Dipl.-Ing. WOLFGANG MATTHES

Für viele umfangreiche Aufgaben der Informationsverarbeitung hat es sich als vorteilhaft erwiesen, Anordnungen aus mehreren miteinander gekoppelten Mikrorechnern einzusetzen, statt größere Rechner oder spezielle Hardware anzuwenden. Mikrorechnerbaugruppen, die sich dafür eignen, sind auf dem Weltmarkt seit einiger Zeit verfügbar. Die Vorteile, die derartige Lösungen in Fertigung und Anwendung bieten, müssen allerdings durch beträchtliche Anstrengungen im Konzeptions- und Entwicklungsprozeß erarbeitet werden. Diese Beitragsfolge soll dazu Erfahrungen und Anregungen vermitteln. Sie wurde im VEB Robotron ZFT Karl-Marx-Stadt erarbeitet, die Erfahrungen wurden im wesentlichen an peripheren Einrichtungen von EDVA gesammelt. Bei den betrachteten Strukturen handelt es sich um Anordnungen, bei denen Mikrorechner zusammen mit ergänzenden speziellen Hardwareeinrichtungen an einen universellen Bus angeschlossen sind, um algorithmisch komplexe Steuerungsaufgaben zu lösen. Die Betrachtungen sind mit dem der DDR verfügbaren Bauelementesortiment verknüpft. Grundlage ist die Mikroprozessorfamilie U 880.

1. Einführung

1.1. Einsatz von Multimikrorechnersystemen

Anordnungen aus mehreren miteinander gekoppelten Mikrorechnern können für bestimmte Aufgabengebiete Einrichtungen, die einen erhöhten Aufwand erfordern würden (z. B. spezielle Hardware, größere Rechenanlagen), vorteilhaft ersetzen. Von der Vielzahl der möglichen Anwendungsfälle sollen hier nur jene betrachtet werden, die die Steuerung physischer Interfaces (zu Tastaturen, Druckern usw., aber auch zu Produktionsmaschinen) zum Gegenstand haben und die eine gewisse Aufwandsgrenze nicht überschreiten (eine Anordnung aus zehn Mikrorechnern zur Steuerung einer Maschine liegt im Bereich der Betrachtungen, nicht aber eine Anordnung aus 256 Mikrorechnern zum Lösen partieller Differentialgleichungen).

1.1.1. Lösung komplexer Problemstellungen

Ist ein Steuerungsproblem nicht allzu trivial, so dürfte beim heutigen Stand der Technik sofort der Einsatz eines Mikrorechners erwogen werden. Erweist es sich, daß dessen Leistungsvermögen nicht ausreicht, müssen andere Wege gesucht werden.

Im allgemeinen stehen zur Auswahl:

- Spezielle Hardware
  Der Entwurf spezieller Schaltungen ist bei Forderungen nach hoher Geschwindigkeit unvermeidlich. Kommt zur Geschwindigkeit noch große Komplexität hinzu, so führt dies zu beachtlichen Aufwendungen und Folgeproblemen (Realisierung, Inbetriebnahme, Änderungen, Service).
- Spezielle mikroprogrammierbare Steuerwerke
  Das Prinzip der Mikroprogrammierung hat sich seit langem bewährt, um komplexe und schnelle Steuerungsabläufe mit verhältnismäßig regulären und übersichtlichen Hardwarestrukturen zu realisieren. Das Ändern von Abläufen, das Ausführen spezieller Testabläufe usw. sind weitgehend unproblematisch. Zur Definition von Mikrobefehlsformaten und zur Organisation der Ablaufsteuerung existiert eine umfangreiche Literatur.
- Rechner ausreichender Leistungsfähigkeit
  Der Einsatz eines entsprechend leistungsfähigen Rechners erübrigt weitgehend die Entwicklung spezieller Hardware und verlagert die Entwicklungsprobleme auf die Softwareebene.
- Multimikrorechneranordnungen
  Anordnungen aus mehreren Mikrorechnern, erforderlichenfalls gekoppelt mit spezieller Hardware geringeren Umfanges zu Anpassungs- bzw. Beschleunigungszwekken, stellen für viele Anwendungsfälle eine wirtschaftliche Alternative zu den vorher genannten Varianten dar.

Die Entscheidung für eine der Varianten hängt zunächst von der Art des Problems (seiner algorithmischen Komplexität) und vom geforderten Realzeitverhalten ab.

Multimikrorechnersysteme sind nur dann sinnvoll, wenn eine Parallelisierung möglich ist (Zerlegung in Teilaufgaben, die parallel von mehreren Mikrorechnern bearbeitet werden können, wobei jede Teilaufgabe so beschaffen ist, daß der betreffende Mikrorechner dem geforderten Realzeitverhalten gerecht werden kann). Gelingt dies nicht, so muß eine andere Lösung gewählt werden.

Sofern derartige Anordnungen den technischen und funktionellen Anforderungen dem Prinzip nach genügen, müssen Wirtschaftlichkeitsbetrachtungen angestellt werden, um die am besten geeignete Lösung auszuwählen.

Bei algorithmisch sehr komplexen Aufgaben steht praktisch nur noch die Alternative des einzelnen, entsprechend leistungsfähigen Rechners zur Auswahl. Im Falle einer einmaligen Anwendung dürften die kommerzielle Verfügbarkeit und die existierende Softwareumgebung die dominierende Rolle spielen.

Für den Einsatz in Erzeugnissen, die in größeren Stückzahlen hergestellt werden, sind die reinen Hardwarekosten maßgebend, so daß in der Regel das Multimikrorechnersystem bevorzugt werden dürfte, selbst wenn dafür umfangreichere Entwicklungsarbeiten zu leisten sind.

Wird statt eines Multimikrorechnersystems ein spezielles mikroprogrammiertes Steuerwerk in Erwägung gezogen, so ist zu bedenken, daß dieses hinsichtlich des Hardware-Aufwandes durchaus vorteilhaft sein kann. Ein Mikrorechner, der für den Einsatz in Multimikrorechnersystemen vorgesehen ist (mit U 880 CPU, 32 Kbyte RAM, Hardware für Buskopplung, Diagnostik usw.), erfordert etwa 140 Schaltkreise. Eine Anordnung aus drei Mikrorechnern umfaßt somit 420 Schaltkreise. Mit diesem Aufwand ließe sich auch ein leistungsfähiges mikroprogrammiertes Steuerwerk aufbauen. Dem steht entgegen, daß das Steuerwerk keine so reguläre Struktur wie die Mikrorechneranordnung hat. Die Befehlsliste von Mikroprozessoren ist im allgemeinen reichhaltiger als die spezieller Steuerwerke. Damit ist gewährleistet, daß man einen Mikrorechner von vornherein universell programmieren kann, während bei einem speziellen Steuerwerk die algorithmische Flexibilität durch die konkrete Gestaltung der Hardware bestimmt wird. Weiterhin ist die Befehlsliste einer Mikroprozessor-Schaltkreisfamilie von vornherein bekannt

(und es sind in der Regel entsprechende Entwicklungshilfen wie Entwicklungssysteme, Cross-Assembler u. dgl. verfügbar), so daß bereits nach der Wahl des Mikroprozessorsystems die Bearbeitung der Software beginnen kann.
Hingegen erfordert ein spezielles Steuerwerk die Schaffung eigener Entwicklungshilfen und zwingt zu einer Serialisierung des Entwicklungsprozesses: Es ist entweder zuerst die Hardware zu definieren, um nachfolgend die gewünschten Funktionen mit den Mitteln der Hardware implementieren zu können (mit Rückwirkungen auf die Hardwarestruktur, wenn es sich z. B. herausstellt, daß zusätzliche Mikroanweisungen erforderlich sind), oder es ist zuerst das zu lösende Problem algorithmisch zu bearbeiten, bevor die Hardware definiert wird, um sicher zu sein, daß die Mikroanweisungen, Datenwege, Speicher usw. dem Problem angemessen sind.
Daraus ergibt sich, daß in der Regel ein Multimikrorechnersystem, sofern es bei angemessenem Aufwand für den betreffenden Zweck geeignet ist, einer speziellen Hardware-Lösung (selbst bei Programmierbarkeit dieser Hardware) vorgezogen werden sollte.

1.1.2. Besonderheiten von Multimikrorechnersystemen

Dem Anwender fällt zunächst der vergleichsweise geringe Hardwareumfang eines Multimikrorechnersystems auf. Dahinter verbergen sich jedoch beachtliche Entwicklungsaufwendungen. Sämtliche funktionellen Eigenschaften sind durch die Software definiert. Dies hat zwei Aspekte:

- Man kann (darf) programmieren
  Gegenüber speziellen Hardwarelösungen ergibt sich ein sehr hoher Freiheitsgrad zur Realisierung zusätzlicher Funktionen, zur Implementierung komplexer Algorithmen, zur Schaffung eines hohen Bedienungskomforts usw.
- Man muß programmieren
  Die Entwicklungsarbeit ist mit der Bereitstellung der Hardware nicht beendet, sondern es muß zusätzlich eine (unter Umständen sehr umfangreiche und komplexe) Software geschaffen werden, um die gewünschten Funktionen zu realisieren.

Zu Beginn einer Entwicklung verleitet die offensichtliche Möglichkeit des Programmierens üblicherweise zu einer gewissen Euphorie. Da im Prinzip alles möglich ist, werden zunächst alle irgendwie wünschbaren Eigenschaften vorgesehen. Die konkrete Realisierung führt dann zu erheblichen Terminproblemen, um die funktionelle Komplexität zu beherrschen. Demnach ist eine sorgfältige Analyse der Erfordernisse des Anwenders zu Beginn der Softwarespezifikation unbedingt zu empfehlen.
Namentlich bei den Aspekten, die mit dem Bedienungskomfort zusammenhängen, ergaben sich folgende Beobachtungen:

- Es ist zunächst unproblematisch, eine Vielzahl komplexer Funktionen vorzusehen, sofern ausreichend Speicherplatz vorhanden ist.
- Mit zunehmender Komplexität wachsen die Schwierigkeiten beim Testen stark an.
- Ein Teil der Funktionen wird von den Anwendern nie oder nur äußerst selten benutzt (weil im praktischen Betrieb kein Bedarf danach besteht, weil die Bedienung ungewohnt oder schlecht dokumentiert ist usw.).

Multimikrorechnersysteme erfordern weiterhin zusätzliche Entwicklungsleistungen für Diagnose, Wartung und Service. Auch hier gibt es zwei Aspekte:

- Man kann umfangreiche und komfortable Vorkehrungen treffen. Durch die freie Programmierbarkeit können an sich beliebige Testabläufe implementiert werden, so daß es nicht erforderlich ist, normale Betriebsabläufe zu Testzwecken zu benutzen. Damit kann die Servicefreundlichkeit und somit die Verfügbarkeit beim Einsatz von Multimikrorechneranordnungen beträchtlich erhöht werden, ohne daß dazu erhebliche Hardwareaufwendungen erforderlich sind.
- Man muß die entsprechenden Vorkehrungen treffen. Da alle Abläufe durch Programme realisiert sind, wobei in vielen Fällen erst das Zusammenwirken mehrerer Programme, die in mehreren Mikrorechnern laufen, die gewünschte Funktion ergibt, wird es zur zwingenden Notwendigkeit, das Testen vom normalen Betrieb zu trennen.

  Die Abläufe im Normalbetrieb sind nicht mehr mit konventionellen Mitteln zu kontrollieren. Spezielle Prüfmittel sind teuer und bei komplexen Systemen nicht mehr zu verwenden, selbst wenn die Kosten akzeptiert werden. So ist es durchaus vertretbar, wenn für Testzwecke an einen Mikrorechner ein Logikanalyzer angeschlossen werden muß. Der Anschluß von acht Analyzern an eine Anordnung mit acht Mikrorechnern ist dagegen weder praktikabel noch hinsichtlich der Kosten zu rechtfertigen. Damit muß für jede Entwicklung ein entsprechender Anteil an Hardware- und Softwarevorkehrungen für Diagnose, Testen und Service vorgesehen werden. Die Hardwareaufwendungen dürften in der Regel gering sein (10 bis 15 % der Gesamtaufwendungen). Sie betreffen im wesentlichen Vorkehrungen zum Einspeisen bzw. Abfragen diagnostischer Signale sowie Bedien- und Anzeigemittel zur Testablaufsteuerung und Fehleranzeige. Hingegen kann die Testsoftware ohne weiteres den Umfang der Anwendungssoftware erreichen.

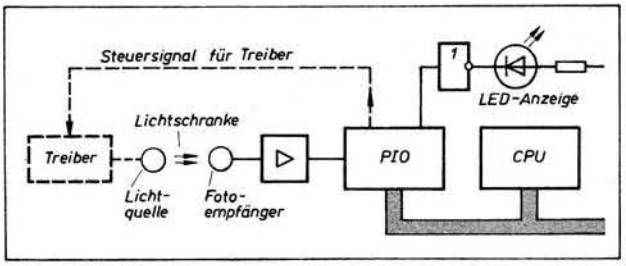


Bild 2: Realisierung der Funktion nach Bild 1 durch Anschluß an einen PIO-Schaltkreis eines Mikrorechners.

Testmöglichkeiten: Programmiertes Ein- und Ausschalten der Lichtquelle (gestrichelt); programmierte Anzeige des aktuellen Zustandes mit LED

Zur Veranschaulichung seien zwei Beispiele diskutiert.

Beispiel 1:

Die Impulse aus einer Lichtschrankenanordnung sind zu zählen (z. B. Transportband). Der Zählbereich soll binär 16 bit betragen. In einer konventionellen Schaltung ist eine Zählschaltung direkt an die Lichtschranke angeschlossen. Sie kann mit vier D 193 einfach realisiert werden (Bild 1).

