL ist folglich als Parameter der Routine anzugeben, um die Übergabe des von HL aktuell repräsentierten Wertes (d. h. eben dieser erforderlichen Adresse) an DECM zu gewährleisten. Realisierungsbeispiele von 2-Byte-Befehlen zeigen

bei $(q2 = 0) \wedge (q0 = 6) \wedge (q1 \neq 6)$ Routine LDRN

bei $(q2 = 0) \wedge (q0 = 6) \wedge (q1 = 6)$ Routine LDMN(HL)

Eingangs der Ausführungsteile beider Routinen adressiert der Befehlszähler PC die Bytekonstante n, die damit als „Quelloperand“ der Ladebefehle beschafft werden kann. Auch hier wird unverzüglich das Weiterschalten von PC abgeschlossen.

Die Analyse der Registerladebefehle (bei $q2 = 1$) ist als Entscheidungstabelle formuliert, das erlaubt auch ein bequemes Erkennen der Halteoperation.

Die HALT-Routine bewirkt einen ZVE-Stopp (inklusive Beeinflussung künftiger Unterbrechbarkeit) und damit das Anhalten der Arbeitsprogramm-Abarbeitung, wobei sich tatsächlich der Befehlszähler als schon weiterschaltet erweist.

Als 3-Byte-Befehl wird der unbedingte Sprung gezeigt (s. 3.3.1.13.):

bei $(q2 = 3) \wedge (q0 = 3) \wedge (q1 = 0)$ Routine JPUNB

Musterbeispiele für den Rückgriff von Realisierungsroutinen auf eine Hilfsroutine (hier CND) sind

bei $(q2 = 3) \wedge (q0 = 0)$ Routine RETBED

bei $(q2 = 3) \wedge (q0 = 2)$ Routine JPBED

Beide haben ein bestimmtes Statusbit Fi auszutesten, wobei q1 den Kode der Bedingung darstellt und angibt, welches F-Bit gemeint und mit welcher Bitkonstanten b zu vergleichen ist. In JPBED z. B. kann daher nach vorherigem Aufruf von CND die Alternative $Fi = b$ entschieden werden: die Erfüllung dieser Bedingung realisiert durch Aktivieren von JPUNB einen Sprung, die Nichterfüllung führt zum bloßen Weiterschalten von PC, d. h. effektiv zum Ignorieren der ganzen Operation.

Weiter zeigt Bild 3.3.2. die vier Fälle, bei denen $(q2, q1, q0)$ noch nicht das op-Byte, sondern erst ein Signalbyte darstellt. Die Aktivierung von Routine IND realisiert z. B. die Fälle (3.3.1.18.); dabei ist die IND-Routine selbst als eigenständiger dreidimensionaler Verteiler aufgebaut, der das ausgelesene neue op-Byte analysiert. Als Parameter braucht IND nicht den per Bezeichnung xy repräsentierten Wert des betreffenden Indexregisterinhalts, weil damit lediglich die indizierte Adressierung realisierbar wäre (wobei e durch die DISPLACEMENT-Routine bereitgestellt wird), sondern eine Art Kurzadresse für xy, damit auch der Inhalt des Indexregisters selbst manipulierbar ist (z. B. mit INC IX).

Die nur angedeuteten Routinen LISTE 2 und LISTE 3 realisieren schließlich entsprechend (3.3.1.19.) die zusätzlichen Teile des Befehlsschlüssels.

Zusammenfassende Wertung

- Die Dekodierarbeiten zur Kodeanalyse sind recht umfangreich, obwohl hier nicht behauptet wird, daß das beschriebene Modell in dieser Beziehung die Hardwarelösung widerspiegelt.
- Die geringe Anzahl verschiedener externer Operationssymbole fächert in der Dekodierung zu einem breiten Spektrum differenzierter realisierender Routinen auf (z. B. „LD“ zu LDRR, LDRM, LDMR, LDRN, LDMN u. a. m.).
- Zentraler Teil jeder Realisierungsroutine ist die Ausführung derjenigen Operation, die in der Befehlsbeschreibung als „wesentlicher Bedeutungsinhalt“ angegeben wird (z. B. $r := 1$ in DECR als Bedeutung des Befehls DEC r).

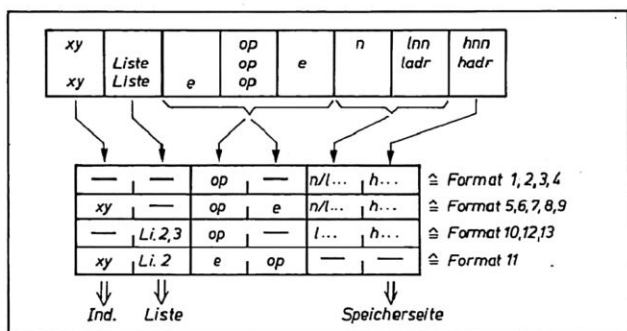


Bild 3.3.3.: Analyse der Befehlsformate

Programm für µP/µR Z80		Beispiele für die Kodierung von Befehlen aller Formate		Betrieb:	
Anfadr=1000H				Bearb.:	
Endadr=1025H				Datum:	
Start = --H				Blatt: von	
LOC	OBJECT-CODE	STMT	SOURCE STATEMENT		
	$\overline{xy}, \overline{op}, \overline{e}, \overline{n/l}, \overline{h} \dots$ $\overline{xy}, \overline{op}, \overline{e}, \overline{n/l}, \overline{h} \dots$ $\overline{xy}, \overline{L1}, \overline{e}, \overline{op}, \overline{--}$	Zelle	Name	Operat.	Operanden Kommentar
1000	47	1	LD	B, A	B:=A
01	C6, 10	2	ADD	10H	A+=16
03	11, E8, 03	3	LD	DE, 03E8H	DE:=1000
06	32, 00, 30	4	LD	(3000H), A	(3000H):=A
09	DD, 23	5	INC	IX	IX+=1
0B	FD, 21, 04, 30	6	LD	IX, 3004H	IX:=3004H
0F	DD, 22, 01, 30	7	LD	(3001H), IX	((3002H), (3001H)):=IX
13	FD, 86, FE	8	ADD	(IX-2)	A+=((IX-2))
16	FD, 36, 01, 18	9	LD	(IX+1), 18H	(IX+1):=24
1A	CB, 77	0	BIT	6, A	Z:=A ₆
1C	DD, CB, 00, FE	1	SET	7, (IX+0)	(IX)7:=1B
20	ED, 44	2	NEG		A:=-A (d.h. A+1)
22	ED, 53, 03, 30	3	LD	(3003H), DE	((3004H), (3003H)):=DE
		4			
		9			
		0			
1026H= 6+4+13+3+7+5+1000H					
Folgeadr = Länge + Anfadr					

Bild 3.3.4.: Entwurf eines Programmformulars für die manuelle Programmverschlüsselung

- Realisierende Routinen können stark untergliedert sein und von einer Vielzahl gemeinsamer Hilfsroutinen Gebrauch machen (CND, DISPLACEMENT, INSTACK, OUTSTACK u. a. m.).
- Die Signalbyte-Methode führt zur weiteren Erhöhung des Dekodierungsaufwandes und damit neben höherem Speicherbedarf infolge langer Codes auch zu einer gewissen Minderung der Abarbeitungseffektivität.
- Der Ausdruck „Interpretation des Arbeitsprogramms“ als Bedeutungsinhalt der Programmabarbeitung als Ganzes erscheint motiviert. Das vervollständigte Modell (einschließlich solcher Details wie Statusanzeige im Register F, z. B. bei DECR) kann als Vorlage für die programm- (d. h. software)mäßige Nachbildung (= Interpretator) eines Mikroprozessors auf einem völlig anderen Rechner („Wirtsrechner“) dienen.

3.3.1.7. Programmformular

Auf der Grundlage einer Analyse der vorhandenen Befehlsformate kann man ein Programmformular für die manuelle Kodierung eines mit symbolischen Befehlen geschriebenen Programms entwickeln. Durch Zusammenfassung jeweils mehrerer der insgesamt 13 Befehlsformate des Z80 lassen sich vier Grundmuster herausarbeiten (s. Bild 3.3.3.), bei denen die Bytes eines Befehlskodes maximal vier von sechs eingerichteten Spalten belegen. Dabei ist die richtige Reihenfolge der Einspeicherung beachtet, was ja bezüglich e insofern zu Schwierigkeiten führt, als e das dritte Befehlsbyte sein muß und daher sowohl vor wie auch nach dem op-Byte stehen kann. Die vorgenommene Spalteneinteilung sichert dann folgende drei visuelle Kontrollmöglichkeiten:

- Indizierung mit xy wird ganz linksbündig angegeben, so daß in dieser Spalte nur die Hexakodes DD, FD auftreten können.

- Ergänzende Befehlslisten sind in der zweiten Spalte kenntlich, in der daher nur die Hexakodes CB, ED auftreten können.
- Rechtsbündig sammeln sich in Spalte 6 alle höherwertigen Hälften von doppeltlangen Konstanten und Adressen, hnn bzw. hadr. Im Fall von Adressen bezeichnet jedes hadr eine unter den maximal 256 Speicherseiten (bei Z80) zu je 256 Byte, innerhalb der dann „ladr“ die Relativadresse des adressierten Bytes angibt. Die Vertauschung der Reihenfolge der Bytes beim Übergang von der externen Notation (hnn, lnn) bzw. (hadr, ladr) zur Speicherbelegung wird damit übersichtlicher. Mehr noch, da sich die Belegungen B_{AP} und B_w eines Programms gewöhnlich auf wenigen Speicherseiten zusammendrängen, dürfen in der rechten Formarspalte nur die Nummern dieser Seiten auftreten, was leicht zu überblicken ist.

Ein Programmformular mit Beispielen für die Kodierung von Z80-Befehlen aller Formate ist im Bild 3.3.4. gegeben. Die Grundmuster der Formate sind am Kopf der Spalte OBJECT-CODE aufgeführt. In der linken Spalte LOC figurieren nur Hexadressen, die den jeweiligen Speicherplatz des ersten Bytes eines Befehlskodes angeben (s. dazu bereits Abschnitt 3.1.4.). Eventuellen weiteren Bytes des Befehlskodes sind die folgenden Hexadressen „stumm“ zugeordnet, wobei Leerspalten zwischen den kodierten Bytes keinerlei Auswirkungen auf diese stumme Zählung oder auf die Speicherbelegung haben, sondern lediglich die genannten visuellen Kontrollen sichern. Die in der LOC-Spalte aufgeführten Adressen springen dann in genau der gleichen Weise, wie der Befehlszähler bei der Programmverarbeitung gemäß Beziehung (3.3.1.12.) schaltet.

Um mögliche Fehler in der LOC-Spalte zu erkennen, ist am Fuß des Formulars eine weitere Kontrolle vorgesehen. Zu diesem Zweck ist in jeder Spalte des OBJECT-CODE die Anzahl der in dieser Formarspalte kodierten Bytes zu ermitteln; die Summe über alle Spalten ergibt die Länge des Programmteils. Bezeichnet man mit Anfadr die Adresse des Kopfbytes eines Programmformulars, mit Folgeadr die Adresse des dem Schlußbyte folgenden Bytes (das zum Kopfbyte des folgenden Formulars werden könnte), so muß die Balance gelten:

$$\text{Folgeadr} = \text{Länge} + \text{Anfadr} \quad (3.3.1.20.)$$

Im rechten Formarteil werden unter SOURCE STATEMENT die Befehle in externer Form notiert und, durch Semikolon abgesetzt, kommentiert. In der STMT-Spalte wird die Zeilennummer angegeben. Damit ist das Formular dem Protokoll eines Assemblerprogramms (s. Bild 3.1.10.) weitgehend ähnlich. Die Spalte Name wird in der Assemblersprache zur Definition symbolischer Bezeichnungen genutzt; in der MS-Programmierung muß infolge Fehlens dieses Apparates mit Direktadressen aus der LOC-Spalte gearbeitet werden.

3.3.2. Herkömmliche Maschinenbefehle

In dieser Zusammenstellung nach Befehlsgruppen wird in gedrängter Form eine Übersicht über diejenigen Maschinenbefehle von Mikroprozessoren (am Beispiel des Z80) gegeben, die vom allgemeinen EDV-Standpunkt als herkömmlich gelten können und auch weitgehend frei von Raffinesse sind. Zur Beschreibung des Bedeutungsinhaltes werden umfassend Kürzel verwendet, wobei ihre von Mal zu Mal wechselnden Bedeutungen genau ausgewiesen sind. Mit Hilfe dieser Kürzel sind auch die externen Befehlsformen selbst notiert; durch Aufsuchen der abgeleiteten Befehle in den Befehlsschlüsseln (Tafeln 3.2.2., 3.2.4., 3.3.1.) können die Implementierung der Operation auf dem jeweiligen Mikroprozessor und auch der Operationscode selbst ermittelt werden. Insbesondere gelten folgende Kürzel:

- s, ss $\triangleq$ „Quelloperand“, zweiter Operand einer dyadischen Operation
- t, tt $\triangleq$ „Zieloperand“, Resultat, erster Operand einer dyadischen Operation.

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

3.3.2.1. Setzen von Anfangsständen

LD t,n ; t:=n, t ≡ r, (xx)	(10 Befehle)
LD tt,nn ; tt:=nn, tt ≡ BC, DE, xx, SP	(6 Befehle)
XOR A ; (Z, CY, A) := (1B, 0B, 00H)	(1 Befehl)
SUB A ; (Z, CY, A) := (1B, 0B, 00H)	(1 Befehl)
SET i,t ; t _i :=1B, 0 ≤ i ≤ 7, t ≡ r, (xx)	(64 Befehle)
RES i,t ; t _i :=0B, 0 ≤ i ≤ 7, t ≡ r, (xx)	(64 Befehle)
SCF ; CY:=1B	(1 Befehl)
JP adr ; PC:=adr	(1 Befehl)

Zunächst sind alle Operationen genannt worden, die zum Setzen definierter Werte in Architekturelementen dienen können, denn wenigstens bei Beginn der Programmabarbeitung sind diese Elemente (außer der Speicherbelegung B_{AD} und B_{CD}) ja unbestimmt. Da es sich um keine homogene Befehlsgruppe handelt, erscheinen diese Befehle an entsprechenden anderen Stellen noch einmal in ihrem natürlichen Zusammenhang.

Beispiele 1, 2:

Problem: Löschen der Register r (durch Setzen auf 00H)

LD A,00H	LD A,00H
LD B,00H	LD BC,0000H
...	LD DE,0000H
LD L,00H	LD HL,0000H
(14 Byte)	(11 Byte)

3.3.2.2. Laden, Byte

LD t,n ; t:=n, t ≡ r, (xx)	(10 Befehle)
LD r1,s ; r1:=s, s ≡ r, (xx)	(70 Befehle)
LD (xx),r ; (xx):=r	(21 Befehle)
LD A,s ; A:=s, s ≡ (BC),(DE),(adr),I,R	(5 Befehle)
LD t,A ; t:=A, t ≡ (BC),(DE),(adr),I,R	(5 Befehle)

Laden ist der allgemeine Begriff für eine Transportoperation; er umfaßt die differenzierten Begriffe Lesen und Speichern (Datenbewegung in der Richtung Speicher → allg. Register → Akku bzw. umgekehrt) sowie Umspeichern (Datenbewegung innerhalb des Speichers oder gleichartiger Register). Eine Kombination mit Testen ist nicht vorgesehen, so daß Flag F stets unbeeinflusst bleibt.

Beispiele 3 bis 5:

Problem: Löschen der Register r (durch Umspeichern)

LD A,00H	LD A,00H	LD BC,0000H
LD B,A	LD B,A	LD D,B
LD C,A	LD C,B	LD E,C
...	...	...
LD L,A	LD L,H	LD A,B

3.3.2.3. Laden, Doppelbyte (ohne Kelleroperationen)

LD tt,nn ; tt:=nn,	tt ≡ BC, DE, xx, SP	(6 Befehle)
LD tt,(adr) ; tt:=((adr+1),(adr)),	tt ≡ BC, DE, xx, SP	(6 Befehle)
LD (adr),ss ; ((adr+1),(adr)):=ss,	ss ≡ BC, DE, xx, SP	(6 Befehle)
LD SP,xx ; SP:=xx		(3 Befehle)

Das Laden erfolgt im internen Ablauf byteweise in (auch zeitlich) genauer Reihenfolge in bezug auf nieder- und höherwertiges Byte. Zum Beispiel gilt:

rr:=nn ≡ (hrr,lrr):=(hnn,lrr) ≡ lrr:=lrr; hrr:=hnn

Häufig wird ein geladenes Doppelregister anschließend als Adressensponder verwendet.

Beispiele 6 bis 9:

Problem: ((8005H), ..., (8000H)) := (H,L,D,E,B,C)

Direktadressierung

LD A,H
LD (8005H),A
LD A,L
LD (8004H),A
LD A,D
LD (8003H),A
...
LD A,C

LD (8000H),A
(24 Bytes)
Indizierung
LD IX,8005H
LD (IX+0),H
LD (IX-1),L
...
LD (IX-5),C
(22 Bytes)

Adressensponder

LD A,H
LD (8005H),A
LD A,L
LD HL,8004H
LD (HL),A
LD L,03H
LD (HL),D
...
LD L,00H
LD (HL),C
(21 Bytes)
Doppelbytes
LD (8004H),HL
LD (8002H),DE
LD (8000H),BC
(11 Bytes)

3.3.2.4. Zählen

INC t ; t+=1	INC tt ; tt+=1	(10+6 Befehle)
DEC t ; t-=1	DEC tt ; tt-=1	(10+6 Befehle)
t ≡ r, (xx)	tt ≡ BC, DE, xx, SP	

Das in der EDV typische Fortschalten (Erhöhen, Erniedrigen) des Wertes einer Größe t um den Wert einer anderen Größe s wird neuerdings mit Schaltoperationen angegeben:

$t := s \oplus t := t + s$; $t := s \oplus t := t - s$

Zählen ist Schalten um den impliziten Operanden Eins. Das Resultat hat kein Vorzeichen, sondern wird gegebenenfalls einer Modulorechnung unterworfen, z. B. gilt für einen Bytezähler:

$(255 + := 1) \bmod 2^8 = 0$, $(0 - := 1) \bmod 2^8 = 255$

Bytezähler beeinflussen zwar das Flagbit CY nicht, für das Flagbit Z gilt aber

t += 1	t -= 1	Z
(t += 1) = 0	(t -= 1) = 0	1
(t += 1) ≠ 0	(t -= 1) ≠ 0	0

Doppelbytezähler beeinflussen das Flagwort gar nicht.

Beispiele 10 bis 13, 14 bis 16:

Problem: (8000H+i) := 255-i für i = 0(1)5 bzw. i = 5(-1)0

Adressen erhöhen

LD HL,8000H	LD HL,7FFFH
LD A,0FFH	LD A,00H
LD (HL),A	INC HL
INC HL	DEC A
DEC A	LD (HL),A
...	...

Adressen erniedrigen

LD HL,8005H	LD HL,8006H
LD A,0FAH	LD A,0F9H
LD (HL),A	DEC HL
DEC HL	INC A
INC A	LD (HL),A
...	...

Schaltoperationen finden insbesondere in den Substrukturen von Zyklen ihre Anwendung, worauf die durch Klammerung hervorgehobenen stereotypen Sequenzen schon nachdrücklich hinweisen.

Die Lösungen in der U-808-D-Version fallen komplizierter aus, weil Doppelbytebefehle für Laden und Zählen fehlen.

Tafel 3.3.1.: Z-80-Befehlsschlüssel in drei Listen

Liste 1

oktal	r	rr	Relat.-Sprung	rr:=nn; xx+=rr	(rr):=A	rr+=1	r+=1	r-=1	r:=n	Verschieben
00.	B	BC	00 NOP	01 LD BC,nn	02 LD (BC),A	03 INC BC	04 INC B	05 DEC B	06 LD B,n	07 RLCA
01.	C		08 EX AF,AF'	09 ADD xx,BC	0A LD A,(BC)	0B DEC BC	0C INC C	0D DEC C	0E LD C,n	0F RRCA
02.	D	DE	10 DJNZ e	11 LD DE,nn	12 LD (DE),A	13 INC DE	14 INC D	15 DEC D	16 LD D,n	17 RLA
03.	E		18 JR e	19 ADD xx,DE	1A LD A,(DE)	1B DEC DE	1C INC E	1D DEC E	1E LD E,n	1F RRA
04.	H	xx	20 JR NZ,e	21 LD xx,nn	22 LD (adr),xx	23 INC xx	24 INC H	25 DEC H	26 LD H,n	27 DAA
05.	L		28 JR Z,e	29 ADD xx,xx	2A LD xx,(adr)	2B DEC xx	2C INC L	2D DEC L	2E LD L,n	2F CPL
06.	(xx)	SP	30 JR NC,e	31 LD SP,nn	32 LD (adr),A	33 INC SP	34 INC (xx)	35 DEC (xx)	36 LD (xx),n	37 SCF//SETC
07.	A		38 JR C,e	39 ADD xx,SP	3A LD A,(adr)	3B DEC SP	3C INC A	3D DEC A	3E LD A,n	3F CCF//CPLC

r1	r	B	C	D	E	H	L	(xx)	A
10.	B	40 LD B,B	41 LD B,C	42 LD B,D	43 LD B,E	44 LD B,H	45 LD B,L	46 LD B,(xx)	47 LD B,A
11.	C	48 LD C,B	49 LD C,C	4A LD C,D	4B LD C,E	4C LD C,H	4D LD C,L	4E LD C,(xx)	4F LD C,A
12.	D	50 LD D,B	51 LD D,C	52 LD D,D	53 LD D,E	54 LD D,H	55 LD D,L	56 LD D,(xx)	57 LD D,A
13.	E	58 LD E,B	59 LD E,C	5A LD E,D	5B LD E,E	5C LD E,H	5D LD E,L	5E LD E,(xx)	5F LD E,A
14.	H	60 LD H,B	61 LD H,C	62 LD H,D	63 LD H,E	64 LD H,H	65 LD H,L	66 LD H,(xx)	67 LD H,A
15.	L	68 LD L,B	69 LD L,C	6A LD L,D	6B LD L,E	6C LD L,H	6D LD L,L	6E LD L,(xx)	6F LD L,A
16.	(xx)	70 LD (xx),B	71 LD (xx),C	72 LD (xx),D	73 LD (xx),E	74 LD (xx),H	75 LD (xx),L	76 HALT	77 LD (xx),A
17.	A	78 LD A,B	79 LD A,C	7A LD A,D	7B LD A,E	7C LD A,H	7D LD A,L	7E LD A,(xx)	7F LD A,A

ω	r	B	C	D	E	H	L	(xx)	A
20.	A+=r	80 ADD B	81 ADD C	82 ADD D	83 ADD E	84 ADD H	85 ADD L	86 ADD (xx)	87 ADD A
21.	A+=(r+CY)	88 ADC B	89 ADC C	8A ADC D	8B ADC E	8C ADC H	8D ADC L	8E ADC (xx)	8F ADC A
22.	A-=r	90 SUB B	91 SUB C	92 SUB D	93 SUB E	94 SUB H	95 SUB L	96 SUB (xx)	97 SUB A
23.	A-=(r+CY)	98 SBC B	99 SBC C	9A SBC D	9B SBC E	9C SBC H	9D SBC L	9E SBC (xx)	9F SBC A
24.	A:=AAr	A0 AND B	A1 AND C	A2 AND D	A3 AND E	A4 AND H	A5 AND L	A6 AND (xx)	A7 AND A
25.	A:=Aθr	A8 XOR B	A9 XOR C	AA XOR D	AB XOR E	AC XOR H	AD XOR L	AE XOR (xx)	AF XOR A
26.	A:=Avr	B0 OR B	B1 OR C	B2 OR D	B3 OR E	B4 OR H	B5 OR L	B6 OR (xx)	B7 OR A
27.	F:=F(A-r)	B8 CP B	B9 CP C	BA CP D	BB CP E	BC CP H	BD CP L	BE CP (xx)	BF CP A

cmd	rr	Rückk. bed.	Entkellern	Sprung bedingt	E/A; <=> Unterbr.	Aufruf bedingt	Kellern u. a.	A:=A n	Restart
30.	Z=0 BC	C0 RET NZ	C1 POP BC	C2 JP NZ,adr	C3 JP adr	C4 CALL NZ,adr	C5 PUSH BC	C6 ADD n	C7 RST 0
31.	Z=1	C3 RET Z	C9 RET	CA JP Z,adr	CB „Liste 2“	CC CALL Z,adr	CD CALL adr	CE ADC n	CF RST 8
32.	CY=0 DE	D0 RET NC	D1 POP DE	D2 JP NC,adr	D3 OUT (n),A	D4 CALL NC,adr	D5 PUSH DE	D6 SUB n	D7 RST 10H
33.	CY=1	D8 RET C	D9 EXX	DA JP C,adr	DB IN A,(n)	DC CALL C,adr	DD „IX“	DE SBC n	DF RST 18H
34.	P/V=0 xx	E0 RET PO	E1 POP xx	E2 JP PO,adr	E3 EX (SP),xx	E4 CALL PO,adr	E5 PUSH xx	E6 AND n	E7 RST 20H
35.	P/V=1	E8 RET PE	E9 JP xx	EA JP PE,adr	EB EX DE,HL	EC CALL PE,adr	ED „Liste 3“	EE XOR n	EF RST 28H
36.	S=0 AF	F0 RET P	F1 POP AF	F2 JP P,adr	F3 DI	F4 CALL P,adr	F5 PUSH AF	F6 OR n	F7 RST 30H
37.	S=1	F8 RET M	F9 LD SP,xx	FA JP M,adr	FB EI	FC CALL M,adr	FD „IX“	FE CP n	FF RST 38H

xx	HL	IX	IY	(xx)	(HL)	(IX+e)	(IY+e)
op	DD	op	FD	op	DD	op e	FD op e

Liste 2; Signalbyte: CB

oktal	Verschieb.	B	C	D	E	H	L	(xx)	A
00.	CY←r←CY	00 RLC B	01 RLC C	02 RLC D	03 RLC E	04 RLC H	05 RLC L	06 RLC (xx)	07 RLC A
01.	→r→CY	08 RRC B	09 RRC C	0A RRC D	0B RRC E	0C RRC H	0D RRC L	0E RRC (xx)	0F RRC A
02.	CY←r←CY	10 RL B	11 RL C	12 RL D	13 RL E	14 RL H	15 RL L	16 RL (xx)	17 RL A
03.	→r→CY	18 RR B	19 RR C	1A RR D	1B RR E	1C RR H	1D RR L	1E RR (xx)	1F RR A
04.	CY←r←CY	20 SLA B	21 SLA C	22 SLA D	23 SLA E	24 SLA H	25 SLA L	26 SLA (xx)	27 SLA A
05.	→r→CY	28 SRA B	29 SRA C	2A SRA D	2B SRA E	2C SRA H	2D SRA L	2E SRA (xx)	2F SRA A
06.		30	31	32	33	34	35	36	37
07.	0→r→CY	38 SRL B	39 SRL C	3A SRL D	3B SRL E	3C SRL H	3D SRL L	3E SRL (xx)	3F SRL A

BIT i,r // TEST i,r	RES i,r // CLR i,r	SET i,r
10. 0 40 41 42 43 44 45 46 47	20. 0 80 81 82 83 84 85 86 87	30. 0 C0 C1 C2 C3 C4 C5 C6 C7
11. 1 48 49 4A 4B 4C 4D 4E 4F	21. 1 88 89 8A 8B 8C 8D 8E 8F	31. 1 C8 C9 CA CB CC CD CE CF
12. 2 50 51 52 53 54 55 56 57	22. 2 90 91 92 93 94 95 96 97	32. 2 D0 D1 D2 D3 D4 D5 D6 D7
13. 3 58 59 5A 5B 5C 5D 5E 5F	23. 3 98 99 9A 9B 9C 9D 9E 9F	33. 3 D8 D9 DA DB DC DD DE DF
14. 4 60 61 62 63 64 65 66 67	24. 4 A0 A1 A2 A3 A4 A5 A6 A7	34. 4 E0 E1 E2 E3 E4 E5 E6 E7
15. 5 68 69 6A 6B 6C 6D 6E 6F	25. 5 A8 A9 AA AB AC AD AE AF	35. 5 E8 E9 EA EB EC ED EE EF
16. 6 70 71 72 73 74 75 76 77	26. 6 B0 B1 B2 B3 B4 B5 B6 B7	36. 6 F0 F1 F2 F3 F4 F5 F6 F7
17. 7 78 79 7A 7B 7C 7D 7E 7F	27. 7 B8 B9 BA BB BC BD BE BF	37. 7 F8 F9 FA FB FC FD FE FF

(xx)	(HL)	(IX+e)	(IY+e)
CB	op	DD CB e op	FD CB e op

Liste 3; Signalbyte: ED

oktal	r	rr	Eingabe (C)	Ausgabe (C)	HL+=rr+CY	(adr):=rr u. a.	A:=A	Rückk. spez.	Unterbr.-mod.	I,R; Schieben
10.	B	BC	40 IN B,(C)	41 OUT (C),B	42 SBC HL,BC	43 LD (adr),BC	44 NEG	45 RETN	46 IM 0	47 LD I,A
11.	C		48 IN C,(C)	49 OUT (C),C	4A ADC HL,BC	4B LD BC,(adr)	4C	4D RETI	4E	4F LD R,A
12.	D	DE	50 IN D,(C)	51 OUT (C),D	52 SBC HL,DE	53 LD (adr),DE	54	55	56 IM 1	57 LD A,I
13.	E		58 IN E,(C)	59 OUT (C),E	5A ADC HL,DE	5B LD DE,(adr)	5C	5D	5E IM 2	5F LD A,R
14.	H	HL	60 IN H,(C)	61 OUT (C),H	62 SBC HL,HL	63	64	65	66	67 RRD
15.	L		68 IN L,(C)	69 OUT (C),L	6A ADC HL,HL	6B	6C	6D	6E	6F RLD
16.		SP	70 TEST(C)	71	72 SBC HL,SP	73 LD (adr),SP	74	75	76	77
17.	A		78 IN A,(C)	79 OUT (C),A	7A ADC HL,SP	7B LD SP,(adr)	7C	7D	7E	7F

Grupp.-Op.	Laden	Vergleichen	Eingeben	Ausgeben
24. +=1, einf.	A0 LDI	A1 CPI	A2 INI	A3 OUTI
25. -=1, einf.	A8 LDD	A9 CPD	AA IND	AB OUTD
26. +=1, zykl.	B0 LDIR	B1 CPJR	B2 INIR	B3 OTIR
27. -=1, zykl.	B8 LDDR	B9 CPDR	BA INDR	BB OTDR

Außerdem ist Lösung 11 nicht praktikabel, weil die Adresse in HL sofort einen Wechsel der Speicherseite erfordert. Zählen im Akku fehlt ebenfalls. Andererseits ist indirekte Adressierung die einzige Möglichkeit für Speicherzugriff (s. auch Zusammenhang 3.3.1.7.):

MVI H,80H	MVI H,80H	MVI H,80H
MVI L,00H	MVI L,05H	MVI L,06H
MVI B,0FFH	MVI B,0FAH	MVI B,0F9H
MOV M,B	MOV M,B	INR B
INR L	DCR L	DCR L
DCR B	INR B	MOV M,B

Die beiden vorstehenden Gruppen von Beispielen illustrieren überzeugend die faktischen und verständnisfähigen Erleichterungen im Programmieren mit einem differenzierten Befehlssatz.

3.3.2.5. Arithmetik, Bytes

ADD s ;A+=s ADC s ;A+=(s+CY) (je 11 Befehle)
 SUB s ;A-=s SBC s ;A-=(s+CY) (je 11 Befehle)
 s ≡ n, r, (xx)

Die Byte-Arithmetik ist von der Hardware her auf Addition und Subtraktion beschränkt, wobei die Operanden ganze Zahlen ohne Vorzeichen aus dem Bereich $0 \leq t, s \leq 255$ darstellen. Wegen der Festlegung $t \equiv A$ können diese Operationen als Schalten des Akkumulators um den Wert des zweiten Operanden s bzw. $s+CY$ angegeben werden. Die Subtraktion ist meist als Addition des Zweierkomplements dieses zweiten Operanden realisiert. Resultate haben kein Vorzeichen; um sie im angegebenen Wertebereich zu halten, werden sie erforderlichenfalls einer Modulo-2⁸-Rechnung unterworfen (in den genannten Befehlsbedeutungen weggelassen). Das Flagwort wird vollständig Neubestimmt und erlaubt u. a. auch das etwa eingetretene und bereits kompensierte Über- oder Unterschreiten des Wertebereiches noch im Nachhinein zu diagnostizieren.

Beispiele 17, 18 :

Problem: Kompensieren und nachträgliche Diagnose des Einhaltens des Wertebereiches bei Addition und Subtraktion:

LD A,80H LD A,80H ;A:=128
 ADD s SUB s ;s ≡ n, n = 127, 128, 129

Addition:

A _{alt}	=	10000000	10000000	10000000
+s	=	01111111	10000000	10000001
A _{alt} +s	=	01111111	10000000	10000001
CY _{neu}	=	0	1	1
(Z,CY,A _{neu})	:=	(0,0,11111111)	(1,1,00000000)	(0,1,00000001)

Subtraktion:

A _{alt}	=	10000000	10000000	10000000
-s	=	10000000	01111111	01111110
A _{alt} -s	=	1	1	1
A _{alt} +s+1	=	10000001	10000000	01111111
CY _{neu}	=	1	1	0
(Z,CY,A _{neu})	:=	(0,0,00000001)	(1,0,00000000)	(0,1,11111111)

Verallgemeinernd kann man die Beeinflussung der Flagbits Z und CY wie folgt zusammenfassen:

Addition

A+=s	A+=(s+CY)	Z	CY
A+s>256	A+s+CY>256	0	1
A+s=256	A+s+CY=256	1	1
0<A+s<256	0<A+s+CY<256	0	0
0=A+s	0=A+s+CY	1	0

Subtraktion

A-=s	A-=(s+CY)	Z	CY
A-s>0	A-s-CY>0	0	0
A-s=0	A-s-CY=0	1	0
-256<A-s<0	-256<A-s-CY<0	0	1
-256=A-s	-256=A-s-CY	1	1

Die Befehle ADD und SUB sind erste Beispiele für die dynamische Bestimmung des CY-Flagbits. Das CY-Bit kann von ADC oder SBC unter erneuter Bildung eines CY weiterverarbeitet werden.

Beispiele 19, 20 :

Problem: $tt \pm nn$ mit $tt := ((adr+1), (adr))$, $0 \leq tt \pm nn < 2^{16}$

LD HL,adr	LD HL,adr	;HL:=adr
LD A,(HL)	LD A,(HL)	;A:=lth
ADD Inn	SUB Inn	; (CY,A) := lth ± Inn
LD (HL),A	LD (HL),A	; lth := (lth ± Inn) mod 2 ⁸
INC HL	INC HL	;HL:=adr+1
LD A,(HL)	LD A,(HL)	;A:=hth
ADC hnn	SBC hnn	; (CY,A) := (0, hth ± (hnn+CY))
LD (HL),A	LD (HL),A	;

Abarbeiten dieser Befehlssequenzen mit geeigneten Daten, z. B.

```

tt = (00000010,10000000)
+) nn = (00000000,10000001)
+) CY = 1
-----
tt+nn = (00000011,00000001)
tt = (00000010,10000000)
-) nn = (00000000,10000001)
-) CY = 1
-----
tt-nn = (00000010,11111111)

```

zeigt, daß das CY auch den Einer-Übertrag zu den bzw. das Einer-Borgen von den höherwertigen Resultatstellen leistet und damit Softwarelösungen für Mehrbyte-Arithmetik ermöglicht.