Das Testen dieser Schaltung in der realen Einsatzumgebung bringt einige Probleme mit sich. Ein vollständiges Durchzählen erfordert 64 K Zählimpulse. Somit müßte die Lichtschranke 65 536mal unterbrochen werden. Das ist auf einfache Weise nicht möglich (man kann nicht etwa 65 000 Teile über das Transportband befördern, nur um die Schaltung zu testen), so daß in der Praxis nur die Möglichkeit bleibt, die gesamte Baugruppe auf Verdacht zu

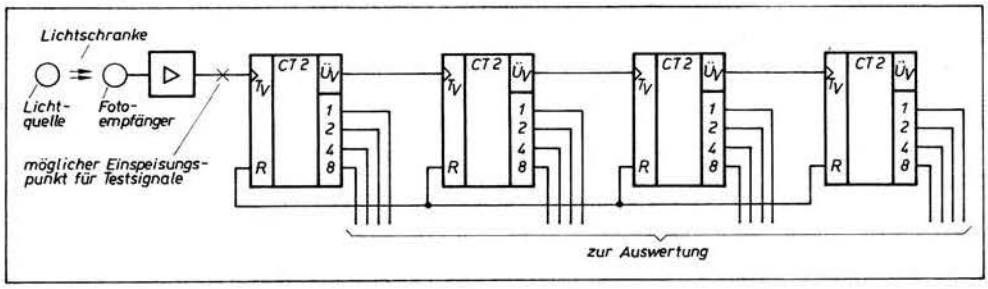


Bild 1: Lichtschranke mit nachgeschaltetem Zähler aus vier Zählerschaltkreisen

wechseln (es sei denn, man benutzt zusätzliche Meßmittel, um den Fehler genauer zu lokalisieren).
Bei Einsatz eines Mikrorechners wird die Lichtschranke ausgangsseitig direkt an eine E-A-Schaltung (z. B. eine PIO) angeschlossen (Bild 2). Das Testen wird damit sehr einfach: Eine Testroutine prüft die Funktionsfähigkeit von CPU, Speichern usw. Eine weitere prüft, ob die PIO das Signal der Lichtschranke korrekt empfängt (entweder vollautomatisch, indem die Lichtquelle über einen anderen E-A-Anschluß programmtechnisch ein- und ausgeschaltet wird, oder durch manuellen Eingriff, indem das Ausgangssignal der Lichtschranke mit einer Leuchtdiode angezeigt wird, die ihrerseits programmtechnisch angesteuert wird).

Beispiel 2:

Es ist eine Anordnung aus zwei Mikrorechnern vorgesehen, wobei einer eine Tastatur abfragt und Steuerinformation an einen zweiten übermittelt, der die jeweiligen Aktionen ausführt (Bild 3).

Die Überprüfung auf Funktionsfähigkeit und die Lokalisierung eines Fehlers würden (bei herkömmlichem Vorgehen)

- den Anschluß eines Logikanalyzers an jeden der Mikrorechner sowie an die Verbindungsleitungen (je nach Kanalzahl zwei bis vier Geräte)
- die genaue Kenntnis der Programmabläufe in beiden Mikrorechnern

erfordern. Das ist nicht mehr praktikabel und erzwingt Testroutinen für die Prüfung der internen Betriebsfähigkeit beider Mikrorechner, die Prüfung der Verbindungen zwischen ihnen, die Prüfung der Tastaturabfrage und für die Prüfung der Erregung der Aktionssignale.

1.2. Prinzipien der Funktionsaufteilung

Die Zerlegung des zu lösenden Problems in Teilaufgaben und deren Verteilung auf die Hardwarekomplexe (Mikrorechner + ergänzende Spezialhardware) ist ein wesentlicher Aspekt bei der Konzipierung eines Multimikrorechnersystems.

Folgende Parameter haben auf die Zerlegung entscheidenden Einfluß:

- Leistungsfähigkeit der einzelnen Mikrorechner
- Organisation und Leistungsfähigkeit der Koppelstellen zwischen den Mikrorechnern
- Interfaces hinsichtlich ihrer Datenrate, geforderter Reaktionszeiten und Kompliziertheit der Steuerfolgen
- algorithmische Komplexität der Verarbeitungsaufgaben.

Die Zuordnung der physischen Interfaces ist relativ einfach möglich. Geht man davon aus, daß alles, was von einem Mikrorechner bewältigt werden kann, auch damit realisiert werden sollte (um Sonderhardware zu vermeiden), können zunächst alle Interfaces, die direkt über LSI-Schaltkreise wie PIO, SIO, CTC gesteuert werden sollen, entsprechenden Mikrorechnern zugeordnet werden. Dies betrifft sowohl Interfaces, die nur eine einfache Anpassung an die LSI-IS erfordern, als auch solche, bei denen die Anpassungsschaltungen autonom arbeitende Komplexe enthalten.
Für Interfaces, die auf diese Weise nicht betrieben werden können, müssen autonome Steuerschaltungen vorgesehen werden. Damit ist die minimal notwendige Hardwarestruktur definiert.
Es ist nun zu entscheiden, ob diese ausreicht oder ob weitere Mikrorechner (für Verarbeitungszwecke) hinzugefügt werden müssen. Prinzipiell kann unter Steuerung eines entsprechenden Betriebssystems eine Vielzahl von Programmen zeitmultiplex simultan auf einem Mikrorechner abgearbeitet werden. Die praktischen Grenzen werden durch die Speicherkapazität und durch die gewünschten Antwortzeiten gesetzt. (Für einen Mikrorechner mit U 880 und 32-Kbyte-RAM sind durchaus bis zu acht Prozesse bzw. „Partitions" im Multiprogramming praktikabel.) Für die Umschaltung zwischen zwei Prozessen bzw. für das Intervall zwischen der physischen Auslösung und dem Start des entsprechenden Anwendungsprogrammes sollten (bei einem komfortablen Echtzeitbetriebssystem) etwa 1...2 ms veranschlagt werden. Damit läßt sich abschätzen, wieviele Mikrorechner für die Bewältigung der Verarbeitungsaufgaben vorgesehen werden müssen.
Eine Anordnung aus Ein- und Ausgabe- und Verarbeitungsmikrorechnern kann recht umfangreich werden und hinsichtlich der Hardwareausnutzung im praktischen Betrieb nicht sehr effizient sein, denn die meisten Mikrorechner sind häufig nicht beschäftigt. Somit sollte in einem weiteren Schritt untersucht werden, inwiefern sich Verarbeitungs- und E-A-Steuerprozesse auf einem Mikrorechner ausführen lassen.

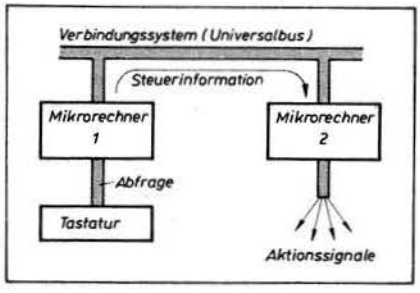


Bild 3: Anordnung aus zwei gekoppelten Mikrorechnern

Die Kommunikation zwischen einem Anwendungsprogramm und einer E-A-Steuerroutine kann nach zwei Prinzipien erfolgen:

- Unterbrechung (Interrupt)
- Abfrage (Polling).

Die Behandlung des Interrupts kostet im Rahmen eines Echtzeitbetriebssystems 1...2 ms (gemessen von der physischen Auslösung bis zum Start des betreffenden Anwendungsprogramms).
Reserviert man die Austauschregister des U 880 für eine schnelle Interruptbehandlung, so sind Reaktionszeiten von 50...100 µs erreichbar. Natürlich können Reaktionszeiten in dieser Größenordnung (sogar kürzere) auch durch direkte („physische") Programmierung erreicht werden. Andererseits ist gerade bei hoher funktioneller Komplexität ein wohldefiniertes und komfortables Betriebssystem eine entscheidende Voraussetzung, um das Entwickeln und Testen der Software zu beherrschen. (In diesem Sinne verdienen neuere Entwicklungen besondere Beachtung, bei denen wesentliche Betriebssystemkomponenten direkt auf der Mikroprozessor-IS realisiert werden [1] [2].)
Eine Abfrageschleife im Rahmen eines Anwendungsprogrammes hat üblicherweise kürzere Reaktionszeiten als eine Unterbrechung, da keine Kontextumschaltung erforderlich ist. Abfrageschleifen haben aber die Eigenschaft, daß das betreffende Programm den Mikrorechner auch dann belegt, wenn keine externen Reaktionen auftreten, sondern lediglich erwartet werden. Damit werden andere lauffähige Programme verzögert, oder beim Ausbleiben der abgefragten Reaktion kommt die gesamte Verarbeitung zum Erliegen. Als Ausweg bietet sich eine zeitgesteuerte Umschaltung zum nächsten Prozeß (Time Slicing) bzw. eine übergeordnete Zeitüberwachung (Watchdog) an. Für die Dauer einer Zeitscheibe beim Time Slicing hat sich eine Größenordnung von 30 ms bewährt.
Derartige Techniken gestatten zwar eine sehr gute Ausnutzung der Hardware, sind aber natürlich nur dann anwendbar, wenn die zeitlichen Verzögerungen durch die Echtzeitumgebung toleriert werden.

1.3. Prinzipien der Hardwareaufteilung

Weitere Überlegungen sind erforderlich, wenn die Hardwarekomponenten nicht fertig bezogen werden können, sondern selbst entworfen werden müssen. Dabei wird normalerweise von vornherein eine gewisse Standardisierung von Leiterplattenformaten, Kontaktbelegungen usw. angestrebt werden, um eine modulare Erweiterbarkeit des Systems zu gewährleisten, um Servicefreundlichkeit zu gewährleisten und um den Prozeß des Entwurfes und der Inbetriebnahme besser zu beherrschen.
Die gewünschte Modularisierung ist nur dadurch zu erreichen, daß auf den einzelnen Leiterplatten funktionelle Einheiten untergebracht werden, die über reguläre Interfaces (z. B. Bussysteme) miteinander in Verbindung stehen. Damit ist eine gewisse Mindestgröße der Leiterplatten festgelegt. Bei einem Multimikrorechnersystem besteht eine wesentliche

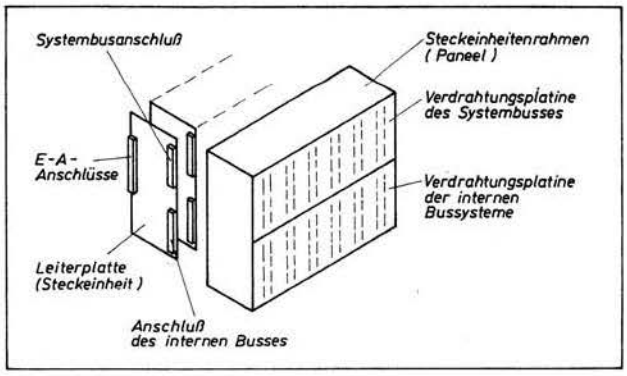


Bild 4: Multimikrorechnersystem in Paneelbauweise

Entscheidung darin, ob der einzelne Mikrorechner in sich modular erweiterbar sein soll oder ob die Baugruppen komplette Mikrorechner sein sollen, die durch ein reguläres Interface mit dem System gekoppelt werden.
Im ersten Fall müssen die einzelnen Leiterplatten an zwei Interfaces angeschlossen werden, nämlich an den internen Bus des einzelnen Mikrorechners und an das Koppelinterface des Systems. Der Vorteil besteht darin, daß die Mikrorechner in sich erweiterungsfähig bzw. modifizierbar sind. Dies erfordert konstruktive Vorkehrungen dafür, daß die einzelnen Mikrorechner aus einer unterschiedlichen Anzahl von Leiterplatten bestehen können. Damit ergibt sich zwangsläufig eine Paneelbauweise gemäß Bild 4.
Gewisse Nachteile bestehen darin, daß entweder die Leitungsführung des internen Busses oder des Koppelinterfaces dem Benutzer überlassen werden muß (Wickelverdrahtung) und daß die E-A-Anschlüsse praktisch nur an der Gegenseite der Leiterplatten herausgeführt werden können, wodurch der Aufbau unübersichtlich und der Service umständlich wird. Des weiteren ist diese konstruktive Ausführung relativ platz- und materialintensiv.
Der Stand der Technik ermöglicht es, einen kompletten Mikrorechner mit recht umfangreicher Ausstattung komplett auf einer einzigen Leiterplatte unterzubringen. Die Anschlüsse für das Koppelinterface des Systems und für die E-A-Verbindungen können auf einer Seite der Leiterplatte herausgeführt werden. Anstelle der Paneelbauweise ist auch eine Anordnung in Buchform möglich, wie dies im Bild 5 dargestellt ist.
Eine derartige Anordnung ist sehr kompakt und materialsparend. Ein weiterer Vorteil besteht darin, daß erforderliche Messungen praktisch im laufenden Betrieb möglich sind. Problematisch ist allerdings die Unterbringung der Schaltmittel für die E-A-Anpassung. Da die großen Leiterplatten in der Entwicklung sehr teuer sind, verbietet es sich, anwendungsspezifische Schaltungen auf ihnen unterzubringen. Bei mäßiger Anzahl derartiger Anpassungsbaugruppen können diese jeweils an Ort und Stelle befestigt werden (ein Drukkeradapter beispielsweise an der Rückwand des betreffenden Druckers). Dies verschlechtert allerdings die Servicefreundlichkeit, da das Auswechseln im Vergleich zu üblichen Steckeinheiten schwieriger wird, und bringt zusätzliche Probleme bei der Funkentstörung, da sich eine derart verteilte Anordnung schlechter abschirmen läßt.