Abschließend soll auf den speziellen Charakter der Befehle für $s \equiv A$ hingewiesen werden:

ADD A ;(CY,A):=(A₇, (A₆, ..., A₁, A₀, 0)), kurz: A:=2·A
 ADC A ;(CY,A):=(A₇, (A₆, ..., A₁, A₀, CY))
 SUB A ;(Z,CY,A):=(1,0,00000000), kurz: CY,A:=0
 SBC A ;(Z,CY,A):=(CY,CY,-CY)

3.3.2.6. Vergleich, Bytes

CP s ;F:=(FA-s), s ≡ n, r, (xx) (11 Befehle)

Die Vergleichsoperanden $t \equiv A$ und s werden arithmetisch interpretiert und zur Bildung der Differenz $A-s \equiv A+(s+1)$ herangezogen. Im Effekt wird das Flagwort in der gleichen Weise bestimmt, wie das SUB s tun würde, lediglich der Akkumulator bleibt bei CP s unverändert. Damit diagnostizieren Z und CY für eine fiktive (oder tatsächlich noch nachfolgende) Subtraktion im Vorhinein das Einhalten des Wertebereiches, und das läuft auf den Vergleich von A mit s hinaus.

A-s ≥ 0 bzw. A ≥ s	Z	CY	
A-s > 0 bzw. A > s	0	0	
A-s = 0 bzw. A = s	1	0	; stets bei CP A
A-s < 0 bzw. A < s	0	1	

3.3.2.7. Arithmetik, Doppelbytes

ADD xx,ss ;xx+:=ss, ss ≡ BC, DE, xx, SP (3-4 Befehle)

ADC HL,ss ;HL+:=ss+CY, ss ≡ BC, DE, HL, SP (4 Befehle)

SBC HL,ss ;HL-:=ss+CY, ss ≡ BC, DE, HL, SP (4 Befehle)

Doppelbyte-Arithmetik ist hardwaremäßig auf Addition und Subtraktion beschränkt, wobei die Operanden als ganze Zahlen ohne Vorzeichen aus dem Bereich $0 \leq tt, ss \leq 2^{16}-1$ interpretiert werden. Das ermöglicht auch und vor allen Dingen arithmetische Operationen mit Adressen. Resultate werden, um sie im angegebenen Wertebereich zu halten, erforderlichenfalls einer Modulo-2¹⁶-Rechnung unterworfen. Das Flagwort wird bei ADC und SBC vollständig Neubestimmt, bei ADD wird nur CY dynamisch geändert. Die Einhaltung des Wertebereiches ist folgendermaßen zu diagnostizieren:

xx+:=ss	Z	CY
xx+ss ≥ 2 ¹⁶	-	1
0 ≤ xx+ss < 2 ¹⁶	-	0

HL+:=ss+CY	Z	CY	HL-:=ss+CY	Z	CY
HL+ss+CY > 2 ¹⁶	0	1	HL-ss-CY < 0	0	0
HL+ss+CY = 2 ¹⁶	1	1	-2 ¹⁶ < HL-ss-CY > 0	1	0
0 < HL+ss+CY < 2 ¹⁶	0	0	HL-ss-CY = 0	0	1
0 = HL+ss+CY	1	0	-2 ¹⁶ = HL-ss-CY	1	1

Beispiele 21, 22, 23:

Problem: $xx := SP$ als „inverse“ Operation zu LD SP, xx.
 Problem: $tt \pm nn$ mit $tt = ((adr+1), (adr))$, $0 \leq tt \pm nn < 2^{16}$.

```
LD xx,0000H      LD HL,(adr)      LD HL,(adr)
ADD xx,SP         LD BC,nn         LD BC,nn
                  ADD HL,BC         CP A ;CY:=0
                  LD (adr),HL      SBC HL,BC
                  LD (adr),HL
```

3.3.2.8. Dyadische Logik, Bytes

AND s ;A:=A ∧ s }
 XOR s ;A:=A ⊕ s } s ≡ n, r, (xx) (je 11 Befehle)
 OR s ;A:=A ∨ s }

Jeder dieser Befehle interpretiert seine beiden Operanden $t \equiv A$ und s als Listen von Wahrheitswerten (kodiert als Null-Eins-Muster) und führt die logische Operation parallel über allen einander entsprechenden Bitpaaren aus:

$t = (t_7, \dots, t_i, \dots, t_0)$ $7 \geq i \geq 0$
 $s = (s_7, \dots, s_i, \dots, s_0)$ $\omega \in \{\wedge, \oplus, \vee\}$

$t \omega s = (\dots, t_i \omega s_i, \dots)$ $t_i \omega s_i = s_i \omega t_i$

Einer der Operanden (meist s) wird als „Maske“ angesehen; die Einsen seines Codes heißen Maskenbits. Die den Maskenbits entsprechenden Bits des anderen Operanden heißen maskiert, jedoch bezeichnet man meist den ganzen Operanden als maskiert. Mit diesen Begriffen können die logischen Operationen folgendermaßen charakterisiert werden:

	$\wedge$	$\oplus$	$\vee$
maskierter Operand:	abcd1010	abcd1010	abcd1010
Maske:	11001100	11001100	11001100
Resultat:	ab001000	ābcd0110	11cd1110

- AND reproduziert die maskierten Bits, annulliert die übrigen
- XOR komplementiert die maskierten Bits, reproduziert die übrigen
- OR setzt die maskierten Bits, reproduziert die übrigen.

Das Flagwort wird Neubestimmt:

$S := t_i \omega s_i$; $CY := 0$
 $t_i \omega s_i = 0$ gilt Z $t_i \omega s_i = 1$ gilt P
 für alle i 1 für gerade Anzahl von Malen 1
 nicht für alle i 0 für ungerade Anzahl von Malen 0

Eine spezielle Wirkung hat

XOR A ;(S,Z,P,CY,A):=(0B,1B,1B,0B,00H)

Hauptanwendungsgebiet der logischen Operationen ist die Modellierung von Schaltfunktionen (s. bereits Abschnitt 3.1.5. bzw. die Bilder 3.1.14. und 3.1.15.), jedoch ermöglichen Operandenvortausch auf Grund des Kommutativgesetzes, Interpretation des Operandenbitmusters in einer anderen Wertart (z. B. arithmetisch) und der übrige Befehlsreichtum oft eine Art Simultanverarbeitung mit überraschenden Lösungen.

Beispiele 24 bis 27:

Problem: Sei $A = 000000yx$. Man erzeuge in A ein Bit $z = y \wedge x$

```
LD C,A ;A:=000000yx
ADD A ;A:=000000yx0
AND C ;A:=000000z0
CP 03H ;A:=000000yx, CY:=yx<3
ADC A ;A:=000000yxz
XOR 01H ;A:=000000yxz
CP 03H ;A:=000000yx
SBC A ;A:=zzzzzzzz
INC A ;A:=0000000z
INC A ;A:=000000zvz, v=y⊕x
```

Die Maskierungstechnik bietet Möglichkeiten für eine Symbolmanipulierung bzw. Textverarbeitung.

Beispiele 28 bis 30:

Problem: Sei $B = (hn, ln)$, $C = (h, 0)$; gefordert: $D = (h, ln)$, $E = (hn, 0)$

```
LD A,B ;A:=(hn,ln)
AND 0FH ;A:=(0,ln)
OR C ;A:=(h,ln)
LD D,A ;
LD A,B ;A:=(hn,ln)
AND 0F0H ;A:=(hn,0)
LD E,A ;
LD A,B ; LD A,B
AND 0FH ; AND 0F0H
OR C ; LD E,A
LD D,A ; XOR B
XOR C ; OR C
XOR B ; LD D,A
LD E,A ;
```

3.3.2.9. Monadische Logik und spezielle Arithmetik

SCF ;CY:=1, alias SETC (1 Befehl)
 CCF ;CY:=CY, alias CPLC (1 Befehl)
 CPL ;A:=A, d. h. $A := (\bar{A}_7, \dots, \bar{A}_1, \bar{A}_0)$ (1 Befehl)
 NEG ;A:=-A, d. h. $A := 00H - A \equiv -A + 1$ (1 Befehl)
 DAA ;Dezimalkorrektur im Akku bei BCD-Verarbeitung (1 Befehl)

Während das Flagbit CY leicht gelöscht werden kann, z. B. mit CPA, ist für das Setzen ein eigener Befehl notwendig. Bestimmte Anwendungen erfordern ein komplementiertes CY-Bit.

Beispiele 31 und 32:

Problem: Varianten der Beispiele 25 und 26

```
CP 03H ;A:=000000yx
CCF ;CY:=¬yx≥3, d. h. CY:=z
ADC A ;A:=000000yxz
CP 03H ;A:=000000yz
CCF ;CY:=z
SBC A ;A:=zzzzzzzz
```

Das bitweise Komplement des Bytes im Akkumulator liefert CPL. Im häufigsten Anwendungsfall der Bildung des Zweierkomplementes von der im Akku stehenden ganzen Zahl ohne Vorzeichen, $-A \equiv A + 1$, wird CPL beim Z 80 durch den Spezialbefehl NEG verdrängt. Während CPL die wesentlichen Flagbits nicht berührt, bestimmt NEG das Flagwort neu; insbesondere gilt:

$-A$ Z CY
 $A_{alt} > 0$ 0 1
 $A = 0$ 1 0

Beispiele 33 bis 35:

Problem: $rr := -rr$, Angabe des Endstandes des CY-Flagbits.

```
LD A,rr XOR A LD A,rr
CPL SUB rr NEG
LD rr,A LD rr,A LD rr,A
LD A,hrr SBC A LD A,00H
CPL SUB hrr SBC hrr
LD hrr,A LD hrr,A LD hrr,A
INC rr
```

[CY unverändert] [CY:=(hrr≠0) ∧ (rr=0)] [CY:=(rr≠0)]

Der DAA-Befehl für Dezimalkorrektur des Akku gestattet eine Arithmetik im gepackten BCD-Format. Über die bereits vom 8080 her bekannte Ausführung der Addition hinaus wird beim Z 80 dank dem speziell eingerichteten Flagbit N auch die korrekte Subtraktion möglich.

3.3.2.10. Verschieben

a) Linksverschieben um eine Dualstelle

RLC s ;(CY,s):=(s₇,(s₆,...,s₁,s₀,s₇)) } s ≡ r, (xx)
 RL s ;(CY,s):=(s₇,(s₆,...,s₁,s₀,CY)) }
 SLA s ;(CY,s):=(s₇,(s₆,...,s₁,s₀,0)) } (je 10 Befehle)
 RLCA ;wie RLC A, aber nur CY beeinflusst (1 Befehl)
 RLA ;wie RL A, aber nur CY beeinflusst (1 Befehl)

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

b) Rechtsverschieben um eine Dualstelle

RRC s ;(s,CY) ::= ((s ₀ , s ₇ , s ₆ , . . . , s ₁), s ₀)	} (je 10 Befehle)
RR s ;(s,CY) ::= ((CY, s ₇ , s ₆ , . . . , s ₁), s ₀)	
SRA s ;(s,CY) ::= ((s ₇ , s ₇ , s ₆ , . . . , s ₁), s ₀)	
SRL s ;(s,CY) ::= ((0, s ₇ , s ₆ , . . . , s ₁), s ₀)	
RRCA ;wie RRC A, aber nur CY beeinflusst	(1 Befehl)
RRR ;wie RR A, aber nur CY beeinflusst	(1 Befehl)

c) Links-/Rechtsverschieben um eine BCD-Stelle ($\cong 4$ Dualstellen)

RLD	$; (A, (HL)) := ((hA, h(HL)), (l(HL), lA))$	(1 Befehl)
RRD	$; (A, (HL)) := ((hA, l(HL)), (lA, h(HL)))$	(1 Befehl)

Das Verschieben um eine Dualstelle umfaßt das sogenannte Rotieren oder zyklische Verschieben der Operandenbits entweder allein („zirkulär“) oder einschließlich des CY-Flagbits und das sogenannte Shiften oder lineare Verschieben in den Versionen arithmetisch und logisch. Damit wird das Rücken des Bitmusters eines Bytes auf seinem Standort möglich: eine Art Justieren in neuer Position unter Auffangen des bisherigen links- bzw. rechtsbündigen Bits im CY, meist für den Zweck einer nachfolgenden Weitergabe an ein „Anschlußbyte“. Während sich die Rechtsshift-Versionen deutlich in bezug auf die Behandlung des linksbündigen Bits s_7 unterscheiden (SRL: linksbündiges Nachziehen von OB; SRA; „Ausziehen“ von s_7 nach rechts, so daß auch für eine Zweierkomplementdarstellung mit s_7 als Vorzeichenstelle die Operation als Division durch 2 wirkt), ist die ausdrückliche Spezifizierung des Linksshifts als arithmetisch (= Verdoppelung, d. h. Multiplikation mit 2) ungerechtfertigt. Diese Operation verkörpert genauso die logische Version (einheitliches rechtsbündiges Nachziehen von OB), lediglich eine Auswertung der neubestimmten Flagbits $CY = s_{7,alt}$, $S = s_{6,alt}$ erlaubt eine Differenzierung der Versionen.

Interpretation von SLA s	Flag-Anforderungen
--------------------------	--------------------

	logische Version	keine
arithmetische Version	ganze Zahl ohne Vorzeichen	kein Überlauf $CY = 0$
	Zweierkomplement	Überlauf $CY = 1$
		kein Überlauf $CY \sim S$
		Überlauf $CY \oplus S$

Beispiele 36 bis 38:

Probleme: $(CY,s):=(s_7,s)$, sog. Pseudooperation „SCF 7, s“;
 $(s,CY):=(s,s_0)$, sog. Pseudooperation „SCF 0, s“;
 $(rr,CY):=((CY,rr_{i5},\dots,rr_{i1}),rr_0)$, genannt „RR rr“

RLC s	$;(s_7, (s_{i_1}, \dots, s_{i_l}, s_7))$	RRC s	RR hrr
RRC s	$;((s_7, s_{i_1}, \dots, s_{i_l}), s_7)$	RLC s	RR lrr

Beispiele 39, 40:

Probleme: $A := (A_7, B_7, \dots, L_7, 0)$; Pseudooperation „RRC rr“.

RLCA ; A = (. . . , A₀, A₇) RRC lrr } siehe „SCF 0, s“
RLC B ; RLC lrr }

RRC B	; siehe „SCF 7,s“	RR hrr	} siehe „RR rr“
RLA		RR lrr	

```
SLA A ;A=(A7,B7,...,L70)
```

Das Rotieren einer BCD-Ziffer im Akku mit den beiden Ziffern eines Speicherbytes erlaubt effektive Abwicklung von Hilfsoperationen bei der BCD-Arithmetik. Hier eine andere Illustration für diese Operationen:

Beispiele 41 und 42:

Probleme: Umstellen bzw. Doppeln der Bytehälften

$$(\text{adr}) = h_1 h_0 H.$$

LD HL,adr	;A	(adr)	LD HL,adr	;A	(adr)
LD A,(HL)	;h ₁ h ₀	h ₁ h ₀	LD A,(HL)	;h ₁ h ₀	h ₁ h ₀
RRD	;h ₁ h ₀	h ₀ h ₁	RLD	;h ₁ h ₁	h ₀ h ₀

3.3.2.11. Unbedingte Sprünge

JP	adr	;PC:=adr	(1 Befehl)
JP	xx	;PC:=xx, s. Anm. in Tafel 3.3.1.	(3 Befehle)
JR	e	;PC+=e	(1 Befehl)

Unbedingte Sprünge laden oder schalten den Befehlszähler PC bedingungslos und bestimmen auf diese Weise explizit das erste oder einzige Byte ihres Nachfolgebefehls (vergleiche Abschnitt 3.3.1.4.).

Wie allgemein üblich, soll künftig mit dem Symbol * die eigene Adresse eines beliebigen Befehls bezeichnet werden, wie sie sich aus der Belegung B_{AP} ergibt und in der LOC-Spalte des Programmformulars (s. Bild 3.3.4.) ablesbar ist. Ein direkter Sprung JP adr ist bei folgenden Gegebenheiten in einen Relativsprung JR e umwandelbar (s. Bild 3.3.5.a):

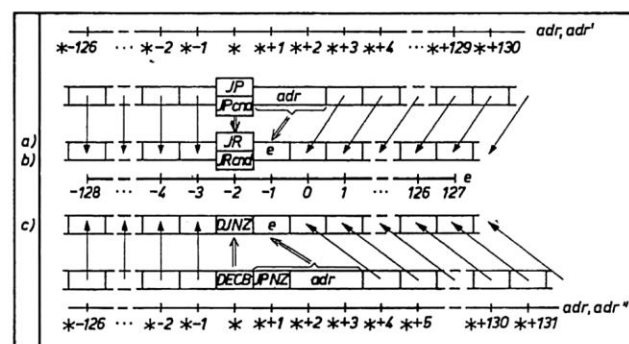
$$\begin{aligned} \text{JP adr} & \quad ; * - 126 \leq \text{adr} \leq * + 130 \\ \Rightarrow \text{JR e} & \quad ; -128 \leq e \leq 127, \quad * - 126 \leq \text{adr}' \leq * + 129 \end{aligned}$$
$$e = \text{adr}' - (*+2), \text{adr}' = \begin{cases} \text{adr, bei } \text{adr} < * \\ \text{(Rückwärtssprung)} \\ \text{adr}-1, \text{ bei } \text{adr} > *+2 \\ \text{(Vorwärtssprung)} \end{cases}$$


Bild 3.3.5.: Zur Umwandlung direkter Sprünge in a) unbedingte; b) explizit-bedingte; c) implizit-bedingte Relativsprünge

Dabei bezeichnet beim Vorwärtssprung $adr' = adr - 1$ die neue und endgültige Adresse des Nachfolgebefehls von JR e, weil infolge des kürzeren Befehlskodes von JR e die gesamte Bytefolge oberhalb $*+2$ um einen Speicherplatz zurückzuversetzen ist.

Unbedingte Sprünge treten beim Kodieren regulärer Struktogramme ausschließlich als interne und integrale Bestandteile der Kodeäquivalente von Fallauswahl und Zyklus auf. In nicht durchgängig, d. h. in irregulär strukturierten Algorithmen stellen sie als Rückwärts- oder Vorwärtssprünge die MS-Abbildungen von sog. Übergängen dar (s. linkes Teilbild 3.3.6.), wobei die dort mit $\leftarrow ?$ kenntlichen Stellen ihrerseits ebenfalls nur mit Übergängen erreichbar sind.

3.3.2.12. Bedingte Sprünge

JP cnd,adr	;Wenn cnd erfüllt, dann PC:=adr	(8 Befehle)
JR cnd,e	;Wenn cnd erfüllt, dann PC+=e	(4 Befehle)
DJNZ e	;Wenn (B-:=1)≠0, dann PC+=e, dabei keinerlei Flag-Beeinflussung	(1 Befehl)

Bei Erfüllung der in die Befehlsbedeutung integrierten Bedingung [explizit: cnd, implizit: $(B - := 1) \neq 0$] wirkt der Befehl wie ein äquivalenter unbedingter Sprung, sonst wird der Befehl ignoriert, d. h., sein textlicher Nachfolgebefehl wird aktiviert. Der genaue Ablauf ist für den Fall JP cnd, adr bereits am Ende von Abschnitt 3.3.1.4. sowie in der Abarbeitungsroutine JPBED vom Bild 3.3.2. angegeben. Beim Kodieren regulärer Struktogramme treten bedingte

	unbedingt		bedingt	
	rückwärts	vorwärts	rückwärts	vorwärts
a)				
b)	JP adr adr < * *+2 < adr		JP cnd, adr ; cnd = Bk adr < * *+2 < adr	
c)	JR e -128 ≤ e < -2 0 ≤ e ≤ 127		JR cnd, e ; cnd = Bk -128 ≤ e < -2 0 ≤ e ≤ 127	

Bild 3.3.6.: a) Übergänge; b) deren Abbildung in direkte Sprünge; c) Relativsprünge

Sprünge ebenfalls nur in den Kodeäquivalenten von Fallauswahl und Zyklus auf. In irregulären Struktogrammen stellen sie als bedingte Rückwärts- oder Vorwärtssprünge die MS-Abbildungen von sog. bedingten Übergängen dar (s. rechtes Teilbild 3.3.6.). Für die dort gezeigte Grundform und eine Reihe konkreter Bedingungen Bk ist die unmittelbare Kodierbarkeit als bedingter Sprung gegeben, wie die Zusammenstellung im Bild 3.3.7. ausweist. Für einen Teil der Bk sind die erhaltenen direkten bedingten Sprünge JP cnd, adr bei gleichen Gegebenheiten wie im Fall von JP adr in bedingte Relativsprünge umwandelbar (s. Bild 3.3.5.b). Eine weitere Transformationsmöglichkeit ist (s. Bild 3.3.5.c):

{ DEC B ; u.a. Z := (B-1) = 0, * -126 ≤ adr ≤ *+131
JP NZ, adr ; dabei ist * die Eigenadresse des DEC-Befehls
⇒ DJNZ e ; -128 ≤ e ≤ 127, * -126 ≤ adr ≤ *+129
e = adr - (*+2), adr = { adr, bei adr < * (Rückwärtsprung)
adr-2, bei adr > *+3 (Vorwärtssprung)

Eliminierung der Flag-Beeinflussung.

Vorwärtssprung: Rückversetzen der Bytefolge oberhalb *+3 um zwei Speicherplätze, Nachfolgebefehl mit Adresse adr".

Notationsform von Bk			äquivalente bedingte Sprünge		
sym&F-Bit	log. Variable	Vergleich	JP cnd, adr cnd	JR cnd, e cnd	implizite Bedingung
Fi	Z	Z=1	Z	Z	—
	CY	CY=1	C	C	—
	P,V	P=1, V=1	PE, VS	—	—
	S	S=1	M	—	—
Fi	Z	Z=0	NZ	NZ	—
	CY	CY=0	NC	NC	—
	P,V	P=0, V=0	PO, VC	—	—
	S	S=0	P	—	—
—	—	(B-1) ≠ 0	DEC B JP NZ, adr	DEC B JP NZ, e'	DJNZ e

Bild 3.3.7.: Grundform des bedingten Übergangs und Fälle unmittelbarer Kodierbarkeit als bedingte Sprünge

Um künftig Äquivalenzen, Transformationen und Optimierungen von Befehlskode einheitlich wiederzugeben, soll in der Notation von Sprungbefehlen die wirkliche, endgültige, resultative eigene Adresse des Nachfolgebefehls bef figurieren können. Dabei hat man sich bef (s. Abschnitt 3.3.1.2.) als Zuordnung in B_{AP} zu vergegenwärtigen und diesen Umstand explizit als sog. Markieren von bef anzugeben:

adr; bef

Die Platzfixierung oder -änderung von bef in B_{AP} fixiert oder ändert den Wert von adr und erfordert das systematische Einbringen dieses Wertes in alle Befehle, die auf bef mittels explizit angegebenem „adr“ Bezug nehmen:

JP adr JP cnd, adr
JR adr-(*)+2 JR cnd, adr-(*)+2 DJNZ adr-(*)+2

Bild 3.3.8. enthält die Transformationen bedingter Übergänge und der ihnen entsprechenden Befehlsformen zu oder aus den Grundformen JP Fi, adr und JP NF_i, adr. Damit werden vorgelegte Befehlsformen formal transformierbar.

Beispiel 43:

Problem: Umformung einer gegebenen Befehlsfolge.

DEC B DEC B DJNZ 3
JP Z, adr JP NZ, *+6 JP adr

Übergänge	Bk			
	1	0	0	1
symbolischer Kode	JP Bk, *+6 JP adr	JP NBk, *+6 JP adr	JP NBk, adr	JP Bk, adr
Bk = Fi		JP NF _i , *+6 JP adr		JP Fi, adr
Bk = F _i Bk = Fi, NBk = NF _i	JP Fi, *+6 JP adr		JP NF _i , adr	

Bild 3.3.8.: Transformation bedingter Übergänge und der ihnen entsprechenden Befehlsformen

3.3.2.13. Bit-Manipulieren

BIT i, s ; Z := s_i, alias TEST i, s
RES i, t ; t_i := 0B, alias CLR i, t
SET i, t ; t_i := 1B
7 ≥ i ≥ 0
t, s ≡ r, (xx)
(je 80 Befehle)

Neben der Möglichkeit, auf beliebige Speicherstellen (außer im ROM) beliebige Bits t_i laden zu können (Löschen = Rücksetzen: t_i := 0B; Setzen: t_i := 1B), besteht mit BIT i, s auch die uneingeschränkte Abfragemöglichkeit via Z-Flagbit:

s_i Z
s_i = 1 0 ; s_i
s_i = 0 1 ; s_i

Die Anwendungen dieses Bit-Manipulierens reichen vom Bit-Transport über Abfrage von Einzelbit-Indikatoren bis zum softwaremäßigen Nachbilden ganzer Statusworte.

Beispiele 44, 45:

Problem: Bit-Transport r1_j := r_i, r1 ≠ r ≠ A, 1 ≤ j-i ≤ 4

LD A, r ; A := r
CPL ; A := r
AND 2ⁱ ; A_i := r_i
RLCA ; (j-i) mal
... ; A_j := r_i
RLCA ;
OR r1 ; A := r1 ∨ r_i
LD r1, A ; r1_j := r_i
RES j, r1 ; r1_j := 0B
BIT i, r ; Z := r_i
JR NZ, 2 ; bei Z = r_i = 0
SET j, r1 ; r1_j := 0, r1_j := 1B
; r1_j := r_i

Beispiele 46, 47:

Probleme: Z := CY; Z := CY

RLA ; A₀ := CY
BIT 0, A ; Z := A₀
RRA ; Regenerieren
CCF ; CY := CY
RLA ; A₀ := CY_{alt}
BIT 0, A ; Z := A₀ (= CY_{alt})
RRA ; Regenerieren
CCF ;

3.3.2.14. Keine Operation, Halten

NOP ; Keine Operation, Zeitdauer 1 M-Zyklus (1 Befehl)
HALT ; Stopp der ZVE, PC bereits erhöht (1 Befehl)

Die Eingliederung beider Operationen in die Programmabarbeitung ist aus Bild 3.3.2. und Ende Abschnitt 3.3.1.4. zu ersehen.

Wird fortgesetzt

Verkaufe
„Funktechnik“ 1974 - 78,
„radio fernsehen
elektronik“ 60 - 78,
im Schraubdeckel

Zuschr. an
Fil. 119 980 DEWAG,
1054 Berlin

Dr.-Ing. WOLFGANG HERRMANN

3.3.2.15. Übrige Befehle

Die übrigen Befehle für Kellerarbeit, Austausch, Unterprogrammaufruf und -rückkehr, Unterbrechungsbehandlung, Ein- und Ausgabe sowie die Gruppenoperationen sind untrennbar mit bestimmten Programmiermethoden verbunden und sollen in diesem Zusammenhang im Abschnitt 3.5. erläutert werden.

3.3.2.16. Flag-Beeinflussung

Vorbedingung für die Auswertung von Flag-Informationen ist die genaue Kenntnis der Bedeutung der Flagbits und ihrer gezielten oder auch unvermeidbaren Beeinflussung durch Befehle (s. hierzu Tafel 3.3.2.).

Die allgemeine Bedeutung der Flagbits ist folgende:

- S $\triangleq$ Vorzeichen (sign), Kopie des resultativen Bits A_7 . Repräsentiert echtes Vorzeichen des Resultats nach arithmetischen Operationen mit Operanden in Zweierkomplementdarstellung: $S:=1$, wenn Resultat < 0 , Abfrage mit $\text{cnd} = M$.
- Z $\triangleq$ Null (zero). Bei resultativer Null ($\triangleq$ z. B. Operandengleichheit in CP-Befehlen): $Z:=1$.
- H $\triangleq$ Halbübertrag (half carry; bei 8080: adjusted carry, AC). Gesetzt bei Übertrag zwischen Resultatbit A_3 und A_4 . Nur durch DAA intern auswertbar.
- N $\triangleq$ Subtraktion. $N:=1$ bei Subtraktion oder Operation mit subtraktivem Charakter. Durch DAA zum Zweck einer korrekten Befehlsausführung der Dezimalkorrektur herangezogen.

Tafel 3.3.2.: Beeinflussung des Flagregisters F

Befehl mit F-Beeinflussung	S	Z	H	P/V	N	C	Bemerkungen
INC t	S	Z	H	V		0	s.3.3.2.4.
DEC t	S	Z	H	V		1	s.3.3.2.4.
ADD s; ADC s	S	Z	H	V		0	C s.3.3.2.5.
SUB s; SBC s; CP s; NEG	S	Z	H	V		1	C s.3.3.2.5. 6. 9.
ADD xx,ss				x		0	C s.3.3.2.7.
ADC HL,ss	S	Z	x	V		0	C s.3.3.2.7.
SBC HL,ss	S	Z	x	V		1	C s.3.3.2.7.
AND s	S	Z	1	P		0	0 s.3.3.2.8.
XOR s; OR s	S	Z	0	P		0	0 s.3.3.2.8.
SCF alias SETC				0		0	1 s.3.3.2.9.
CCF alias CPLC				x		0	C s.3.3.2.9.
CPL				1		1	s.3.3.2.9.
DAA	S	Z	H	P			C s.3.3.2.9.
RLCA; RLA; RRCA; RRA				0	P	0	C s.3.3.2.10.
RLC s; RL s; RRC s; RR s	S	Z	0			0	C s.3.3.2.10.
SLA s; SRA s; SRL s	S	Z	0	P		0	C s.3.3.2.10.
RLD; RRD	S	Z	0	P		0	s.3.3.2.10.
BIT i,s alias TEST i,s	x	Z	1	x		0	s.3.3.2.13.
POP AF	x	x	x	x	x	x	F:=(SP)
LD A,i; LD A,R	S	Z	0	P	V	0	
IN r,(C); TEST (C)	S	Z	H	P		0	
LDI; LDD	x	x	0	P	V	0	
LDIR; LDDR	x	x	0	0		0	
CPI; CPD	x	Z	x	P	V	1	Z:=(A=(HL))
CPIR; CPDR	x	Z	x	P	V	1	— " —; Z:=1/V,P,V=0
INI; IND; OUTI; OUTD	x	Z	x	x		1	Z:=((B—:=1)=0)
INIR; INDR; OTIR; OTDR	x	1	x	x		1	

S, Z, H, P/V, C = betreffendes F-Bit gemäß Resultat der Operation aktualisiert (= dynamisch Neubestimmt)

P/V $\triangleq$ bei LD A,i; LD A,R: P/V:=IFF2
bei LDI; LDD; CPI; CPD; CPIR; CPDR:
P/V:=((BC—:=1) $\neq$ 0)

P = P/V-Bit gemäß Parität des Resultats aktualisiert
V = P/V-Bit gemäß Überlaufsituation des Resultats aktualisiert

0 = betreffendes F-Bit von Operation gelöscht

x = betreffendes F-Bit von Operation unbestimmt beeinflusst (in Tafel 3.2.3./4. durch I gekennzeichnet)

bei POP AF: Flagregister aus Keller neu geladen

1 = betreffendes F-Bit von Operation gesetzt

C $\triangleq$ Übertrag CY (carry). Übertrag aus Akku nach Addition, komplementärer Übertrag (= geborgtes Bit) nach Subtraktion, herausgeschobenes Bit nach Verschieben. Mittels SCF und CCF auch ausdrücklich setz- und komplementierbar.

P/V $\triangleq$ Parität/Überlauf (parity/overflow), multivalenter Charakter:

1) Zeigt Parität P nach logischen und Verschiebeoperationen und nach $\text{IN } r,(C)$ an: $P:=1$ bei gerader Anzahl resultativer Bit-Einsen, Abfrage mit $\text{cnd} = PE$; Abfrage einer ungeraden Anzahl mit $\text{cnd} = PO$.

2) Zeigt im Gegensatz zu CY echten Überlauf V des Resultats nach einer arithmetischen Operation mit Operanden in Zweierkomplementdarstellung an. Abfrage: $\text{cnd} = VS \equiv PE$.

3) Sichert nach LD A,I und LD A,R das Bit aus IFF2.

4) Fixiert nach LD- und CP-Gruppenoperationen die Aussage, ob der Bytezähler BC nach Erniedrigung ungleich Null ist, d. h. $P/V:=(BC—:=1) \neq 0$. Abfrage mit $\text{cnd} = VS \equiv PE$.

Die Auswertung der Flag-Informationen geschieht

- durch bedingte Operationen mittels Abfrage eines Einzelbits (Z, C, P/V, S) auf den Wert 0 oder 1 entsprechend der Verschlüsselung im Befehlsbestandteil cnd
- durch DAA als interne Abfrage von H, N zur Dezimalkorrektur im Akku bei BCD-Verarbeitung
- durch Zugriff zu einer ganzen Kopie des F-Bytes mittels der indirekten Adresse (SP) nach vorheriger Kelleroperation PUSH AF.

3.4. Programmierung nach Struktogrammen

Als Vorlage für die Aufstellung zunächst eines Programmteils oder -abschnitts und danach auch einer ganzen Programmseinheit dient hier die grafische Darstellung des Lösungsalgorithmus in Form eines Struktogramms nach Abschn. 3.1.3. Der Vorgang des Umsetzens eines solchen Struktogramms in ein Programm (Kodieren) kann und soll soweit formalisiert werden, daß er auch in manueller Ausführung bloße Routinearbeit bleibt. Alle schöpferischen Kräfte sind daher auf eine hohe Qualität des zugrundgelegten Struktogramms zu richten, was sehr wohl heißt, daß man bei dessen Abfassung auch die Eigenarten der Mikroprozessoren bzw. des für die spätere Programmabarbeitung ausgewählten Mikroprozessors zu beachten hat.

Finalprodukte des Kodierens sind Programme oder Programmteile auf einem Niveau, das der symbolischen Notationsform wegen der Assemblersprache weitgehend nahekommt. Struktogrammvorlage einerseits und Formalisierung des Kodierungsverfahrens andererseits garantieren, daß verschiedene Kodierer nach ein und demselben Struktogramm Finalprogramme aufstellen, die höchstens in der Effektivität der Kodeäquivalente für Anweisungs-„Kästchen“ differieren, was aus unterschiedlicher Einsicht in die operationellen Möglichkeiten des Mikroprozessors rührt und meist ein Werturteil über die Raffinesse der Maschinensprache oder das Können des Kodierers erlaubt.

3.4.1. Kodierung der Grundstrukturen

Die formale Struktogrammumsetzung stützt sich auf die Verwendung von drei Tafeln, deren Gebrauch zunächst in einem Umfang erläutert wird, daß damit alle Grundstrukturen routinemäßig kodiert werden können.