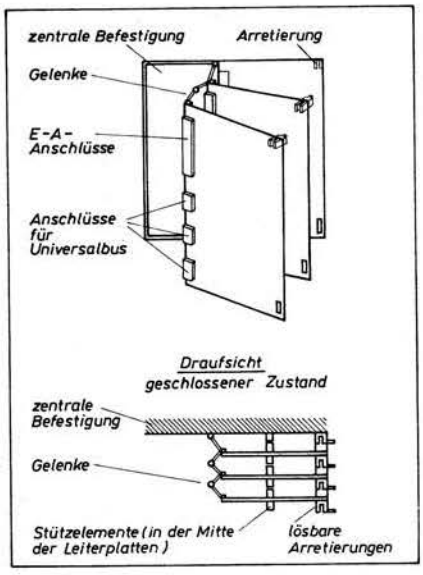


Bild 5: Leiterplattenanordnung in Buchform

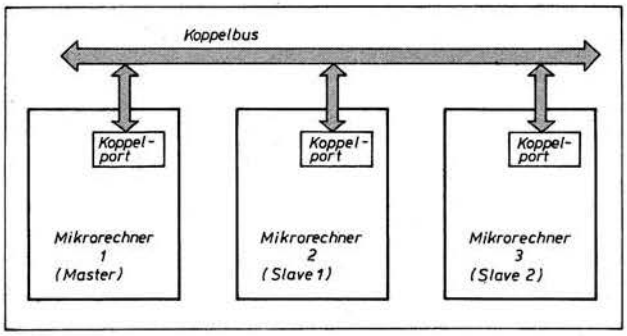


Bild 6: Multimikrorechnerkopplung durch E-A-Ports [3]

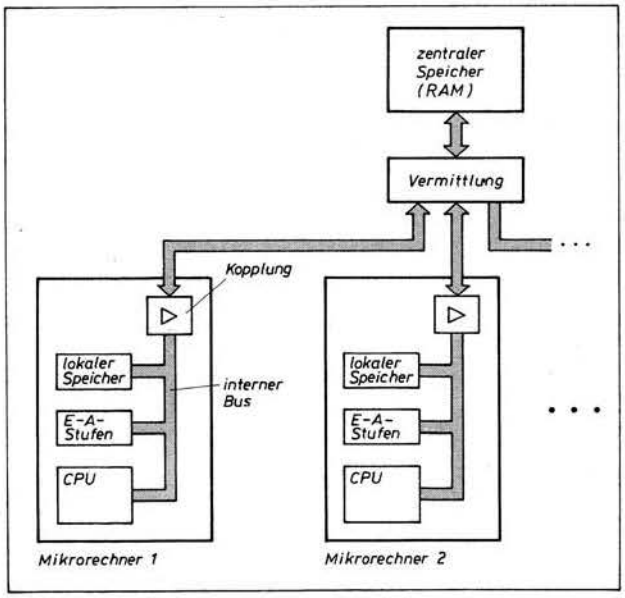


Bild 7: Multimikrorechnerkopplung durch zentralen Speicher

In der Regel dürfte man sich bei Systemen, die sehr stark E-A-orientiert sind, für die Paneelbauweise entscheiden und die Mikrorechner in sich modular aufteilen. Das führt zu Leiterplattenformaten von etwa 200 mm × 170 mm, die sich sowohl zur Unterbringung von Mikrorechnermoduln als auch von prozeßspezifischen E-A-Anpaßschaltungen eignen.
Ist das System überwiegend für interne Verarbeitungsaufgaben vorgesehen, so kann die Unterbringung kompletter Mikrorechner auf jeweils einer einzigen Leiterplatte ein Mittel sein, die Materialökonomie wirksam zu verbessern, da Paneelrahmen, Rückverdrahtungs-Leiterplatten sowie die meisten Steckverbinder entfallen und auch weniger Leiterplatten-Basismaterial sowie weniger Schaltkreise benötigt werden.

1.4. Koppelprinzipien zwischen mehreren Mikrorechnern

1.4.1. E-A-Kopplung

Mehrere komplette Mikrorechner können über E-A-Anschlüsse miteinander verbunden werden. Die Verbindungen können parallel oder seriell organisiert sein. Beispielsweise sind beim Mehrrechnersystem K 1520 die Verbindungen parallel organisiert [3] (Bild 6).

1.4.2. Gemeinsamer Speicher

Gemäß Bild 7 sind mehrere Mikrorechner an einen gemeinsamen Speicher angeschlossen, der von jedem der Mikrorechner aus als Teil des jeweiligen Adressenraumes zugänglich ist.

1.4.3. Gemeinsamer Bus

Es ist ein Bussystem vorgesehen, an das die einzelnen Mikrorechner in einheitlicher Weise angeschlossen sind (Bild 8).

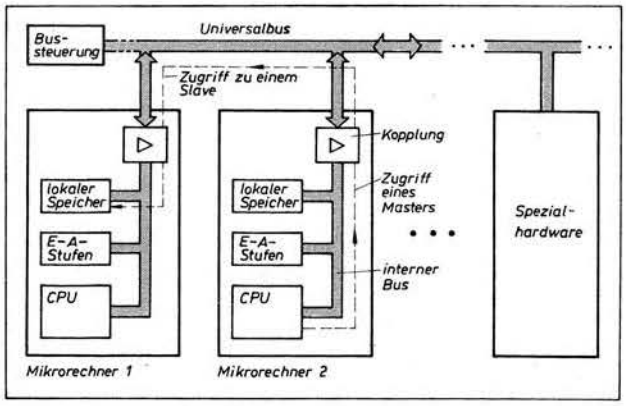


Bild 8: Multimikrorechnerkopplung durch gemeinsamen Bus

1.4.4. Auswahlkriterien

Die Auswahl wird wesentlich durch den Anwendungsfall bestimmt, wobei die zu implementierende Softwarekonzeption und das gewünschte Realzeitverhalten entscheidenden Einfluß haben.

Die Kopplung über E-A-Ports erfordert sehr geringe Hardwareaufwendungen, bedingt aber eine softwaremäßige Steuerung der Kopplungsprozeduren und des Datenaustausches. Jeder Mikrorechner muß in seinem Speicher alle Programme haben, die zum aktuellen Zeitpunkt lauffähig sein müssen. Der Zugriff zu Programmen und Datenbereichen in anderen Mikrorechnern ist nur durch Transportoperationen möglich; die Verarbeitungsbefehle können nicht auf Daten angewendet werden, die in Speicherbereichen anderer Mikrorechner stehen.

Auf einen gemeinsamen Speicher sind an sich beliebige Zugriffe der angeschlossenen Mikrorechner möglich. Ein echter sternförmiger Anschluß (Ausgestaltung der Hardware gemäß Bild 6) bringt einige technische Probleme mit sich. Dem Speicher muß ein Vermittlungsnetzwerk vorgeschaltet werden; dieses hat bei k Anschlußleitungen je Mikrorechner und n Mikrorechnern k · n Kontakte. Demzufolge ist die sternförmige Konfiguration wenig realisiert worden.

Ein gemeinsamer Speicher wird in der Literatur auch eher als Mailbox zur Kommunikation zwischen Softwareprozessen angesehen (geringe Speicherkapazität ausreichend), denn als Mittel zur Unterbringung größerer gemeinsamer Dateien bzw. Programme. Durch ein gemeinsames Bussystem für alle Mikrorechner wird der gemeinsame Speicher technisch praktikabel.

Der nächste Schritt besteht darin, Teile des lokalen Speichers in jedem Mikrorechner für Buszugriffe zugänglich zu machen, ohne daß dadurch die internen Zugriffsmöglichkeiten eingeschränkt werden (Dual Port Memory, s. Bild 9).

Der globale Adressenraum des so gebildeten Multimikrorechnersystems umfaßt die entsprechend zugänglichen Speicherbereiche der einzelnen Mikrorechner sowie eventuell angeschlossene zusätzliche Speichereinrichtungen. Damit gibt es praktisch keine Beschränkung des Zugriffs. Es ist möglich, daß jeder Mikrorechner beliebige Zugriffe zum globalen Adressenraum ausführen kann, so daß sich die verfügbare Speicherkapazität gut ausnutzen läßt. In der Regel werden die Mikrorechner technisch in einheitlicher Weise realisiert sein, aber verschiedene Aufgaben auszuführen haben. Das hat zur Folge, daß die lokalen Speicher unterschiedlich belegt sind. Datenbereiche bzw. Programme, die in einem Mikrorechner keinen Platz mehr finden, können in freien Speicherbereichen anderer Mikrorechner untergebracht werden. Der Zugriff dazu ist nur unwesentlich langsamer als der zum lokalen Speicher. Bei U-880-Systemen werden auch Blocktransporte bedeutend schneller ausgeführt, da dafür LDIR/LDDR-Befehle benutzt werden können anstelle der OUTIR/INIR-Befehle bei einer E-A-Kopplung.

Der Vorteil der Universalität wird durch höhere Hardwareaufwendungen erkauft. Es ist ein Bussystem mit „Multi Master Capability" vorzusehen (jeder der angeschlossenen Mikrorechner muß den Bus als Master belegen können), und in jeder der angeschlossenen Einrichtungen müssen die Schaltmittel für die Buskopplung vorgesehen werden. Weiterhin ist es unumgänglich, für die Probleme der gegenseitigen Zugriffsverriegelung, der Prioritätssteuerung usw. besondere Schaltmittel anzuordnen. Die Inbetriebnahme und Testung eines derartigen Systems bringt weitere Schwierigkeiten mit sich. Es ist deshalb verständlich, daß manche Autoren eine lose Kopplung bevorzugen [4]. Dabei sollen die Mikrorechner über intelligente Steuerschaltkreise an den gemeinsamen Verbindungsweg angeschlossen sein.

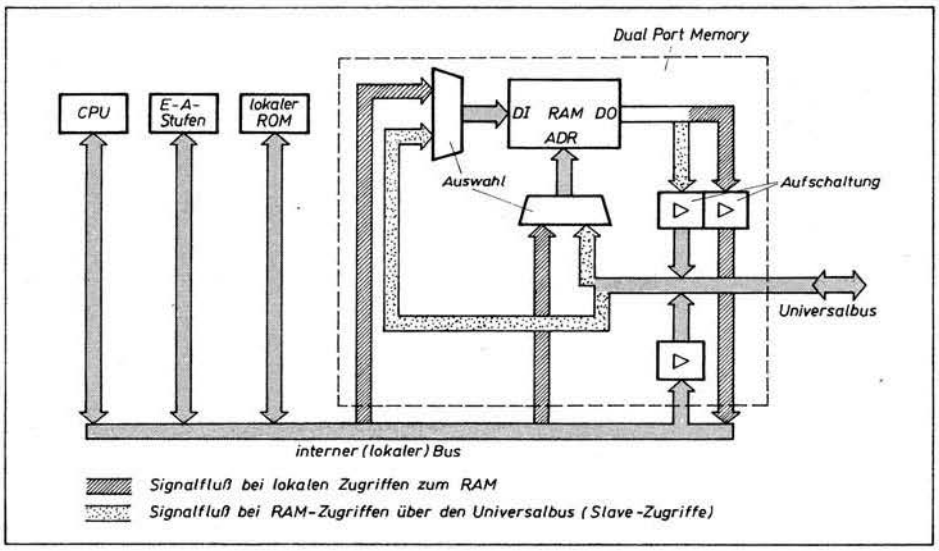


Bild 9: Prinzip eines Mikrorechners mit „Dual Port Memory"

Um ein stark gekoppeltes System durch ein lose gekoppeltes zu ersetzen, müßte der Stand der Technik es ermöglichen, ausreichende Verarbeitungsleistungen und Speicherkapazitäten in jedem Mikrorechner vorzusehen, die eine Aufteilung von Ressourcen zwischen den Mikrorechnern erübrigen. Künftige Entwicklungen dürften dies gewährleisten.

1.5. Einsatz spezieller Hardware

Die Verarbeitungsleistung von Multimikrorechnersystemen kann für bestimmte Aufgaben erhöht werden, indem die Mikrorechner durch spezielle Hardwareeinrichtungen ergänzt werden. Oft ermöglicht eine Kombination aus Mikrorechneranordnungen und (relativ einfacher) Sonderhardware überhaupt erst kostengünstige Alternativen zu ausgesprochenen Spezialösungen.
Als spezielle Hardware bezeichnet man dabei sowohl einfache Zusatzschaltungen als auch komplexe, weitgehend autonom arbeitende Funktionsmoduln.

Es gibt zwei prinzipielle Einsatzgebiete:

- Anschluß peripherer Einrichtungen bzw. Interfaces
- Beschleunigung interner Programmabläufe.

Der Anschluß peripherer Einrichtungen bzw. Interfaces ist unumgänglich und erfordert in jedem Fall einen gewissen Aufwand an spezieller Hardware. Im einfachsten Fall (wenn das betreffende Interface direkt an E-A-Schaltkreise eines Mikrorechners angeschlossen ist) handelt es sich lediglich um die entsprechenden Entkopplungs- und Anpassungsschaltungen.
Umfangreichere Hardware in diesem Zusammenhang kann zwei Aufgaben erfüllen, nämlich die Beschleunigung von Abläufen und die Erleichterung der Programmierung.
Bei der Beschleunigung von Abläufen ist davon auszugehen, daß die maximale Datenrate durch die Eigenschaften der verwendeten Mikroprozessorfamilie definiert ist. Spezielle Hardware kann lediglich dazu beitragen, daß diese Datenrate tatsächlich ausgenutzt werden kann.
Mittel zur Erleichterung der Programmierung sind vorzusehen, wenn Verarbeitungszeit oder Speicherzeit nicht in ausreichendem Maße verfügbar sind.

Beispiel 3

Ein peripheres Gerät benötigt Ausgabezeichen in einem Kode B; der Mikrorechner verarbeitet normalerweise den Kode A. Die Kodewandlung mit Hilfe des Programmes über eine Tabelle ist an sich trivial. Die Geschwindigkeit einer derartigen Ausgabe-Routine (in Schleifenform) ist allerdings begrenzt. Schaltet man gemäß Bild 10 einen ROM zwischen PIO und Geräteinterface, so wird das Programm von der Kodewandlung entlastet, der Speicherplatz für die Tabelle und die Wandlungsroutine wird eingespart, und die Ausgabe kann für Blöcke mit einer Länge bis zu 256 Bytes mit einem OUTIR-Befehl programmiert werden, wodurch die maximale Datenrate erreichbar ist.
Allerdings kostet jede Zusatzhardware zusätzlichen Aufwand, so daß sich eine detaillierte Untersuchung stets lohnt. Im Beispiel kann auf den ROM verzichtet werden, wenn genügend RAM-Speicherplatz zur Verfügung steht und wenn das Zeitintervall zwischen der Übertragung von zwei Datenblöcken hinreichend groß ist. Dann kann der Ausgabedatenblock im RAM des Mikrorechners mit einer

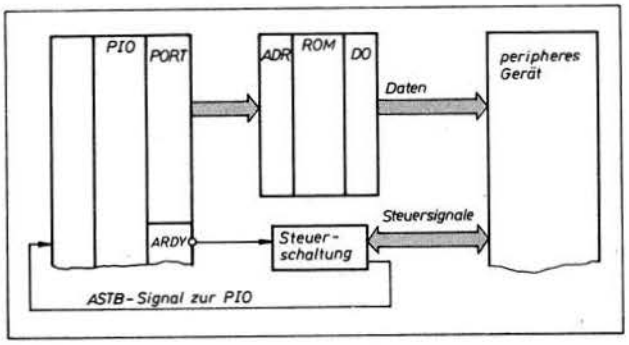


Bild 10: Anschluß eines peripheren Gerätes an eine PIO mit zwischengeschaltetem ROM zur Kodewandlung

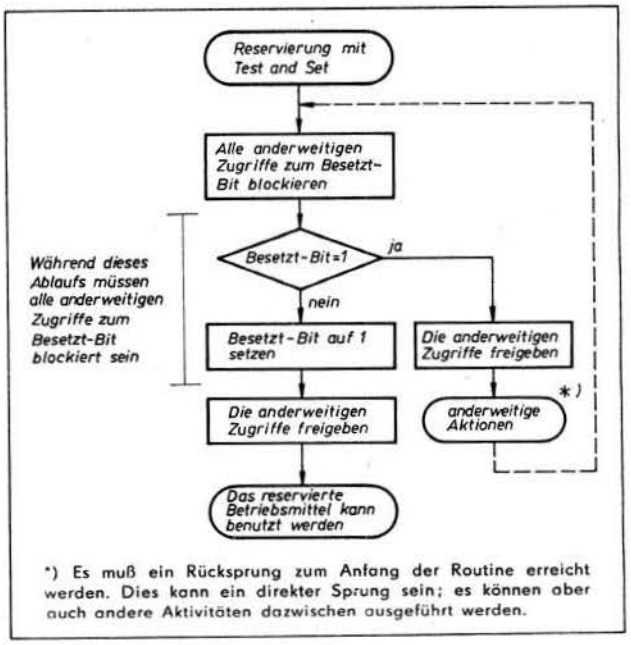


Bild 11: Ablauf der Reservierung eines Betriebsmittels mit „Test and Set"

Wandlungsroutine aufgebaut und anschließend mit dem OUTIR-Befehl ausgegeben werden.
Eine weitere Geschwindigkeitsgrenze bei E-A-Anschlüssen ist dadurch gegeben, daß praktisch nur ein Austausch von Bytes möglich ist. Selektive Reaktionen, etwa auf Belegungen einzelner Steuerleitungen, erfordern entsprechende Softwareroutinen, so daß sich Befehlsfolgen der Art IN (Steuersignale) . . . Abfrageroutine, Entscheidung . . .OUT (Reaktionssignale) ergeben. Sind die damit erreichbaren Zeiten dem zu betreibenden Interface nicht angemessen, so müssen Schaltmittel vorgesehen werden, die in der Lage sind, die zeitkritischen Abläufe autonom zu steuern. Derartige Anordnungen werden zweckmäßigerweise an den Universalbus des Multimikrorechnersystems oder an den internen Bus eines Mikrorechners angeschlossen.
Einrichtungen zur Beschleunigung interner Programmabläufe sind beispielsweise in Form spezieller Arithmetikprozessoren bekannt, die an das Bussystem eines Mikrorechners angeschlossen werden. Es kann sich aber auch um Einrichtungen handeln, die zur Benutzung durch mehrere Mikrorechner vorgesehen sind.
Ein wesentlicher Aspekt derartiger Zusatzeinrichtungen besteht darin, daß sie für die Software als Betriebsmittel anzusehen sind und entsprechend verwaltet werden müssen. Dies kann eine triviale Aufgabe sein, wenn die Art der Zusatzhardware einen konkurrierenden Zugriff verhindert oder wenn es im gesamten System tatsächlich nur einen Prozeß gibt, der die Zusatzhardware benötigt. Ansonsten sind für das Reservieren und Freigeben der Zusatzeinrichtungen programm- oder gerätetechnische Mittel vorzusehen. Prinzipielle Möglichkeiten:

- Verwaltung
- Blockierung.

Die Verwaltung erfordert, daß stets vor der Benutzung der Spezialhardware ein Reservierungsablauf und nach der Benutzung ein Freigabeablauf ausgeführt werden. Das bedingt folgende Programmstruktur:
RESERVE . . . Zugriff zur Spezialhardware . . . RELEASE.
Der einfachste Reservierungsablauf ist der vom Typ „Test and Set". Dabei wird für jede Einrichtung ein Besetzt-Bit verwaltet, wie dies im Bild 11 veranschaulicht ist. Die Freigabe besteht im Ausschalten des Besetzt-Bits. Eine Blokkierung wird dadurch erreicht, daß während der Belegung der Spezialhardware verhindert wird, daß ein anderer Prozeß, der die Hardware ebenfalls belegen könnte, Laufzeit erhält.
Laufen die konkurrierenden Prozesse auf einem Mikrorechner ab, so reicht u. U. bereits ein Verhindern von Unterbrechungen (DI-Befehl) zur Blockierung aus. Sind mehrere Mikrorechner beteiligt, so kann das Verfahren darin bestehen,

Tafel 1: Verwaltung und Blockierung

Methode	Vorteile	Nachteile
Verwaltung	Prozesse, die im betreffenden Zeitraum nicht um die Spezialhardware konkurrieren, werden nicht verzögert	Software-Overhead; es vergeht eine gewisse Zeit (durch die Reservierungsroutine), bis das Betriebsmittel benutzt werden kann
Blockierung	kein Software-Overhead	Prozesse, die in den blockierten Einrichtungen aktiv sind, werden verzögert

die konkurrierenden Mikrorechner durch spezielle Hardware in den WAIT-Zustand zu versetzen.
Die Vor- und Nachteile beider Verfahren sind in der Tafel 1 dargestellt. Es ist ersichtlich, daß das Verwaltungsprinzip in den Fällen besser geeignet ist, in denen das Betriebsmittel (Spezialhardware) jeweils längere Zeit belegt wird. Bei nur kurzzeitigen Belegungen kann hingegen das Blockierungsverfahren zeitliche Vorteile bringen.
Die Blockierung konkurrierender Prozesse ist zudem Voraussetzung für das Verwaltungsverfahren: Wie aus Bild 11 ersichtlich ist, muß für den Verwaltungsablauf gewährleistet sein, daß zwischen Abfragen und Setzen des Besetzt-Bits andere Einrichtungen nicht darauf zugreifen können. Somit sind die entsprechenden Mittel obligatorisch vorzusehen, und es ist eine Frage der konkreten Anwendung, ob diese in direkter Weise oder zur Implementierung einer softwaremäßigen Verwaltung eingesetzt werden.

2. Bussystem

2.1. Prinzipien

Das Bussystem soll mehrere Mikrorechner und erforderlichenfalls spezielle Funktionsmoduln zu einem System verbinden, wobei im Rahmen eines definierten Adressenraumes Zugriffe von beliebigen Einrichtungen nach beliebigen Einrichtungen möglich sein sollen. Die Übertragung ist nach dem Master-Slave-Prinzip organisiert, d. h., daß zu einem beliebigen Zeitpunkt jeweils nur eine Einrichtung (der aktuelle Master) den Bus belegen und zu jeweils einer anderen Einrichtung (dem aktuellen Slave) zugreifen kann. Im Sinne der Universalität soll es das Bussystem ermöglichen, daß prinzipiell jede der angeschlossenen Einrichtungen Master werden kann (engl. multi master capability).

Ein solches Bussystem muß zur Ausführung folgender Operationen eingerichtet sein:

- Auswahl des aktuellen Masters
- Auswahl des aktuellen Slave
- Steuerung von Funktion und Übertragungsrichtung (Lesen, Schreiben usw.)
- Übertragung von Adressensignalen vom Master zum Slave
- Übertragung von Datensignalen vom Master zum Slave (beim Schreiben) oder vom Slave zum Master (beim Lesen)
- Beenden des Buszyklus.

Dazu kommen noch einige Sonderfunktionen wie Prioritätssteuerung, Fehlersignalisierung, Einstellen besonderer Zustände usw.
Für die Implementierung der einzelnen Funktionen des Bussystems gibt es jeweils mehrere prinzipielle Alternativen. Eine ausführliche systematische Übersicht über die wesentlichen Lösungswege ist in [5] enthalten. Die folgenden Betrachtungen sind an dieser Darstellung orientiert.

2.1.1. Auswahl des Masters

Eine Einrichtung, die Master werden will, muß die Kontrolle über den Bus anfordern, und es müssen Schaltmittel vorhanden sein, die zu einem gegebenen Zeitpunkt einer der anfordernden Einrichtungen die Kontrolle über den Bus zusprechen. Dabei müssen Zugriffskonflikte so gelöst werden, daß der Datendurchsatz des Bussystems möglichst wenig beeinträchtigt wird. Nach der Anordnung der entsprechenden Schaltmittel unterscheidet man zwischen zentralisierter und dezentraler Bussteuerung. Eine zentralisierte Bussteuerung ist dadurch gekennzeichnet, daß eine zentrale Funktionseinheit die Busanforderungen erkennt und die Buszyklen den anfordernden Einrichtungen zuteilt. Bei dezentralen Bussteuerungen sind diese Schaltmittel auf die einzelnen Einrichtungen verteilt.
Für das Erkennen von Busanforderungen und die Zuteilung von Busbelegungen (Zyklen) gibt es drei verschiedene Prinzipien, zwischen denen auch Kombinationen möglich sind:

Literatur

[1] Bunata, T.; Huber, B.: 16-Bit-Mikroprozessor für Echtzeit-Multitask-Betrieb. Elektronik, München 32 (1983) 7, S. 53–57
[2] Geyer, J.: 32-Bit-Mikrocomputer besitzt neuartige Architektur. Elektronik, München 30 (1981) 5, S. 59–66
[3] Wollenberg, G.; Kehrer, E.: Multimikrorechnerkopplung über Ein- und Ausgabeports. radio fernsehen elektronik, Berlin 29 (1980) 8, S. 489–494
[4] Faggin, F.: VLSI verändert Computer-Strukturen. Elektronik, München 27 (1978) 12, S. 57-61, 112

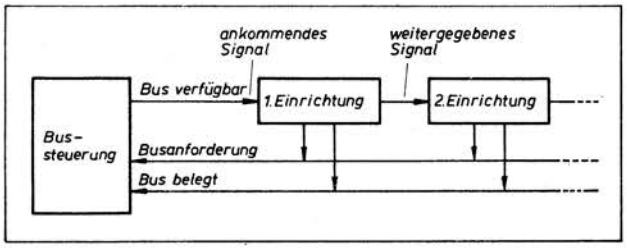


Bild 12: Busvermittlung nach dem Daisy-Chain-Prinzip

1. Daisy Chaining
2. Abfrage (Polling)
3. unabhängige Anforderungen.

Diese drei Möglichkeiten werden zunächst für den Fall einer zentralisierten Bussteuerung betrachtet. Bild 12 zeigt dazu das Daisy-Chain-Prinzip.

Jede Einrichtung kann eine Anforderung durch Belegen der Leitung „Busanforderung" signalisieren. (Diese Leitung ist als Wired-OR beschaltet.) Erkennt die Bussteuerung, daß diese Leitung aktiv ist, so gibt sie ein Signal auf der Leitung „Bus verfügbar" ab. Einrichtungen, die keine Busanforderung gestellt haben, leiten dieses Signal zur jeweils nächsten Einrichtung weiter. Trifft das Signal an einer Einrichtung ein, die eine Busanforderung gestellt hat, so wird es nicht weitergeleitet. Die Einrichtung erregt die Bus-belegt-Leitung (ebenfalls als Wired-OR ausgeführt), schaltet das Anforderungssignal aus und beginnt die Datenübertragung. Nach Beendigung des Buszugriffs wird das Bus-belegt-Signal abgeschaltet. Damit ist der Bus wieder frei, und eine neue Vermittlung kann beginnen.

Bild 13 zeigt das Prinzip der Abfrage (Polling). Die Einrichtungen stellen ihre Busanforderungen in derselben Weise, wie dies bereits beschrieben wurde. Statt durch das beschriebene Durchreichen des Bus-Verfügbar-Signals wird die Einrichtung, die den Bus zugesprochen bekommt, dadurch ermittelt, daß die Bussteuerung die einzelnen Einrichtungen nacheinander daraufhin abfragt, ob sie eine Anforderung gestellt haben. Die erste Einrichtung, die auf diese Weise identifiziert wurde, erhält den Bus zugesprochen.

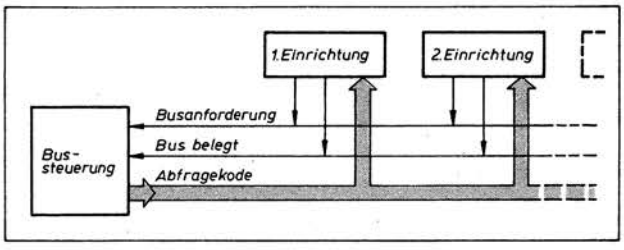


Bild 13: Busvermittlung nach dem Abfrageprinzip

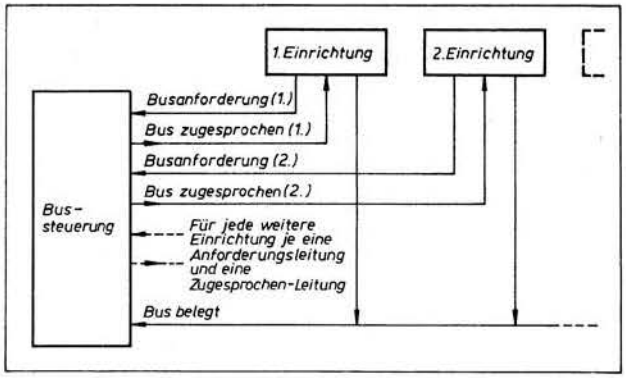


Bild 14: Busvermittlung nach dem Prinzip der unabhängigen Anforderungen

Das Prinzip der unabhängigen Anforderungen wird durch Bild 14 veranschaulicht. Jede Einrichtung hat eine eigene Busanforderungsleitung und erhält über eine eigene Bus-zugesprochen-Leitung den Bus zugeteilt. Die Bussteuerung wählt nach einem Prioritätsschema aus, welche Einrichtung aus der Menge derer, die eine Anforderung gestellt haben, den Bus zugesprochen bekommen soll und erregt die entsprechende Leitung. Daraufhin schaltet die betreffende Einrichtung ihre Anforderungsleitung ab und aktiviert für die Dauer der Datenübertragung die Bus-belegt-Leitung. Wird „Bus belegt" abgeschaltet, so schaltet die Bussteuerung die betreffende Bus-zugesprochen-Leitung ab und beginnt erneut mit der Vermittlung.