3.4.1.1. Kode-Generierungsmuster der Grundstrukturen

Zunächst sei festgestellt, daß die Sequenz wegen ihres elementaren algorithmischen Ablaufs nicht betrachtet zu wer-

den braucht, denn sie geht problemlos und so effektiv, wie der Kodierer die operationellen Möglichkeiten der Ziel-Programmiersprache beherrscht, in ein „Geradeaus“-Programmstück umzusetzen.

Fallauswahl und Zyklus bedingen durch Vorhandensein auszuwählender Fälle bzw. durch zyklisches Repetieren und auszuwählende Appendizes einen wesentlich komplizierteren algorithmischen Ablauf, der adäquat in den Kode des Finalprogramms abzubilden ist. Diese Kodierung wird in einer einzigen vertikalen Durchmusterung jedes Strukturblocks erledigt, die daher in dem Sinne „sequentiell“ verläuft, als man sich immer auf die laufende (und evtl. vorausschauend noch auf die folgende) Zeile beschränken kann und ausdrücklich nie den Strukturblock als Ganzes vor Augen haben muß.

Aus diesem Grunde können Muster für das Generieren (Erzeugen und Notieren in der anfallenden Reihenfolge) von Codes bereits für Einzelelemente von Grundstrukturen angegeben werden (Positionen 1 bis 12 in Tafel 3.4.1.). Aus diesen elementaren Kodemustern baut man das Kode-Generierungsmuster eines ganzen Strukturblocks genauso auf, wie sich letzterer aus seinen Strukturelementen zusammensetzt

Tafel 3.4.1.: Kodierung von Struktogrammen

- 1, 2 Bedingung/Anweisung am Anfang eines Strukturelementes
- 3 Bedingung Bi
- 4 Abschluß des Hauptteiles einer Fallauswahl
- 5, 6 Bedingung/Anweisung am Abschluß des Zyklushauptteiles
- 7, 8 Fall Ai vorhanden/fehlt
- 9 textlich letzter Fall (und Schlußmarke zx)
- 10, 11, 12 optimale Fallanordnung
- 13 bedingter Übergang
- 14, 15 unvollständige/vollständige Alternative
- 16, 17, 18 Zyklen (inwendig, kopf-, endseitig gesteuert)
- 19, 20 allgemeine Fallauswahl mit optimaler Fallanordnung
- 21 allgemeiner Zyklus mit optimaler Fallanordnung

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	21

(Positionen 13 bis 21 in Tafel 3.4.1.). Der Vollständigkeit halber wurde als Position 13 dabei auch der bedingte Übergang (als irreguläre Struktur nach Bild 3.3.6.) aufgenommen. Außerdem ist der Kopf einer Fallauswahl in Position 19/20 gegenüber Bild 3.1.5 b1 vorteilhaft modifiziert worden.

In allen Tafelpositionen ist also rechts neben einem Strukturelement oder -block dessen prinzipielles Kodeäquivalent angegeben. Dabei bedeuten:

Aa, Ab, ..., An – Codes für Anweisungen des Zyklushauptteils

A0, A1, Ai, Ak, Am – Codes für Fälle bzw. Appendizes

B1, Bi, Bk, Bm – Codes für Bedingungen

JP f, ..., bzw. JP f, ... – sog. Verzweigungsoperatoren

JP z bzw. JP z0 bzw. JP zx – unbedingte Sprünge der MS

z: bzw. z0: usw. – Markierungen (s. Abschn. 3.3.2.12.)

z, z0, z1, zi, zk, zm, zx – symbolische Adressen.

Die Haupteigenschaft der Generierungsmuster sticht sofort ins Auge: die zeilenmäßige (textliche) Reihenfolge der Kodeäquivalente für Anweisungen A... und Bedingungen B... ist identisch mit der textlichen Reihenfolge der A- und B-Kästchen im Struktogramm. Alles übrige sind „Zutaten“ für die Realisierung des Steuerflusses im Finalprogramm. Zusammenfassend läßt sich sagen: Das Umsetzen jedes Strukturblocks beginnt mit seiner Identifizierung unter den Positionen der Tafel 3.4.1. und der Annahme des nebenstehenden Kode-Generierungsmusters als aktueller Leitfaden dieser Kodierung.

3.4.1.2. Kodeäquivalente für Bedingungen

Die in den Strukturblocks auftretenden Bedingungen der Fall- und Appendixauswahl oder einfach nur des Beendigungs eines Strukturblocks sind in Befehle des Finalprogramms umzusetzen, die eine dynamische Aktualisierung des Prozessorzustandes durch vollständige oder teilweise Neubestimmung des Flagwortes F bewirken.

Tafel 3.4.2. führt 63 Formen nach sog. Testgruppen geordnet an, die unter Berücksichtigung konkreter Operanden für s, t insgesamt mehrere Hundert denkbarer Bedingungen in B-Kästchen von Struktogrammen repräsentieren. Jede dieser Bedingungen ist infolge ihres elementaren Charakters in einen Einzelbefehl umzusetzen, den man der Tafel entnehmen kann (bis auf Nr. 5 und 6).

Darüber hinaus gibt Tafel 3.4.2. in ihrer rechten Spalte ein einzelnes Flagbit F1 bzw. seine Negation F1 oder eine logische Funktion zweier Flagbits F1, F2 als sog. Zustand f an. Das ist in folgendem Sinne zu verstehen: Der kodierte Einzelbefehl aktualisiert das Flagwort derart, daß der Zustand f (evtl. neben weiteren, jedoch nicht gegensätzlichen Zuständen) garantiert dann vorliegt, wenn die ursprüngliche Bedingung des Struktogramms erfüllt wäre.

Zusammenfassend:

Das Umsetzen einer Bedingung besteht in ihrer Identifizierung mit einer der Formen in Tafel 3.4.2., dem Notieren des Codes des entsprechenden Einzelbefehls („kodierte Bedingung“) als laufende Textzeile des entstehenden Programms und dem einstweiligen Merken der zugehörigen Zustandsfunktion f.

Die Tabellenformen 5 und 6, $(B \rightarrow \equiv 1) = 0$ bzw. $\neq 0$, werden so behandelt, als hätten sie keine Kodeäquivalente; sie aktualisieren auch das Flagwort nicht, sondern sind „original“ als Funktion f zu merken.

Alle anderen Formen zerfallen in drei Klassen, als deren typische Vertreter Vergleich, Zähltest und Summentest (z. B. in Gestalt der Formen 9, 3, 19) anzusehen sind. Der Kode der Bedingung $A \leq s$ hat den einzigen Zweck der Herbeiführung einer Zustandsänderung bei unveränderten Operanden. Der Kode von $(t \rightarrow \equiv 1) = 0$ liefert eine Aussage über den in t geschalteten neuen Wert. Der Kode von $A + s \geq 256$ bedeutet die Ausführung der Addition, also die Veränderung von A, um a posteriori den Test auf Überlauf bzw. Übertragsbildung beim Summieren vornehmen zu können.

Diese Feinheiten lassen sich im Rahmen des Strukturblocks eines Zyklus als Paraphrasen der ursprünglichen Bedingungen deutlich machen (s. Bild 3.4.1.). Die resultierenden B-Kästchen im Bild 3.4.1. enthalten unmittelbar f-Funktionen;

Fortsetzung auf Seite 377

**Tafel 3.4.2.: Kodierung von Bedingungen Bi;
Angabe resultativer Zustände f**

Testgruppe	Nr. Bedingung Bi im Struktogramm	Bedingung kodiert	Zustand f
$t \pm := 1$	1 $(t+:=1) \bmod 256 = 0$	INC t	Z
	2 $(t+:=1) \neq 0$	INC t	Z
	3 $(t-:=1) = 0$	DEC t	Z
	4 $(t-:=1) \neq 0$	DEC t	Z
	5 $(B-:=1) = 0$	—	$(B-:=1) = 0$
	6 $(B-:=1) \neq 0$	—	$(B-:=1) \neq 0$
$A \nabla s$	7 $A < s$	CP s	CY
	8 $A = s$	CP s	Z
	9 $A \leq s$	CP s	$CY \vee Z$
	10 $A > s$	CP s	$CY \wedge Z$
	11 $A \neq s$	CP s	Z
	12 $A \geq s$	CP s	CY
$A - s$	13 $A - s < 0$	SUB s	CY
	14 $A - s = 0$	SUB s	Z
	15 $A - s \leq 0$	SUB s	$CY \vee Z$
	16 $A - s > 0$	SUB s	$CY \wedge Z$
stets gilt $-256 < A - s$	17 $A - s \neq 0$	SUB s	Z
	18 $A - s \geq 0$	SUB s	CY
$A + s$	19 $A + s \geq 256$	ADD s	CY
	20 $A + s > 256$	ADD s	$CY \vee Z$
	21 $A + s = 256$	ADD s	$CY \wedge Z$
	22 $0 \leq A + s < 256$	ADD s	CY
	23 $0 < A + s \leq 256$	ADD s	$CY \wedge Z$
	24 $0 = A + s$	ADD s	$CY \wedge Z$
	25 $0 < A + s \leq 256$	ADD s	$CY \sim Z$
	26 $(A+:=s) \bmod 256 = 0$	ADD s	Z
$HL + ss + CY$	27 $(A+:=s) \bmod 256 \neq 0$	ADD s	Z
	28 $xx + ss \geq 2^{16}$	ADD xx,ss	CY
$xx + ss$	29 $0 \leq xx + ss < 2^{16}$	ADD xx,ss	CY
	30 $-256 \leq A - s - CY < 0$	SBC s	CY
$A - s - CY$	31 $-256 < A - s - CY < 0$	SBC s	$CY \wedge Z$
	32 $-256 = A - s - CY$	SBC s	$CY \wedge Z$
	33 $A - s - CY \geq 0$	SBC s	CY
	34 $A - s - CY > 0$	SBC s	$CY \wedge Z$
	35 $A - s - CY \leq 0$	SBC s	$CY \wedge Z$
	36 $-256 < A - s - CY \leq 0$	SBC s	$CY \oplus Z$
	37 $(A-:=s+CY) \bmod 256 = 0$	SBC s	Z
	38 $(A-:=s+CY) \bmod 256 \neq 0$	SBC s	Z
$-A (A \geq 0)$	39 $-A < 0$	NEG	$CY, \overline{Z}$
	40 $-A = 0$	NEG	CY
$2 \cdot s$	41 $2 \cdot s \geq 256$	SLA s	CY
	42 $2 \cdot s < 256$	SLA s	CY
$A_i \wedge s_i$ ($i=7(-1)0$)	43 $\Sigma(A_i \wedge s_i) = 0$	AND s	Z
	44 $\Sigma(A_i \wedge s_i) \neq 0$	AND s	Z
ähnlich: $A_i + s_i, A_i, s_i$	45 $\Sigma(A_i \wedge s_i) = 2k$	AND s	P ($\triangleq PE$)
	46 $\Sigma(A_i \wedge s_i) = 2k+1$	AND s	P ($\triangleq PO$)
Bit-Test	47 $A_{7,alt} = A_{0,neu} = 1$	RLCA	CY
	48 $A_{7,alt} = 1$	RLA	CY
49 $s_{7,alt} = s_{0,neu} = 1$	50 $s_{7,alt} = 1$	RLC s	CY
	51 $s_{7,alt} = 1$	RL s	CY
	52 $s_{6,alt} = s_{7,neu} = 1$	SLA s	CY
53 — " —	54 — " —	RLC s	S
	55 $s_{0,alt} = s_{7,neu} = 1$	RL s	S
	56 $s_{0,alt} = 1$	SLA s	S
57 $s_{0,alt} = s_{7,neu} = 1$	58 $s_{0,alt} = 1$	RRCA	CY
	59 — " —	RRA	CY
	60 — " —	RRC s	CY, S
	61 $s_{7,alt} = s_{7,neu} = s_{0,neu} = 1$	RR s	CY
	62 $s_i = 0$	SRA s	CY
	63 $s_i = 1$	SRL s	CY
Bit-Test s_i ($0=i=7$)	64 $s_i = 0$	BIT i,s	Z
	65 $s_i = 1$	BIT i,s	Z

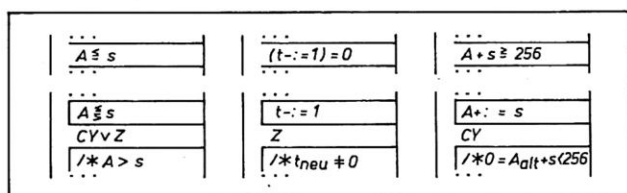


Bild 3.4.1.: Paraphrasieren elementarer Bedingungen

Tafel 3.4.3.: Verzweigungsoperator „JP f, →“

f	JP f, →	f	JP f, →
F1	JP F1, →	$\overline{F1}$	JP NF1, →
$F1 \vee F2$	JP F1, → JP F2, →	$\overline{F1} \wedge \overline{F2}$	JP F1, *+6 JP NF2, →
$F1 \vee \overline{F2}$	JP F1, → JP NF2, →	$\overline{F1} \wedge F2$	JP F1, *+6 JP F2, →
$\overline{F1} \vee F2$	JP NF1, → JP F2, →	$F1 \wedge \overline{F2}$	JP NF1, *+6 JP NF2, →
$\overline{F1} \vee \overline{F2}$	JP NF1, → JP NF2, →	$F1 \wedge F2$	JP NF1, *+6 JP F2, →
$F1 \sim F2$	JP F1, *+9	$\overline{F1} \oplus \overline{F2}$	JP F1, *+9
$\overline{F1} \oplus \overline{F2}$	JP F2, *+9	$F1 \oplus F2$	JP NF2, *+9
$F1 \oplus F2$	JP —	$\overline{F1} \sim \overline{F2}$	JP —
$F1 \oplus F2$	JP F2, →	$\overline{F1} \sim F2$	JP NF2, →
$(B-:=) \neq 0$ DJNZ → — (*+2) $(B-:=) = 0$ DJNZ 3 JP →			

Kodiervorschrift:

Fi	JP Fi, ...	JP NFi, ...
Z	JP Z, ...	JP NZ, ...
CY	JP C, ...	JP NC, ...
P	JP PE, ...	JP PO, ...
S	JP M, ...	JP P, ...

Fortsetzung von Seite 368

deren Umsetzung entspricht dem Ritual bezüglich der Formen 5 und 6 und ist im übrigen völlig offensichtlich. Die in dieser Art angegebenen Strukturböcke sind natürlich stark MS-orientiert, bieten damit aber auch die u. U. wichtige Möglichkeit, das Aktualisieren des Flagwortes von der Zustandsauswertung abzutrennen, wie später auch illustriert werden soll.

3.4.1.3. Kodierung des Steuerflusses

Der Steuerfluß ist in den Generierungsmustern der Tafel 3.4.1. komplett in Form der JP-Zeilen enthalten und braucht vom Kodierer nicht verstanden zu werden. Die anfallenden JP-Zeilen sind einfach in das entsprechende Finalprogramm zu kopieren. Dabei ist für den etwa auftretenden Parameter f (bzw. $\overline{f}$) die von der Tafel 3.4.2. angegebene f-Funktion (bzw. ihre Negation) einzusetzen, die sich der Kodierer beim eben beendeten Umsetzen der vorangehenden Bedingung gemerkt hatte.

Die denkbaren Formen dieses sog. Verzweigungsoperators „JP f, ...“ bzw. „JP $\overline{f}$, ...“ sind in Tafel 3.4.3. detailliert angegeben, wobei die Operation der Negation von f lediglich das Ablesen der detaillierten Form in der Nachbarposition der anderen Tafelhälfte erfordert. F1 und F2 sind die im Abschn. 3.4.1.2. erwähnten Flagbits der f-Funktion. Der zweite Operand des Verzweigungsoperators in der suggestiven Gestalt eines Pfeils ist durch die vom Generierungsmuster jeweils vorgeschriebene symbolische Adresse zu ersetzen.

Unter dem Hauptteil der Tafel 3.4.3. ist die Kodierungsvorschrift für die vorkommenden bedingten Sprünge hinzugefügt (vergleiche auch Bild 3.3.7.). Im Fall der Flagbits Z, CY kann man bei Z80 besser die entsprechenden Relativsprünge (nach Abschn. 3.3.2.12.) mit ausgerechnetem oder symbolischem Displacement e notieren, das nur im Bereich $-128 \leq e \leq 127$ liegen muß.

Das Strukturelement in Position 5 von Tafel 3.4.1. läßt schließlich eine Transformation seines Generierungsmusters zu (sei $k \neq m$):

B_m	B_m	B_m
JP f,z,m	JP f,z	JP f,z
JP z	JP z,m	JP z,x
zk: ...	kein Fall m	zk: ...

3.4.1.4. Markierung und symbolische Adressen

Die von den Generierungsmustern vorgeschriebenen Markierungen decken den tatsächlichen Bedarf an symbolischen Adressen garantiert und routinemäßig ab. Eine gewisse Redundanz beim Markieren ist für das Gliedern des Finalpro-

gramms und insbesondere beim Kodieren hierarchischer Strukturen von Vorteil. Nach Setzen der Schlußmarke z : eines Generierungsmusters können sämtliche generierten Markierungen, auf die kein Bezug durch symbolische Adressen genommen wird oder zu denen das Displacement berechnet wurde, als redundant gestrichen werden. Die beim Kodieren eines Strukturblocks eingeführte Anfangsmarke z : stellt eine beliebige, aber beim laufenden Programmierprozeß bislang noch nicht vorgekommene Bezeichnung (d. h. einen Buchstaben, dem weitere Buchstaben und/oder Ziffern in begrenzter Anzahl folgen dürfen) dar. Alle anderen Markierungen bzw. symbolischen Adressen (belanglos, was zuerst vom Generierungsmuster verlangt wird) werden durch Anhängen der Ziffer für die Fallnummer $0, 1, \dots, k, \dots, m$ (vorausgesetzt $m \neq 9$) oder von Buchstaben (z. B. x bei der Schlußmarke) an die gewählte Bezeichnung z der Anfangsmarke gebildet.

3.4.1.5. Programmierbeispiele

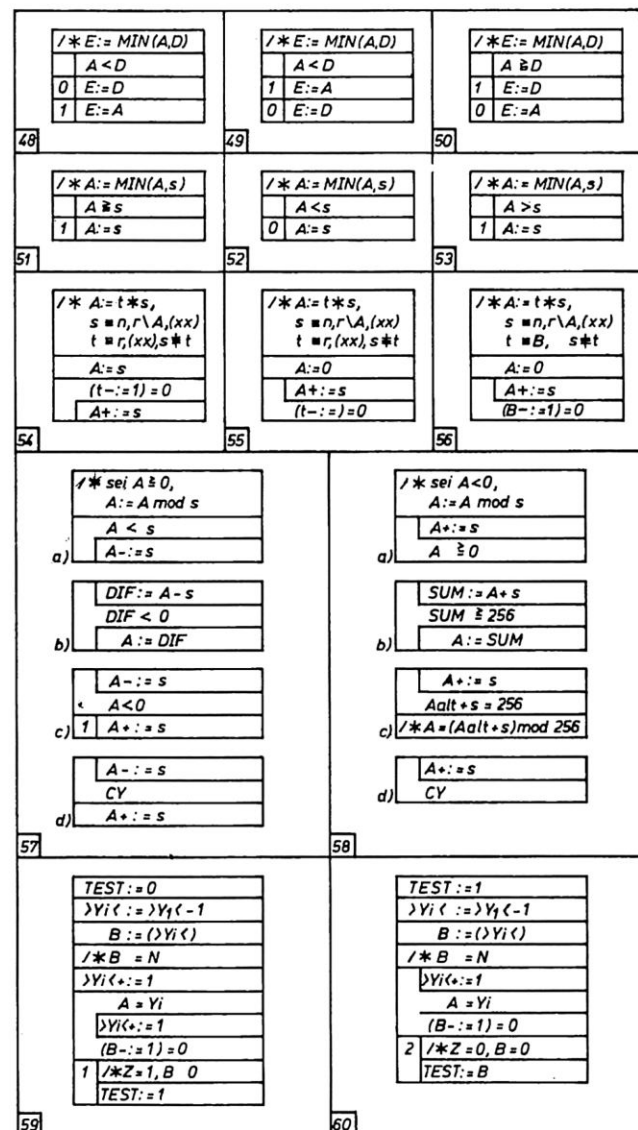
Zunächst soll die Umsetzung konkreter Strukturblocke in Programmabschnitte gezeigt werden, die auf Z 80 abgearbeitet werden können. Auch die Tafeln 3.4.2. und 3.4.3. sind ja entsprechend den Möglichkeiten dieses Mikroprozessors ausgelegt.

Die Einbeziehung von Ein- und Ausgabevorgängen in die Beispiele wird zunächst vermieden. Vielmehr wird angenommen, daß Operanden, insbesondere Werte in Registern, schon bereitstehen und Resultate auch weitgehend dort lassen werden können, wo sie anfallen.

Beispiele 48 bis 50 (vollständige Alternativen):

Problem: Minimum zweier Größen, $E := \min(A, D)$.

Gegeben: Zwei ganze Zahlen ohne Vorzeichen in A, D .



Lösung in drei Varianten, verkörpert durch die Struktogramme Nr. 48 bis 50 im Bild 3.4.2. Die letzte Variante erhält man dabei aus der ersten durch Negieren der Bedingung und Vertauschen der Fallnummern, während die die Fälle darstellenden beiden A-Kästchen ihre textliche Reihenfolge unverändert beibehalten.

Kode-Generierungsmuster lt. Position 15 in Tafel 3.4.1.; Wahl der Bezeichnungen mi, mj, mk als jeweiliges z .

$mi: CP D$	$mj: CP D$	$mk: CP D$
$JP CY,mi1$	$JP CY,mj0$	$JP CY,mk0$
$mi0: LD E,D$	$mj1: LD E,A$	$mk1: LD E,D$
$LD mix$	$JP mix$	$JP mkx$
$mi1: LD E,A$	$mj0: LD E,D$	$mk0: LD E,A$
$mix:$	$mjx:$	$mkx:$

Streichen überflüssiger Marken; Kodieren der Verzweigungsoperatoren:

$CP D$	$CP D$	$CP D$
$JP C,mi1$	$JP NC,mj0$	$JP C,mk0$
$LD E,D$	$LD E,A$	$LD E,D$
$JP mix$	$JP mix$	$JP mkx$
$mi1: LD E,A$	$mj0: LD E,D$	$mk0: LD E,A$
$mix:$	$mjx:$	$mkx:$

Umwandeln der Sprünge in Relativsprünge mit symbolischem Displacement:

$CP D$	$CP D$	$CP D$
$JR C,mi1-$	$JR NC,mj0-$	$JR C,mk0-$
$(*)+2$	$(*)+2$	$(*)+2$
$LD E,D$	$LD E,A$	$LD E,D$
$JR mix-(*)+2$	$JR mix-(*)+2$	$JR mkx-(*)+2$
$mi1: LD E,A$	$mj0: LD E,D$	$mk0: LD E,A$
$mix:$	$mjx:$	$mkx:$

Berechnung der Displacements; Streichen aller Marken, Kommentierung:

$CP D ; A < D$	$CP D ; A < D$	$CP D ; A \geq D$
$JR C,3$	$JR NC,3$	$JR C,3$
$LD E,D ; 0:E=D$	$LD E,A ; 1:E=A$	$LD E,D ; 1:E=D$
$JR 1$	$JR 1$	$JR 1$
$LD E,A ; 1:E=A$	$LD E,D ; 0:E=D$	$LD E,A ; 0:E=A$

Beispiele 51 bis 53 (unvollständige Alternativen):

Problem: Minimum zweier Größen, $A := \min(A, s)$.

Gegeben: Zwei ganze Zahlen ohne Vorzeichen in A, s ($s \leq n, r, (xx)$).

Verfahren: Der Wert in A ist bedingt durch den Wert von s zu ersetzen. Lösung in drei Varianten, s. Struktogramme! Kode-Generierungsmuster lt. Position 14 in Tafel 3.4.1.

$mr: CP S$	$ms: CP s$	$mt: CP s$
$JP CY,mrx$	$JP CY,msx$	$JP CY/\bar{Z},mtx$
$mr1: LD A,s$	$ms0: LD A,s$	$mt1: LD A,s$
$mrx:$	$msx:$	$mtx:$

Streichen redundanter Marken; Kodieren der Verzweigungen:

$CP s$	$CP s$	$CP s$
$JP C,mrx$	$JP C,msx$	$JP C,mtx$
$LD A,s$	$LD A,s$	$JP Z,mtx$
		$LD A,s$

$mrx: \quad msx: \quad mtx:$

Ein Variantenvergleich hat das interessante Ergebnis, daß die wegen des scharfen Testes $A > s$ „korrekteste“ Variante Beispiel 53 zu einer komplizierteren f-Funktion führt, woraus mehr Code resultiert. Da die detaillierten Formen der Verzweigungsoperatoren jederzeit der Tafel 3.4.3. entnommen werden können, wollen wir auf die fernere Angabe dieses Programmierschritts verzichten, ebenso auf die evtl. Umwandlung von Sprüngen in Relativsprünge. Displacement-Berechnung würde sowieso entfallen, denn nur bei konkretem s wäre die Befehlslänge von $LD A,s$ angebar, so daß die Sprungdistanz zur Schlußmarke berechnet werden könnte.

Bild 3.4.2.: Struktogramme zu den Beispielen 48 bis 60 in Abschnitt 3.4.1.5.

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

Beispiele 54 bis 56 (Zyklen):

Problem: Elementare Produktbildung durch mehrfaches Addieren:

$$A := \underbrace{s + s + \dots + s}_{t \text{ mal}} = t \cdot s$$

Gegeben: Zwei ganze Zahlen ohne Vorzeichen in t, s ($s \equiv A$) derart, daß $0 \leq t \cdot s \leq 255$ garantiert ist. Gefordert wird, daß der Wert von s erhalten bleibt.

Lösung in drei Varianten, siehe Struktogramme!

Kode-Generierungsmuster lt. Positionen 17, 18 in Tafel 3.4.1.

```
LD A,s      XOR A      XOR A
pr: DEC t    ps: ADD s   pt: ADD s
JP Z,prx    DEC t      JP (B--:=1)≠0,pt
ADD s       JP Z,ps     pt:
JP pr       psx:
```

prx:

Die Variante nach Beispiel 56 führt für Z 80 mit Abstand zur günstigsten Lösung, weil der Verzweigungsoperator in diesem Fall einem Spezialbefehl des Z 80 äquivalent ist (s. Abschn. 3.3.2.12):

JP (B--:=1)≠0, ... ⇒ DJNZ ...-(*)+2

Dieser Umstand wird künftig im Klartext ausgewiesen. Die optimale Lösung des gestellten Problems präsentiert sich daher in der Gestalt des Beispiels 56 (nach Streichen von ptx:) als

```
XOR A
pt: ADD s
DJNZ pt-(*)+2
```

Beispiele 57, 58 (Zyklen mit paraphrasierter Bedingung):

Problem: Modulo-Operation, $A := A \bmod s$.

Gegeben: Wert in A, definitiv erklärt als nicht-negativ (Beispiel 57) bzw. negativ und daher im Zweierkomplement vorliegend (Beispiel 58). Für den Wert von s gelte garantiert $s > 0$.

Lösung: Transformation des Wertes in A in das nach rechts offene Intervall $[0, s]$ durch fortlaufendes Erniedrigen bzw. Erhöhen um s bis schließlich gilt: $0 \leq A < s$ (siehe Struktogramme).

Kode-Generierungsmuster zu den Lösungsvarianten 57a, 58a lt. Positionen 17, 18 in Tafel 3.4.1.:

```
modp: CP s      modn: ADD s
      JP CY,modpx ?      ; A ≥ 0"
      SUB s      JP f,modn
      JP modp     modnx:
```

modpx:

Es erweist sich, daß die algorithmische Darstellung im Struktogramm 58a nicht mikroprozessorbezogen ist: Das Umschlagen des zu Beginn des Verfahrens als negativ erklärten Akkumulatorinhaltes in einen nicht-negativen und damit erst überhaupt elementaren Wert kann nicht mit der mathematischen Bedingung $A \geq 0$ wiedergegeben werden. Durch Paraphrasieren gelangt man über die Zwischenschritte b) und c) zum Struktogramm 58d. Das Paraphrasieren bei 57a hat das Ziel einer Effektivierung und führt zum Struktogramm 57d, was einen Wechsel des Generierungsmusters nach sich zieht.

Zusammenfassend.

```
modp: SUB s      modn: ADD s
      JP CY,modp  JP CY,modn
modpx: ADD s      modnx:
```

Beispiele 59, 60 (Zyklen mit mehreren Bedingungen):
Problem: Test auf „Enthaltensein“.

Gegeben: Ein Bytevektor $Y [1:N]$ mit N Komponenten Y_i , $i = 1(1)N$.

Speicherbelegung: fortlaufend, so daß für die Adressen gilt:

$$\rangle Y_1 \langle, \rangle Y_2 \langle = \rangle Y_1 \langle + 1, \dots, \rangle Y_N \langle = \rangle Y_{N-1} \langle + 1$$

Die Angabe der Komponentenanzahl N wird in der Speicherbelegung der ersten Komponente vorangestellt, d. h.

$$\rangle N \langle = \rangle Y_1 \langle - 1.$$

Weiterhin gegeben: Argumentwert in A.

Gefordert: Resultatwert TEST derart, daß gilt:

$$\text{TEST} = \begin{cases} 1, & \text{wenn } A \in Y[1:N], \text{ d. h., für mindestens ein } i \text{ gilt: } A = Y_i \\ 0, & \text{wenn } A \notin Y[1:N], \text{ d. h., für alle } i \text{ gilt: } A \neq Y_i \end{cases}$$

Lösung in zwei Varianten, siehe Struktogramme!

Verwendung einer Größe $\rangle Y_i \langle$ als Spender der Adresse von Y_i . Initiieren des Adressensenders mit $\rangle Y_1 \langle - 1$, d. h. mit $\rangle N \langle$.

Ressourcenbelegung: $C \triangleq \text{TEST}$, $HL \triangleq \rangle Y_i \langle$, B für Abzählen.

Kode-Generierungsmuster: für Beispiel 59 u. a. lt. Position 5, modifiziert entsprechend Abschn. 3.4.1.3.; für Beispiel 60 lt. Position 21 in Tafel 3.4.1. (dabei $m = 2$):

```
LD C,00H      LD C,01H
LD HL, \rangle Y_1 \langle - 1  LD HL, \rangle Y_1 \langle - 1
LD B, (HL)    LD B, (HL)
INC HL        et: INC HL
et: CP (HL)   CP (HL)
JP Z,et1     JP Z,etx ;kein FALL1
INC HL       DJNZ et-(*)+2
DJNZ et-(*)+2 et2: LD C,B ;FALL2
JP etx ;kein FALL2 etx:
et1: LD C,01H;FALL1
etx:
```

Es fällt auf, daß außer im Register C das Testergebnis auch im Flagbit Z zur Verfügung steht, da ja DJNZ das Flagwort nicht mehr ändert.

Anwendungsmöglichkeit der Problemlösung:

Sei KDNF die kanonische disjunktive Normalform einer Schaltfunktion, die für $k + 1$ unabhängige Variable $x_k, \dots, x_0$ definiert ist

$$\text{KDNF}(x_k, \dots, x_0) = \bigvee_{i \in I} m_i$$

mit den Mintermen m_i und der Index-Trägermenge I mit der Kardinalität $\text{card}(I)$. Sei X eine Variablenbelegung. Dann berechnet jeder der Programmteile Beispiele 59, 60 die $\text{KDNF}(X)$, wenn die schaltalgebraischen Größen folgendermaßen mit den algorithmischen Größen identifiziert werden:

$$N \equiv \text{card}(I), Y[1:N] \equiv I, A \equiv \widetilde{X}, \text{TEST} \equiv \text{KDNF}(X).$$

Kapazitätsbeschränkungen: $k \leq 7, N \leq 255$.

Aufbau der Variablenbelegung: $X = (0, \dots, x_k, \dots, x_0)$.

3.4.2. Kodierung hierarchischer Strukturen

Struktogramme wesentlicher Algorithmen enthalten stets hierarchisch angelegte Strukturen, die durch Superposition (= Überlagerung) oder – wie man in der Programmierungstechnik üblicherweise sagt – durch Verschachtelung von Grundstrukturen entstehen. Dieser Vorgang ist dann zulässig und resultiert erneut in einer regulären hierarchischen Struktur (künftig Hierarchie genannt), wenn in einer gegebenen regulären Struktur ein A-Kästchen einer Sequenz oder einer der Strukturtypen 13 bis 21 (s. Tafel 3.4.1.) (künftig Typen genannt) wiederum durch einen dieser Typen – möglicherweise denselben – ersetzt wird. Dabei erhöht sich der Verschachtelungsgrad der aus dem substituierten Typ stammenden A-Kästchen in der hierarchischen Unterordnung um Eins.

Kritisch ist bei diesem Verfahren die sichere Wiederaufnahme des jeweils zuletzt unterbrochenen Musters, was beim



manuellen Programmieren durch folgende zwei Schritte (auch ohne das Führen einer Merklste nach dem Keller- oder LIFO-Prinzip) möglich wird:

- Rekonstruktion des wiederaufzunehmenden Musters durch Reidentifizierung des Typus derjenigen Struktur, die der eben umgesetzten Substruktur unmittelbar übergeordnet ist
- Ermittlung der konkreten Wiederaufnahmestelle im Muster dieses Typus als Platz hinter demjenigen A-Element, für das soeben der vollständige Code der Substruktur substituiert wurde. Dazu ist bei systematisch durchgeführtem Markieren nur diejenige Marke auszuwerten, die das eingefügte Kodestück der Substruktur anführt (z. B. im Bild 3.4.3.: bei W2 ist Muster 15 oben mit dem Generieren von „JP z0x“ wiederaufzunehmen, weil der Code für z00:A00 komplett substituiert wurde usw.).

Mit dieser Technik wird eine Hierarchie „in einem Zug“ umsetzbar. Zur Demonstration noch einmal Beispiel 62 aus Bild 3.4.4. mit Kommentierungen als Bezug auf Bild 3.4.3.:

```

z: B1 ; Hauptmuster entspr. Typ 14 ob.
  JP f,zx
z0: B01 ; U1: Substruktur vom Typ 15 ob., z' ≅ z0
  JP f,z01
z00: B001 ; U2: Substruktur vom Typ 14 ob., z' ≅ z00
  JP f,z00x
z000:A000
z00x: ; W2: Typ 15 ob., hinter z...0:A...0
  JP z0x
z01: A01
z0x: ; W1: Typ 14 ob., hinter z0:A0
zx: ; Hierarchie umgesetzt

```

Der routinierte Kodierer kann ganz beiläufig weitere Verbesserungen vornehmen, die anderenfalls durch eine Nachbereitung zu erledigen wären:

- Anhängen einer Wiederaufnahmestelle an die Schlußmarke der umgesetzten Substruktur (in Bsp. 62: W2 an z00x:)
- Eliminieren von Mehrfachmarkierungen (insbesondere bei verschachtelten Alternativen) zugunsten der Marke mit der kürzesten Bezeichnung. Parallel dazu sind symbolische Adressen, die auf die eliminierten Markierungen bezug nahmen, bezeichnungsmäßig auf die verbliebene Markierung umzustellen (meist durch „Verkürzen“ des Mittelteils ihrer Bezeichnungen)
- Evtl. Streichen redundanter Einzelmarkierungen. Allerdings können solche Marken fakultativ verbleiben, denn sie kennzeichnen übersichtlich Programmbestandteile, insbesondere Anfang und Schluß einer Hierarchie oder ihrer Substrukturen.

Bild 3.4.4. enthält die Struktogramme einiger ausgewählter Hierarchien sowie deren Umsetzungen und Nachbereitungen. Der abstrakte Charakter dieser Beispiele in Termini von A- und B-Kästchen bringt es mit sich, daß die hier nachfolgend zusammengestellten Endfassungen nicht so sehr Finalprogramme als eben die hier einmal notationell fixierten Code-Generierungsmuster der vorgelegten Hierarchien darstellen.

Beispiele 61 bis 63 (verschachtelte Alternativen)

z: B1 JP f,zx	z: B1 JP f,zx	z: B1 JP f,z1
z0: B01 JP f,zx	z0: B01 JP f,z01	z0: B01 JP f,zx
z00: B001 JP f,z001	z00: B001 JP f,zx	z00: B001 JP f,zx
z000:A000 JP zx	z000:A000 JP zx	z000:A000 JP zx
z001:A001	z01: A01	z1: A1
zx:	zx:	zx:

Beispiele 64 bis 66 (verschachtelte Zyklen)

64: Verschachtelte, kopfseitig gesteuerte Zyklen können die inneren Ausgänge optimieren, wenn ihre Zyklusenden in dieselbe Struktogrammzeile fallen.

65, 66: Verschachtelte Zyklen mit einem endseitig gesteuerten Zyklus als Hauptstruktur beginnen mit Doppelmarkierungen,

wenn ihre Zyklusanfänge in dieselbe Struktogrammzeile fallen oder dort eine andere Substruktur beginnt. Es fällt bei diesen Beispielen auf, daß der generierte Code günstiger durch eine Nachbereitung zu verbessern ist.

z: B1 JP f,zx Ab1	z: Aa1 Ba1 JP f,z zax: Aa2 B1 JP f,z zx:	z: Ba1 JP f,zax zab: Bab1 JP f,z zab1:Aab1 JP z zax: B1 JP f,z zx:
-------------------------	--	--

Beispiele 67 bis 69 (gehäufte gleiche Fälle)

67: Der mit z10 redundant markierte Fall kann nach Streichen von z1x: getilgt werden, so daß Fall 2 für ihn eintritt.

68b: Struktogrammfassung äquivalent zu 68a. Die mit z2 bis z1 markierten Fälle werden bei gleichzeitiger Änderung der auf sie bezugnehmenden symbolischen Adressen in z1 gestrichen. Erst dann kann der mit zm redundant markierte Fall getilgt werden, so daß auch für ihn Fall 1 eintritt. (Das Struktogramm 68a führt etwas mühevoller zum gleichen Finalprogramm.)

69: Die Hierarchie 69a läßt sich in eine Grundstruktur umformulieren. In 69b ist dann folgende Transformationsregel zu benutzen:

$\overline{B_i}$	$\overline{B_i}$	B_i
JP f,...	JP f,...	JP f,...

Die beschriebenen Algorithmen lauten in Kurzbezeichnung:

„A bei $B_1 \oplus B_2$ “	„A bei $\cup B_i$ “	„A bei $\cap B_i$ “
z: B1 JP f,z1 B2 JP f,z2 JP zx z1:B2 JP f,zx z2:A zx:	z: B1 JP f,z1 B2 JP f,z1 ... B1 JP f,z1 Bm JP f,zx z1:A zx:	z: B1 JP f,zx B2 JP f,zx ... B1 JP f,zx Bm JP f,zx z0:A zx:

3.4.2.4. Programmierbeispiele

Durch Anwendung des im vorigen Abschnitt angegebenen Verfahrens der Hierarchie-Umsetzung auf die Kodierung einiger konkreter „struktogrammierter“ Algorithmen soll der wichtige didaktische Schritt von der prinzipiellen Befähigung zur wirklichen Fertigkeit gefördert werden.

Die Struktogramme aller hier behandelten Hierarchiebeispiele sind im Bild 3.4.5. zusammengestellt. Stets wird der komplette „in einem Zug“ generierte Code angegeben, wobei Hierarchiemarken „z“ wahlfrei gelassen sind. An beiläufigen Aufbereitungen wird nur das Anhängen einer unmarkierten Wiederaufnahmestelle an eine unmittelbar zuvor generierte Schlußmarke praktiziert sowie in einem textlich letzten Fall der überflüssige Schlußsprung unterdrückt. Alle anderen Auf- und Nachbereitungen müssen hier aus drucktechnischen Gründen fortbleiben – ihre Ausführung sei dem Leser empfohlen. Nur in Ausnahmefällen wird die dabei erhaltene Finalform dem generierten Code gegenübergestellt.

Beispiele 70 und 71 (verschachtelte Alternativen)

Problem: Minimum dreier Größen, $E := \min(B, C, D)$; s. a. bereits Bild 3.1.7.!

70: Benutzung von drei verschiedenen Mustern.

71: Zweimalige rekursive Benutzung von Muster 15 unten.

z: LD A,B CP C JP CY,z0 z1: CP D JP CY,z11 z10: LD E,D JP z1x z11: LD E,B z1x: JP zx	z: LD A,B CP C JP CY,z0 z1: CP D JP CY,z10 z11: LD E,B JP z1x z10: LD E,D z1x: JP zx
--	--

```

z0: LD E,C
    LD A,C
z0a: CP D
     JP CY,z0ax
z0a0: LD E,D
z0ax:
zx:

```

```

z0: LD A,C
    CP D
    JP CY,z00
z01: LD E,C
     JP z0x
z00: LD E,D
z0x:
zx:

```

Beispiele 72 und 73 (endseitig gesteuerte Zyklen)
Probleme: Erzeugung der Null- und der Einheits-(N,N)-Matrix.

Speicherbelegung der quadratischen Matrizen: zeilenweise fortlaufend, für die Adressen der Byte-Komponenten X_{ij} gilt $(i,j = 1(1)N)$:

$$X_{12} = X_{11} + 1, \dots, X_{1N} = X_{1,N-1} + 1, X_{21} = X_{1N} + 1$$

usw.

Es erfolgt keine Berechnung der Adressen von X_{ij} gemäß

$$X_{ij} = X_{11} + (i-1) \cdot N + (j-1);$$

vielmehr wird ein einziger Adressenspender, der in der algorithmischen Beschreibung mit X_{ij} bezeichnet ist, fortwährend geschaltet. Das gesamte Verfahren wird durch verschachteltes Herunterzählen $N(-1)0$ (count down) gesteuert, im inneren Zyklus natürlich mit dem DJNZ-Befehl.

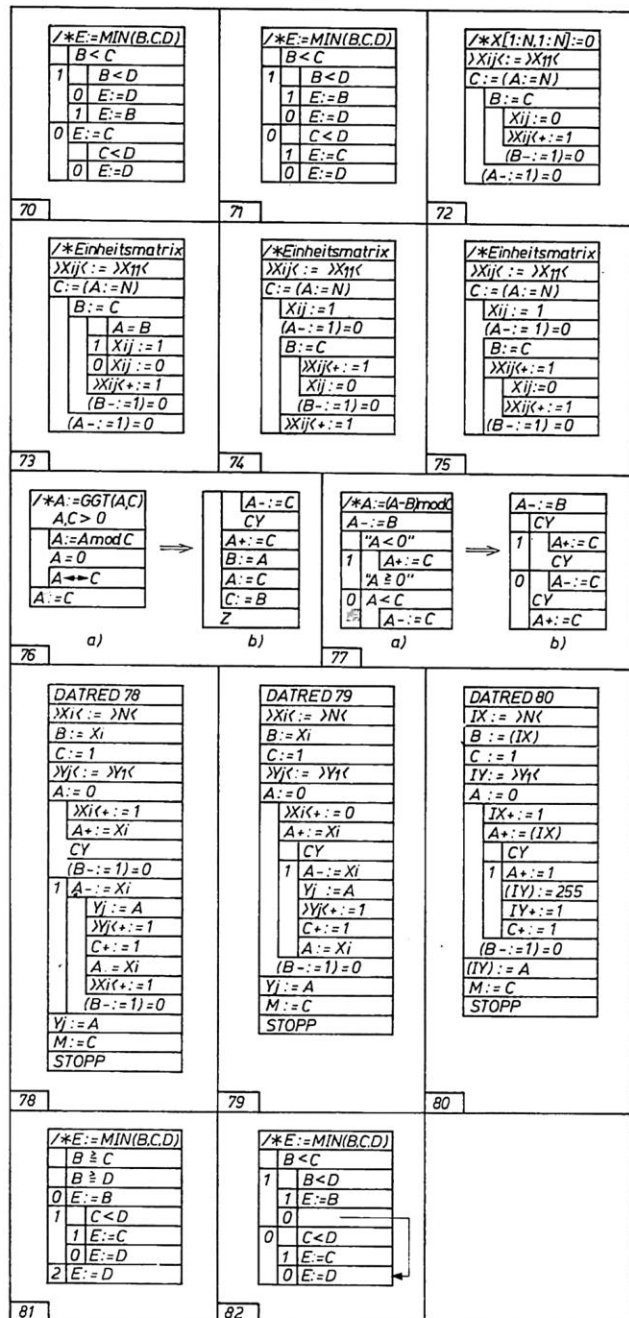


Bild 3.4.5.: Hierarchische Struktogramme zu den Beispielen 70 bis 82 von Abschn. 3.4.2.2.

Der Zugriff zu der Größe N soll jederzeit über eine unabhängige Speicheradresse möglich sein.

73: Hier erfolgt das Abspeichern von Eins, wenn die Count-down-Zähler koinzidieren; genau dann bezeichnet der Adressenspender ein Diagonalelement X_{ij} . Bekanntlich ist die Einheitsmatrix definiert als

$$X_{ij} = \begin{cases} 1, & \text{wenn } i = j \\ 0, & \text{wenn } i \neq j \end{cases}$$

```

n72: LD HL,X11(
     LD A,(N)
     LD C,A
z: LD B,C
za: LD (HL),0
    INC HL
    DJNZ za-(*+2)
zax: DEC A
     JP Z,z
zx:

e73: LD HL,X11(
     LD A,(N)
     LD C,A
z: LD B,C
za: LD B,C
zax: CP B
     JP Z,zax0
zax1: LD (HL),1
     JP zaxax
zax0: LD (HL),0
zaxax: INC HL
     DJNZ zax-(*+2)
zax: DEC A
     JP Z,z
zx:

```

Beispiele 74 und 75 (verschachtelte Zyklen)

Problem: Varianten der Erzeugung der Einheits-(N,N)-Matrix. Speicherbelegung wie in Beispiel 73. Zunächst erfolgt $X_{11} := 1$. Dann wird $(N-1)$ -mal eine Folge aus N Nullen und einer Eins abgespeichert.

```

e74: LD HL,X11(
     LD A,(N)
     LD C,A
z: LD (HL),1
    DEC A
    JP Z,zx
    LD B,C
    INC HL
    LD (HL),0
    DJNZ zb-(*+2)
zbx: INC HL
     JP z
zx:

e75: LD HL,X11(
     LD A,(N)
     LD C,A
z: LD (HL),1
    DEC A
    JP Z,zx
    LD B,C
    INC HL
    LD (HL),0
    DJNZ zb-(*+2)
zbx: INC HL
     JP z
zx:

```

Beispiele 76 und 77 (Hierarchien mit Paraphrasen)

76: Das Problem besteht darin, A durch den Größten Gemeinsamen Teiler aus A und C zu ersetzen: $A := \text{GGT}(A,C)$, $A,C > 0$. Für die Teiloperation $A \text{ mod } C$ wird die Lösung aus Beispiel 57d verwendet. Die Auswertung der bei $A+:=C$ implizite angefallenen Zero-Aktualisierung wird hinter den Tausch $A \leftrightarrow C$ verschoben.

77: Mit $C > 0$ wird $A := (A-B) \text{ mod } C$ aus den beiden früheren Teillösungen 58d, 57d komponiert.

```

z: m77: SUB B
za: SUB C
   JP CY,z0
zax: ADD C
     LD B,A
     LD A,C
     LD C,B
     JP Z,z
zx:

z1: ADD C
   JP CY,z1
z1x: JP zx
z0: SUB C
   JP CY,z0
z0x: ADD C
zx:

```

Beispiele 78 bis 80

Problem: Modellfall einer Datenreduktion (3 Varianten). Gegeben: Bytevektor $X[1:N]$ mit einer Speicherbelegung wie in Beispiel 59, dessen Komponenten $X_i > 0$ als Summanden einer gewissen Summe $\sum X_i$ aufzufassen sind. Durch geeignete Bildung von Teilsummen Y_j , $0 < Y_j \leq 255$, aus diesen X_i ist ein neuer Bytevektor $Y[1:M]$ aufzubauen, für den bei Erhalt der Eigenschaft $\sum X_i = \sum Y_j$ möglichst $M < N$ gilt (obwohl evtl. $M = N$ nicht zu verhindern ist).

78: Bildung von Y_j und danach bloßes Umspeichern aller evtl. noch nicht summierten X_i als weitere Y_j . Die Programmierung sei hier dem Leser überlassen!

79: Bildung so vieler Y_j , bis alle X_i verbraucht sind.

80: Setzen so vieler Y_j auf den Wert 255, daß gilt:

$$\sum X_i = (M-1) \cdot 255 + Y_M$$

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

Illustrationsbeispiel mit konkreten Werten:

Speicher-
verteilung :N X₁ X₂ X₃ X₄ X₅ X₆ X₇ X₈ ... X_N
X[1:N] :10, 40, 70, 90, 80, 220, 100, 60, 120, 110, 100
Speicher-
verteilung :M Y₁ Y₂ Y₃ Y₄ Y₅ Y₆ Y₇ Y₈ ...
Y[1:M],
Bsp. 78 :8, 200, 80, 220, 100, 60, 120, 110, 100
Bsp. 79 :6, 200, 80, 220, 160, 230, 100
Bsp. 80 :4, 255, 255, 255, 225

<pre> d79: LD HL,N(LD B,(HL) LD C,1 LD DE,Y1(XOR A z: INC HL ADD (HL) za: JP CY,zax za1: SUB (L) LD (DE),A INC DE INC C LD A,(HL) zax: DJNZ z-(*)+2) zx: LD (DE),A LD A,C LD (M),A HALT </pre>	<pre> d80: LD IX,N(LD B,(IX) LD C,1 LD IY,Y1(XOR A z: INC IX ADD (IX) za: JP CY,zax za1: INC A LD (IY),255 INC IY INC C zax: DJNZ z-(*)+2) zx: LD (IY),A LD A,C LD (M),A HALT </pre>
--	--

Der Variantenvergleich zeigt, daß im Beispiel 80 die Inanspruchnahme der Indexregister ungerechtfertigt war (und außerdem zu einem bytemäßig längeren Kode führen würde). Zu Indexregistern muß gegriffen werden, wenn die Universal-Doppelregister nicht ausreichen und/oder mit Displacement oder operationeller Verknüpfung beim Zugriff gearbeitet werden muß.

Zum Abschluß sollen noch einmal zwei Varianten des Beispiels 71, E:=MIN(B,C,D) aufgegriffen werden, bei denen die eigenwilligen Struktogrammvorlagen (man überprüfe z. B. Nr. 81) im Ergebnis der Nachbereitung des generierten Kodes zu überraschenden Finalprogrammen führen.

Beispiel 81 (Variante mit gehäuften gleichartigen Fällen)

<pre> z: LD A,B CP C JP CY,z1 CP D JP CY,z2 z0: LD E,B JP zx z1: LD A,C CP D JP CY,z10 ;ändern! z11: LD E,C JP z1x ;kürzen! z10: LD E,D ;streichen! z1x: JP zx ;streichen! z2: LD E,D zx: </pre>	<pre> z: LD A,B CP C JP NC,z1 CP D JP NC,z2 z0: LD E,B JP zx z1: LD A,C CP D JP NC,z2 z11: LD E,C JP zx z2: LD E,D zx: </pre>
---	--

Beispiel 82 (expliziter Übergang vorhanden)

<pre> z: LD A,B CP C JP CY,z0 z1: CP D JP CY,z10 ;ändern! z11: LD E,B JP z1x ;kürzen! </pre>	<pre> z: LD A,B CP C JP NC,z0 z1: CP D JP NC,z00 z11: LD E,B JP zx </pre>
---	--

<pre> z10: JP z00 ;streichen! z1x: JP zx ;streichen! z0: LD A,C CP D JP CY,z00 z01: LD E,C JP z0x ;kürzen! z00: LD E,D z0x: ;streichen! zx: </pre>	<pre> z0: LD A,C CP D JP NC,z00 z01: LD E,C JP zx z00: LD E,D z0x: ;streichen! zx: </pre>
---	--

Vergleicht man nun die in den beiden Beispielen 81 und 82 nachbereiteten Finalprogramme, so zeigt sich deren Übereinstimmung (bis auf die konkrete Wahl der Bezeichnungen für die Marken bzw. symbolischen Adressen). Zeichnet man eine solche Finalvariante in einen herkömmlichen Programmablaufplan um, so führt das auf Bild 3.1.3b rechts – also auf ein „Gegenbeispiel“ zu einer regulären Programmstruktur. Beispiele der Art Nr. 67, 68a, 81 u. ä. m. führen damit zur offenen Frage, ob eine nachbereitende Optimierung auch ohne Umstellungen in der textlichen Reihenfolge der Befehle eine Programmlösung sozusagen „irregularisieren“ und damit vom Standpunkt der strukturierten Programmierung abwerten kann.

3.4.3. Algorithmische Verfeinerung

Die Angabe des Lösungsalgorithmus eines Problems „in einem Zuge“ ist wohl nur in einfachsten Fällen bzw. bei bereits bekannter oder ähnlich gelagerter Thematik möglich und setzt auch dann noch viel Routinekönnen voraus. Das charakteristische Bild eines als Kodiervorlage geeigneten Struktogramms in Gestalt einer hierarchischen Struktur legt nun einen Gedanken nahe, nämlich die Verschachtelung der mitwirkenden Grund- oder Teilstrukturen selbst als Leitfaden für die planmäßige, schrittweise Entwicklung des Algorithmus in die Struktogrammform zu nutzen. Die Verwirklichung dieser Idee führt zur Aufstellung des Algorithmus durch Zerlegung einer anfänglichen Grobformulierung in Teilprobleme und deren allmähliche Ausgestaltung auf tieferen Niveaus. Dieses Präzisierungsverfahren bis hinab zu einem kodierbaren Endstand heißt „algorithmische Verfeinerung“; es ist zu einer Basistechnologie der Programmierung geworden.

3.4.3.1. Elemente und Ausführungstechniken der Verfeinerung

In Anwendung auf die im Kapitel 3. eingeführten „vertikalen“ Struktogramme soll ein einzelner Verfeinerungsschritt so verstanden werden, daß ein bisheriges A- oder B-Kästchen mindestens durch eine einzelne Grundstruktur substituiert und dadurch sowohl inhaltlich wie auch ablaufmäßig präzisiert wird.

Ein genaueres Erfassen der Sachlage wird möglich, wenn die Anzahl horizontaler Einrückungen an irgendeiner Stelle des Struktogramms (in den Abschnitten 3.1.3. und eingangs 3.4.2. intuitiv „Verschachtelungsgrad“ genannt) als Niveau p der lokalen algorithmischen Verfeinerung ($p \geq 0$) angesehen wird. Im Bild 3.4.6. sind hierzu einige Grundelemente zusammengestellt. So soll die zu substituierende Präzisierung kurz „Verfeinerung“ heißen. Ihr Anfangskästchen identifiziert sie als Substitut des gleichbezeichneten Kästchens der Grobstruktur; ihr Schlußkästchen INSERT schreibt die Substitution vor, in deren Ergebnis eine Fein- oder jedenfalls „feinere“ Struktur anfällt. Ist die Verfeinerung eines A-Kästchens mit Niveau p eine Sequenz, so kommt es zu keiner Niveauerhöhung, und die Verfeinerung hebt sich nach der Substitution nicht mehr aus ihrer Umgebung ab (s. Bild 3.4.6.b). Man könnte daher folgendes vereinbaren: Mit

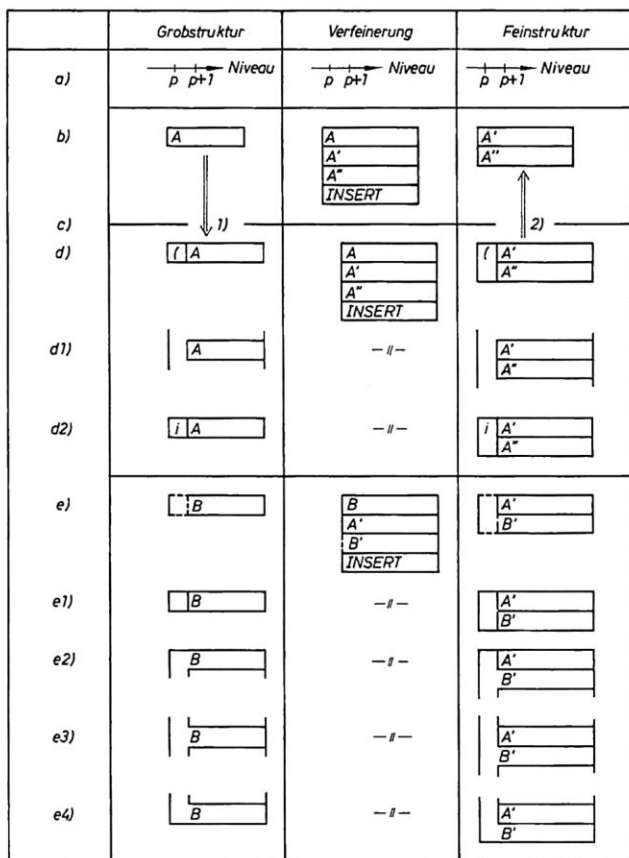


Bild 3.4.6.: Elemente algorithmischer Verfeinerung.
a) Verschachtelungsgrad (= Niveau);
b) A-Verfeinerung auf laufendem Niveau p;
c) formale Niveauerhöhung (1) bzw. -senkung (2);
d) A-Verfeinerung auf nächsthöherem Niveau p+1;
d1), d2) A-Verfeinerungen im Zylushauptteil und Fall;
e) B-Verfeinerung;
e1) bis e4) B-Verfeinerungen in Fallauswahl und Zyklus

einem Operator (hier: öffnende Klammer) wird das Einrücken eines Kästchens und damit seine formale Niveauerhöhung um Eins möglich; verfeinert werden können nur Kästchen auf dem Niveau p+1; die Verfeinerungen ihrerseits dürfen nur noch einmal eingerückte Kästchen enthalten. Bild 3.4.6. zeigt weiter, daß nach der Substitution die Verfeinerung von dem auf Niveau p vorhandenen Ablaufelement (z. B. Klammer, Fallnummer) umfaßt wird. Eine umspannende Klammerung kann bei Wunsch auch wieder rückgängig gemacht werden (Niveausenkung).

Die Identifizierung zwischen zu verfeinerndem Kästchen und seiner Verfeinerung erfolgt, wie gesagt, durch Gleichbezeichnung. Es ist allgemein wünschenswert, auf allen Niveaus ähnliche inhaltliche Ausgestaltung (Betextung) vorzunehmen. Wir werden eine Art gehobene symbolische Befehlsnotation praktizieren, d. h. formale oder symbolische Operationsbezeichnungen verwenden und Operanden bzw. Parameter aufführen.

Bild 3.4.7. illustriert die Methode der algorithmischen Verfeinerung am konkreten Problem der Bestimmung des Größten Gemeinsamen Teilers aus den in A und C befindlichen Werten, $A := \text{GGT}(A, C)$ (vgl. hierzu bereits Beispiel Nr. 76).

Zwei unterschiedliche Ausführungstechniken sind denkbar. So können (wie in den Teilbildern 3.4.6.a, b) zunächst alle Verfeinerungen ausgeführt werden, ehe man das Rückwärts-Einsetzen vornimmt. Dagegen zeigt Teilbild 3.4.6.c die etappenweise Entwicklung einer kodierbaren Feinstruktur, wobei die auf laufendem Niveau erarbeiteten Verfeinerungen sofort substituiert werden. In den Teilbildern 3.4.6.b, c, e wird auch die formale Niveauerhöhung und -senkung vorgeführt. An der Feinstruktur im Bild 3.4.6.c ist gegenüber Bild 3.4.6.d noch deutlich zu erkennen, daß die Anweisung $A := C$ Bestandteil einer Verfeinerung gewesen sein muß, während sie das im Teilbild d nicht zu sein brauchte.

Es versteht sich, daß die attraktivste Form dieser Programmierungstechnologie die Entwicklung der Struktogramme

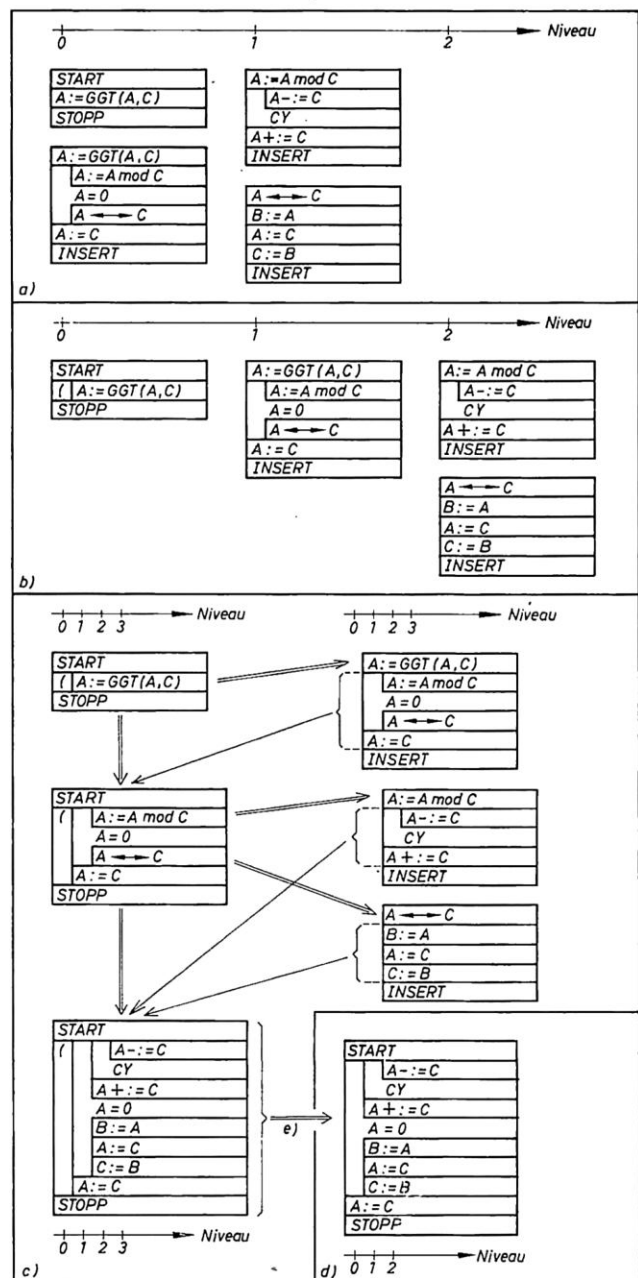


Bild 3.4.7.: Methoden algorithmischer Verfeinerung (s. Beispiel Nr. 76 im Bild 3.4.5.).

- a) Verfeinerungen auch auf laufendem Niveau (hier: 0) zulässig;
- b) Verfeinerungen nur auf nächsthöherem Niveau;
- c) etappenweise Entwicklung einer Feinstruktur entsprechend b);
- d) Feinstruktur entsprechend a);
- e) formale Niveaueinrückung ($\Rightarrow$ Transformationen, Verfeinerungen; $\rightarrow$ Substitutionen)

auf einem Bildschirm wäre, wobei Eingaben, Korrekturen, Verwaltung von Grobstrukturen und Verfeinerungen, Substitutionen und Niveauänderungen durch Kommandos befohlen und rechnergestützt ausgeführt werden.

3.4.3.2. Anweisungsverfeinerung

Ausgehend von der zweifachen Natur des Struktogramms, nämlich Algorithmusdokumentation und Kodievorlage zu sein, kann die Frage nach dem erforderlichen Verfeinerungsniveau nur pragmatisch entschieden werden: Unter Wahrung der algorithmischen Korrektheit ist so lange zu verfeinern, bis die Kodierbarkeit in der gewählten Programmiersprache (hier der Mikroprozessor-Maschinensprache in symbolischer Notation) offensichtlich ist.

Hieraus entwickelt sich das Bestreben, erarbeitete Teilprobleme in einer Entwicklungsbibliothek abzulegen und bei Bedarf dort zu entnehmen. Als einfachste Illustrationen könnten die im Bild 3.4.7. benutzten Operatoren $A := A \text{ mod } C$ und $A := \text{GGT}(A, C)$ bzw. ihre bereits früher angegebenen effektiveren Varianten nach den Beispielen 57d und 76b dienen. Paradebeispiele sind weiter scheinbare Aufstockun-

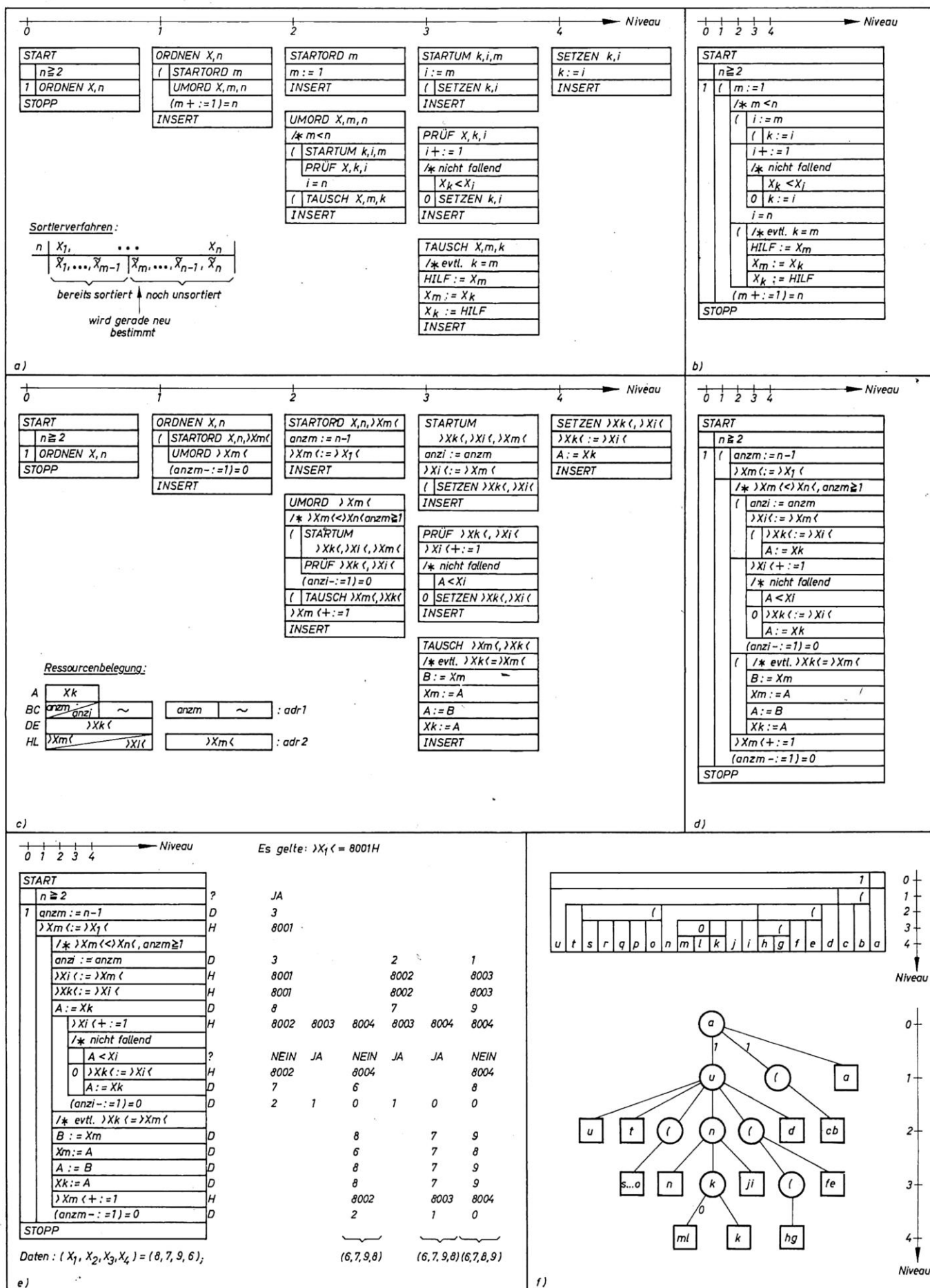


Bild 3.4.8.: Varianten algorithmischer Problemaufbereitung (zum Beispiel 83: Sortieren).

- Mathematische Formulierungsebene;
- resultierende Feinstruktur aus a);
- MS-orientierte Formulierungsebene;
- Feinstruktur nach c);
- komprimierte Feinstruktur mit Ablaufprotokoll;
- Korrespondenz zwischen Struktogramm und Programmbaum

gen der operationellen Möglichkeiten von installierten Prozessoroperationen: man verwendet deren Bezeichnung mit anderen oder zusätzlichen Operanden und notiert im Finalprogramm dafür die definierende Befehlsfolge (s. bereits Beispiele 36 bis 38 und 40). So könnte man statt mehrfacher Einzelverschiebungen RLA oder RRA z. B. die Makrooperation RLA k mit $1 \leq k \leq 8$ einführen und unter Ausnutzung der Eigenschaft $RLA k \triangleq RRA 9-k$ definieren:

RLA 1 ... RLA 3 RLA 4 RLA 5 ... RLA 8

RLA	...	RLA	RLA	RRA	...	RRA
		RLA	RLA	RRA		
		RLA	RLA	RRA		
			RLA	RRA		

Die algorithmische Aufbereitung eines größeren Problems mit Variantenvergleich zeigt Bild 3.4.8. Es geht um das Sortieren eines arithmetischen Vektors $X[1:n]$, wobei das Neuordnungsprinzip für die Komponenten (nicht fallend, steigend, fallend, nicht steigend) erst auf möglichst tiefem Niveau in die algorithmische Verfeinerung einfließen soll (gewählt wurde „nicht fallend“ infolge effektiver Realisierbarkeit in Maschinensprache).

Die Betextung der Verfeinerungen ist stark durch die gewählten Programmiersprachen gefärbt, wie ein Vergleich der Teilbilder 3.4.8.a, b mit den Teilbildern c, d zeigt. Wesentlichste Unterscheidungselemente sind die in den Bildern 3.4.8.c, d eingeführten Adressensponder und die Komponentenabzählungen (mit Blick auf die MS-Realisierung durch DJNZ). Teilbild 3.4.8.e zeigt, daß neben einem vertikalen Struktogramm platzgünstig der Ablauftest notiert werden kann.