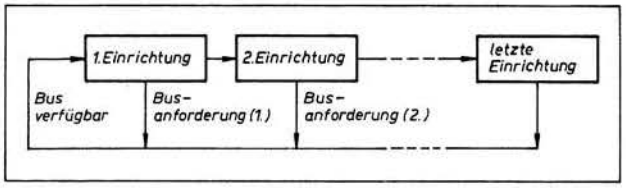


Bild 15: Dezentrale Daisy-Chain-Vermittlung (Pegel des Vermittlungssignals werden ausgewertet)

Diese Prinzipien lassen sich auch im Rahmen dezentraler Bussteuerungen realisieren. In der Anordnung nach Bild 15 fordert eine Einrichtung den Bus an, indem sie ihre Leitung „Busanforderung" erregt, wenn die entsprechende ankommende Bus-verfügbar-Leitung inaktiv ist. Wird ein Bus-verfügbar-Signal empfangen, so wird es von Einrichtungen, die keine Anforderung gestellt haben, weitergeleitet. Die erste Einrichtung mit Busanforderung leitet das Bus-verfügbar-Signal nicht weiter und hält die Busanforderung solange erregt, bis die Datenübertragung beendet ist. War die betreffende Einrichtung die einzige, die eine Anforderung gestellt hatte, so wird „Bus verfügbar" ebenfalls abgeschaltet. Anderenfalls halten die anderen Anforderungssignale die Leitung „Busanforderung" weiterhin aktiv, so daß das Bus-verfügbar-Signal zur nächsten anfordernden Einrichtung weitergereicht wird.

Es ist offensichtlich, daß bei diesem Vermittlungsprozeß Wettlauferscheinungen auftreten, so daß zu deren Vermeidung zusätzliche Maßnahmen nötig sind. Ein einfaches Mittel ist die Synchronisierung des Vermittlungsablaufes durch einen gemeinsamen Takt, der über das Bussystem an alle Einrichtungen verteilt wird.

In der Anordnung nach Bild 16 werden keine statischen Belegungen, sondern Signalwechsel durch die angeschlossenen Einrichtungen weitergeleitet bzw. nicht weitergeleitet. Wird von einer Einrichtung ein Signalwechsel (L-H oder H-L) auf der ankommenden Bus-verfügbar-Leitung erkannt, und die Einrichtung benötigt den Bus nicht, so wird der Signalwechsel zur nächsten Einrichtung weitergegeben (d. h., es wird auf der abgehenden Bus-verfügbar-Leitung ebenfalls ein Signalwechsel ausgelöst). Damit oszilliert das Bus-verfügbar-Signal ständig, wenn keine Anforderungen bestehen. Die erste Einrichtung, die den Bus benötigt und einen Signalwechsel empfängt, gibt diesen nicht weiter (die abgehende Bus-verfügbar-Leitung bleibt statisch auf ihrem Potential) und belegt den Bus. Nach Ende der Datenübertragung veranlaßt die Einrichtung einen Signalwechsel auf der abgehenden Bus-verfügbar-Leitung.

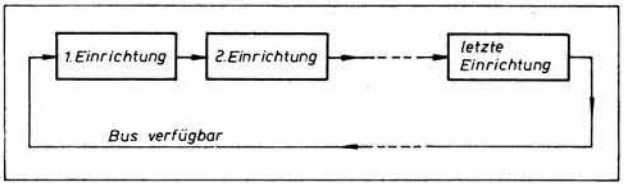


Bild 16: Dezentrale Daisy-Chain-Vermittlung (Änderungen des Vermittlungssignals werden ausgewertet)

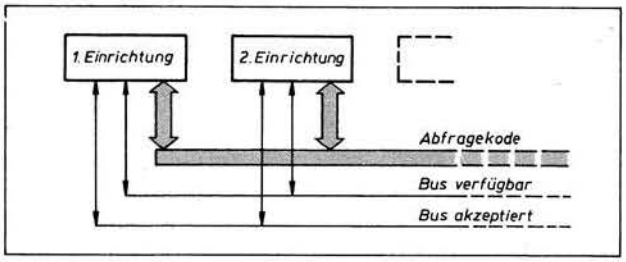


Bild 17: Abfrageprinzip bei dezentraler Bussteuerung (eine der Einrichtungen ist als abfragende Einrichtung konfiguriert)

Bild 17 veranschaulicht das Abfrageprinzip bei dezentraler Bussteuerung. Eine Einrichtung, die Buszugriffe vermitteln will, belegt die Abfragekode-Leitungen mit einem entsprechenden Wert und schaltet „Bus verfügbar" ein. Hat eine andere Einrichtung, in der eine Busanforderung anhängig ist, denselben Kode, so antwortet diese mit „Bus akzeptiert". Die steuernde Einrichtung gibt die Leitungen „Abfragekode" und „Bus verfügbar" frei, und die zweite Einrichtung beginnt mit der Datenübertragung. Empfängt die steuernde Einrichtung kein Akzeptiertsignal, so verändert sie den Abfragekode gemäß einem Vermittlungsalgorithmus und startet eine erneute Abfrage.

Bei diesem Prinzip muß eine der Einrichtungen als steuernde Einrichtung initialisiert werden. Damit entspricht es im wesentlichen der zentralisierten Abfragesteuerung. Der Vorteil besteht darin, daß bei Ausfall der einen abfragenden Einrichtung eine andere diese Aufgabe übernehmen kann (dazu sind zusätzliche Busleitungen zur Fehlersignalisierung, zur Steuerung des Umschaltens usw. erforderlich).

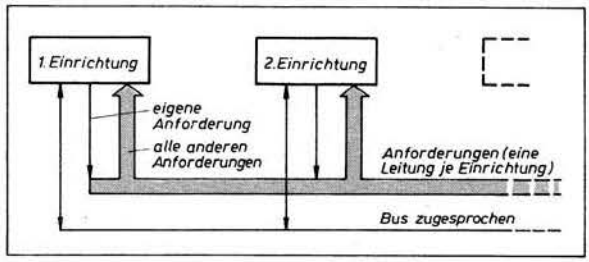


Bild 18: Prinzip der unabhängigen Anforderungen bei dezentraler Bussteuerung

Gemäß Bild 18 ist auch beim Prinzip der unabhängigen Anforderungen eine dezentrale Steuerung möglich. Jede Einrichtung, die den Bus belegen will, erregt ihre eigene Anforderungsleitung. Alle Einrichtungen können alle Anforderungsleitungen auswerten. Die Einrichtung, die sich selbst zu einem gegebenen Zeitpunkt als die mit der höchsten Priorität erkennt, belegt den Bus, indem sie die Bus-zugesprochen-Leitung erregt. Daraufhin schalten alle anderen Einrichtungen ihre Anforderungssignale aus. Nach dem Ende der Datenübertragung wird „Bus zugesprochen" ausgeschaltet, worauf die Vermittlung von neuem beginnt.

Diskussion

Diese Erläuterungen veranschaulichten nur das jeweilige Prinzip. In der Praxis werden zusätzliche Busleitungen und Modifikationen des Signalspiels eingeführt werden müssen, um Wettlauferscheinungen zu vermeiden, elektrischen Anforderungen gerecht zu werden und auch das gegebene Schaltkreissortiment zweckmäßig einsetzen zu können.
Die Zeit zur Auswahl des Masters bestimmt (neben dem Ablauf der Datenübertragung) wesentlich den Durchsatz des Bussystems. Das dafür gewählte Prinzip hat entscheidenden Einfluß auf die Struktur des Bussystems im allgemeinen, auf die Anzahl der Steuerleitungen, die Aufwendungen zur Busankopplung in den einzelnen Funktionseinheiten und auf die Zuverlässigkeit.

Bei der Definition dieses Teiles des Bussystems müssen folgende Aspekte berücksichtigt werden:

- Schaltkreissortiment

  Müssen die Schaltmittel zur Bussteuerung mit SSI- und MSI-Schaltkreisen aufgebaut werden, stehen fertige LSI-Schaltkreise zur Verfügung, oder ist die Entwicklung spezieller LSI-Schaltkreise möglich? Dadurch werden die Aufwendungen und die obere Grenze der Arbeitsgeschwindigkeit bestimmt.

- Leistungsanforderungen

  Aus geforderter Datenrate (in Buszyklen/s) und gefordertem Wechselspiel zwischen Datenübertragung und Masterauswahl (neue Masterauswahl nach jedem übertragenen Byte oder nach längeren Datenblöcken) läßt sich die Zeit ermitteln, die zur Masterauswahl zur Verfügung steht.

- Systemstruktur

  Können viele bzw. alle Einrichtungen des Systems oder nur einige bzw. eine begrenzte Anzahl als Master aktiv werden? Muß das System erweiterungsfähig sein? Diese Betrachtungen sind erforderlich, um zu bestimmen, inwiefern sich Verfahren mit festem Kodeumfang (Abfragekodes, Prioritäten o. ä.) prinzipiell eignen, wieviel Abfragekodes bzw. Prioritätsniveaus vorgesehen werden müssen und wieviel Einrichtungen beim Daisy-Chain-Prinzip vom Auswahlsignal durchlaufen werden.

- Systemgröße

  Ist das System kompakt (elektrisch kurze Busleitungen) oder verteilt (lange Busleitungen) aufgebaut?

- Technologische Realisierung

  Wieviel Kontakte je Einrichtung stehen für das Bussystem zur Verfügung? Wie ist die Leitungsführung des Bussystems realisiert (gedruckte Verbindungsleiterplatte, Flachbandkabel o. ä.)? Hat man sich für eine bestimmte Organisationsform der Datenübertragung entschieden, so bestimmt die verbleibende Kontaktzahl oft darüber, welche Prinzipien der Masterauswahl realisiert werden können (stehen nur wenig Kontakte zur Verfügung, so kommt nur eine Form des Daisy Chaining in Frage). Starre Leitungsführungen (gedruckte Verbindungsleiterplatten mit regulärer Struktur, Flachbandkabel) erschweren die Realisierung individueller Anschlüsse, z. B. von Prioritätsleitungen, die von den einzelnen Einrichtungen zu einer zentralen Bussteuerung führen.

- Zuverlässigkeitsanforderungen

  Muß der Betrieb auch bei Ausfall einer beliebigen Einrichtung fortgesetzt werden können oder ist die Erkennung von Busausfällen (mit nachfolgender Fehlerreaktion) ausreichend? Dieser Aspekt bestimmt wesentlich über die Wahl zwischen zentraler und dezentraler Bussteuerung. Bei extremen Zuverlässigkeitsanforderungen hat man die Wahl zwischen Zentralisierung mit Parallelredundanz (d. h. praktisch doppelter Aufbau) oder dezentraler Organisation mit entsprechenden Rettungsvorkehrungen (automatische Umschaltung im Fehlerfall o. ä.). Die Entscheidung sollte nicht auf Grund einer isolierten Betrachtung des Bussystems getroffen werden, sondern Teil der Konzipierung des gesamten Systems sein (beispielsweise nützt ein extrem zuverlässiges Bussystem nicht viel, wenn bei Ausfall einer bestimmten Funktionseinheit wesentliche periphere Einrichtungen nicht mehr angesteuert werden können). Somit ist zunächst das Ausfallverhalten des Gesamtsystems zu bestimmen und die Redundanzstruktur festzulegen. Daraufhin sollte das Bussystem definiert werden.

- Betriebsbedingungen

  Ist es erforderlich zu gewährleisten, daß einzelne Einrichtungen aus dem System entfernt werden können? Ein einfaches und recht wirkungsvolles Mittel der Fehlersuche besteht darin, verdächtige Funktionseinheiten aus dem System zu entfernen und (mit entsprechenden Testprogrammen) die Funktionsfähigkeit des verbleibenden Systems zu überprüfen. Dabei müssen Daisy-Chain-Leitungen in der Regel überbrückt werden. Ein ähnliches Problem entsteht in der Fertigung, wenn die Systeme in unterschiedlicher Weise mit Funktionseinheiten bestückt sind. In welchem Maße muß der Informationsaustausch über den Bus beobachtbar und steuerbar sein (Fehlersuche, Testen von Software)? Bussysteme mit dezentraler Steuerung bieten in dieser Hinsicht größere Schwierigkeiten. Bei zentralen Steuerschaltungen ist es weniger problematisch, entsprechende Mittel (z. B. Ablaufspeicher, Vergleichseinrichtungen o. ä.) fest einzubauen oder Anschlüsse für externe Prüfmittel, wie etwa Logikanalysatoren, vorzusehen.

Für die zentralisierte Form spricht die geringere Bauelementezahl bei Verwendung von LSI- und MSI-IS. Wettlauferscheinungen lassen sich auf einfache Weise und mit geringer Zeiteinbuße verhindern. Die Einschwingzeiten sind kürzer, und Toleranzprobleme sind leichter beherrschbar. Mittel für Überwachung und Testen sind zentral leicht anzuordnen bzw. extern anzuschließen.
Eine dezentrale Bussteuerung kann vorteilhaft sein, wenn die Steuerschaltungen mit LSI-Bauelementen realisiert werden. Es ist damit eher möglich, Vermittlungsschaltungen in jeder Funktionseinheit vorzusehen. Dies kann zu einer weitergehenden Standardisierung beitragen und eine freizügigere Konfigurierbarkeit ermöglichen. Mit einem physisch verteilten Aufbau wird eine Vielzahl von Steuerleitungen zwischen den Funktionseinheiten und einer zentralen Bussteuerung vermieden.
Zwischen den Alternativen sind Kombinationen möglich, etwa zwischen dezentraler Vermittlung und zentralisierter Fehlerüberwachung.
Bisweilen handelt es sich auch darum, daß die Schaltmittel zwar dezentral angeordnet sind (auf jeder Funktionseinheit), aber als zentralisierte Steuerung betrieben werden (nur eine Funktionseinheit ist als Bussteuerung initialisiert; s. a. Bild 17). Eine derartige Lösung, die namentlich bei LSI-Realisierung näher untersucht werden sollte, hat folgende Vorteile:

- Weitgehende Standardisierung (besondere Bussteuerbaugruppen sind nicht mehr erforderlich)
- freizügige Konfigurierbarkeit (jede Einrichtung kann als Bussteuerung deklariert werden)
- höhere Zuverlässigkeit (bei Ausfall einer Bussteuerschaltung steht entsprechende Redundanz auf anderen Funktionseinheiten zur Verfügung).

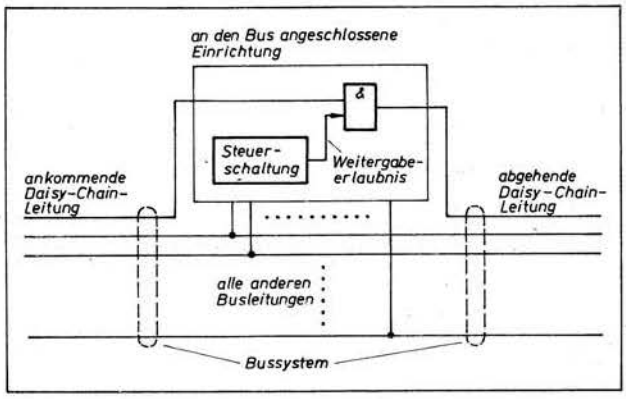


Bild 19: Anschluß von Busleitungen an eine Funktionseinheit

Weiterhin ist das Steuerprinzip für die Masterauswahl zu definieren. Dazu werden die beschriebenen Prinzipien im einzelnen betrachtet.

2.1.1.1. Daisy Chaining

Dieses Prinzip erfordert wenigstens eine Daisy-Chain-Verbindung und eine Weitergabeschaltung in jeder Funktionseinheit. Die anderen Steuerleitungen erfordern keine logische Trennung in den Funktionseinheiten (Bild 19).

Zur Vermeidung von Wettlauferscheinungen müssen entsprechende Maßnahmen vorgesehen werden (gemeinsamer Bustakt, Verzögerungsschaltungen in den Funktionseinheiten).

Dem Vorteil der geringen Anzahl von Steuerleitungen stehen entgegen:

- Die Priorität einer Funktionseinheit hängt von ihrer physischen Position am Bus ab (bei einer zentralisierten Steuerung dadurch ausgedrückt, wieviel Einrichtungen zwischen der Steuerung und der betreffenden Funktionseinheit angeordnet sind; bei dezentraler Steuerung ergibt sich ein Prioritätsverhalten nach Art des zyklischen Weiterschaltens).
- Fehler in Funktionseinheiten, die die Weitergabe des Daisy-Chain-Signals verhindern, führen zum Ausfall des gesamten Systems.
- Werden Funktionseinheiten aus dem System entfernt, müssen die Daisy-Chain-Anschlüsse überbrückt werden.
- Die Vermittlungszeit wächst mit der Länge des Bussystems und der Anzahl der angeschlossenen Einrichtungen.

Entscheidung für Daisy Chaining wegen

- relativ kostengünstiger und einfacher Systeme, die erweiterungsfähig sein sollen
- kompakten Aufbaus
- wenig Steuerleitungen
- mittlerer Vermittlungszeiten (etwa 200 ns...2 μs).

2.1.1.2. Abfrage (Polling)

Die Reihenfolge der Abfrage (und damit die Prioritätszuordnung) ist auf einfache Weise modifizierbar (durch entsprechendes Einstellen des Abfragezählers in der steuernden Einrichtung). Die physische Anschlußreihenfolge hat keinen Einfluß auf die Priorität. Wettlauferscheinungen treten praktisch nicht auf. Die Anzahl der anschließbaren Einrichtungen wird durch die Anzahl der Abfragekodeleitungen bestimmt; man kann aber an ein Bussystem mit n derartigen Leitungen 2^n Funktionseinheiten anschließen.

Werden Funktionseinheiten aus dem System entfernt, so sind keine besonderen Maßnahmen erforderlich, um das reduzierte System weiter betreiben bzw. testen zu können.

Nur Fehler, die dazu führen, daß die Bus-belegt-Leitung (Bild 13) ständig erregt bleibt, bewirken einen Ausfall des Systems.

Nachteilig ist, daß der Vermittlungsablauf relativ lange dauern kann (bei n Einrichtungen bis zu n Vermittlungstaktzyklen, im Mittel n/2).

Entscheidung für Abfrage wegen

- komplexer Systeme mit höheren Anforderungen an die Zuverlässigkeit
- Forderung nach flexibler Steuerung der Prioritätszuordnung
- längeren zulässigen Vermittlungszeiten ($> 1\ \mu$s).

2.1.1.3. Unabhängige Anforderungen

Dieses Prinzip ermöglicht eine maximale Geschwindigkeit der Vermittlung, bringt aber folgende Nachteile mit sich:

- Große Leitungs- und Kontaktanzahl, begrenzte Anzahl anschließbarer Einrichtungen (das Bussystem muß für jede Anschlußposition zwei Leitungen enthalten)
- Änderung der Prioritätszuordnung schwieriger als beim Abfrageprinzip (statisch durch Ändern der Verdrahtung oder des Prioritätsnetzwerkes, dynamisch durch Umsteuerung des Prioritätsnetzwerkes).

Entscheidung für unabhängige Anforderungen wegen

- Systemen hoher Leistungsfähigkeit
- kompakten Aufbaus
- kurzer Vermittlungszeiten (max. 200 ns)
- Begrenzung der maximal zu installierenden Funktionseinheiten (etwa 8 bis 16); spätere Erweiterungen können von vornherein ausgeschlossen werden.

Die Bedingungen im praktischen Betrieb sind ähnlich günstig wie im Falle des Abfrageprinzips (Funktionseinheiten können problemlos entfernt werden, nur wenige Arten von Fehlern führen direkt zum Ausfall des Systems).

2.1.2. Steuerung der Datenübertragung

Für die Steuerung des Datenaustausches zwischen Master und Slave müssen Steuersignalleitungen vorgesehen und Steuerfolgen definiert werden. Steuersignale können als Impulse definierter Länge abgegeben (sie werden als Strobesignale bezeichnet) oder in ihrer Länge durch eine Verriegelung mit Antwortsignalen bestimmt werden (engl. interlocking). Dazu werden im folgenden einige Varianten vorgestellt.

2.1.2.1. Strobeimpuls von der sendenden Einrichtung (Bild 20)

Die sendende Einrichtung legt die Information auf den Bus und gibt nach einem gewissen Zeitintervall einen Strobeimpuls ab, der die Information als gültig kennzeichnet. Die Verzögerung muß so groß gewählt werden, daß die Daten an der empfangenden Einrichtung (unter Berücksichtigung der Laufzeiten durch die Buskoppelbaustufen) sicher angekommen sind.

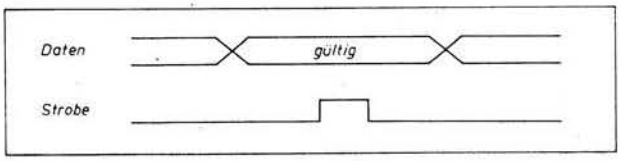


Bild 20: Informationsübertragung mit Strobeimpuls von der sendenden Einrichtung

2.1.2.2. Strobeimpuls mit Antwortsignal (engl. non-interlocked handshaking, Bild 21)

Die Informationsübertragung an sich läuft genau so ab wie in der Variante nach Abschnitt 2.1.2.1. Sie wird aber nicht nach einem definierten Zeitintervall beendet, sondern erst, wenn die empfangende Einrichtung einen Quittungsimpuls abgegeben hat.

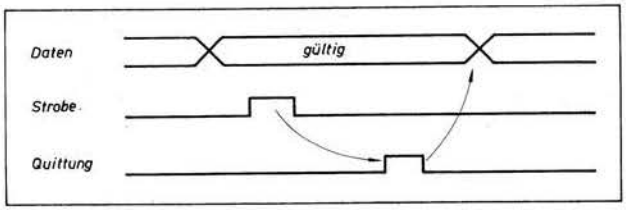


Bild 21: Informationsübertragung mit Strobeimpuls und Quittungsimpuls

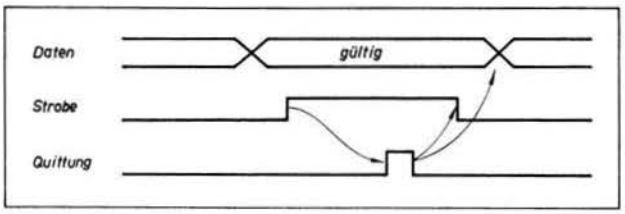


Bild 22: Informationsübertragung mit teilweiser Verriegelung

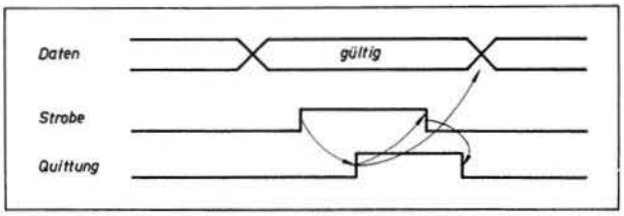


Bild 23: Informationsübertragung mit vollständiger Verriegelung

2.1.2.3. Teilweise Verriegelung des Strobesignals mit dem Antwortsignal (engl. half interlocked handshaking, Bild 22)

Das Strobesignal wird nicht als Impuls definierter Länge gebildet, sondern so lange aktiv gehalten, bis der Quittungsimpuls von der empfangenden Einrichtung eintrifft.

2.1.2.4. Vollständige Verriegelung (engl. fully interlocked handshaking, Bild 23)

In diesem Fall wird auch das Quittungssignal nicht als Impuls definierter Länge abgegeben, sondern so lange aktiv gehalten, bis die gewünschte Wirkung (das Abschalten des Strobesignals) eingetreten ist.

Diskussion

Die Variante nach 2.1.2.1. erfordert für jede Übertragungsrichtung (vom Master zum Slave und umgekehrt) nur jeweils eine Strobeleitung. Es gibt keine Antwortsignale von der jeweils empfangenden Einrichtung. Das hat den Vorteil, daß hohe Geschwindigkeiten (Datenraten) möglich sind, da die Signallaufzeit über die Busleitungen nur einmal je Zyklus wirksam wird. Dieser Vorteil läßt sich aber nur dann nutzen, wenn alle Einrichtungen am Bussystem für die gewünschte maximale Datenrate ausgelegt sind. Es ist jedoch bei vielen Bussystemen ausdrücklich wünschenswert, daß Einrichtungen mit unterschiedlichen (bisweilen stark variierenden) Arbeitsgeschwindigkeiten bzw. Datenraten angeschlossen werden können, ohne daß ein Zusatzaufwand für Pufferzwecke erforderlich ist.
Des weiteren kann man nur dann unbedenklich auf Quittungssignale von der empfangenden Einrichtung verzichten, wenn von der Auslegung des gesamten Systems her klar ist, daß angebotene Daten stets akzeptiert werden. Im allgemeinen kann eine empfangende Einrichtung auch anderweitig auf angebotene Daten reagieren, z. B. durch Zurückweisung (sie ist momentan nicht in der Lage, Daten zu empfangen) oder durch Fehlersignalisierung (empfangene Daten sind inkorrekt). Für diese Fälle sind Quittungssignale von Vorteil. Prinzipiell haben alle nicht vollständig verriegelten Steuerabläufe den Nachteil, daß bei zu langen Impulsen oder bei einem Dauersignal anstelle eines Impulses das Signalspiel am Bus zum Erliegen kommt. Die Definition der zulässigen Impulsbreiten wird durch die Verzögerungszeiten der Signale über das Bussystem beeinflußt. Sie kann problematisch werden, wenn einerseits diese Zeiten stark variieren (sehr unterschiedliche Busleitungslängen möglich, große Abweichungen der Verzögerungszeiten in den einzelnen Funktionseinheiten) und andererseits gewisse Mindestdatenraten erreicht werden müssen.
Bei vollständig verriegelten Steuerabläufen können die einzelnen Funktionseinheiten prinzipiell jede beliebige interne Geschwindigkeit haben, und der Bus darf beliebig lang sein. Für die Ermittlung der maximal möglichen Datenrate ist allerdings die vierfache Kabellaufzeit (unter Berücksichtigung der Koppelbaustufen) des Bussystems in Rechnung zu stellen.