Die Darstellungsart der Testdaten ist übrigens rechts neben den Struktogrammzeilen durch Kennungen angegeben.

Beispiel 83 (s. Bild 3.4.8.d oder e)

Problem: Sortieren von $X[1:n]$ auf eigenem Speicherplatz.

s:	LD	A, (n)	;A:=n, Gen.-Muster Pos. 14
	CP	2	;n $\geq$ 2, f=CY
	JP	C, sx	;JP f, ...
s1:	DEC	A	;A=n-1
	LD	B, A	;B=anzm
	LD	HL, X _i	;HL:=X _i , HL $\equiv$ X _m
s1a:	LD	(adr1), BC	;Retten von anzm, B $\equiv$ anzi,
	LD	(adr2), HL	;Retten von X _m , HL $\equiv$ Xi
	LD	D, H	
	LD	E, L	;X _k :=X _i , DE $\equiv$ X _k
	LD	A, (DE)	;A=X _k
s1aa:	INC	HL	;X _i ++:=1, Muster Pos. 18
s1aaa:	CP	(HL)	;A<X _i , Muster Pos. 14 oben,
	JP	C, s1aaax	;JP f, ... mit f=CY
s1aaa0:	LD	D, H	
	LD	E, L	;X _k :=X _i
	LD	A, (DE)	;A=X _k
s1aaax:	DJNZ	s1aa--(*+2)	
s1aax:	LD	HL, (adr2)	;Holen von X _m ,
	LD	B, (HL)	;B:=X _m
	LD	(HL), A	;X _m :=A
	LD	A, B	;A:=B
	LD	(DE), A	;X _k :=A
	INC	HL	;X _m ++:=1
	LD	BC, (adr1)	;Holen von anzm, B $\equiv$ anzm
	DJNZ	s1a--(*+2)	
s1ax:			;redundante Marke
sx:	HALT		;STOPP

Interessanteste Programmelemente sind die dynamischen Auslagerungen von BC, HL in den Speicher infolge Mehrfachnutzung dieser Doppelregister; dies ist später durch Kellierung zu effektivieren.

Das Teilbild 3.4.8.f zeigt die enge Korrespondenz zwischen dem vertikalen (hier aber um 90° gedrehten) Struktogramm und der Darstellungsmethode des Programmabbaums. Dieser würde die Anzahl der möglichen Verbindungen zwischen den Programmbestandteilen als minimal ausweisen, wenn das Finalprogramm nicht monolithisch aufgebaut wäre, sondern tatsächlich aus einzelnen sogenannten Moduln be-

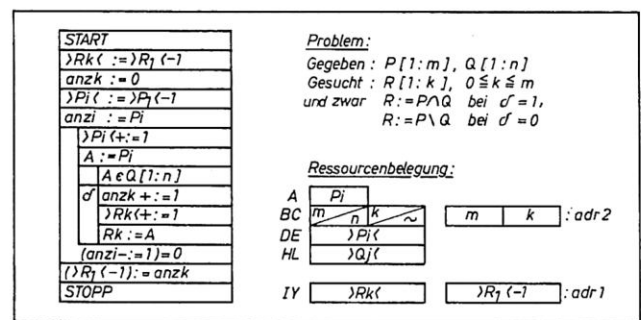


Bild 3.4.9.: Struktogramm und Ressourcenbelegung zum Beispiel 84 (Vektorzerlegung)

stände. Endknoten sind optisch von Steuerknoten (Kreise) unterschieden; auf jedem Ast-Niveau ist für alle vom selben Steuerknoten ausgehenden Kanten von rechts nach links fortzuschreiten; eine Bezeichnung im Steuerknoten gibt an, welcher gleichbezeichnete Endknoten das Entscheidungskriterium liefert; Kantenbewertung entspricht Fallnummern (gleiche Nummern bedeuten unbedingtes Hintereinanderschalten der entsprechenden Zweige). Das Struktogramm beinhaltet also alle Vorteile des Programmabbaums, ist aber nur in der Vertikalrichtung platzintensiv und gestattet die vollständige inhaltliche Angabe des Algorithmus an Ort und Stelle.

3.4.3.3. Bedingungsverfeinerung bzw. -transformation

Die formale Seite der Abwicklung einer B-Verfeinerung ist aus den Teilbildern 3.4.6.e, e1 bis e4 zu ersehen. Wesentlichste Eigenschaft ist, daß auf dem Verfeinerungsniveau $p+1$ eine Bedingung B nur wieder zu einer einzigen Bedingung B' wird, die Bedingungsanzahl (die den Fallnummernmechanismus regelt) sich also nicht ändert.

Rückblickend ist festzustellen, daß die im Abschnitt 3.4.1.2. und im Bild 3.4.1. vorgestellten Paraphrasen elementarer Bedingungen nur Verfeinerungen mit A' in Form einzelner MS-Operationen und B' $\equiv$ f sind.

Von Bedeutung ist weiter der Fall des gezielten Ladens nur eines einzigen Zustandsbits F1 durch eine Bedingung B, so daß die im Teilbild 3.4.6.e angegebene Prinziplösung folgendermaßen zu verstehen ist:

$$A' \equiv F1 := B, B' \equiv F1$$

Die Identität A' stellt eine Ergibtanweisung für die logische Zustandsvariable F1 dar, die ihrerseits in Maschinensprache durch bedingte Operationen auswertbar ist. In der Tafel 3.4.4. sind zwei allgemeine und 24 konkrete Fälle für $F1 := B$ aufgeführt. Die Fälle 3 bis 14 haben u. U. weniger selbständige Bedeutung, als daß sie in andere Fälle eingehen. Zum Beispiel führt der Intervalltest $A \in [s1, s2]$, d. h. $s1 \leq A < s2$, auf den logischen Ausdruck

$$B \equiv A \geq s1 \wedge A < s2$$

der gemäß Fall Nr. 2 behandelt wird und unter Nr. 20 eingetragen ist; Operanden könnten z. B. $s1 \equiv (IX)$, $s2 \equiv (IX+1)$ sein.

Aus diesen Überlegungen ist zu erkennen, daß die Bedingungsverfeinerung zu einer Reihe von Standardlösungen führt, die ebenfalls Bibliothekscharakter haben. Als Folge kann man in Struktogrammen recht komplexe Bedingungen formulieren und sie dennoch als bereits kodierbar ansehen. Musterhaft ist dafür der Enthaltenseins-Test Nr. 26 in Tafel 3.4.4., der auf den Beispielen 59, 60 fußt: Ist die Größe in A unter den Komponenten eines Vektors $Q[1:n]$ enthalten? Mit diesem Komplextest läßt sich ein interessantes Problem der Vektorzerlegung

$$R := P \cap Q \text{ („Durchschnitt“)}, R := P \setminus Q \text{ („Differenz“)}$$

wie im Bild 3.4.9. formulieren. Mit $A \equiv P_i$, $1 \leq i \leq m$, braucht nur die Lösung für $Z := A \in Q[1:n]$ herangezogen zu werden, um entscheiden zu können, ob bei fixiertem δ -Fall das P_i in den Resultatvektor R kommt oder nicht.

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN

Beispiel 84 (s. Bild 3.4.9.)

Problem: Vektorzerlegung entsprechend Fall δ

$\delta = 1: R := P \cap Q$

$\delta = 0: R := P \setminus Q$

```

vz: LD  IY,R1(-1) ;Rk(=)R1(-1)
     LD  C,0      ;k:=0
     LD  DE,P1(-1) ;Pi(=)P1(-1)
     LD  A,(DE)    ;A:=m
     LD  B,A       ;B:=anzi=m
     LD  (adr1),IY ;Retten>Rk(
s:  INC  DE        ;Pi(+=)1
     LD  A,(DE)    ;A:=Pi
     LD  (adr2),BC ;Retten (m,k)
    
```

```

t: ... ;Z:=AeQ[1:n], Fall Nr. 26, T. 3.4.4.
tax:LD  BC,(adr2) ;Holen (m,k)
r:  JP  f,rx       ;f:=Z, f=0: f=Z
rδ: INC  C         ;k+=1
     INC  IY        ;Rk(+=)1
     LD  (IY),A     ;Rk:=A
rx: DJNZ s-(*)+2)
sx: LD  IY,(adr1)  ;Holen>R1(-1)
     LD  (IY),C     ;Abspeichern von k
     HALT
    
```

Auch hier ist die Auslagerung von Doppelregistern in den Speicher bemerkenswert; sie ist durch Kellierung zu effektivieren.

Tafel 3.4.4.: Transformation von Bedingungen

- 1, 2 Sukzessiver Abbau logischer Funktionen von Bedingungen
 3 bis 6 Reduzierung der Anzahl signifikanter Flagbits
 7, 8 Umspeichern von Flagbits

Nr.	F1:=B	F1:=B	Kode-Äquivalent
1.	F1:=B1∨B2	F1:=B1 F1 0 F1:=B2	z: "SET F1,B1" ;F1:=B1 JP F1,zx zx: "SET F1,B2" ;F1:=B2
2.	F1:=B1∧B2	F1:=B1 F1 1 F1:=B2	z: "SET F1,B1" ;F1:=B1 JP NF1,zx zx: "SET F1,B2" ;F1:=B2
3.	CY:=CY∨Z	CY 0 Z 1 CY:=1	JR C,3 ;CY=1 JR NZ,1 ;CY=Z=0 CCF ;CY=1
4.	CY:=CY∧Z	CY 1 Z 0 CY:=0	JR NC,3 ;CY=0 JR Z,1 ;CY=Z=1 CCF ;CY=0
5.	CY:=CY~Z	CY 1 Z 0 CY:=0 0 Z 0 CY:=1	JR NC,5 ;CY=0 JR Z,6 ;CY=Z=1 CCF ;CY=0 JR 3 JR Z,1 ;CY=0,Z=1 CCF ;CY=1
6.	Z:=Z∨CY	Z 0 CY 1 Z:=1	JR Z,3 ;Z=0 JR NC,1 ;Z=CY=0 CP A ;Z:=1
7.	Z:=CY	Z:=CY	RLA ;A0:=CY BIT 0,A ;Z:=A0 RRA ;CY:=A0
8.	Z:=CY	CY:=CY Z:=CY CY:=CY	CCF RLA BIT 0,A RRA CCF
9.	CY:=A<s	CY:=A<s	CP s
10.	CY:=A=s	A<s 1 CY:=CY	CP s ;f=CY Z JR C,2 ;JP f,... JR NZ,1 CCF
11.	CY:=A≤s	A=s 1 CY:=CY	CP s ;f=Z JR NZ,1 ;JP f,... CCF
12.	CY:=A>s	A#s 1 CY:=CY	CP s ;f=Z JR Z,1 ;JP f,... CCF
13.	CY:=A#s	A>s 1 CY:=CY	CP s ;f=CY∧Z JR C,3 ;JP f,... JR Z,1 CCF
14.	CY:=A≥s	CY:=A<s	CP s CCF

- 9 bis 17 Laden nur eines Flagbits entsprechend Vergleich
 18 bis 25 Laden eines Flagbits entsprechend Intervalltest
 26 Laden eines Flagbits entsprechend Enthaltensein-Test

Nr.	F1:=B	F1:=B	Kode-Äquivalent
15.	Z:=A=s	Z:=A=s	CP s
16.	Z:=A≤s	A<s 1 Z:=1	CP s ;f=CY JR NC,1 ;JP f,... CP A ;Z:=1
17.	Z:=A≥s	A>s 1 Z:=1	CP s ;f=CY∧Z JR C,3 ;JP f,... JR Z,1 CP A ;Z:=1
18.	CY:=Ae[s1,s2]	CY:=Ae[s1,s2] CY 1 CY:=A≤s2	CP s1 ;s.Nr.14 CCF z: JR NC,zx-(*)+2 CP s2 ;s.Nr.11 JR NZ,1 CCF zx:
19.	CY:=Ae(s1,s2)	CY:=A>s1 CY 1 CY:=A≤s2	CP s1 ;s.Nr.12 JR Z,1 CCF z: JR NC,zx-(*)+2 CP s2 ;s.Nr.11 JR NZ,1 CCF zx:
20.	CY:=Ae(s1,s2)	CY:=A≤s1 CY 1 CY:=A<s2	CP s1 ;s.Nr.14 CCF z: JR NC,zx-(*)+2 CP s2 ;s.Nr.9 zx:
21.	CY:=Ae(s1,s2)	CY:=A>s1 CY 1 CY:=A<s2	CP s1 ;s.Nr.12 JR Z,1 CCF z: JR NC,zx-(*)+2 CP s2 ;s.Nr.9 zx:
22.	CY:=Ae[s1,s2]	CY:=A<s1 CY 0 CY:=A>s2	CP s1 ;s.Nr.9 z: JR C,zx-(*)+2 CP s2 ;s.Nr.12 JR Z,1 CCF zx:
23.	CY:=Ae(s1,s2)	Methodik der Lösung s.Nr.18 bis 22 (hier: auf der Basis von Nr.1)	
24.	CY:=Ae(s1,s2)		
25.	CY:=Ae(s1,s2)		
26.	Z:=AeQ[1:n]	HL:=Q1(-1) B:=(HL) I#B=n HL+=1 A=(HL) (B-=1)=0 2/*AeQ,Z=0 1/*AeQ,Z=1	t: LD HL,Q1(-1) LD B,(HL) ;B=n ta: INC HL CP (HL) ;Z:=A=(HL) JR Z,2 DJNZ -6 tax: ;Z:=AeQ

3.5. Spezielle Programmiermethoden

In der Übersicht über die Maschinenbefehle von Mikroprozessoren (am Beispiel des Z 80) waren in Abschnitt 3.3.2.15. einige Befehlsgruppen mit der Bemerkung zurückgestellt worden, daß ihre Anwendung untrennbar mit speziellen Programmiermethoden verbunden sei. Mit ihrer Behandlung erhält das Kapitel über elementare Programmierungstechnik seinen Abschluß.

3.5.1. Kellertechnik

3.5.1.1. Kellerprinzip

Im RAM kann nach Vorgabe durch das Bediensystem eines Mikrorechners oder auch nach freiem Ermessen eines Anwenders eine zusammengehörige, einseitig begrenzte Folge von Speicherworten als „Keller“ (engl. stack) eingerichtet werden. Im Keller können Informationen nach dem hardwaremäßig unterstützten Kellerprinzip verwaltet werden. Bild 3.5.1.a illustriert dies in Gegenüberstellung zur „Warteschlange“ am Modell eines simplen Akten-Dully. Bei der Warteschlange gelangt die zuerst eingespeicherte Information auch als erste zur Verarbeitung (FIFO = first in first out), und alles nachfolgend Gespeicherte muß erst auf den Boden „hinabsinken“. Beim Kellerprinzip kommt dagegen die zuletzt eingespeicherte Information als erste zur Verarbeitung (LIFO = last in first out), und alles früher Gespeicherte muß erst infolge Verarbeitung sämtlicher aufliegenden Informationen wieder „auftauchen“.

Die Realisierung beider Speicherprinzipien in einer linear angeordneten Folge von Speicherworten (s. Bild 3.1.5.b) offenbart den Nachteil der Warteschlange: Verwaltung jedes ihrer beiden Enden durch einen eigenen Zeiger und „Wandern“ der Schlange innerhalb der Speicherbelegung (bzw. Umspeichern). Der Keller dagegen wird mit einem einzigen Kellerzeiger SP (engl. stackpointer) verwaltet, der auf das pulsierende Ende weist und die zwanglose Neubelegerung freigelegener Worte ermöglicht.

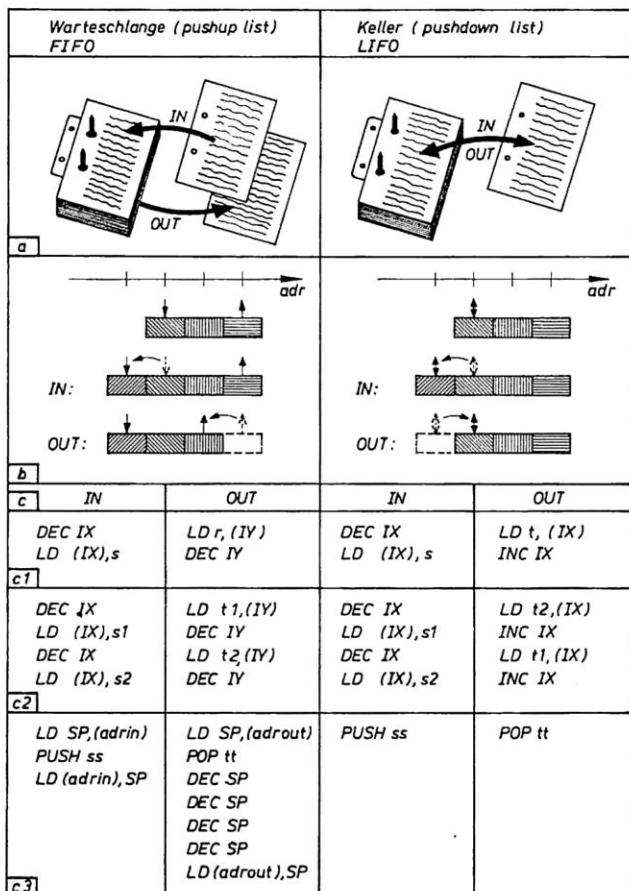


Bild 3.5.1.: Gegenüberstellung der Speichertechniken „Warteschlange“ und „Keller“. a) Bildhafte Verdeutlichung des Einspeicherns und Auslesens; b) Realisierung der Speicherprinzipien auf einer linearen Folge von Speicherworten; c) programmtechnische Modellierung von Einzelbytes (c1), von Doppelbytes (c2), von Doppelbytes mit Kelleroperationen (c3).

Es bedeuten: s, s1, s2 $\equiv$ n, r; t1, t2 $\equiv$ r; ss, tt $\equiv$ BC, DE, xx, AF; adrin, adout = Adressen von Doppelbyte-Hilfsworten zum Retten von SP

Bei Mikroprozessoren gilt für gewöhnlich:

- Der Keller wächst beim Einspeichern in Richtung niedrigerer Adressen.
- Der Kellerzeiger SP weist auf den jüngsten Wert.

In den Bildern 3.1.5.c1 und 3.1.5.c2 sind programmtechnische Modellierungen des Ein- und Ausladens von Byte- und Doppelbytwerten nach dem FIFO- und LIFO-Prinzip gezeigt, wobei zunächst noch Indexregister die Rolle benötigter Zeiger spielen.

3.5.1.2. Keller- und Austauschoperationen

a) Operationen mit dem Kellerzeiger SP

Laden: LD SP, nn ; SP $\leftarrow$ nn
LD SP, (adr) ; SP $\leftarrow$ ((adr+1), (adr))
LD (adr), SP ; ((adr+1), (adr)) $\leftarrow$ SP
LD SP, xx ; SP $\leftarrow$ xx, xx $\equiv$ HL, IX, IY } s. Abschn. 3.3.2.3.

Zählen: INC SP ; SP $\leftarrow$ SP+1
DEC SP ; SP $\leftarrow$ SP-1 } s. Abschn. 3.3.2.4.

Arith.: ADD xx, SP ; xx $\leftarrow$ SP, xx $\equiv$ HL, IX, IY
ADC HL, SP ; HL $\leftarrow$ (SP+CY)
SBC HL, SP ; HL $\leftarrow$ (SP+CY) } s. Abschn. 3.3.2.7.

Retten: s. Beispiel 21 ; xx $\leftarrow$ SP

b) Kellern und Entkellern

PUSH ss ; ss $\equiv$ BC, DE, xx, AF, s. Bild 3.5.2.a
POP tt ; tt $\equiv$ BC, DE, xx, AF, s. Bild 3.5.2.b

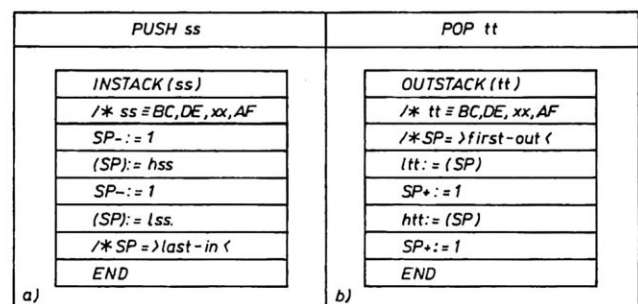


Bild 3.5.2.: Ablauf der Kelleroperationen. a) Kellern; b) Entkellern

c) Austauschen

EX (SP), xx ; xx $\equiv$ HL, IX, IY, s. Bild 3.5.3.a
EX DE, HL ; s. Bild 3.5.3.b

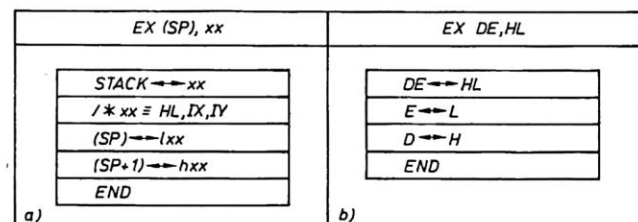


Bild 3.5.3.: Ablauf der Austauschoperationen. a) Zwischen Kellerspitze und xx-Register; b) zwischen den Doppelregistern DE und HL

Der Kellerzeiger SP ist ein eigenständiges Architecturelement (s. bereits Bild 3.2.1.), das anfangs mit der Adresse des „Kellerbodens“ geladen und (in Spezialfällen) explizit gezählt sowie arithmetischen Operationen als zweiter Operand zugeführt und softwaremäßig gerettet werden kann. Die Bedeutung des Kellerzeigers liegt in der indirekten Adressierung von Kellerworten beim Kellern und Entkellern von Doppelbytes gemäß den hardwaremäßig realisierten Standardroutinen INSTACK(ss) und OUTSTACK(tt), die der Programmierer mit den Maschinenbefehlen PUSH und POP anweist. Bild 3.5.2. zeigt ihren Ablauf in Struktogrammform und macht das implizite Schalten von SP deutlich.

Das Teilbild 3.5.1.c3 zeigt, wie einfach die Verwaltung des Kellers mit diesen Operationen wird und wie man sie sogar für die Bedienung einer Warteschlange umfunktionieren kann.

Unter Abschnitt 3.5.1.2.(c) werden schließlich noch zwei Austauschoperationen angeführt: der sachlich hierher gehö-

rige Austausch zwischen „Kellerspitze“ und xx-Register sowie der methodisch hier anführbare und vom Mikroprozessor 8080 überkommene Austausch DE $\leftrightarrow$ HL (s. Bild 3.5.3.).

3.5.1.3. Elementare Anwendungen der Kelleroperationen

Mit Kelleroperationen kann eine Reihe von kleinen Bausteinlösungen für vielfältig einsetzbare Miniaturprobleme aufgestellt werden. Der Kellerzeiger gelte stets als passend initiiert.

Beispiel 84 bis 87

Probleme: Programmierbares erweitertes Kellern und Entkellern.

84: „PUSH nn“, d. h. INSTACK(nn)

85: „PUSH (adr)“, d. h. INSTACK(((adr+1),(adr)))

```
PUSH xx          PUSH xx
LD  xx,nn        LD  xx,(adr)
EX  (SP),xx      EX  (SP),xx
```

86: „PUSH SP+nn“, d. h. INSTACK(SP+nn)

87: „POP (adr)“, d. h. OUTSTACK(((adr+1),(adr)))

```
PUSH xx          EX  (SP),xx
LD  xx,nn        LD  (adr),xx
ADD xx,SP        POP  xx
EX  (SP),xx
```

Beispiele 88 und 89

Probleme: Kellern „EXPUSH xx“ und Entkellern „POPEX xx“ hinter der Kellerspitze mit Schieben des Spitzenelementes im Keller.

```
expush:EX  (SP),xx    popex:POP  xx
        PUSH  xx      EX  (SP),xx
```

Beispiel 90

Problem: „EX (SP),(adr)“, d. h. Austausch Keller $\leftrightarrow$ Speicher

„EXPUSH xx“ ;Kellern hinter Kellerspitze, s. Beispiel 88

LD xx,(adr) ;Austausch von

EX (SP),xx ;Kellerspitze und

LD (adr),xx ;adressiertem Doppelbyte

„POPEX xx“ ;Entkellern hinter Kellerspitze, s. Beispiel 89

Beispiele 91 bis 94

Probleme: Programmierte Austauschoperationen

91: „EX tt,xx“, d. h. tt $\leftrightarrow$ xx, tt $\equiv$ BC,DE,xx,AF

92: „EX BC,rr“, d. h. BC $\leftrightarrow$ rr, rr $\equiv$ DE, HL

```
PUSH tt          PUSH BC
EX  (SP),xx      LD  B,rrr
POP  tt          LD  C,rrr
                POP  rr
```

93: „EX tt,ss“, d. h. tt $\leftrightarrow$ ss, tt,ss $\equiv$ BC,DE,xx,AF

94: „EX htt,ltt“, d. h. htt $\leftrightarrow$ ltt, z. B. B $\leftrightarrow$ C

```
PUSH tt          PUSH tt ;tt  $\equiv$  BC,DE,xx,AF
PUSH ss          PUSH tt
POP  tt          INC  SP
POP  ss          POP  tt ;htt $\leftrightarrow$ ltt
                INC  SP
```

Beispiele 95 und 96

Probleme: Abspeichern und Laden des Flagregisters F

95: „LD t,F“, d. h. t:=F mit t $\equiv$ A,B,C,D,E, (xy+e)

96: „LD F,s“, d. h. F:=s mit s $\equiv$ n,A,B,C,D,E, (xy+e)

```
PUSH AF          PUSH AF
EX  (SP),HL      EX  (SP),HL
LD  t,L          LD  L,s
POP  HL          EX  (SP),HL
                POP  AF
```

Beispiel 97

Problem: Zyklisches Verschieben einer Kellerliste aus k+1 Doppelbytes über die Kellerspitze hinweg.

Der programmtechnische Ansatz ist im Bild 3.5.4. skizziert.

3.5.1.4. Keller als temporärer Hilfsspeicher

Die programmierstrategische Bedeutung des Kellers liegt in seiner Benutzung als zeitweiliger Hilfsspeicher für Doppelbytwerte bei impliziter – also anonymer – Adressierung, wenn nur die folgenden zwei Forderungen erfüllt sind:

- Ein- und Ausladen beim Zwischenspeichern von Informationen gehen streng nach dem LIFO-Prinzip vor sich.

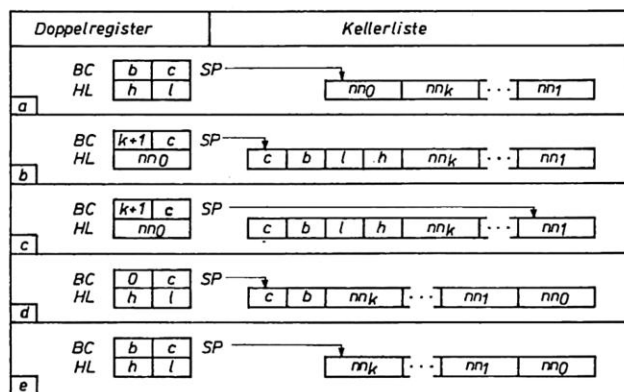


Bild 3.5.4.: Bildvorlage für das zyklische Verschieben einer Kellerliste über die Kellerspitze hinweg (s. Beispiel 97). a) Anfangszustand; b) c) d) Arbeitsphasen; e) Endzustand

- Auch bei maximaler Belastung steigt der Keller nicht in Speicherbereiche herab, die für andere Belegungen vergeben sind. (Achtung: Hardwarekontrollen hierüber fehlen!)

Als erste Anwendungsbeispiele seien hier die fragmentarischen Wiederholungen zweier früherer Beispiele geboten.

Beispiele 98 und 99

Probleme: Varianten der Beispiele 83 und 84 mit zeitweiliger Auslagerung von Doppelregisterinhalten in den Keller.

```
s: LD  A,(n);A:=n    vz: LD  IY,R1<-1
...                  ...
LD  HL,X1<          PUSH IY ;Retten>R1<-1
s1a: PUSH BC ;Retten anzm s: ...
    PUSH HL ;Retten>Xm<    PUSH BC ;Retten (m, k)
...                  t: ...
s1aax:POP  HL        tax:POP  BC
...                  ...
POP  BC              sx:POP  IY
DJNZ s1a-(*)+2      LD  (IY),C
sx: HALT              HALT
```

Bei radikaler Kellernutzung könnte man auch nur mit BC und HL auskommen.

3.5.2. Unterprogrammtechnik

3.5.2.1. Unterprogrammprinzip

Bei der programmtechnischen Umsetzung eines algorithmischen Ablaufes ist die Kodierung eines Teilablaufes als sog. geschlossenes Unterprogramm dann geboten, wenn

- das Teilprogramm an mehreren Stellen desselben Gesamtprogramms mit jeweils wechselnden Parametern dynamisch in den Ablauf einzuschalten ist bzw.
- das Teilprogramm in mehreren Problemen Verwendung findet oder für diese Möglichkeit potentiell befähigt werden soll (in diesem Fall kann auch ein komplexer Algorithmus in die UP-Form umgesetzt werden)
- beim Ausbalancieren zwischen dem Vorteil des exemplarischen (d. h. einmaligen physischen) Vertretenseins im Speicher und dem erhöhten zeitlichen Aufwand für die Parameterversorgung und den Aufruf- und Rückkehrvorgang der Vorzug dem Speicherkriterium gegeben wird.

Schwerpunkte der Konzipierung von Unterprogrammen sind:

- Versorgung des gerufenen UP mit den Eingangsparametern und Übergabe der Resultate an das rufende Programm
- Übergabe der Programmsteuerung an das UP unter Sicherstellung der Rückkehrmöglichkeit an eine definierte Stelle des rufenden Programms (vermittels einer „Link“-Adresse)
- unproblematische dynamische Verschachtelbarkeit von UP derart, daß ein UP ein weiteres (evtl. sich selbst, $\triangleq$ rekursiv) aufrufen kann.

Im Struktogramm soll das UP-Prinzip durch Parameterangaben in der Identifizierungszeile und durch die Schlußzeile RETURN angezeigt werden.

3.5.5.2. UP-Ruf- und Rückkehroperationen

- a) Ruf, unbedingt
 CALL adr ;INSTACK(*+3), PC:=adr (1 Befehl)
 RST hhH ;INSTACK(*+1), PC:=00hhH (8 Befehle)
 hhH ∈ {0, 8, 10H, 18H, 20H, 28H, 30H, 38H}
- b) Ruf, bedingt (Bedingung „cnd“ wie bei „JP cnd,adr“)
 CALL cnd,adr ;wenn cnd erfüllt,
 dann INSTACK(*+3), PC:=adr (8 Befehle)
- c) Rückkehr, unbedingt
 RET ;OUTSTACK(PC) (1 Befehl)
- d) Rückkehr, bedingt („cnd“ wie bei „JP cnd,adr“)
 RET cnd ;wenn cnd erfüllt,
 dann OUTSTACK(PC) (8 Befehle)

Bei unbedingten oder erfüllten bedingten UP-Ruf- und Rückkehroperationen akzeptieren die Standardroutinen INSTACK und OUTSTACK den Befehlszähler PC als Parameter. Beim Rufen ist das der bereits um die Befehlslänge erhöhte Adreßwert: er bezeichnet das auf den Rufbefehl folgende Byte und stellt bei einem bestimmten Modus der Parameterversorgung die Verbindungs- oder „Link“-Adresse dar, d. h. bezeichnet die Rückkehrstelle in das rufende Programm. Diese Linkadressen werden also nach dem LIFO-Prinzip im allgemeinen Keller verwaltet.

Die bedingten Operationen lassen sich als Ergebnisse bestimmter Befehlstransformationen auffassen, z. B.:

JP cnd,link }
 CALL adr } ⇒ CALL cnd,adr
 link: ... } link: ...

Der Ein-Byte-Rufbefehl RST dient zum Rufen einiger weniger ausgezeichneter UP bzw. UP-Anfänge auf definierten Plätzen und kommt besonders bei der Unterbrechungstechnik zur Anwendung.

3.5.5.3. Einführungsbeispiele für die UP-Konzipierung

Wenden wir uns der Konzipierung eines universellen UP zur Realisierung beliebiger elementarer Schaltfunktionen zweier Variablen $u = f_i(x, y)$, $0 \leq i \leq 15$, zu (s. bereits Abschn. 3.1.5., wobei die Übertragbarkeit des Verfahrens auf den Fall dreier Variablen gesichert ist).

Sei $X = (000000xy)$ die in ein Byte gepackte Variablenbelegung, interpretiert als Dualdarstellung einer Zahl, so daß (dezimal gesehen) gilt: $0 \leq X \leq 3$. Der Index i der Schaltfunktionen ist nun so definiert, daß mit den Ziffern seiner Dualdarstellung

$$i = u_3 \cdot 2^3 + u_2 \cdot 2^2 + u_1 \cdot 2 + u_0 = u_3 u_2 u_1 u_0 B$$

die fundamentale Beziehung gilt:

$$u_X = f_i(X).$$

Hieraus folgt die Möglichkeit, den Wertevorrat von $f_i(x, y)$ entweder zu einem einzigen Byte gepackt oder aber als Tafel mit vier Funktionswerten von Byteformat darzustellen:

X	$f_i(X)$
(000000xy)	gepackt
(00000000)	($u_0 \dots u_0$)
(00000001)	($u_1 \dots u_1$)
(00000010)	($u_2 \dots u_2$)
(00000011)	($u_3 \dots u_3$)

Bei Vertafelung dient X zum Auslesen des Funktionsbytes gemäß

$$f_i(X) = (u_X \dots u_X) = (f_i(00000000) \{+X\}),$$

wobei jede elementare Schaltfunktion f_i durch eine eigene Tafel mit der Anfangsadresse $f_i(00H) \{$ repräsentiert ist. Nachstehend werden drei UP-Varianten des Tafelverfahrens mit unterschiedlicher Parameterversorgung angegeben:

Beispiele 100 bis 102

Probleme: UP-Varianten zur Realisierung beliebiger elementarer Schaltfunktionen $u = f_i(x, y)$ mit vertafeltem Wertevorrat.

Parameterübergabe:

bei allen UP: Eingangswert X und Resultat $f_i(X)$ im Akku A
 100: $f_i(00H) \{$ in einem Doppelregister

101: $f_i(00H) \{$ in der Anschlußzeile, d. h. als textlicher Nachfolger des Rufbefehls. Definition des Adreßwertes mit der Pseudooperation „DW $f_i(00H) \{$ “ (2 Bytes, kein op-Kode).

102: $f_i(00H) \{$ im Keller

Aufrufbeispiele (Parameterversorgung):

LD BC, $f_i(00H) \{$ LD A, 000000xyB LD tt, $f_i(00H) \{$
 LD A, 000000xyB CALL schfu2 PUSH tt
 CALL schfu1 DW $f_i(00H) \{$ LD A, 000000xyB
 link: ... link: ... link: ...
 CALL schfu3

UP-Varianten:

schfu1: AND 03H schfu2: POP HL schfu3: POP HL
 LD L, A LD C, (HL) EX (SP), HL
 LD H, 00H INC HL AND 03H
 ADD HL, BC LD B, (HL) LD C, A
 LD A, (HL) INC HL LD B, 00H
 RET PUSH HL ADD HL, BC
 AND 03H LD A, (HL) LD A, (HL)
 LD L, A RET

Beispiele 103 und 104

Probleme: Konzipierung von „Flag-Prozeduren“, d. h. von UP mit einem bestimmten Flagbit oder dem Flag insgesamt als Resultat.

103: Schaltfunktion $f_i(x, y)$ mit gepacktem Wertevorrat:
 CY := u_X durch X-maliges Rechtsrotieren zum Flagbit CY.

104: Enthaltenseinstest nach Beispiel 84 bzw. 99.

Aufrufbeispiele:

LD C, i ;C:=i LD A, arg ;A:=arg
 LD A, 000000xyB LD HL, $Q_i \{-1$
 CALL schfu CALL enth

Unterprogramme (mit Retten von BC):

schfu: PUSH BC enth: PUSH BC
 AND 03H ;Xmod4 LD B, (HL); B:=n
 LD B, A entha: INC HL ;O_j{
 INC B CP (HL)
 schfua: RRC C ;CY:= u_k JR Z, 2
 DJNZ -4 ;A=xy DJNZ -6
 schfuax: POP BC ;C:=i enthax: POP BC
 RET ;CY= u_X RET ;Z=arg ∈ Q[1:n]

Beispiele 105 und 106 (UP-Verschachtelung)

Problem: Modifizierung der Lösungen 101 und 102 zwecks interner Angleichung der Parameterübergabe an den Modus des UP SCHFU1 (ohne Retten weiterer Doppelregister)

schfu20: POP HL schfu30: POP HL
 LD C, (HL) POP BC ;BC:= $f_i(00H) \{$
 INC HL PUSH HL
 LD B, (HL) CALL schfu1
 INC HL RET
 PUSH HL ;HL=link
 CALL schfu1
 RET

3.5.5.4. Systematisierung der Parameterübergabe

Die Parameterübergabe zwischen rufendem und gerufenem Programm vollzieht sich im wesentlichen nach den folgenden Grundschemata, die natürlich auch vermischelt auftreten können:

- in Registern bzw. Doppelregistern
- auf absolut oder relativ fest vereinbarten Plätzen im Bereich der Speicherbelegung
 - $B_w^{(HP)}$ des rufenden Programms („Hauptprogramm“)
 - $B_w^{(UP)}$ des gerufenen Programms (Unterprogramm)
- textlich hinter dem Rufbefehl (d. h. innerhalb B_{AP}) durch
 - Angabe der Parameterliste PL aus Werten und/oder Adressen
 - Angabe der Adresse dieser Parameterliste PL ($\hat{=}$ Adresse der Adresse des ersten Parameters in PL; meist $PL \in B_w^{(HP)}$)
- im Keller (unmittelbar hinter der gekellerten Link-adresse).

Wird fortgesetzt

Dr.-Ing. WOLFGANG HERRMANN,
Dr.-Ing. JÜRGEN SCHLECHTER und
Dr.-Ing. ISTVAN WINDISCH

Betrachten wir anhand des Problems der Vektorzerlegung „vz“ nach Beispiel 84 mit dem Enthaltenseinstest in Form der Flagprozedur nach Beispiel 104 folgende zwei Grundtypen der Konzipierung von „vz“ als UP mit drei Adreßparametern:

Beispiele 107 und 108

107: Parameterübergabe in Doppelregistern

108: Annahme, daß das UP die Parameter in folgender Liste innerhalb $B_w(UP)$ vorfindet:

adad1:DS 2;R₁←1 Anm.: DS ist eine weitere
adad2:DS 2;P₁←1 Pseudooperation zur Reservie-
adad3:DS 2;Q₁←1 rung von Speicherplatz (hier:
2 Bytes)

Aufrufmuster:

LD IY,R₁←1 ... ;Parameterliste
LD DE,P₁←1 ... ;in $B_w(UP)$ gelte
LD HL,Q₁←1 ... ;als gefüllt!
CALL vz1 CALL vz2

link:...

link:...

Unterprogramme (aufeinanderfolgende PUSH/POP symbolisch angeben):

vz1: PUSH AF,BC LD C,0 LD A,(DE) LD B,A PUSH IY ;Retten>R₁←1 s: INC DE LD A,(DE) PUSH HL ;Retten>Q₁←1 t: CALL enth ;Beispiel 104 tax: POP HL r: JP f,rx rd: INC C INC IY LD (IY),A rx: DJNZ s-(*)+2 sx: POP IY LD (IY),C POP BC,AF RET	vz2: PUSH AF,BC,IY,DE,HL LD IY,(adad1) LD C,0 LD DE,(adad2) LD A,(DE) LD B,A s: INC DE LD A,(DE) LD HL,(adad3) t: CALL enth r: JP f,rx rd: INC C INC IY LD (IY),A rx: DJNZ s-(*)+2 sx: LD IY,(adad1) LD (IY),C POP HL,DE,IY,BC,AF RET
---	--

Beispiele 109 und 110 (PL nach Rufofbericht)

109: Ohne Registerrettung

110: In $B_w(UP)$ ein Hilfswort adsp:DS 2; SP-Rettung

Gemeinsames Aufrufmuster:

CALL vz3 bzw. vz4
pl: DW >R₁←1
DW >P₁←1
DW >Q₁←1
link:...

vz3: EX (SP),IX ;IX:=pl LD DE,0006H ADD IX,DE ;IX:=link LD E,(IX-6) LD D,(IX-5) PUSH DE POP IY ;IY:=>R₁←1 LD E,(IX-4) LD D,(IX-3) LD L,(IX-2) LD H,(IX-1) EX (SP),IX CALL vz1	vz4: EX (SP),HL ;HL:=pl LD (adsp),SP;Retten SP LD SP,HL ;SP:=HL POP HL LD (adad1),HL POP HL LD (adad2),HL POP HL LD (adad3),HL LD HL,0 ;SP:=pl+6 ADD HL,SP ;HL:=link LD SP,(adsp);Holen SP EX (SP),HL
--	--

RET

CALL vz2

RET

Beispiele 111 und 112 (PL abgetrennt in $B_w(HP)$)

111: Ohne Registerrettung

112: In $B_w(UP)$ zwei Hilfs Worte adsp:DS 2 ;SP-Rettung
adlk:DS 2 ;Link-Rettung

Gemeinsames Aufrufmuster:

CALL vz5 bzw. vz6
adpl:DW pl
link:...

vz5: EX (SP),IX ;IX:=adpl INC IX INC IX ;IX:=link LD E,(IX-2) LD D,(IX-1);DE:=pl EX (SP),IX ;SP:=link PUSH DE EX (SP),IX ;IX:=pl LD E,(IX) LD D,(IX+1) PUSH DE POP IY LD E,(IX+2) LD D,(IX+3) LD L,(IX+4) LD H,(IX+5) POP IX CALL vz1 RET	vz6: EX (SP),HL ;HL:=adpl LD (adsp),SP LD SP,HL ;SP:=adpl POP HL ;HL:=pl LD (adlk),SP;SP:=link LD SP,HL ;SP:=pl POP HL LD (adad1),HL POP HL LD (adad2),HL POP HL LD (adad3),HL LD HL,(adlk) LD SP,(adsp) EX (SP),HL CALL vz2 RET
--	---

Beispiele 113 und 114 (Parameter im Keller)

113: Ohne Registerrettung

114: In $B_w(UP)$ ein Hilfs Wort adhl:DS 2 ;HL-Rettung

Gemeinsames Aufrufmuster

LD tt,R₁←1 PUSH tt LD tt,P₁←1 PUSH tt LD tt,Q₁←1 PUSH tt CALL vz7 bzw. vz8 link:...	vz8: LD (adhl),HL POP HL ;HL:=link EX (SP),HL LD (adad3),HL POP HL ;HL:=link EX (SP),HL LD (adad2),HL POP HL ;HL:=link EX (SP),HL LD (adad1),HL LD HL,(adhl) CALL vz2 RET
---	--

Als Resümee ergibt sich folgende Konzipierung der Parameterübergabe:

- für einfache UP und oder wenig Parameter: in den Registern, nach Muster von „vz1“
- für komplexe UP und/oder viele Parameter: in einer UP-eigenen Liste nach Muster von „vz2“. Das konzipierte UP ist dann logisches Kernstück eines „Hüllprogramms“, das die gängigen Parameterübergabefälle nach den Mustern vz4, vz6 und vz8 verarbeitet und jene Liste füllt. Das ist zwar aufwendig, sichert aber durchgehende Registerrettung und ist leicht automatisierbar; die symbolischen Adressen erfordern jedoch im Prinzip bereits Assembler-niveau der Programmierung.

3.5.3. Anmerkung

Eine Übersicht über Kommunikationsprobleme (Ein- und

Ausgabe, Unterbrechungstechnik) sowie die Einschätzung der arg Z 80 implementierten Gruppenoperationen für Laden, Vergleichen, Ein- und Ausgeben muß hier zugunsten von Applikationsbeschreibungen unterbleiben, die diese Probleme zum Teil konkret angehen. Es sei lediglich auf [1] [2] verwiesen.

4. Anwendungen von Mikrorechnern

4.1. Anwendungsgebiete und -möglichkeiten

Die Einsatzgebiete der Mikroprozessor- bzw. Mikrorechner-technik sind äußerst vielgestaltig, sie umfassen sowohl traditionelle Fachgebiete der Elektronik, wie Rechentechnik und Datenverarbeitung, Steuer- und Regelungstechnik, Nachrichtentechnik und Meßtechnik mit ihren Teilgebieten, als auch neu erschlossene Anwendungen, wie z. B. Spezialgebiete der Land- und Forstwirtschaft, der Medizin- und Labortechnik, der Bürotechnik und der Automatisierung von Fertigungsprozessen aller Art. International ergibt sich dabei zur Zeit die im Bild 4.1.1. dargestellte Aufteilung.

Der internationale Trend der Mikroelektronikanwendung sieht für Spezialanwendungen die Nutzung kompletter Standardbaugruppen (wie z. B. Konfigurationen des K 1510, K 1520) vor, während für Massenstückzahlen Spezialentwicklungen der Mikroelektronik sowohl mit universellen als auch mit kundenspezifischen LSI-Schaltkreisen genutzt werden.

Doch gerade die Möglichkeit, immer komplexere Funktionen auf einem Chip zu integrieren, und die Entwicklung universeller, programmierbarer, modular erweiterbarer Baugruppen lassen diese Trennung zwischen beiden Varianten verschwinden. So bieten z. B. die Einkartenrechner (Single Board Computer, SBC), unter die auch die zentrale Recheneinheit (ZRE) des K 1520 einzuordnen ist, als vollständige, universelle Rechner bereits die Möglichkeit für eine effektive Nutzung auch bei größeren Stückzahlen.

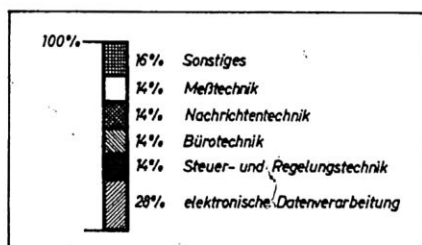


Bild 4.1.1.: Mikrorechneranwendungen

4.1.1. Mikroelektronik in Gebieten mit traditioneller Elektronik-Nutzung

Hier erfolgt der Einsatz mikroelektronischer Bauelemente insbesondere unter folgenden Aspekten:

- Nutzung des günstigeren Preis-Leistungsverhältnisses
- geringere Masse, Volumeneinsparung, niedrigerer Energiebedarf
- Einführung neuer Leistungs- und Servicemerkmale bei gleichem oder niedrigerem Aufwand
- Verkürzung der Entwicklungszeiten
- vereinfachter Schaltungsentwurf, da hochintegrierte Schaltkreise viele Schaltungsdetails, die bisher vom Gerätehersteller konzipiert werden mußten, bereits enthalten
- günstigere Erweiterungsmöglichkeiten durch modulare Gestaltung der Hard- und Software.

4.1.2. Mikroelektronik in Gebieten mit Relais- und diskreter Technik

Hier sind beim Anwender Elektronikgrundlagen vorhanden, es ist jedoch ein an die Bedingungen der modernen Mikroelektronik gebundenes Umdenken erforderlich.

Insbesondere gewinnt verstärkt der Umgang mit fertig konzipierten Funktionseinheiten, die mit entsprechenden Anschlußwerten für den Nutzer repräsentant sind, an Bedeutung. Außerdem geht die Software-Komponente (insbesondere auch die softwaremäßige Implementierung von Logik-

Literatur

- [1] Blomeyer-Bartenstein, H. P.; Mindermann, B.: Peripheriebausteine für den Mikroprozessor Z 80 (Anschluß- und Programmierhinweise). elektronikpraxis (1977) H. 9, S. 18-29
- [2] Nicoud, J. D.: Microprocessor ZILOG Z 80: Hardware and software. MicroScope II (1977) H. 9, S. 1-20

Hardware bei zeitunkritischen Prozessen) in die gesamte Systemkonfiguration ein.

Die Umsetzungsstrategie zur Lösung der Aufgabenstellung mit mikroelektronischen Bauelementen ist gerade für den Lösungseffekt von außerordentlicher Bedeutung. Es muß beachtet werden, daß kein einfaches Umtransformieren von Einzelfunktionen der bisherigen Realisierung in die Möglichkeiten der neuen Technik erfolgen sollte, sondern daß der Weg über die Schaffung an die Rechentechnik angelehnter Algorithmen für den zu realisierenden Gesamtablauf gegangen wird.

Bei der Anwendung sind insbesondere folgende Schwerpunkte zu beachten:

- Erfüllt die mit Mikroelektronik ausgerüstete Gesamtanordnung (also auch mit Meß- und Stellgliedern) gleiche bzw. bessere Leistungsmerkmale (z. B. Durchsatz, Zuverlässigkeit, Störsicherheit, Wartungsarmut)?
- Ist der Aufwand an der direkten Schnittstelle zum Prozeß an das Preis-Leistungsverhältnis der Mikroelektronik angepaßt?
- Werden optimale Meß- und Stellglieder genutzt?
- Welche Zusatzschaltkreise (universelle oder kundenspezifische bei größeren Stückzahlen) sind notwendig, um die Effektivität der Gesamtanordnung zu gewährleisten?

Diese Aspekte sind vor allen Dingen bei der Einführung der Mikroelektronik in allen Gebieten der Automatisierungstechnik zu beachten.

4.1.3. Einsatz der Mikroelektronik in nichtelektronischen Gebieten

Auch in Bereichen der Wirtschaft, in denen bisher aus Preis-, Platz- oder Energiegründen nicht an den Einsatz von Elektronik (bzw. der Rechentechnik) zu denken war, ergeben sich Anwendungsmöglichkeiten, z. B. in der Land- und Forstwirtschaft und in neuen Bereichen der Konsumgüterindustrie. Der Finalproduzent wird zum Erstanwender der Mikroelektronik. Durch interdisziplinäre Zusammenarbeit müssen hier die Erfahrungen anderer Volkswirtschaftszweige genutzt werden. Das erfordert neben einem Gesamtverständnis der Mikroelektronik vor allem eindeutige Schnittstellenmodelle, an denen sich Elektroniker und Nichtelektroniker begegnen.

4.1.4. Anwendungsmodelle

Als allgemeine Modellvorstellung bietet sich, angelehnt an die Angewandte Automatentheorie, ein Steuermodul (oder -automat) $\mathcal{A}$ an, der eine Menge von Eingangszuständen in eine Menge von Ausgangszuständen unter Einnahme einer Menge von inneren Zuständen transformiert (siehe Bild 4.1.2.).

Der Digitalinformation verarbeitende Mikrorechner stellt dabei einen Sonderfall dieses Steuermoduls $\mathcal{A}$ dar. Auch hier wird eine Menge von Eingangsdaten in eine Menge von Ausgangsinformationen überführt. Es erweist sich als notwendig, sich sowohl mit der Hardware als auch der Software und ihrer gegenseitigen Durchdringung vertraut zu machen.

Von praktischer Bedeutung ist insbesondere die Interface-Gestaltung. Gerade bei der Nutzung moderner Interface-Schaltkreise (PIO, SIO, USART, CTC) treffen Hard- und Software besonders zusammen, d. h., mit Software kann die Hardware entsprechend ihrem Verwendungszweck modifiziert werden.

Im folgenden sollen, aufbauend auf diesem Grundprinzip, drei typische Modelle vorgestellt werden.

Modell A

Dieses Modell stellt abgerüstete mikroelektronische Steue-

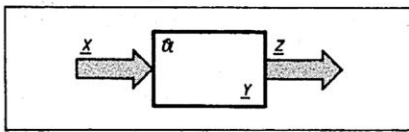


Bild 4.1.2.: Steuermodul α .

X = Menge der Eingangszustände mit $x_i \in X$ (Eingangsvariablen);
 Y = Menge der inneren Zustände mit $y_i \in Y$ (Zustandsvariablen);
 Z = Menge der Ausgangszustände mit $z_i \in Z$ (Ausgangsvariablen)

Bild 4.1.3.: Einsatz mikroelektronischer Schaltkreise für einfache Steuerungsalgorithmen (nach dem Prinzip der Folgesteuern)

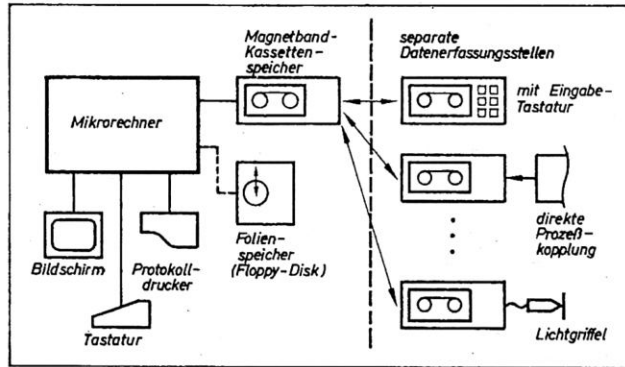
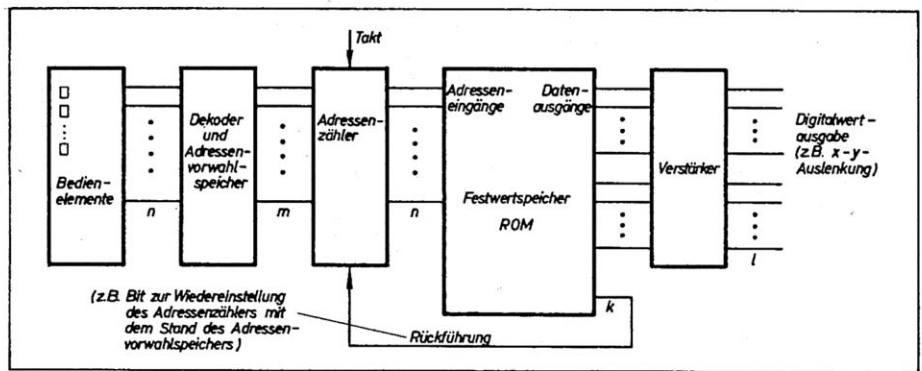


Bild 4.1.4.: Terminalsystem mit Mikrorechner (z. B. K 1520) zur Datenerfassung und -verarbeitung

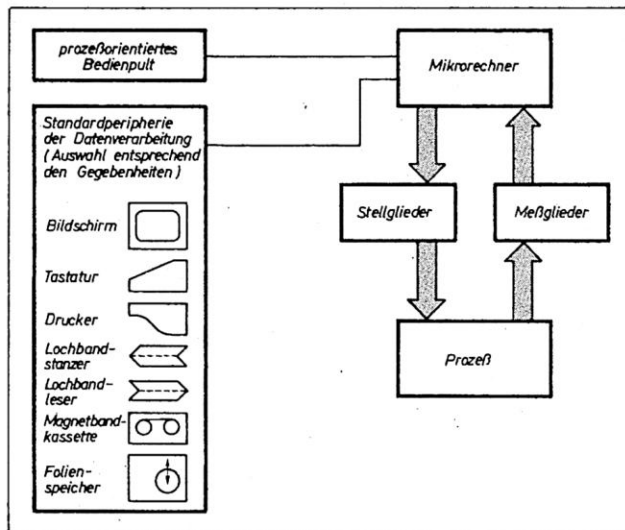


Bild 4.1.5.: Mikrorechnersystem zur On-line-Prozeßsteuerung

rungen dar, die vom Prinzip unter dem Begriff der Folgesteuern erfaßt sind und eigentlich schon nicht mehr in das Gebiet Mikroprozessortechnik eingeordnet werden, da sie ja gar keinen „echten“ Mikroprozessor enthalten. Aber gerade die ökonomische Nutzung dieses Prinzips in sehr vielen Gebieten (u. a. Konsumgüterelektronik) soll eine Einordnung an dieser Stelle rechtfertigen. Außerdem werden von diesen Anordnungen, die oftmals durch kundenspezifische Schaltkreise realisiert werden, ebenfalls die Grundkomponenten Verarbeitung, Speicherung, Ein- und Ausgabe erfüllt.

Gerade durch die Nutzung von Festwertspeichern (ROMs oder auch von PLAs) wird ein hoher Grad der Flexibilität bezüglich der Anwendungen erreicht. Allerdings muß bemerkt werden, daß bei ROM-Nutzung die gesamte Lösungsmenge bereits bei der Algorithmenerstellung bekannt sein muß. Ein einfaches Nutzungsbeispiel ist im Bild 4.1.3. gezeigt.

Anwendungsfälle (auch anderer Realisierungsvarianten nach ähnlichem Prinzip) sind z. B.

- Nähmaschinen-, Strickmaschinensteuerungen
- Steuerungen für Grill- und Backgeräte
- Zeitablaufsteuerungen.

Modell B

Dieses im Bild 4.1.4. gezeigte System stellt eine Verkörperung des Prinzips der „dezentralen Intelligenz“ dar. Durch das günstige Preis-Leistungsverhältnis der Mikroelektronik ist es möglich geworden, den Datenverarbeitungsendplatz direkt mit einem Rechner auszurüsten. Dieses System, das natürlich ebenfalls im On-line- oder Off-line-Betrieb mit einem übergeordneten Rechner kommunizieren kann, erfüllt seine Aufgaben dort, wo die Probleme entstehen. Als Datenerfassungs- und Datenverarbeitungssystem bietet es, ausgerüstet mit entsprechender Peripherie, alle Möglichkeiten für einen operativen Einsatz. Bildschirm und Tastatur gestatten günstige Kommunikationsmöglichkeiten zwischen Mensch und System, ein Protokoll-drucker ist oftmals in den typischen Anwendungsfällen unentbehrlich.

Anwendungsfälle sind

- Verwaltungsprozesse bei der Lagerhaltung
- Meßwerterfassung von dezentralen Meßstellen (Konzentration der Datenmenge und Weiterleitung an übergeordneten Rechner)
- Einsatz als Hilfsmittel für operative Entscheidungsfindung (Verwaltung von Umschlagprozessen, Buchungsprozessen usw. in kleineren Dimensionen)
- statistische Zwecke.

Modell C

Hier wird der Mikrorechner, der direkt mit dem Prozeß gekoppelt ist, im Prinzip als (Mikro-)Prozeßrechner genutzt. Als besonderer Aspekt dieser Anordnung (Bild 4.1.5.) ist zu beachten, daß die Art und Weise der Kopplung mit dem Prozeß entscheidend die Effektivität der Gesamtanordnung bestimmt.

Für einfachste Anwendungen ist es günstig, billige Meß- und Stellglieder, die die digitale E-A-Schnittstelle des Mikrorechners möglichst ohne Anpassmaßnahmen nutzen, anzuwenden. Eine A-D- und D-A-Umsetzung sollte in Genauigkeit und Geschwindigkeit angepaßt und möglichst nur dort, wo es unbedingt notwendig ist, vorgenommen werden.

Für die Nutzungs- und Servicefreundlichkeit ist insbesondere der Einsatz eines prozeßorientierten Bedienpults von Bedeutung. Der Nutzer will wie bisher seine Maschine, sein Aggregat und keinen Rechner bedienen, erwartet also durch den Rechnereinsatz keine Verkomplizierung, sondern mehr Bedienkomfort.

Da die Mikrorechner prinzipiell die Anschlußmöglichkeiten für die Standardperipherie der Datenverarbeitung besitzen, ist es möglich, je nach den Gegebenheiten eine Peripherieauswahl anzuschließen.

Anwendungsfälle sind z. B.

- Einsatz in Prozessen der Automatisierungstechnik allgemein (Drehzahlregelung, Industrieroboter)
- meßtechnische Anwendungen (Labor- und Präzisionsmeßgeräte)
- Medizindiagnostik
- Systeme der allgemeinen Sicherheitstechnik (Klimaregelung, elektrische Schließsysteme, Brandwarmanlagen)
- Verkehrsablaufsteuerungen (Post, Bahn, Ampelsteuerung).

4.1.5. Grenzen der Anwendung

Während in den Anfängen der Mikroprozessortechnik oftmals die Verarbeitungsgeschwindigkeit eine ernsthafte

Grenze darstellte, werden jetzt mit Prozessoren der 2. und 3. Generation Werte erreicht, die für den größten Teil der Standardanwendungen völlig ausreichend sind. Für hohe Geschwindigkeitsanforderungen werden international bereits bipolare Mikroprozessor-Konzepte angeboten.

Während sich für Einfachanwendungen die Richtung der Nutzung von Einchip-Mikrorechnern abzuzeichnen scheint, wird andererseits die Schranke der 8-Bit-Wortlänge beseitigt. Hier ist einmal der Weg zur Entwicklung oftmals eng auf Vorgängersysteme bezogener 16-Bit-Standardprozessoren (z. B. Intel 8086, Z 8000), zum anderen auch die Entwicklungslinie der Bit-Slice-Prozessoren zu beachten. Die größere Wortbreite ermöglicht den Einsatz der Mikrorechner-technik in Gebieten, die der bisherigen Kleinrechner-technik vorbehalten waren.

Waren bisherige Rechner an klimatisierte Räume gebunden, so sind als wesentlicher Vorteil auch bei hochintegrierten Schaltungen Temperatur ($\vartheta_s = -65 \dots 125^\circ\text{C}$, $\vartheta_a = 0$ bis 70°C) und Luftfeuchtigkeit relativ unproblematisch.

Kritisch sind aber sowohl für den Schaltkreis als natürlich auch für den Gesamtaufbau mechanische Beanspruchungen, insbesondere Stoßbeanspruchungen.

Das Erreichen einer geringen Störanfälligkeit gegenüber benachbarter Elektronik und Starkstromtechnik zählt für sehr viele Einsatzfälle mit zu einem Schwerpunkt der Entwicklungsarbeiten. Hier bleibt die endgültige Lösung oftmals den in Testreihen ermittelten, für andere Systemkonzepte nicht reproduzierbaren Anordnungen vorbehalten.

Allgemein sollten jedoch folgende prinzipielle Maßnahmen beachtet werden:

Möglichst kurze Leitungsführung; Steuersignalleitungen mit Masseleitungen bei längerer Leitungsführung verdrillt ausführen; Vermeidung einer Parallelführung von Interface-Leitungen mit systemfremden Leitungen in gemeinsamen Kabelschächten; Anwendung von Kabeltreibern und -empfängern (größerer Störabstand des Übertragungspegels); Anwendung der langsamen Störsicheren Logik (LSL) auf der Peripherieseite; Beachtung von Abschirmprinzipien; Anwendung von Schaltungen bzw. logischen Schaltungskonzepten zur Unterdrückung von Störungen.

Zusammenfassend sollten folgende Aspekte bei der Mikroprozessor- bzw. Mikrorechneranwendung beachtet werden:

- Die Mikroelektronik setzt sich nicht im Selbstlauf durch! Die Phase, in der wir uns heute befinden, setzt die schöpferische Mitarbeit der Werkstätten aller Volkswirtschaftszweige voraus. Die Mikroelektronik verlangt durch ihr Eindringen in die unterschiedlichsten volkswirtschaftlichen Bereiche verstärkt die interdisziplinäre Zusammenarbeit.
- Der Mikrorechner ist eigentlich nichts anderes als ein billiger Rechner. Es ergibt sich die günstige Möglichkeit, Probleme der Hardware verstärkt in die Software zu verlagern.
- Der Mikrorechner sollte nicht um jeden Preis eingesetzt werden, sondern nur dort, wo er heute und in Zukunft effektiv Nutzen bringt, vor allen Dingen aber dort, wo es in Zukunft unbedingt notwendig sein wird, um den Prozeß überhaupt entsprechend den Standards zu beherrschen.

4.2. Digitalvoltmeter

Ein besonders großes Anwendungsgebiet von Mikroprozessoren in der Meßtechnik sind Digitalvoltmeter. Wie z. B. das Digitalvoltmeter 6000 [4.2.1.] zeigt, führt der Einsatz eines Mikroprozessors in der Steuerung des Meßgerätes zur Universalität, Senkung des Hardware-Aufwandes und leichteren Bedienung.

Die bezüglich der Zeitparameter nicht kritischen Funktionen lassen sich mit Hilfe der Software realisieren. So werden z. B. das Anzeigeformat mit Vornullunterdrückung, die Ziffernkodierung für Siebensegmentanzeige und das Setzen des Dezimalpunktes mit Programmen durchgeführt. Dementsprechend enthält ein Anzeigeelement nur LEDs, einige Widerstände und ein 8-Bit-Register. Auch die Tastatur-Logik besteht aus nur zwei billigen Schaltkreisen; Abfrage, Entstörung und Kodierung werden durch eine Programmroutine organisiert.

Die Bedienung der in diesem System benutzten IEE-488-Schnittstelle erfordert kurze Reaktionszeiten, die durch Mikroprozessorsteuerung nicht eingehalten werden könnten. Die Abwicklung der Signalfolgen wird deshalb von einem diskret aufgebauten Mikroprogrammsteuerwerk ausgeführt, die Information aber wird mit dem Mikroprozessor bearbeitet.

Das mikroprozessorgesteuerte Digitalvoltmeter bietet gegenüber den herkömmlichen Geräten folgende Vorteile:

- einfache Funktionswahl durch Programme
Die gewünschte Funktion wird entweder mit der Bedientastatur oder durch Kommandos über die IEE-488-Interface eingestellt.
- einfache Bedienung durch automatische Kalibrierung
Bei der Hauptkalibrierung werden die aktuellen Werte der primären Elemente, wie Referenzspannung, Eingangsverstärkung und Spannungsteiler, gemessen und in den Festwertspeicher eingeschrieben. Diese Daten werden dann von dem Mikroprozessor zur Korrektur der Meßwerte verwendet. Dadurch, daß diese Primärelemente keine große Genauigkeit mehr haben müssen, entfällt eine große Anzahl von Justiergliedern. Eine periodische Hauptkalibrierung muß alle 90 Tage ausgeführt werden. In der Zwischenzeit wird die Wirkung von Drift und Änderung der Sekundärelemente mit Programmen der automatischen Kalibrierung korrigiert.
- neue Funktionen
 - Nullpunkt- und Skalenbereichsfestlegung. Diese Funktion konvertiert das Meßergebnis in eine gewünschte Maßeinheit; sie gestattet dem Anwender z. B. eine prozentuale Skala zur gegebenen Basis, eine einem anderen Gerät ähnliche Anzeige und/oder eine Nullpunkt- und Skalenfehlerkorrektur
 - Minimumwert-, Mittelwert- und Maximumwertbestimmung. Sie ermöglichen die Grenzwertfassung auch über längere Zeitdauer. Diese Funktion kann auch als digitales Filter zur Beseitigung der Störungen von Geräuschen benutzt werden.
 - Die Größer-Kleiner-Funktion gestattet dem Benutzer, sieben Meßwerte anzugeben, die als Bereichsgrenzen gespeichert werden. Bei der Messung wird festgestellt und angezeigt, in welchem Bereich das Ergebnis liegt.
 - Die Dezibel-Messung-Funktion zeigt die Meßwerte, auf 1 mW Leistung bezogen, an.
 - Die MATH-Funktion ermöglicht eine freie Wahl eines beliebigen 0-dB-Niveaus bei Anwendung vom AC-Adapter. Dadurch wird es leichter, -3 dB oder andere Meßpunkte zu bestimmen.

Wie vorteilhaft und kostensparend die Anwendung dieser Technik sein kann, zeigt die Tatsache, daß dieses Digitalvoltmeter von sieben Ingenieuren in nur sechs Monaten entwickelt wurde.

Wird fortgesetzt

Dr.-Ing. ISTVAN WINDISCH und
Dipl.-Ing. THOMAS KRÜGER

4.3. Wägetechnik

Die überwiegende Mehrheit der gegenwärtigen Wägeeinrichtungen ist modular aufgebaut, wobei die einzelnen Modulen bestimmte wägetechnische Funktionen haben. Durch zweckmäßigen Zusammenbau der entsprechenden, zueinander kompatiblen Einheiten können verschiedenartigste Anwenderansprüche erfüllt werden. Eine spätere Änderung oder Erweiterung der Funktion ist jedoch nur durch individuellen Umbau des Wäegerätes möglich.

Beim Einsatz von Mikroprozessoren in der Steuerung werden die einzelnen wägetechnischen Aufgaben durch Programmblöcke realisiert. In der Produktion können also gleichartige Zentraleinheiten unabhängig von den Kundenwünschen hergestellt werden. Selbstverständlich werden die peripheren Geräte, Bedienungs- und Anzeigeelemente weiterhin nach Bedarf geliefert, sie sind aber über eine einheitliche Schnittstelle einfach an das System anschließbar.

Neben den herkömmlichen wägetechnischen Funktionen können nun auch solche Aufgaben von den Mikroprozessoren dezentralisiert ausgeführt werden, die bisher den zentralen Prozeßrechner belasteten. Einige Beispiele für die Erweiterung der Leistungen in mikroprozessorgesteuerten Wägeeinrichtungen sollen hier gezeigt werden:

- On-line-Kontrolle der Betriebsfähigkeit der Waage, bei erkanntem Fehler schnelle Fehleranalyse
- Schutz gegen fehlerhafte Bedienung
- Verknüpfung von Taragewichten mit anderen Angaben
- Programmierbarkeit beliebiger Druckbilder für die Buchung
- Operatortätigkeit und Datenausgabe können über UB-Steuerung organisiert werden, d. h., sie stören die Messung nicht
- mehrere, voneinander unabhängige Dosierungszyklen können organisiert werden
- Glaubwürdigkeitsuntersuchungen des Brutto-Meßergebnisses
- bei kontinuierlich arbeitenden Dosierungseinrichtungen kann z. B. der Nachfluß automatisch ermittelt werden, so daß der Bediener nur die gewünschten Komponentengewichte angeben muß
- Rezeptur kann in beliebiger Form angegeben werden
- bei Dosierungs- und Mischungsprozessen können auch äußere Parameter berücksichtigt werden

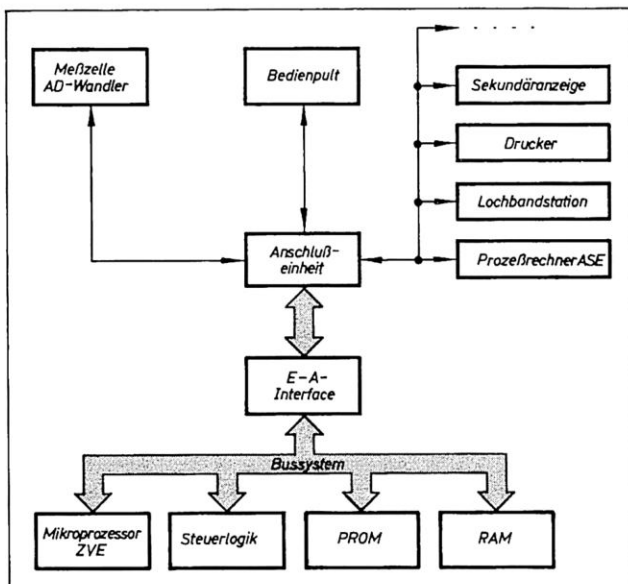


Bild 4.3.1.: Prinzipschaltung des Wägesystems mit Mikroprozessor

- eventuell können mehrere Meßplätze von einem Prozessor bedient werden
- beim Wägen von beweglichen Lasten können die Meßzeiten durch mit dem Mikroprozessor realisierte digitale Filter in realen Grenzen (1 s) gehalten werden.