2.1.2.5. Weitere Varianten

Für ein konkretes Bussystem können die diskutierten Prinzipien modifiziert und kombiniert werden. Einige Möglichkeiten sind:

- Zuordnung des Steuersignalflusses gemäß der aktuellen Master-Slave-Beziehung im jeweiligen Buszyklus (unabhängig von der Richtung der Datenübertragung). Beispielsweise kann ein Steuersignal vom Master abgegeben werden, wenn die Adressen- und Funktionssteuersignale (beim Schreiben zusätzlich die Datensignale)

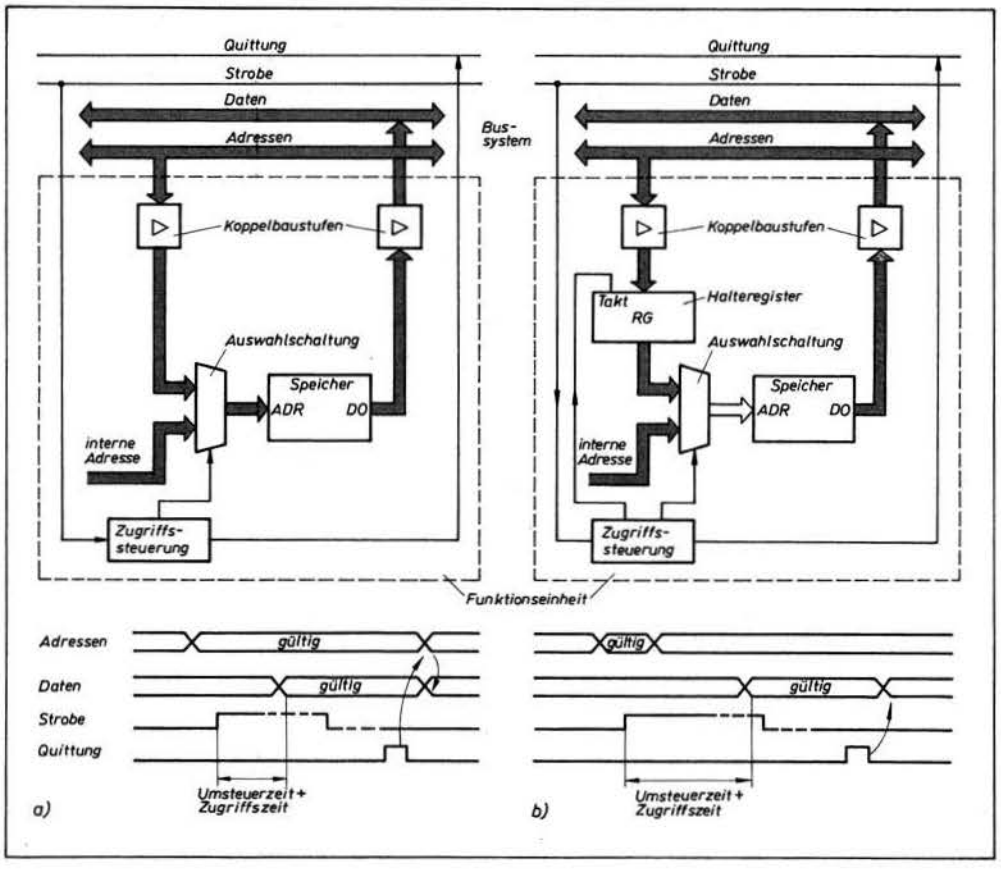


Bild 24: Definition der zeitlichen Verhältnisse für die Informationsübertragung in Abhängigkeit von der Hardware
a) Adressen für Speicherzugriff bis zum Eintreffen des Quittungsimpulses gültig; b) Adressen nur im Bereich der Vorderflanke des Strobeimpulses gültig

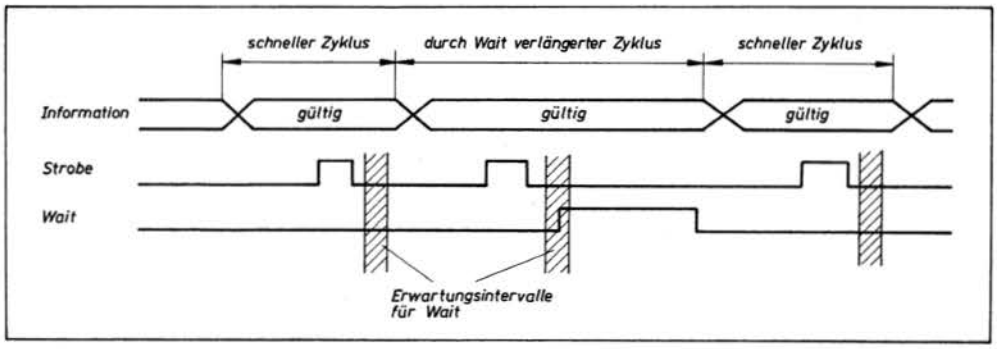


Bild 25: Informationsübertragung, deren Dauer durch ein Waitsignal über das nominelle Maß hinaus verlängert werden kann

auf dem Bus gültig sind. Der Slave kann mit einem Steuersignal antworten, wenn er die Daten akzeptiert hat (beim Schreiben) oder wenn diese auf dem Bus gültig sind (beim Lesen).

- Modifikationen hinsichtlich der Beziehung zwischen Steuersignal und Gültigkeit der Information, z. B.
  - Information gültig in bezug auf die Vorderflanke des Steuersignales
  - Information gültig in bezug auf die Rückflanke des Steuersignales
  - Information gültig in bezug auf die Dauer des Steuersignales.

  Die Festlegung der entsprechenden zeitlichen Verhältnisse ist ein wesentlicher Bestandteil der Definition des Bussystems und erfordert eine sorgfältige Untersuchung der funktionellen Eigenschaften und des zeitlichen Verhaltens der Koppelbaustufen und der wesentlichen Funktionskomplexe in den einzelnen Einrichtungen. Bild 24 veranschaulicht, welche Mehraufwendungen sich ergeben können, wenn das Signalspiel am Bus und die Eigenschaften wesentlicher Hardwarekomponenten nicht aufeinander abgestimmt sind: Halbleiterspeicher erfordern häufig, daß die Adressen- und Freigabesignale während der Zugriffszeit stabil anliegen. Ist in der Definition des Bussystems vorgesehen, daß die Information bereits zu Beginn des Strobesignals gültig sein muß und erst nach Empfang des Quittungssignals abgeschaltet werden darf, können die Buskoppelschaltungen sehr einfach ausgeführt werden (Bild 24a). Besagt hingegen die Busdefinition, daß die Information nach einer gewissen Haltezeit, gemessen von der Vorderflanke des Strobesignals, bereits abschalten darf, werden die Koppelschaltungen aufwendiger (Bild 24b).
- Senden eines Quittungssignales nur bei außergewöhnlichen Bedingungen. Der Verzicht auf Quittungssignale gestattet es, eine höhere Datenrate zu realisieren, erfordert aber, daß alle Funktionseinheiten entsprechend ausgelegt sind. Eine Lösung, die den Anschluß langsamerer Funktionseinheiten ermöglicht, besteht darin, ein Quittungssignal nur dann abzugeben, wenn die an sich vorgegebene Zugriffszeit nicht eingehalten werden kann. Dies entspricht der Waitsteuerung, die bei vielen Mikroprozessoren, z. B. beim U 880, vorgesehen ist. Die betreffenden Einrichtungen können das Quittungssignal sofort erregen, wenn sie als Slave adressiert werden. Es wird erst dann wieder ausgeschaltet, wenn der Zugriff erfolgt ist. Der Master muß zum Beginn des Zyklus erkennen, ob Wait signalisiert wurde, und die Dauer des Zyklus entsprechend beeinflussen (in der festgelegten Zeit beenden, wenn Wait inaktiv ist, bzw. bis zum Abfall von Wait verzögern, s. Bild 25).
- Zeitstarre Steuerung der Informationsübertragung. Statt asynchron durch das Zusammenwirken von Master und Slave kann die Informationsübertragung auch von einer zentralen Bussteuerung aus in festen Zeitintervallen gesteuert werden.

  Dies kann Vorteile wie z. B. eine überschaubare Arbeitsweise, die Konfliktsituationen vermeidet, oder Aufwandsverringerung durch die Zentralisierung von Steuermitteln mit sich bringen. Bei einem universell ausgelegten Bussystem, an dem unterschiedliche Einrichtungen als Master oder Slave arbeitsfähig sein sollen, wird allerdings das Definieren der Zeitbedingungen und Steuerfolgen noch problematischer, denn sowohl Master als auch Slave müssen in die Toleranzbetrachtungen einbezogen werden, und alle Kombinationen der Buskommunikation sind zu berücksichtigen.

2.1.3. Organisation der Daten- und Adressenwege

Um das Leitungssystem zu definieren, das die Daten- und die zugehörige Adresseninformation überträgt, sind folgende grundsätzliche Entscheidungen zu treffen:

- Anzahl der parallel in einem Zyklus zu übertragenden Adressen- und Datenbits
- parallele Übertragung von Adressen und Daten über unabhängige Busleitungen oder zeitmultiplexe Übertragung über gemeinsame Busleitungen
- blockweise Übertragung von Datenbereichen (Paketen) oder Übertragung einzelner Datenworte.

Bild 26 zeigt das Prinzip der parallelen Übertragung von Adressen- und Datensignalen.

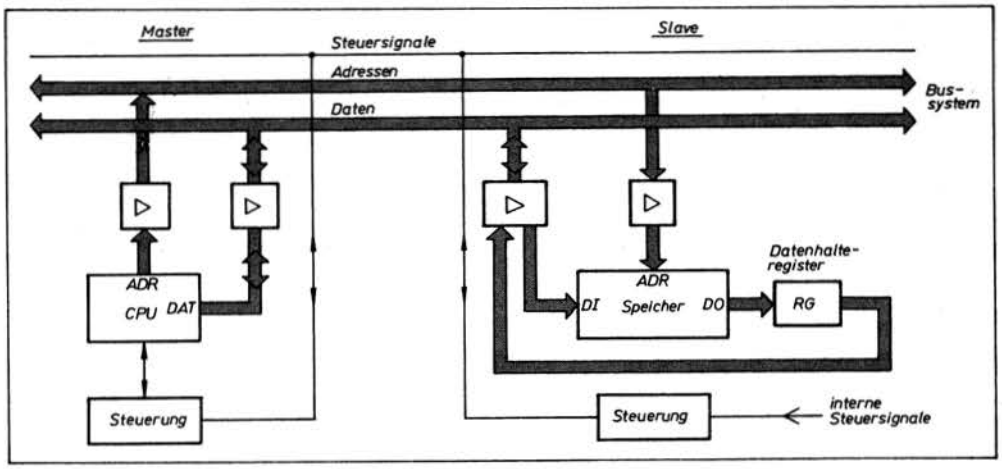


Bild 26: Parallele Übertragung von Daten und Adressen

Im Bild 27 ist dargestellt, wie Adressen und Daten zeitmultiplex über gemeinsame Leitungen übertragen werden können. Der Master legt zunächst die Adresse auf den Bus und veranlaßt mit Adressenstrobe eine Übernahme in das Adressenhalteregister des Slave. Nachfolgend wird die Dateninformation übertragen.

Beim Schreiben legt der Master die Daten auf den Bus und schaltet das Daten-gültig-Signal ein. Danach führt der Slave den Speicherzugriff aus. Beim Lesen legt der Slave die Daten auf den Bus und erregt das Daten-gültig-Signal. Die Steuerschaltungen in Master und Slave müssen für die bidirektionalen Buskoppelschaltungen die entsprechenden Freigabe- und Richtungssteuersignale liefern.

Bild 28 illustriert die paketweise Übertragung von Datenblöcken über ein zeitmultiplex organisiertes Bussystem. Der Master überträgt zunächst die Startadresse (mit Adressenstrobe) und danach die Anzahl der zu übertragenden Datenworte (Längenzählerstrobe) zum Slave. Daraufhin werden die Daten in der gewünschten Richtung übertragen, wobei im Slave jeweils der Adressenzähler erhöht und der Längenzähler vermindert wird. Steht der Längenzähler auf 0, so beendet der Slave die Übertragung mit dem Datenfertig-Signal. Im Master ist für den Datenaustausch ein Pufferspeicher vorgesehen, der während der Übertragung durch einen eigenen Adressenzähler adressiert wird.

Für Multimikrorechnersysteme werden sich die entsprechenden Überlegungen in der Regel an dem Bussystem des jeweiligen Mikroprozessors (bzw. an der im Gesamtsystem am häufigsten eingesetzten Schaltkreisfamilie) orientieren. Ein Ziel könnte z. B. darin bestehen, die Daten-, Adressen- und Steuersignale des internen Mikrorechner-Bussystems praktisch direkt (unter Zwischenschaltung von Koppelbaustufen und ergänzt durch zusätzliche Steuerleitungen) auf den gemeinsamen Bus des System durchzuschalten. Bei einem System, das auf dem U 880 basiert, hätte der Bus somit acht Datenleitungen und wenigstens 16 Adressenleitungen. Diese Vorgehensweise erspart umfangreiche Anpassungsschaltungen und ergibt eine übersichtliche Busstruktur. Im Beispiel müßte dieses Bussystem um

- zusätzliche Adressenleitungen zur Slaveauswahl (sofern der Adressenraum des Systems nicht auf 64 Kbyte beschränkt bleiben soll)
- Steuerleitungen für Richtung und Art der Informationsübertragung (z. B. READ, WRITE)
- Steuerleitungen für die Auswahl des Masters
- Leitungen zur Steuerung der Datenübertragung
- gegebenenfalls zusätzliche Leitungen für Sonderfunktionen, Fehlerkontrolle usw.

ergänzt werden.

Damit ist eine Mindestzahl von Leitungen gegeben. Ist diese auf Grund der Auslegung des Gesamtsystems nicht tragbar, muß ein zeitmultiplexes Übertragungsverfahren definiert werden.

Eine weitgehende Reduktion der Leitungszahl (im Idealfall bis zu der Grenze, die durch die geforderte Mindestdatenrate gegeben ist) lohnt sich dann, wenn die Gelegenheit besteht, die Steuer- und Anpassungsschaltungen als LSI-Schaltungen zu realisieren.

Offensichtliche Vorteile sind die geringere Anzahl an Verbindungen und die höhere Zuverlässigkeit sowohl statisch durch verringerte Komponentenzahl als auch dynamisch durch Reduktion von Störeinflüssen der anderen Signalleitungen. Die Beobachtbarkeit wird jedoch erschwert, so daß beim Entwurf der Steuerschaltungen entsprechende Mittel nicht vernachlässigt werden sollten.

Bei der Analyse der Variante gemäß Bild 28 sollte berücksichtigt werden, daß Pufferregister vorgesehen werden müssen, daß bei Lesezugriffen (Adresse vom Master zum Slave, daraufhin Daten vom Master zum Slave) praktisch kein Zeitverlust entsteht und daß bei Schreibzugriffen ein zusätzliches Zeitintervall benötigt wird.