Die ungarische Waagenfabrik METRIPOND entwickelt auf Grund der vorstehend genannten Vorteile ein Wägesystem mit Mikroprozessorsteuerung [4.2.2.]. Das Bild 4.3.1. zeigt in Blockdarstellung den prinzipiellen Aufbau des Systems.

Literatur

- [4.2.1.] Cruzan, G. R., u. a.: For a versatile digital voltmeter a microprocessor plus software. Electronics 51 (1978) H. 12, S. 133-139
[4.2.2.] Mészáros, L.; Bertyni, A.: Mikroprozessoren in automatischen Wägeeinrichtungen. Feingerätetechnik 27 (1978), H. 5, S. 199 und 200

4.4. Programmierbare Maschinensteuerung¹⁾

Bekannterweise haben Mikroprozessoren und Mikrorechner in nahezu allen Gebieten der Steuerungstechnik ihre Anwendungsfälle gefunden. Im Maschinenbau vollzog sich dieser Applikationsprozeß langsamer als in anderen Bereichen; vor allem wohl deshalb, weil sich erst Spezialisten herausbilden mußten, die sich sowohl in den Belangen des Maschinenbaus und dessen Forderungen als auch in der Technik und im Umgang mit den Mikroprozeßrechnern gleichermaßen gut auskennen.

Im Werkzeugmaschinenbau begann die Prüfung der Eignung der Mikrorechner als Steuerungsbestandteil naturgemäß in der Technik der numerischen Steuerungen (numerical control, NC), denn sie verarbeiten numerisch (digital) verschlüsselte Informationen, die den technologischen und maschinentechnischen Fertigungsablauf beschreiben. Dazu werden ein Rechen- und Steuerwerk sowie mehr oder weniger umfangreiche Datenspeicher benötigt. Es lag daher auf der Hand, bisher festverdrahtete Funktionsschaltkreise durch die universelleren Mikrorechnerbausteine zu ersetzen.

Besonders an die logischen Netzwerke, Ablauf- oder Folgesteuerungen werden heute hohe Anforderungen gestellt, wie sie im Bild 4.4.1. teilweise dargestellt sind. Mitunter kann allerdings die Forderung nach geringstem Platzbedarf den aufgeführten Kriterien übergeordnet sein. Mit den bisher-

¹⁾ Der Abschnitt 4.4. unserer Reihe ist eine Mitteilung aus der Sektion Fertigungstechnik und Werkzeugmaschinenwesen der TU Dresden

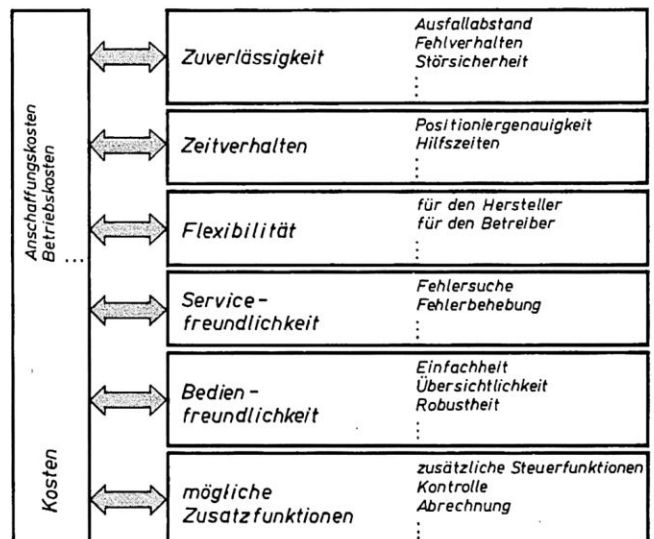


Bild 4.4.1.: Einige Kriterien für Maschinensteuerungen

gen Realisierungsformen (Relais, Diodenmatrizen, Schrittschaltwerke, Gatterbausteine u. a.) ist es nicht möglich, allen Forderungen in gleicher Weise nachzukommen. Daher werden seit einiger Zeit programmierbare Steuerungen (engl. Programmable Controller, PC), gelegentlich auch frei-programmierbare Steuerungen oder programmierbare Logiksteuerungen (PLC) genannt, entwickelt und eingesetzt. Sie sind für den Einsatz als nichtnumerische Steuerungen von Maschinen und Anlagen konzipiert, bei denen der Hauptanteil der zu verarbeitenden Informationen binär ist. Trotzdem setzt sich heute der 8-Bit-Mikroprozessor als Kernstück programmierbarer Steuerungen durch. Die Hauptvorteile dieser programmierbaren Steuerungen bestehen in der Universalität bzw. Flexibilität, in der wesentlich verkürzten und vereinfachten Inbetriebnahme der Maschine bzw. Anlage sowie in der leichten Änderbarkeit des Steuerungsinhaltes. Aus der Literatur sind Fälle bekannt, bei denen der Einsatz einer programmierbaren Steuerung auf Mikrorechnerbasis allein durch die verkürzte Inbetriebnahme gerechtfertigt war.

Im folgenden soll eine Möglichkeit des Aufbaus einer programmierbaren Steuerung aus einem Mikrorechner genauer vorgestellt werden. Gesichtspunkte für ihren Aufbau sind einerseits die obengenannten Forderungen an eine moderne nichtnumerische, hauptsächlich binäre Steuerung und andererseits die besondere Berücksichtigung der Bedingungen im Werkzeugmaschinenbau. Diese Steuerungsform erlaubt es, aus einem handelsüblichen Mikrorechner mit geringstem Aufwand an zusätzliche Hardware und mit universeller Software eine programmierbare Maschinensteuerung zu erhalten. Hauptvorteil der vorzustellenden Software ist neben der Universalität die Anwenderfreundlichkeit, d. h. insbesondere der Wegfall jeglicher Spezialkenntnisse. Konkreter Erprobungsfall war ein Industrieroboter, der in seiner ersten Ausbaustufe zwar relativ einfach war, aber den praktischen Forderungen vollauf entsprach.

4.4.1. Aufbau und Funktion

Beim Aufbau einer Prozeßsteuerung mit Mikrorechner steht man stets vor der Problematik der Aufteilung der informationsverarbeitenden Funktionen in Hardware und Software. Hardwaremäßige Realisierungen arbeiten in der Regel schneller, sind aber weniger flexibel als entsprechende Softwarelösungen. Zwischen beiden bestehen sehr enge Wechselbeziehungen. Im Maschinen- und Anlagenbau tritt dieser Problemkreis z. B. bei der Entprellung von Kontakten, die Eingangssignale für die Steuerung liefern, auf. Einerseits kann man diese Prellerscheinung über Flip-Flops abfangen, also hardwareseitig; andererseits gelingt dies ebenso durch Mehrfachabfrage des Signals mit internem Vergleich, also softwareseitig. Wenn der zu steuernde Prozeß und die Organisation des Programmsystems das ermöglichen, kann die Entprellung auch überflüssig werden. Aus Gründen der Übersichtlichkeit sollen hier trotz des engen Zusammenspiels von Hard- und Software beide Teile getrennt dargestellt werden.

4.4.1.1. Hardware

Für die Hardware-Struktur gelten eine Vielzahl Aspekte und Bedingungen, die aus praktischen Forderungen abgeleitet sind:

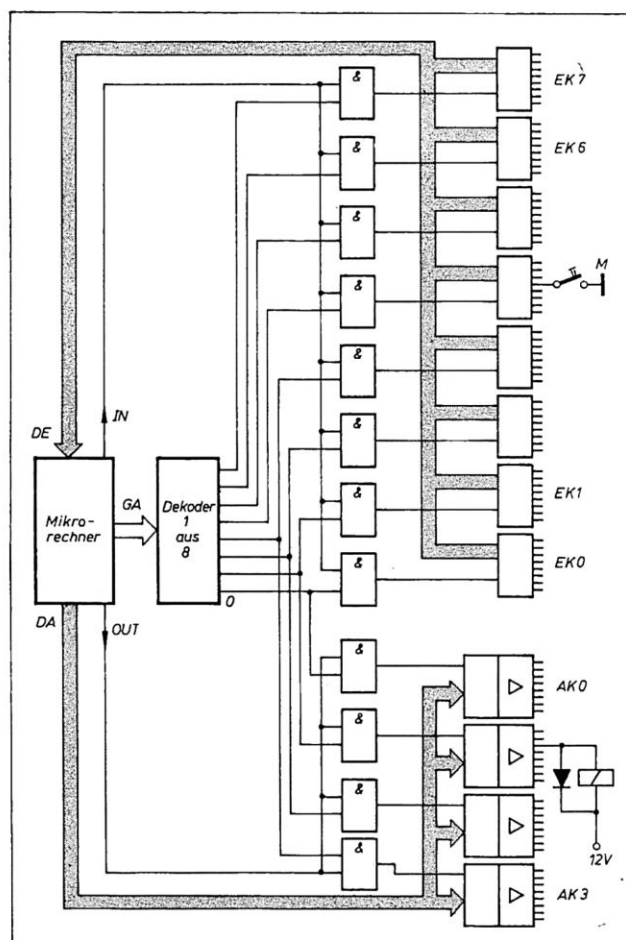
- Eingriffe in den Mikrorechner erfolgen nicht.
- Anschlüsse am Mikrorechner sind
 - 8-Bit-Dateneingabebus
 - 8-Bit-Datenausgabebus
 - ein Teil des Adreßbusses
 - zwei Steuerleitungen (IN, OUT).
- Da Maschinen- und Anlagensteuerungen stets mehr als acht binäre Ein- bzw. Ausgänge verlangen, müssen Datenflußverengung (eingangsseitig) und Datenflußverbreiterung (ausgangsseitig) erfolgen.
- Die Benutzung der Interrupt-Möglichkeiten des Mikrorechners wurde wegen des höheren Hardware-Aufwandes vermieden.
- Die Eingänge in die Hardware vom Prozeß aus sollen kurzschluß- und querschlußunempfindlich sein.
- Um die Leistungsverstärkung für die Stellgliedansteuerung in der Maschine vornehmen zu können, ist am Hardware-Ausgang ein Mindestpegel von 12 V erforderlich, der mindestens bis 100 mA belastbar sein muß.

- Das Verhältnis der Zahl der binären Eingänge vom Prozeß zur Zahl der binären Ausgänge an den Prozeß beträgt im Maschinenbau etwa 2:1.
- Die Hardware-Struktur soll universell anwendbar, einfach und übersichtlich sowie modular auf- und ausbaufähig sein.

Bild 4.4.2. zeigt die gewählte und realisierte Hardware-Struktur als Übersicht. Die dargestellte Ausbaustufe von 64 binären Eingängen und 32 binären Ausgängen ist ein Anwendungsfall und keine obere Grenze (s. u.). Kernstück der Steuerung ist der als Black Box aufgefaßte Mikrorechner (MR) mit seinem 8-Bit-Dateneingabebus (DE), seinem 8-Bit-Datenausgabebus (DA) und den beiden Steuersignalen für Eingabe (IN) und Ausgabe (OUT). Des weiteren wurde der Adreßbus verlängert, um die notwendige Datenflußverengung bzw. -verbreiterung vornehmen zu können. Die Frage nach der Sinnfälligkeit der externen Benutzung des Adreßbusses ist oft gestellt worden; zumeist mit dem Hinweis, die äußere Datenbreite über Schieberegister einfach erreichen zu können. Hier wird jedoch die Adreßbusvariante gewählt, weil die modulare Aufrüstbarkeit bei gleicher Software für die Ein- und Ausgabeorganisation in der Schieberegistervariante schwieriger zu verwirklichen ist. Zudem kann man Steuerungs- und Prozeßelemente, die außerhalb des Mikrorechners liegen, als für den Mikrorechner periphere Geräte ansehen. Und für die Kommunikation zwischen Mikrorechner und Peripherie ist die Adreßsteuerung üblich.

Im Bild 4.4.3. ist der Signallauf an je einem Ein- und Ausgang konkretisiert. Vom Adreßbus werden für die gezeichnete Anzahl von 64 Eingängen und 32 Ausgängen lediglich 3 Bits zur Unterscheidung der acht Eingangskanäle (EK 0 bis EK 7) bzw. der vier Ausgangskanäle (AK 0 bis AK 3) benötigt. Die restlichen Signale dienen der Abgrenzung gegenüber anderen peripheren Geräten wie Lochbandleser u. ä. Die genannten drei Bits werden im Dekoder zum Signalaufbau 1 aus 8 umgesetzt. Es ist somit unschwer zu erkennen,

Bild 4.4.2.: Hardware-Struktur einer programmierbaren Steuerung



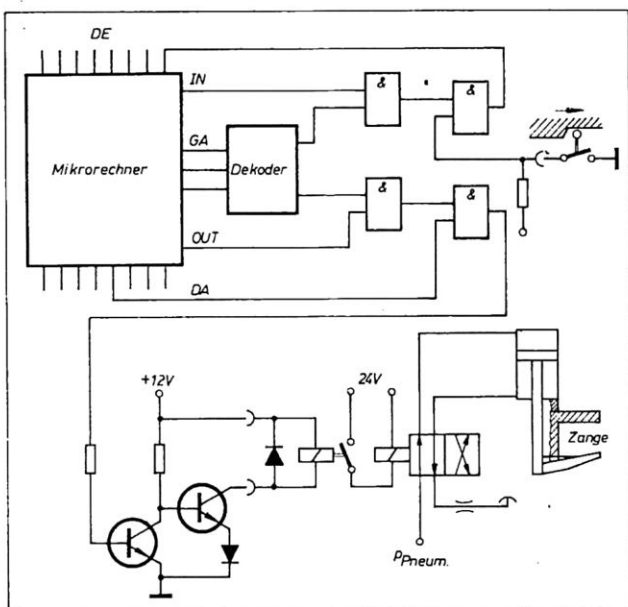


Bild 4.4.3.: Signalfluß je eines Ein- und Ausgangs durch die zusätzliche Hardware

daß die obere Grenze der Ausbaufähigkeit bezüglich der Ein- und Ausgänge lediglich von der benutzten Breite des Adreßbusses abhängt. Mit beispielsweise sechs Bits des Adreßbusses lassen sich also $2^6 = 128$ Ein- bzw. Ausgangskanäle ansteuern, die ihrerseits jeweils mit acht Bit Datenbreite (gleich Datenbusbreite) arbeiten und somit im Beispiel 1024 Eingänge und ebenso viele Ausgänge zulassen. Sowohl die Eingabe- als auch die Ausgabekanäle werden meist als Puffer ausgeführt. Die Eingabeseite sollte außerdem vom zu steuernden Prozeß über Optokoppler o. ä. galvanisch getrennt sein, um das Übergreifen von Störungen zu vermeiden.

Zusammengefaßt kann man die Funktionsweise der ausgeführten Hardware wie folgt beschreiben: Der Mikrorechner gibt an den Dekoder die verschlüsselte Adresse des Kanals, über den die Ein- oder Ausgabe erfolgen soll. Der Dekoder sendet die entsprechenden 1-aus-n-Signale aus, die ihrerseits mit dem Signal IN oder OUT, der die Unterscheidung zwischen Ein- und Ausgabeoperation enthält, verknüpft werden. Diese so gebildeten Signale öffnen den aufgerufenen Eingangs- oder Ausgangskanal, so daß dieser mit dem entsprechenden Datenbus verbunden ist und die anderen Kanäle von den Datenbussen getrennt sind.

4.4.1.2. Software

Jedes Programmpaket, das multivalent nutzbar sein soll, besteht prinzipiell aus zwei Teilen:

- Grundprogramm (-system), das bestimmte Algorithmen nachbildet
- Datensatz, der die speziellen Bedingungen der momentanen Anwendung beschreibt.

Durch diese Zweiteilung und der damit erreichbaren Universalität der Software bleibt dem Anwender die Kenntnis des inneren Aufbaus weitgehend erspart. Er hat lediglich bestimmte Formalitäten bei der Generierung des Datensatzes für seinen Anwendungsfall zu beachten und kann sich darauf verlassen, daß diese Daten ordnungsgemäß verarbeitet werden: Dem prozeßspezifizierenden Charakter der Daten Rechnung tragend, ist die Arbeitsweise des vorliegenden Grundprogramms zur Realisierung einer binären Steuerung interpretativ. Das heißt, unter der Voraussetzung, daß binäre Eingangssignale nach bestimmten logischen Vorschriften zu binären Ausgangssignalen verarbeitet werden, enthält der Datensatz die logischen Vorschriften. Dieser Datensatz, der allein kein selbständig arbeitendes Programm darstellt, wird vom Grundprogramm interpretiert, damit die Gesamtsteuerung so arbeitet, wie es der Datensatz (die Gesamtheit der logischen Vorschriften) vorsieht.

Das vorliegende Mikrorechnerprogrammssystem hat den Namen LOST (abgeleitet von „Logische Steuerung“) erhalten. Der ihm zugrundeliegende Algorithmus wird im Bild 4.4.4.

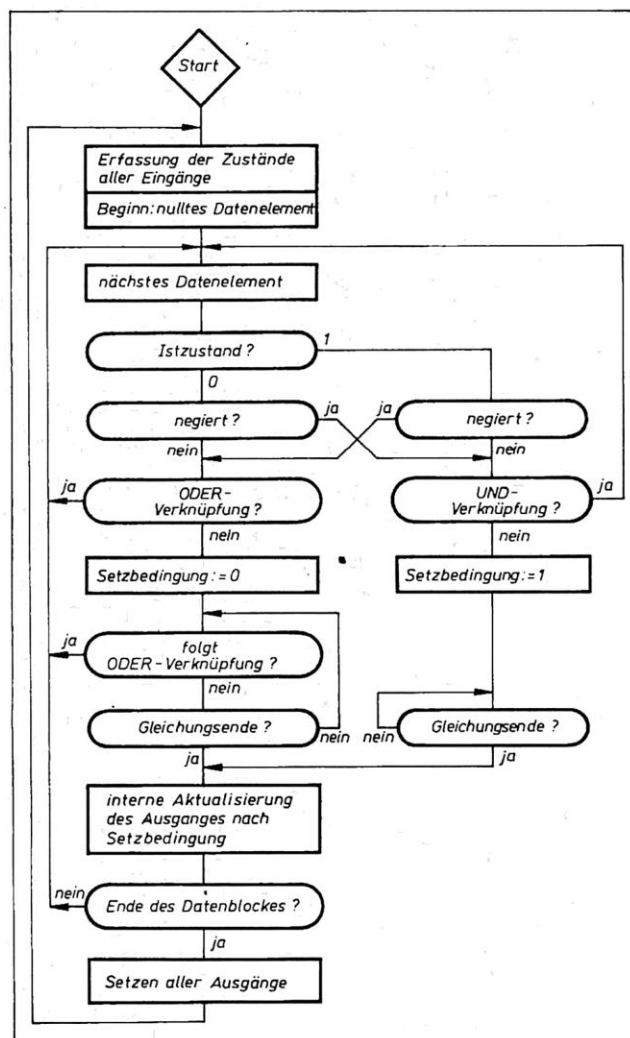
stark vereinfacht (ohne die Klammerbearbeitung) wiedergegeben. Nach dem Start fragt der Mikrorechner alle anliegenden Eingänge ab und speichert deren Istzustände. Anschließend geht er zur Abarbeitung des Datensatzes über und bildet die Entscheidungen der logischen Gleichungen nach. Die Ergebnisse werden abgespeichert, d. h., der Zustand der Stellglieder wird (vorerst nur) intern aktualisiert. Am Ende des Datensatzes (Datenblockes) wird das gesamte intern aktualisierte Ausgangsbelegungsbild vom Rechner an die Maschine (an den Prozeß) übergeben; die Stellglieder werden bestätigt. Dann beginnt der Zyklus von neuem. Wie zu erkennen ist, weist dieser Algorithmus drei Besonderheiten auf:

1. einmalige Istzustandserfassung je Zyklus
2. einmalige Ausgabe der aktualisierten Ausgangsbelegung je Zyklus (wenn der Datensatz aus nur einem Block besteht)
3. Die Zykluszeit hängt vom Umfang des Datenblockes und von der aktuellen Istzustandsbelegung ab.

Durch die ersten zwei Eigenschaften wurde erreicht, daß einerseits auf die recht aufwendige Entprellung mechanischer Kontakte verzichtet werden konnte und daß sich andererseits keine Fehlerwertläufe in der Steuerung ausbilden können. Natürlich verlangt beides relativ kurze Zykluszeiten. Zur Anwendung des Programms LOST in einem realen, noch nicht getesteten Prozeß werden folgende Schritte gegangen:

1. Laden des Grundprogramms LOST in den Mikrorechner, sofern es nicht bereits in einem ROM liegt
2. Formulierung bzw. Korrektur logischer Vorschriften für den zu steuernden Prozeß
3. Kodierung der Logik nach Tafel 4.4.1.
4. Laden des kodierten Datensatzes bzw. Korrektur des Datensatzes über Lochband oder Tastatur

Bild 4.4.4.: Vereinfachter Algorithmus zum Programm LOST



5. Simulations- oder Testlauf des Prozesses mit Korrekturen durch die Schritte 2 bis 4
6. Start der Steuerung (wenn Prozeß in Ordnung).

Die Dokumentation der Steuerung umfaßt für eine komplette Prozeßsteuerung:

- Angaben zum Mikrorechner und zur Zusatz-Hardware
- Lochband oder ROM-Speicher mit LOST
- Kodiertabelle
- Liste zur Bezeichnung und Bezifferung der Anschlüsse bzw. Signale (gleichzeitig als Anschlußplan verwendbar)
- Liste der logischen Vorschriften mit Kodierung, d. h. der Datensatz, evtl. auf Lochband.

Mit dem Wegfall von Bauschaltplänen und Verdrahtungsplänen (einschließlich deren Änderungsdienst) u. ä. wird die Dokumentation erheblich übersichtlicher und einfacher.

Tafel 4.4.1.: Kodierung der Logik bei LOST (unvollständig)

Teil A			Teil B	
Verknüpfung von Eingängen			Bit-Nummer	Kanal- Kode Nummer
			.0 .1 .2 .3 .4 .5 .6 .7	
UND	(E)	, -	00 01 02 03 04 05 06 07	0 00
ODER	(E)	, +	10 11 12 13 14 15 16 17	1 01
NICHT-UND	(E)	, -	20 21 22 23 24 25 26 27	2 02
NICHT-ODER	(E)	, +	30 31 32 33 34 35 36 37	3 03
Verknüpfungen von Ausgängen und Merkern				4 04
				5 05
				6 06
				7 07
				8 08
				9 09
UND	S(M)	, -	80 81 82 83 84 85 86 87	10 0a
ODER	S(M)	, +	90 91 92 93 94 95 96 97	11 0b
NICHT-UND	S(M)	, -	a0 a1 a2 a3 a4 a5 a6 a7	12 0c
NICHT-ODER	S(M)	, +	b0 b1 b2 b3 b4 b5 b6 b7	13 0d
Gleichungsende				14 0e
				15 0f
				16 10
GLEICH	= S(M)		88 89 8a 8b 8c 8d 8e 8f	17 11
				18 12
				19 13
Beispiel:				20 14
E0,3 UND E21,4 ODER M26,3 GLEICH S15,6 wird zu				21 15
			Teil A	Teil B
0,3			23	22 16
21,4 +			14	23 17
M26,3			83	24 18
= S 15,6			8e	25 19
				usw.

4.4.2. Industrieroboter

Zum Sammeln praktischer Erfahrungen beim Mikrorechner an Werkzeugmaschinen (zu denen hier auch deren Handhabungsgeräte und Industrieroboter zählen sollen) wurde das Programm LOST mit den Daten eines relativ einfachen Industrieroboters gekoppelt. Dieser Roboter realisiert vier Bewegungen:

- Linearbewegung
- Drehbewegung
- Anschlag zum Erreichen verschiedener Drehwinkelpositionen
- Öffnen und Schließen der Zange.

Aufgabe des Industrieroboters ist die geordnete Verteilung von zwei verschiedenen Werkstücken, die ungeordnet liegen, auf vier Stationen mit je zwei möglichen Ablagepunkten (Bild 4.4.5.). Die Steuerung muß außer der Generierung des jeweiligen Ablaufes der Bewegungen noch die Entscheidungen fällen, die sich aus der Werkstückunterscheidung (hier geht es um das Vorhandensein einer Rille im Werkstück), Stationsanforderungen und den beiden Ablagemöglichkeiten je Station ergeben. Der steuerungstechnische Teil besteht aus Mikrorechner, Zusatz-Hardware (drei Leiterplatten), Geradeausverstärkung und Leistungsverstärkung, Tastatur als Ersatz für Stationsanforderungen und dem pneumatischen Teil.

Als Mikrorechner wurde eine von der Sektion Informationstechnik der TU Dresden eigentlich für studentische Lehrzwecke aufgebaute 8080-Version benutzt. Er war mit der Aufgabe der Industrierobotersteuerung schätzungsweise nur zu einem Viertel ausgelastet. Die Leiterplatten der Zusatz-Hardware waren für 64 binäre Eingänge und 32 binäre Ausgänge einschließlich deren Vorverstärkung (12 V, 0,1 A)

ausgelegt, obwohl bei weitem nicht alle gebraucht wurden. Wie im Bild 4.4.3. dargestellt ist, wurden Leistungsverstär-

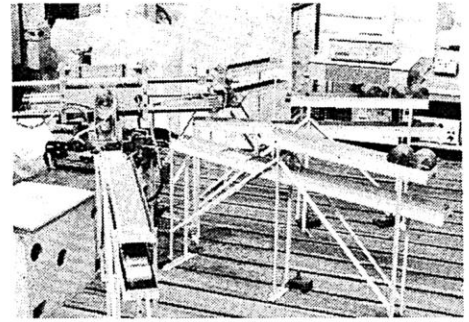


Bild 4.4.5.: Industrieroboter mit Mikrorechnersteuerung

Tafel 4.4.2.: Ausschnitt aus dem Datensatz „Industrieroboter“

Endschalter RüBr	ODER	3,5 +	15 03	3.) Rückwärts- bremse M 7
Selbsthaltung RüBr	UND	S1,6 ·	86 01	
Merk Br.	ODER	M9,4 +	b4 09	
Linearvorlauf = RüBr		S1,4 = S1,6	84 01 8e 01	
Endschalter 0°	UND	3,0 ·	00 03	4.) Drehbremse
Drehung Links	ODER	S1,1 +	b1 01	
Anschlag vorn = Drehbremse		S1,2 = S1,3	82 01 8b 01	
Drehung rechts	UND	S1,1 ·	81 01	
Merk Anschlag = Anschlag vor		M9,3 = S1,2	83 09 8a 01	5.) Drehanschlag vor

kung und galvanische Trennung der Ausgänge der Einfachheit halber über Relais realisiert. Die sich bei Einfügung des Industrieroboters in einen realen Prozeß ergebenden Anforderungssignale der Stationen, auf die die Werkstücke zu verteilen waren, wurden mittels einer einfachen Tastatur simuliert.

Die Inbetriebnahme des Industrieroboters, dessen Steuerungseigenschaften vorher nicht vollends geklärt waren, dauerte knapp eine Woche. Mit einer festverdrahteten Steuerung hätte sie etwa vier Wochen gedauert. Die Vielzahl von Korrekturen während der Inbetriebnahmephase, die durch das Verhalten des Roboters bedingt waren, bereiteten von der Software und vom Mikrorechner her keine ernsthaften Sorgen, da jede Änderung per Papier und Tastendruck und nicht durch Umlöten o. ä. erfolgte und die Dokumentation trotzdem einfach und übersichtlich blieb. Störungen des Mikrorechners durch die Ein- und Ausschaltspitzen benachbarter Geräte wurden vermieden, indem er auf eine separate Phase des Netzes gelegt wurde.

Der Industrieroboter arbeitete mehrere Wochen lang nahezu ununterbrochen im rauen Werkstattbetrieb. Die gesamte Elektronik (Mikrorechner und Zusatz-Hardware) waren niemals Verursacher von Unterbrechungen, obwohl die Zusatz-Hardware nur als Labormuster aufgebaut und außerdem eingangsseitig auf galvanische Trennung, auf Entprellung der mechanischen Kontakte sowie auf Abschirmungen verzichtet worden war. Es erwies sich, daß einige Steckverbinder für den rauen Laborbetrieb zu schwach und zu störanfällig sind.

Wird fortgesetzt

Allwellempfänger
Tesla „Lambda“,
leicht defekt, zu verkaufen
Fil. 151 807 DEWAG, 1054 Berlin

Ing. für Elektronik, Raum Magdeburg, übernimmt ggf. hauptberuflich Service, Reparatur, Bau und Sonderanfertigung (Musterbau, Realisierung Ihrer Ratio-Maßn.) von elektron. Geräten und Baugruppen. Arbeits-u. Lageraum sowie Werkstattgrundausstattung sind vorhanden. Angebote an
TK 5796 DEWAG, 1054 Berlin

Dipl.-Ing. THOMAS KRÜGER und
Dipl.-Ing. HANS-HELGE GEITNER

Als Grundprogramm für diese Aufgabe reichte eine abgerüstete Variante des Programms LOST aus. Dieses Programm belegte lediglich etwa 400 Bytes Speicherplatz. Für den Datensatz dieses Industrieroboters wurden etwa 200 Bytes des Speichers verwendet. Zur Erläuterung der praktischen Anwendbarkeit der Kopplung von Mikrorechner und Programmsystem LOST zeigt Tafel 4.4.2. einen Ausschnitt aus dem Datensatz „Industrieroboter“. Es sind drei Gleichungen und deren Umsetzung dargestellt. Die verbale Formulierung für die Gleichung (3) dazu lautet:

Das Ventil für das Abbremsen der Rückwärtsbewegung der Lineareinheit wird eingeschaltet, wenn der Nocken an der Lineareinheit den entsprechenden Endschalter (der auf Kanal 3 den 5. Kontakt belegt) betätigt. Weiterhin soll dieses Ventil auch immer dann die Bremsstellung einnehmen, wenn sich die Lineareinheit vorwärts bewegt (Pneumatikbedingung). Und endlich soll dieses Ventil M_7 in dieser Stellung gehalten werden, solange das Ergebnis der Merkgleichung $M_{9,4}$, in der praktisch eine Ausschaltbedingung formuliert ist, gleich Null ist.

Im Bild 4.4.6. wird als konstruktives Detail die Aufnahme eines Werkstückes gezeigt. Gut zu erkennen ist die Zange des Industrieroboters mit dem Pneumatikzylinder. In Greifhöhe der Zange sind seitlich zwei Mikrotaster direkt nebeneinander angebracht, die die Meldungen über Vorhandensein und Art des Werkstückes an den Mikrorechner geben.

4.4.3. Zusammenfassung

Nach der Abgrenzung gegenüber numerischen Steuerungen wird eine Anwendungsmöglichkeit zur Steuerung von Maschinen in Form einer „Programmierbaren Steuerung“ vorgestellt. Für die speziellen Fälle von logischen und Ablaufsteuerungen an Maschinen und Anlagen wird das Programm LOST verwendet und beides an einem relativ einfachen Industrieroboter demonstriert. Der gegenwärtig noch notwendige Hardware-Zusatz bei Einsatz eines kaufbaren Mikrorechners ist gering und aus DDR-Bausteinen universell und modular aufrüstbar gestaltet. Ebenso wie die Hardware ist auch das vorgestellte Programm LOST bewußt flexibel und anwendungsfreundlich gehalten.

4.5. Antriebstechnik²⁾

Mit Hilfe der Mikroelektronik können die bereits in den sechziger Jahren mit Prozeßrechnern begonnenen Untersuchungen zum Einsatz direkter digitaler Regelungen (Direct Digital Control, DDC) in der Antriebstechnik unter günstigeren technischen und ökonomischen Bedingungen weitergeführt werden. Die anhaltende stetige Entwicklung der Mikroelektronik [4.5.1.] ermöglicht dabei Lösungen mit sehr hoher Zuverlässigkeit bei verringertem Platzbedarf und sinkenden Kosten. Bei der Auswahl eines Mikrorechners (MR) zur Realisierung von Antriebsregelungen sollte vom Aufgabenumfang, nicht von Forderungen nach möglichst guter Nachbildung kontinuierlicher Verhältnisse ausgegangen werden, da die Regelgüte nicht stetig mit sinkender Zeit des Abtastalgorithmus (t_{ALG}) zunehmen muß. Sie ist von Typ und Näherungsformel des gewählten Regelalgorithmus und von der Abtastzeit (T) abhängig [4.5.2.].

Bei der Einsatzvorbereitung eines Mikrorechners für direkte digitale Regelungen sind u. a. folgende Probleme zu lösen:

- Formulierung der Aufgabenstellung (Regelalgorithmus) in einer Form, die sich zur Programmierung des Rechners eignet
- Optimierung des digitalen Reglers
- Wahl der Informationsdarstellung (Kode, Wortbreite)

²⁾ Der Abschnitt 4.5. unserer Reihe ist eine Mitteilung aus der Sektion Elektrotechnik der TU Dresden

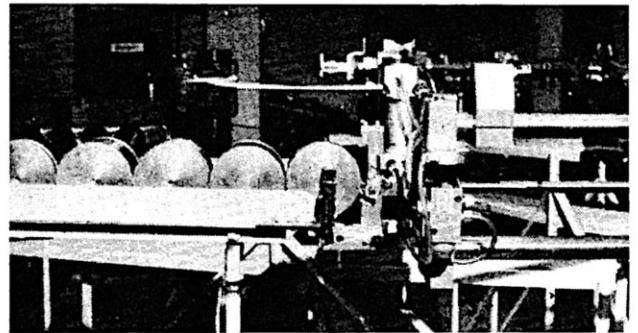


Bild 4.4.6.: Aufnahme eines Werkstückes durch den Industrieroboter

- Verbindung des Computers mit dem zu beeinflussenden Prozeß
- Untersuchung verschiedener Programmstrukturen und der jeweiligen Auslastung des Rechners.

Liegt die zu lösende Aufgabe in Form algebraischer Gleichungen vor, so besteht das Ziel der weiteren Arbeiten in einer Zerlegung des arithmetischen Problems in die Grundrechenarten. Das ist eine Grundvoraussetzung für die Erstellung eines MR-Programmes, da die Rechenzeiten nichtlinear mit der Kompliziertheit der Rechenoperationen ansteigen. Allgemein kann der zu programmierende Algorithmus zur Regelung einer Schleife nach [4.5.3.] folgendermaßen dargestellt werden, wobei die Faktoren c_p nur bei Führungsgrößenaufschaltung auftreten:

$$u_n = a_0 x_{w,n} + a_1 x_{w,n-1} + \dots + a_l x_{w,n-l} + b_1 u_{n-1} + \dots + b_k u_{n-k} + c_0 x_{s,n} + \dots + c_p x_{s,n-p} \quad (4.5.1.)$$

mit

$x_{w,n} = x_{s,n} - x_{i,n}$ = aktuelle Regelabweichung

u_n = zum Zeitpunkt t_{n+1} am MR-Ausgang zur Verfügung stehender Wert

$x_{s,n}$ = Sollwert zum Zeitpunkt t_n

$x_{i,n}$ = Istwert zum Zeitpunkt t_n

a, b, c = Algorithmusparameter, $n = 0; 1; 2 \dots; l \geq k$

Zur Berechnung eines Algorithmus l -ter Ordnung ohne Aufschaltung der Führungsgröße sind folglich maximal $2l + 1$ Multiplikationen und $2l + 1$ Additionen bzw. Subtraktionen notwendig. Dazu müssen die Zeiten für die Ein- und Ausgabeprogrammteile, Umkodierungen, Unterprogrammaufrufe, Speicherkommunikation u. ä. addiert werden, um eine Abschätzung der Algorithmuslaufzeit zu gewährleisten. Bei Regelung mehrerer Größen bestehen folgende Möglichkeiten:

- a) Einmalige Programmierung von Gl. (4.5.1.) und mehrmalige Abarbeitung mit verschiedenen Parametern.
- b) Werden sehr unterschiedliche Regler gefordert, kann die Programmierung jedes einzelnen Regelalgorithmus bei jeweiliger Reduzierung von Gl. (4.5.1.) günstig sein.
- c) Theoretisch besteht außerdem die Möglichkeit der Programmierung nach Gl. (4.5.2.):

$$u_n = u_{s,n} + \sum_{i=1}^l u_{i,n} \quad (4.5.2.)$$

l = Anzahl der Istwerte.

Die einzelnen Anteile $u_{s,n}$ und $u_{i,n}$ sind wiederum nach Gl. (4.5.1.) zu berechnen, wobei stets $c_p = 0$ gilt und statt $x_{w,n}$ jeweils $x_{s,n}$ bzw. $x_{i,n}$ zu setzen ist. Damit würde allen Eingangsgrößen (den verschiedenen Istwerten und dem Sollwert) ein spezieller Algorithmus zugeordnet. Die Program-

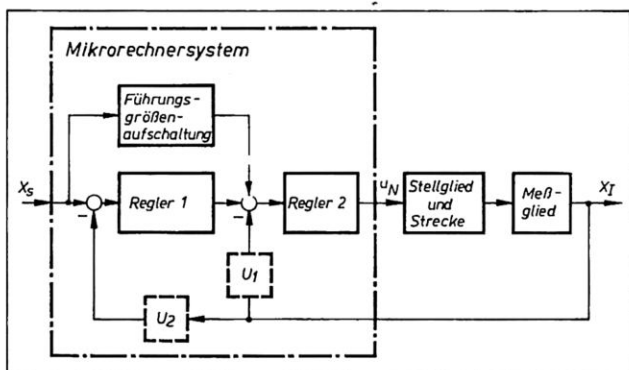


Bild 4.5.1.: Regelstruktur mit einem Meßglied

mierung kann wiederum nach Fall a) oder b) unterschieden werden, wobei der Aufwand an Organisationsprogramm für den Fall a) den jeweiligen Ausschlag gibt. Werden mehrere Größen unter Voraussetzung mehrerer Meßglieder geregelt, so ist eine Programmierung nach c) von vornherein ungünstiger, da gegenüber den Fällen a) und b) stets ein Algorithmus mehr abzuarbeiten ist. Besteht jedoch die Forderung nach Verwendung nur eines Meßgliedes, so ist der Einsatz einer Struktur nach c) zu prüfen. Bild 4.5.1. veranschaulicht zwei mögliche Strukturen bei Einsparung eines zweiten Meßgliedes. Für $u_1 = 1$ und $u_2 = 1/p$ erhält man eine Lageregelung mit unterlagerter Drehzahlregelung, für $u_1 = p$ und $u_2 = 1$ eine Drehzahlregelung mit unterlagerter Beschleunigungsregelung – jeweils bei Messung der Drehzahl. Die Stromregelung wird dabei als analog und unterlagert vorausgesetzt. Die Entscheidung, nach welcher Struktur programmiert wird, ist von der möglichen Einsparung höherer Rechenoperationen sowie der gewünschten Flexibilität des Algorithmus abhängig.

Alle Grundrechenoperationen sollten als Unterprogramme geschrieben werden, da entweder keine entsprechenden MR-Befehle existieren oder diese Befehle nur den Kern der Operation realisieren, auf eine bestimmte Informationsdarstellung festgelegt sind und keine Reaktion auf eine Überschreitung des gewählten Datenbereiches auslösen. Die einfachste Methode, einen Regelalgorithmus nach Gl. (4.5.1.) oder Gl. (4.5.2.) zu erhalten, besteht im direkten Übergang von der Integrodifferential- zur Differenzengleichung.

Integrationen können mittels Rechteck- bzw. Trapezformel und Verwendung von Rekursionsformeln, Differentiationen mit Hilfe eines Ansatzes nach [4.5.3.] nachgebildet werden. Durch Gl. (4.5.3.) ist die Berechnung einer beliebigen μ -ten Ableitung mit einer Genauigkeit 1. Ordnung möglich.

$$\frac{d^\mu x_w}{dt^\mu} \bigg|_{nT} \approx \frac{1}{T^\mu} \sum_{i=1}^{\mu} \binom{\mu}{i} (-1)^i x_{w,n-i} \quad (4.5.3.)$$

T = Abtastzeit des Systems

Mit Hilfe einer Differentiationsnachbildung ν -ter Ordnung kann eine Zeitfunktion, die sich aus Polynomen bis ν -ter Ordnung zusammensetzt, exakt differenziert werden. Gl. (4.5.4.) ermöglicht eine Berechnung der 1. Ableitung mit beliebiger ν -ter Genauigkeit (Ordnung).

$$\frac{dx_w}{dt} \bigg|_{nT} \approx \frac{1}{T} \sum_{i=1}^{\nu} \binom{\nu}{i-1} \frac{(-1)^{i+1}}{i} (w_{w,n} - x_{w,n-i}) \quad (4.5.4.)$$

Für den praktischen Einsatz kann die 3. Ordnung einer Differentiationsnachbildung als Grenze angesehen werden, da der Algorithmus anderenfalls zu viele Multiplikationen enthält. Entsprechende Tabellen mit Gewichtungsfaktoren wurden berechnet.

4.5.1. Informationsdarstellung

Die Wahl der Darstellung rechnerinterner Informationen im Dualcode stellt eine Digitalrechnern angepaßte Lösung dar. Negative Zahlen sind im Zweierkomplement darzustellen. Dieser Code eignet sich sowohl für die interne Informationsdarstellung als auch für die Eingabe von Istwerten als auch die Ausgabe von Ergebnissen. Zur Kommunikation mit dem Anwender sollte der BCD-Code Anwendung finden (z. B. zur Einstellung von Sollwerten und Parametern an Ziffernvor-

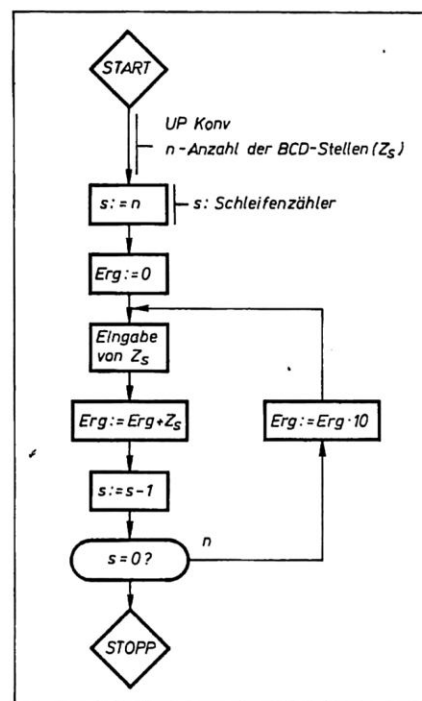


Bild 4.5.2.: PAP zum Prinzip der BCD-Dualcodeumsetzung

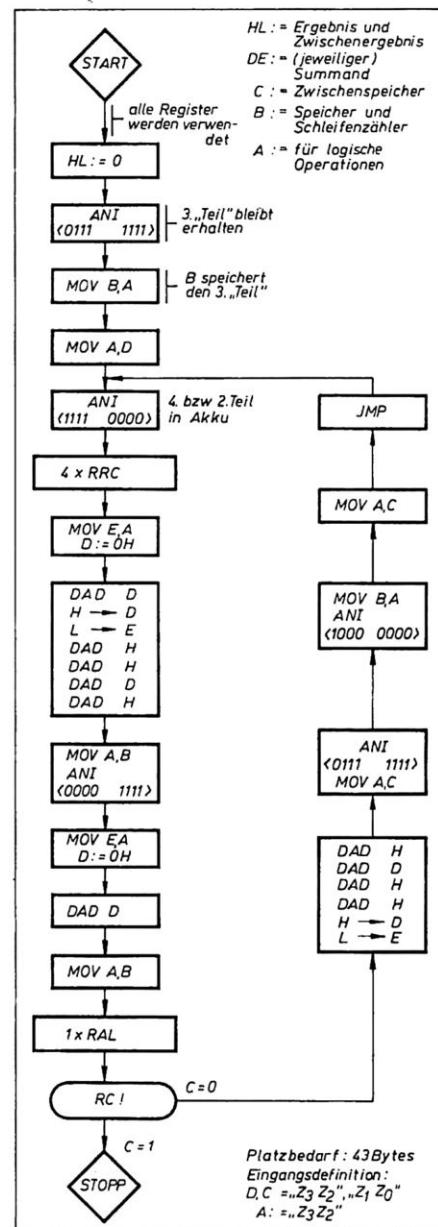


Bild 4.5.3.: Mnemocode für ein Programm zur BCD-Dualcodeumsetzung

Platzbedarf: 43 Bytes
Eingangsdefinition:
D,C = „Z3 Z2“, „Z1 Z0“
A: „Z3 Z2“

Tafel 4.5.1.: Impulsanzahlen (in Mill.) für Lageregelungen bei Verwendung eines inkrementalen rotatorischen Gebers mit 10 000 Impulsen/Umdrehung

maximaler Arbeitsweg		Spindelsteigung			
		20 mm	10 mm	5 mm	
2 m		1	2	4	†
5 m		2,5	5	10	
10 m		5	10	20	

† Mit Dreifach-Wortverarbeitung lösbar

wahlschaltern). Die Bilder 4.5.2. und 4.5.3. zeigen Programmablaufplan und Mnemokode (8080) für eine mögliche Umsetzung von vier BCD-Stellen in den Dualcode.

Die Datenwortbreite der internen Informationsdarstellung ist vom Anwendungsfall abhängig. Ist die Messung des Istwertes schwierig, wird eine Auflösung von etwa 1 ‰ ausreichend sein. Da die Wortbreite der meisten MR 8 bit beträgt, kann in diesen Fällen mit Einfachwortverarbeitung gearbeitet werden (6 bit Betrag ergeben 1,56 ‰ Auflösung). Je nach verwendetem Rechner wird bei Darstellung positiver und negativer Zahlen eine Schutzstelle zwischen Vorzeichen und Betrag notwendig (z. B. 8080) oder nicht (z. B. K 1520).

Doppelwortverarbeitung (16 bit) ist z. B. für eine hochgenaue Drehzahlregelung ausreichend, während zur Erfassung höchster Ansprüche bei Lageregelungen eine Vierfach-Wortverarbeitung notwendig wird. Aus der Tafel 4.5.1. ist die jeweilige Impulsanzahl in Millionen bei Verwendung eines inkrementalen Rotationsgebers mit 10 000 Impulsen/Umdrehung ablesbar. Im einfachsten Fall kann dabei der Lage-Istwert durch Integration des Drehzahl-Istwertes gebildet werden, wobei gleichzeitig ein zweites Meßglied eingespart wird. Dieses Verfahren eignet sich nur für Lageregelungen mit geringen Ansprüchen, da Fehlereinflüsse durch die Übertragungsmechanik zwischen Motor und Tisch unberücksichtigt bleiben.

Es ist zu beachten, daß die Genauigkeit der Regelung allein durch den Soll-Istvergleich bestimmt wird. Damit kann sich an die Berechnung der Lageabweichung direkt eine Begrenzung der Datenwortbreite auf eine dem Mikrorechner angepaßte Wortbreite (1- bzw. 2-Byte-Verarbeitung) anschließen. Die Wahl einer anderen Verarbeitungsbreite als 8, 16 oder 32 Bit erscheint nicht sinnvoll, da kein Gewinn an Rechenzeit erzielbar ist.

4.5.2. Prozeßanschluß

Bei bestimmten Anwendungen kann die Messung einer analogen Größe notwendig werden, die innerhalb der Rechenzeit des Algorithmus starken Schwankungen unterliegt. In diesen Fällen ist die Wandlung der Meßgrößen mit direkten A-D-Wandlern – die in Näherung eine Augenblickswertmessung realisieren – durch eine Mittelwertbildung (z. B. über Spannungsfrequenzwandler) zu ersetzen. Ein Beispiel ist die Strommessung bei Gleichstromreversierantrieben im Lückbetrieb. Eine andere Form der Istwerterfassung ist bei der Drehzahlmessung notwendig. Im Fall der direkten Messung werden die Impulse eines inkrementalen Gebers (IGR) während einer Meßzeit t_M direkt in einen Vor- und Rückwärtszähler eingezählt. Für eine Drehzahl von 1 500 U/min ergibt sich bei Verwendung eines Gebers mit 10 000 Impulsen je Umdrehung und $t_M = 3$ ms eine Impulsanzahl von 750. Der Drehzahlsollwert ist in Inkrementen von 2 U/min wählbar. Wird die Meßzeit auf 6 ms erhöht, steigt das Auflösungsvermögen des Istwertes von 1,34 ‰ auf 0,67 ‰, und der Drehzahlsollwert kann mit einer Genauigkeit von 1 U/min vorgegeben werden. Durch dieses Beispiel wird der Widerspruch zwischen Dynamik und Genauigkeit bei quasistetigen Betrachtungen deutlich sichtbar. Die Langzeitgenauigkeit der Regelung hängt von der Erfassung jedes Impulses durch das Meßglied ab.

Mit der angenommenen Drehzahl ergibt sich eine maximale Übertragungsfrequenz von 250 kHz. Bereits in diesem Frequenzbereich ist mit Schwierigkeiten für eine störungsfreie Signalübertragung zu rechnen. Neben örtlichen Störquellen im Schaltschrank (z. B. Stromrichter) treten im Industrieinsatz lange Übertragungswege als Störquelle in den Vordergrund. Deshalb und bei gleichzeitiger Entlastung des MR-

Systems erscheint die Forderung nach einem IGR mit echtem Digitalausgang und MR-ansteuerbarer Meßzeitumschaltung sinnvoll. Intelligente Peripherie wird durch die Mikroelektronik möglich.

Die Störwahrscheinlichkeit bei der Istwertübertragung kann durch Speicherung der Information im Meßglied verringert und der Einfluß auftretender Störungen durch entsprechende Gestaltung des Algorithmus (Anwendung von Rekursionsformeln) abgeschwächt werden. Es wird deutlich, daß bei der keineswegs maximalen Forderung nach einem Auflösungsvermögen von 1 ‰ das Meßglied der schwächste Teil im System ist.

Die Forderung an das ausgangsseitige Auflösungsvermögen wird durch die Ansprechempfindlichkeit von Stellglied und Strecke bestimmt. 10-Bit-D-A-Wandler sind in der Mehrzahl der Anwendungsfälle ausreichend (Auflösung von 1:1 000). Um zusätzliche Störeinflüsse zu vermeiden (z. B. Drift und Ansprechschwelle von Analogschaltern) muß der Vorzeichenwechsel der Ausgangsspannung auf der Digitalseite bewirkt werden.

4.5.3. Programmstrukturen

Im Bild 4.5.4. sind mögliche Ausbaustufen bei einem MR-Einsatz für direkte digitale Regelungen zusammengefaßt. Im einfachsten Fall berechnet der Mikrorechner einen übergeordneten Algorithmus, dem analoge Glieder folgen; im günstigsten Fall erfolgt die gesamte Regelung digital durch den MR, und hochintegrierte, den Prozessor entlastende Zusatzschaltkreise realisieren die Ansteuerung des Stellgliedes.

Mögliche Programmstrukturen zur Realisierung einer Regelschleife sind im Bild 4.5.5. zusammengestellt. Bei der Abarbeitung dieses Programmablaufplanes sind folgende Entscheidungen zu treffen:

- F1: Sollen in den Programmspeichern ein oder mehrere Anwenderprogramme untergebracht werden?
- F2j: Auswahl des gerade benötigten Anwenderprogrammes, z. B. durch PROM-Auswechslung. Dazu ist das Aufbereitungsprogramm entweder universell zu schreiben oder jeweils ein spezielles Programm zu formulieren.
- F3: Ist ein gesonderter Startbefehl (z. B. Synchronisationsforderung) nötig?
- F4: Parameter- und Sollwertänderungen während des Betriebes werden durch Unterbrechung möglich.
- F5/8: Muß während einer Meßzeit t_M einmal oder konstant n-mal gerechnet werden, oder muß keine Synchronisation mit der Meßzeit erfolgen?
- F2n: Die Auswahl des Algorithmus soll durch einen Großrechner erfolgen. Es wird jedem System ein optimaler Algorithmus zugeordnet.
- F9n: Die Anpassung von Algorithmus und Parametern erfolgt nur einmal.
- F10n: Erfolgt keine „Langzeitadaption“, so läuft die Anpassung an das System nach jedem Schleifendurchlauf erneut.
- F10j: Durch Angabe von Vergleichswerten ist eine einfache, grobstufige Parameterumschaltung bzw. Algorithmusumschaltung möglich (z. B. Wechsel zwischen einfachen „schnellen“ und „langsamen“ komplizierten Algorithmen).

Der Abbruch des Hauptprogrammes kann durch einen HALT-Befehl, Interrupt oder Abschaltung erfolgen.

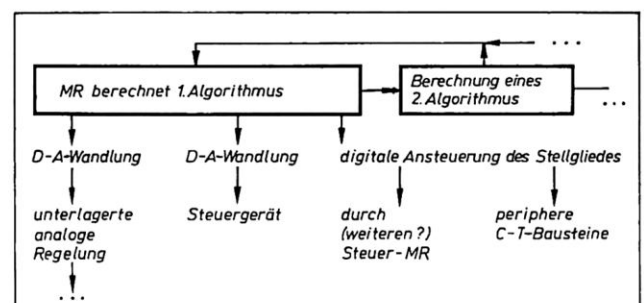


Bild 4.5.4.: Mögliche Ausbaustufen eines Mikrorechners für direkt digitale Regelungen

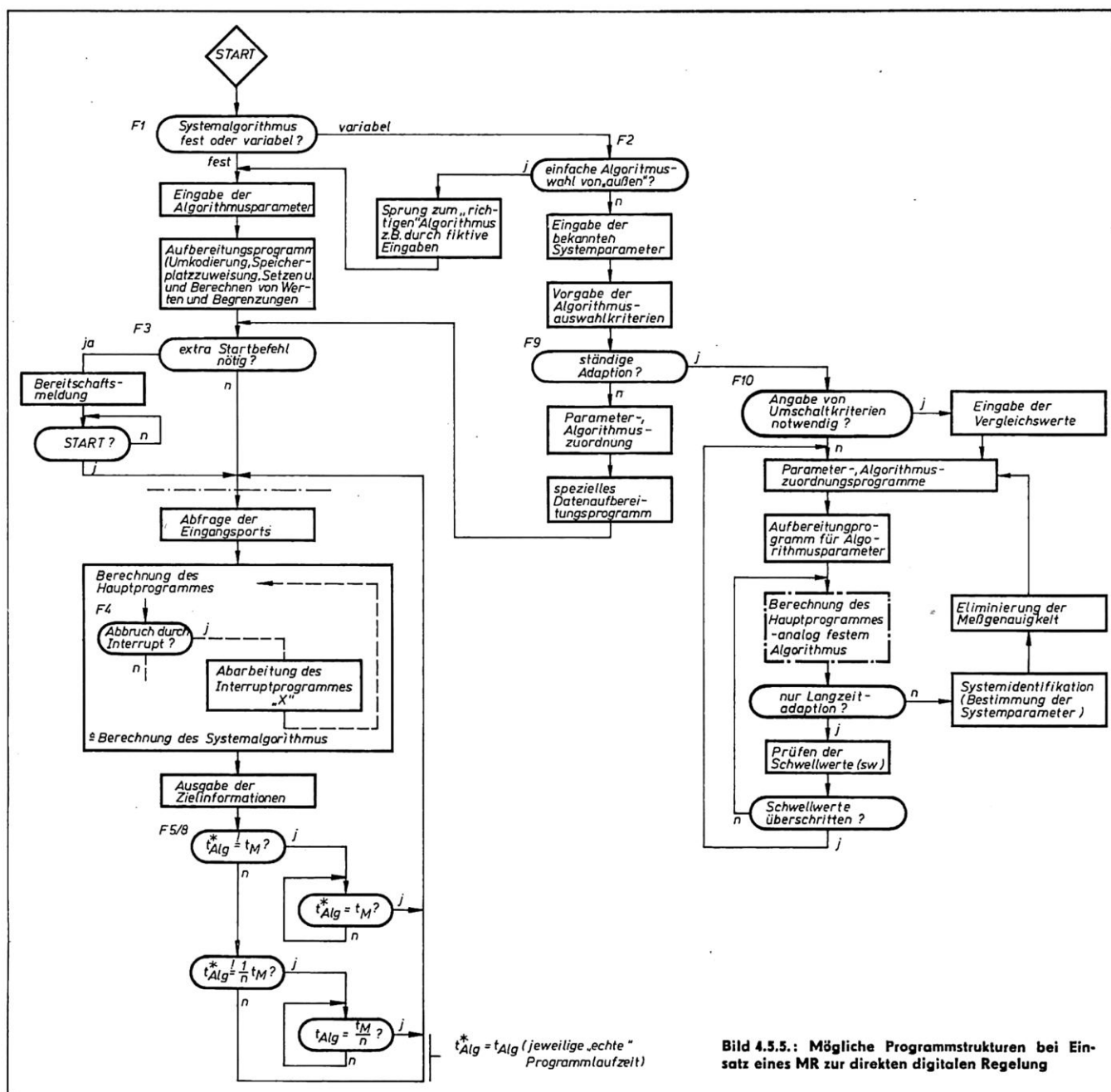


Bild 4.5.5.: Mögliche Programmstrukturen bei Einsatz eines MR zur direkten digitalen Regelung

Die vollständige Bearbeitung dieses Programmablaufplanes durch einen Mikrorechner sollte nicht Ziel des MR-Einsatzes sein. Für diesen sind PAP-Teile geeignet, etwa F1 fest, F2j, F10j und mit Einschränkungen F10n. Der gesamte Programmablaufplan umreißt etwa den Umfang eines automatischen Projektierungsprogrammes bzw. Möglichkeiten bei der Zusammenarbeit mehrerer Computer. Bild 4.5.5. kann keinen Anspruch auf Vollständigkeit erheben, so fehlen die Darstellung einer Algorithmuswahl durch das Nullsetzen von Parametern oder ein eventuelles Sperren bestimmter Programmteile für Interrupt.

Bei Auswahl der notwendigen Programmstruktur ist zu klären, ob und wie eine Synchronisation zwischen Rechenzeit und Meßzeit erfolgen soll (F5/8 im Bild 4.5.5.). Ein Verzicht auf Synchronisation ergibt die einfachsten Lösungen, birgt aber die Gefahr von Schwebungen, wenn sich die beiden Zeiten nur geringfügig unterscheiden.

Im Anschluß an die Strukturwahl wird der entsprechende Algorithmus programmiert und das Zeitverhalten festgestellt. Mit Vorliegen dieser Aussage sind u. a. folgende Entscheidungen zu treffen:

- Komplizierterer Algorithmus zeitmäßig möglich?
- Bedienung mehrerer Regelkreise zeitmäßig möglich?
- Adaption z. B. nur durch Vorgabe tabellierter Werte erreichbar?
- Prozessor ungeeignet, zusätzliche LSI-Schaltkreise notwendig?

- Andere Formen für Ein- und Ausgabe wählen (s. Abschn. 4.5.4.)?

Ziel der Programmierung bei Mikrocomputern für direkte digitale Regelungen ist eine zeitoptimale Programmgestaltung – keine vordergründig speicherplatzoptimalen Lösungen. Ein Zeitgewinn bei zeitkritischen Programmteilen kann auch durch Verschachtelung verschiedener Algorithmen erreicht werden [4.5.5.]. Hierdurch entstehen andererseits unübersichtliche, änderungsunfreundliche und wenig flexible Lösungen. Eine entsprechende Technik sollte nur angewendet werden, wenn die Lösung eines Problems auf anderem Wege unmöglich erscheint.

4.5.4. Realisierung einer digitalen Regelung

Eine Abschätzung der Größe eines MR-Systems für direkte digitale Regelungen ist neben dem Speicherplatzbedarf sehr wesentlich von der Anzahl der Ein-Ausgabe-Ports abhängig. Als Beispiel für den „Kommunikationsaufwand“ soll ein einfaches System mit einer Datenwortbreite von 8 bit bei Doppelwortverarbeitung dienen:

Eingaben:

- 10 + 1 bit: Sollwert und Vorzeichen
- 12 + 1 bit: Istwert und Übergabesignal
- 8 + 7 bit: zwei Parameter
- 3 bit: Steuersignale

Ausgaben:

10 + 1 bit: Stellgröße und Vorzeichen.

Für dieses System sind mindestens acht 1-Byte-Ports notwendig. Da übliche Ein- und Ausgabebausteine 16 bis 24 bits „kontrollieren“ können, sind allein für dieses System etwa vier LSI-Bausteine zur Kommunikation mit der Außenwelt nötig. Trotzdem ist ein wesentlicher Vorteil von Mikrocomputern, daß durch eine Konfiguration von relativ wenigen Bausteinen ein allein durch die Software hoch flexibles System entsteht. Bei einem Vergleich mit analogen Reglern sollte der eventuell durch mehr Bedienelemente bedingte größere Platzbedarf (Einsatz von Ziffernvorwahlschaltern zur Parameter- und Sollwerteinstellung) nicht als Nachteil angesehen werden. Vergleichbar ist lediglich das zur Realisierung der Reglerfunktion beanspruchte Volumen.

Die Bilder 4.5.6. und 4.5.7. zeigen zwei Drehzahlübergangsvorgänge einer im VEB Elektropjekt und Anlagenbau entwickelten digitalen Drehzahlregelung bei Feldschwächung und betragsoptimaler bzw. symmetrischer Reglereinstellung. Die digitale Regelung ist gut an der Änderung der Stellgröße (D-A-Wandlerausgang) zu diskreten Zeitpunkten ($\Delta t = T = 6 \text{ ms}$) zu erkennen. Es wird deutlich, daß auch bei digitaler Regelung dynamisch gute Antriebe möglich sind. Weitere Vorteile sind u. a. hohe Langzeitgenauigkeit ($\leq 0,1 \text{ ‰}$) und driftfreies Arbeiten bei Drehzahl Null.

Bevor eine MR-Regelung in Betrieb genommen wird, empfiehlt es sich, die Ein- und Ausgabestrecken genauestens zu testen. Zur Prüfung der gesamten Ausgabestrecke (Prozessor, Ausgabebaustein, D-A-Wandler) kann z. B. ein Sägezahngenerator programmiert werden.

Im Bild 4.5.8. ist eine mögliche Variante im Mnemokode angegeben. Ist zusätzlich eine analoge Regelung vorhanden, kann bei einer funktionstüchtigen Ausgabestrecke die Istwerterfassung ebenfalls leicht getestet werden, indem bei Betrieb der analogen Regelung der Istwert vom Mikrorechner erfaßt und über den D-A-Wandler ausgegeben wird. Das Ergebnis ist zu oszillografieren und mit der Tachoaussgangsspannung zu vergleichen.

Die Ein- und Ausgabe von Informationen zwischen dem MR-System und seiner Umwelt kann auf vielfältige Art und Weise geschehen:

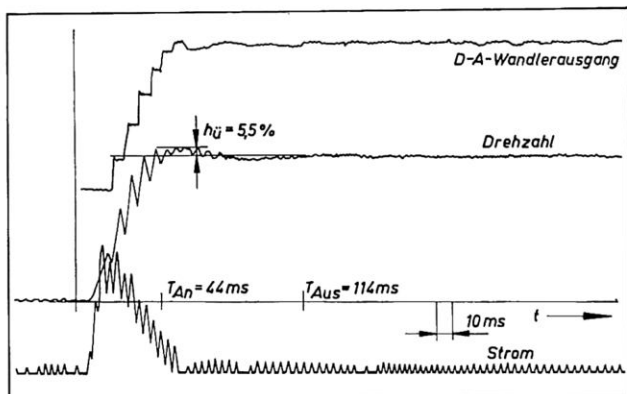


Bild 4.5.6.: Drehzahlübergangsvorgang bei betragsoptimaler Reglereinstellung und Feldschwächung ($T_{AN} = 44 \text{ ms}$, $T_{AUS} = 114 \text{ ms}$, $h_u = 5,5 \text{ ‰}$)

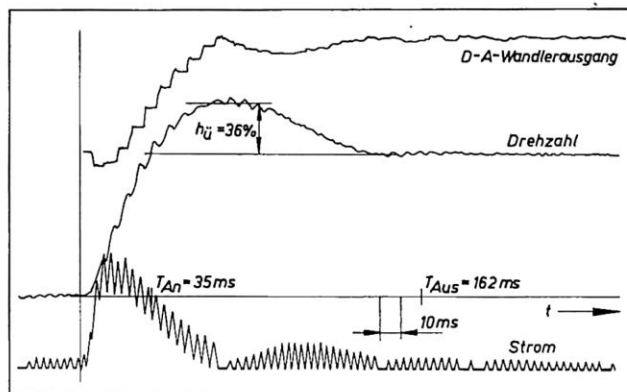


Bild 4.5.7.: Drehzahlübergangsvorgang bei symmetrischer Reglereinstellung und Feldschwächung ($T_{AN} = 35 \text{ ms}$, $T_{AUS} = 162 \text{ ms}$, $h_u = 36 \text{ ‰}$)

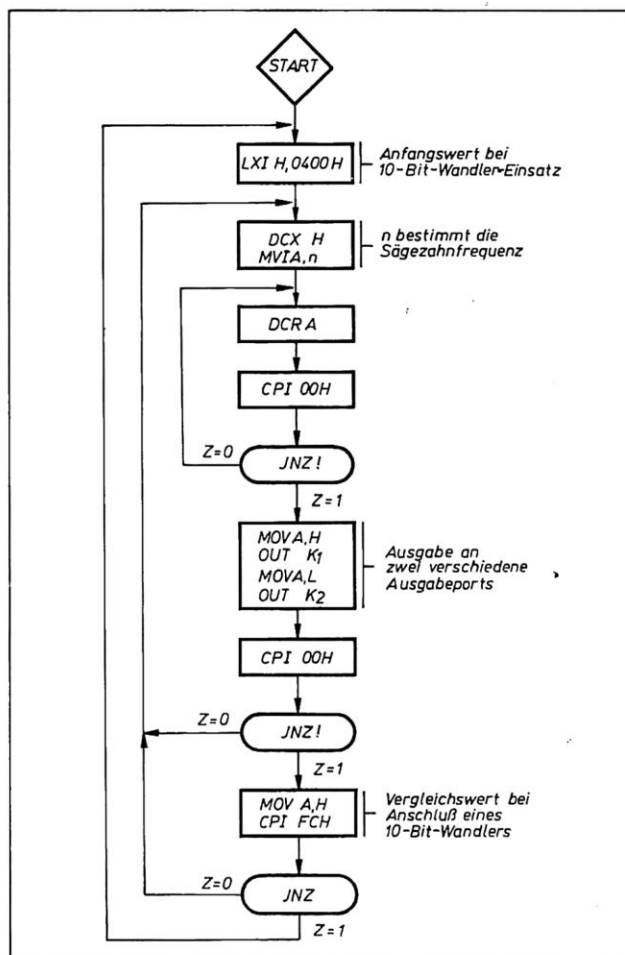


Bild 4.5.8.: Mnemokode für Sägezahngenerator

- parallel oder seriell
- durch Übergabesignale (Interrupt- oder Wartesteuerungen)
- Mehrfachabfrage oder Übergabesteuerung
- ohne Mehrfachabfrage und ohne Übergabesteuerung
- in Abhängigkeit von Sperrbits (hard- und softwaremäßig)
- synchronisiert bzw. regelmäßig oder zu beliebigen Zeitpunkten.

Die jeweilige Wahl der Kommunikationsart hängt von den Anforderungen des zu automatisierenden Prozesses ab, wobei stets die einfachste mögliche Variante gewählt werden sollte.

4.5.5. Ausblick

Gegenwärtig werden digitale Regelungen auf Mikrorechnerbasis noch nicht breit angewendet. Die Entwicklungen stehen erst am Anfang eines aussichtsreichen Weges. Es werden völlig neue Reglerstrukturen möglich; Selbstanpassung von Parametern, Parametersuche und Modellnachbildungen der Strecke im Regler sind keine grundsätzlichen Schwierigkeiten für digitale Regler mit Mikroprozessoren. Da die Entwicklung der Mikroelektronik weiter schnell vorangeht, wird eine rechtzeitige Vorbereitung auf die kommende Technik unerlässlich, um die Vorzüge dieser Technik ohne Verzögerungszeiten praxiswirksam werden zu lassen.

Literatur

- [4.5.1.] Faggin, F.: VLSI verändert Computer-Strukturen. Elektronik 27 (1978) H. 12, S. 57
- [4.5.2.] Schöne, A.: Zum technischen Stand der DDC. Regelungstechnik 15 (1967) H. 7, S. 297
- [4.5.3.] Reiner, A.; Wiegand, R.: Überblick über Algorithmen zur digitalen Regelung. Regelungstechnik 24 (1976) H. 6, S. 181–226; H. 7, S. 227–232
- [4.5.4.] Schönfeld, R.; Krug, H.: Berechnung digitaler Regelstrukturen mit Hilfe modifizierter Frequenzkennlinien. messen steuern regeln 22 (1979) H. 4, S. 186–191
- [4.5.5.] Schnieder, E.: Regelung von Gleichstromantrieben durch Mikroprozessoren. IFAC Symposium 1977, S. 603–608