Der letzte Gesichtspunkt ist vernachlässigbar, wenn über den Bus vorwiegend Befehlszugriffe von Mikrorechnern aus-

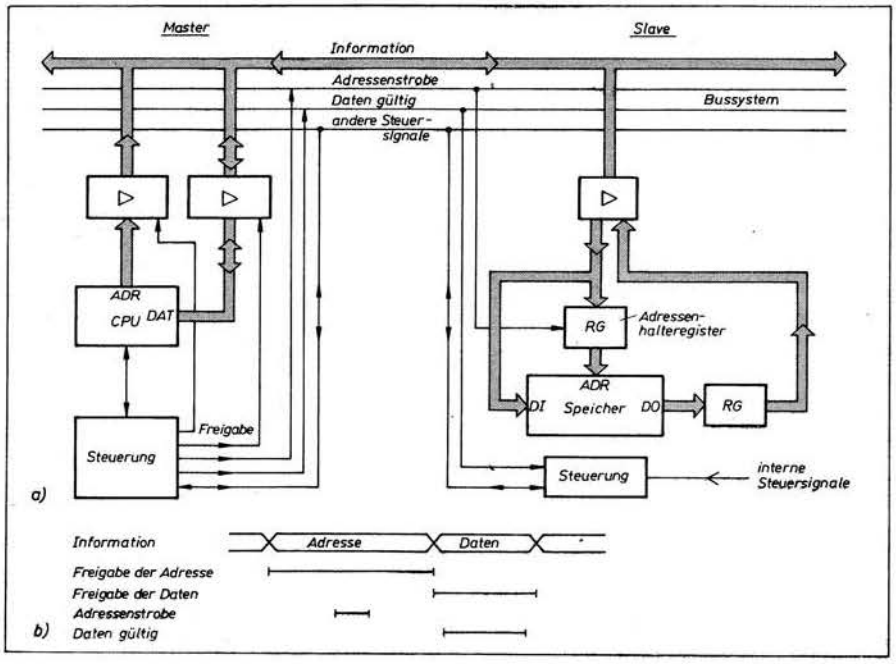


Bild 27: Zeitmultiplexe Übertragung von Daten und Adressen über gemeinsame Busleitungen. a) Prinzipschaltung; b) Zeitdiagramm

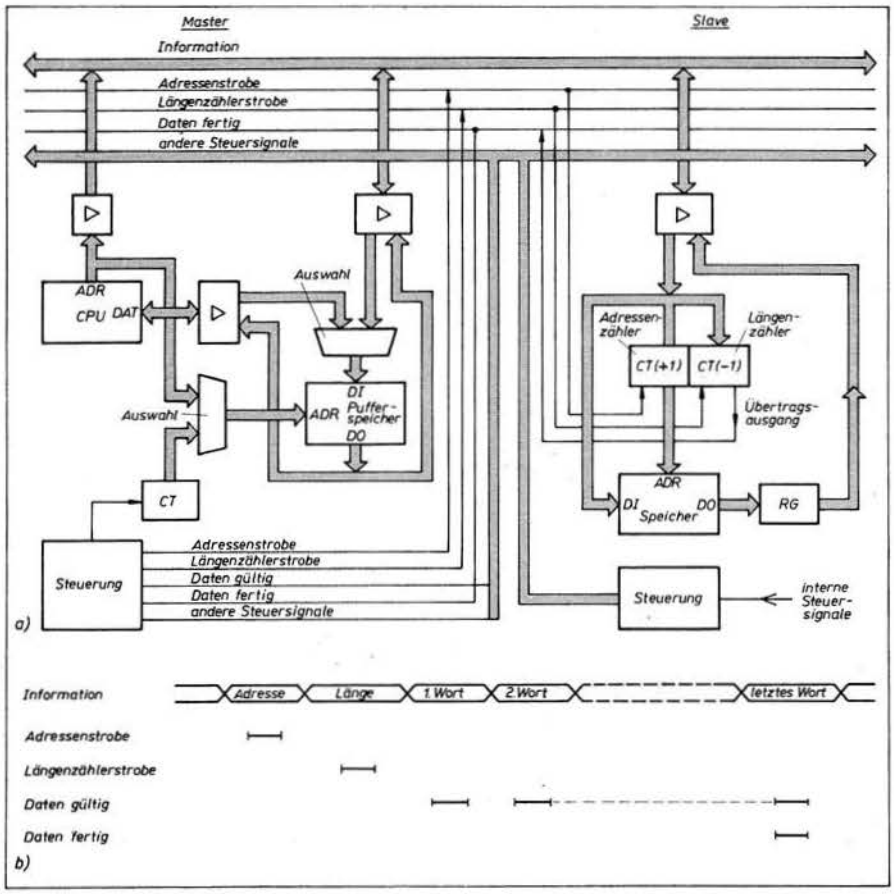


Bild 28: Paketweise Übertragung von Datenblöcken über zeitmultiplex organisiertes Bussystem. a) Prinzipschaltung; b) Zeitdiagramm

geführt werden, da hierbei Lesezugriffe bei weitem überwiegen. Es kann aber eine beträchtliche Leistungsminderung eintreten, wenn wesentliche Abläufe des Systems darin bestehen, daß Mastereinrichtungen (z. B. Mikrorechner) umfangreiche Datenblöcke zu Slaveeinrichtungen transportieren.

Hinsichtlich der blockweisen Übertragung ist zwischen zwei Fällen zu unterscheiden:

- Es handelt sich um eine Art zeitweiliger Monopolisierung des Bussystems, indem der Master den Bus längere Zeit für die Übertragung mehrerer Datenworte belegt, wobei aber zu jedem Datenwort die entsprechende Adresse über den Bus geschickt wird. Ein Anwendungsfall wäre die zeitweilige Blockierung anderer Einrichtungen bzw. Abläufe, wie es in 1.5. erläutert wurde.
- Der Master sendet zur Identifizierung der Information lediglich eine Startadresse und eine Längenangabe. Daraufhin wird ein zusammenhängender Datenstrom (Paket) übertragen. Damit lassen sich schnelle zeitmultiplexe Bussysteme aufbauen, die trotz geringer Leitungsanzahl einen sehr hohen Datendurchsatz gewährleisten. Es ist allerdings erforderlich, daß sowohl Master als auch Slave über autonome Adressenzähler verfügen und mit Pufferspeichern ausgerüstet sein müssen, wenn die interne Verarbeitungsgeschwindigkeit nicht ausreicht, den Datenstrom direkt zu erzeugen bzw. zu akzeptieren.

2.1.4. Wahl eines Bussystems

Falls nicht ein definitiver Zwang zum Einsatz eines bestimmten Bussystems besteht, ist die Entscheidung zu treffen, ob ein standardisiertes Bussystem eingesetzt oder ob ein eigenes entwickelt werden soll.
Standardisierte Bussysteme haben zweifellos Vorteile, nämlich die Einsparung von eigenem Entwicklungsaufwand sowie die Möglichkeit, Baugruppen anderer Hersteller einzusetzen. Dies kann sowohl eigene Entwicklungen erleichtern als auch ein Verkaufsargument sein.

Diese prinzipiellen Vorteile können aber nur dann wirksam werden, wenn folgende Voraussetzungen erfüllt sind:

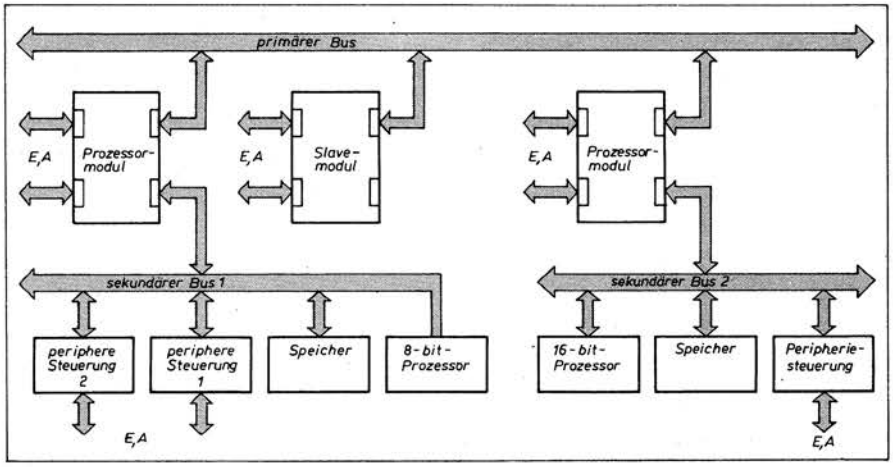
- Neben den logisch-funktionellen und elektrischen können auch die technologischen Bedingungen des Busstandards eingehalten werden (z. B. Steckverbinder, Leiterplattenformate).
- Die standardgemäße Funktion des Bussystems ist verifizierbar. (Alle Regeln des Standards sind bekannt; es stehen Baugruppen fremder Hersteller zur Überprüfung der Kompatibilität zur Verfügung usw.)
- Die Möglichkeiten zum Anschluß fremder Baugruppen sind auch kommerziell gegeben. (Die Anwender haben die tatsächliche Möglichkeit, dies zu tun; es stehen Testhilfen zur Verfügung; der Kundendienst ist darauf eingerichtet usw.)

Demnach ist es kaum zweckmäßig, ein bekanntes standardisiertes Bussystem mit eigenen Leiterplattenformaten, Steckverbindern usw., womöglich auf Grundlage unvollständiger Beschreibungen, im Sinne irgendwelcher Forderungen nach Kompatibilität zu realisieren.
In derartigen Fällen erscheint es sinnvoller, ein eigenes Bussystem zu definieren, bei dem Leistungsvermögen und Aufwand den konkreten Gegebenheiten entsprechend ausgelegt werden können. Dazu sollten die bewährten Lösungen standardisierter Bussysteme sorgfältig studiert werden.
Bei extremen Anforderungen hinsichtlich Leistungsvermögens oder Aufwandsminimierung wird eine spezielle Lösung stets unvermeidlich sein.
Welche Bedeutung eigene Bussysteme haben, geht z. B. daraus hervor, daß trotz jahrelanger Bemühungen um Standardisierung allein in Westeuropa über 140 Busstrukturen verwendet werden [6].
In komplexen Systemen können auch verschiedene Bussysteme gleichzeitig eingesetzt werden, um ein günstiges Verhältnis von Aufwand und Leistungsfähigkeit zu erzielen [6]. Bild 29 zeigt ein System, in dem unterschiedliche Bussysteme (eigene oder standardisierte) eingesetzt werden können.
Ein derartiges System könnte beispielsweise eigene Bussysteme in den Teilkomplexen erhalten, wo extreme Datenraten erforderlich sind, und standardisierte Bussysteme in den Teilen, in denen Universalität und Erweiterungsfähigkeit gewünscht werden, etwa zum Betreiben verschiedener Bildschirmgeräte, Drucker, Prozeß-Interfaces usw.

2.2. Standardisierte Bussysteme

Im folgenden werden einige international eingeführte Bus-

Bild 29: Komplexes Multimikrorechnersystem mit unterschiedlichen Bussystemen [6]



systeme vorgestellt, die für Multimikrorechnersysteme vorgesehen sind [7].

2.2.1. S-100-Bus (IEEE-696)

Auslegung

asynchron, parallele Übertragung von Adressen und Daten

Datenpfad

Es können 8-bit- und 16-bit-Worte übertragen werden. Dafür sind zwei unidirektionale 8-bit-Leitungsbündel vorgesehen. Beide zusammen bilden für 16-bit-Übertragungen einen bidirektionalen 16-bit-Bus.

Adressenpfad

Eine Mastereinrichtung muß mindestens 16 und darf höchstens 24 Adressenbits liefern. Zusätzlich werden Steuersignale abgegeben, die die Art des Buszyklus und die jeweilige Verwendung der Adresse kennzeichnen. Es gibt folgende Steuersignale:

MEMORY READ, INTERRUPT, OP CODE FETCH, ACKNOWLEDGE, INPUT, HALT ACKNOWLEDGE, OUTPUT, DATA TRANSFER REQUEST, WRITE CYCLE.

zulässige Leitungslänge

etwa 64 cm

zulässige Anzahl von Mastereinrichtungen

16

Prinzip der Masterauswahl

Zentrale Steuerung. Die Einrichtungen stellen Masteranforderungen mit Hilfe der Hold-Request-Leitung. Die Priorität wird mit vier Leitungen signalisiert. Die zentrale Steuerung erregt die Hold-Acknowledge-Leitung, wenn der Bus frei ist, worauf die Einrichtung mit der momentan höchsten Priorität den Bus zugesprochen erhält. Vier zusätzliche Leitungen steuern den Datentransport über den Bus: eine signalisiert den Beginn eines neuen Buszyklus. Die zweite zeigt an, daß die Adressen- und Statussignale auf dem Bus gültig sind. Eine Read-Strobe-Leitung veranlaßt die Datenübertragung vom Slave zum Master, und eine Write-Strobe-Leitung steuert die Übernahme von Busdaten in den Slave.

Interruptstruktur

Es sind acht Prioritätsebenen für Interrupts verfügbar.

2.2.2. Multibus (IEEE-795)

Auslegung

asynchron, parallele Übertragung von Adressen und Daten

Datenpfad

16 bit

Adressenpfad

bis zu 24 bit

zulässige Leitungslänge

etwa 38 cm

zulässige Anzahl von Mastereinrichtungen

16

Prinzip der Masterauswahl

Das Prinzip der unabhängigen Anforderungen ist anwendbar. Dies erfordert eine Anordnung aus Prioritätskodierer-IS (z. B. 74 148) und Dekodern (z. B. 74S138). Es kann auch das Daisy-Chain-Verfahren benutzt werden. Dazu wird die Prioritätseingangsleitung der höchstpriorisierten Einrichtung mit L-Potential verbunden und die Prioritätsausgangsleitung an die Eingangsleitung der Einrichtung mit der nächstniederen Priorität angeschlossen usw. Üblicherweise werden nicht mehr als drei Einrichtungen auf diese Weise zusammengeschaltet, da ansonsten die Verzögerungszeiten übermäßig anwachsen würden.

Interruptstruktur

acht Ebenen und zwei Prinzipien (engl. non-bus-vectored, bus vectored); bei letzterem werden die Adressenleitungen zur Übertragung von Interruptvektor-Adressen benutzt.

2.2.3. VME-Bus

Auslegung

asynchron, parallele Übertragung von Adressen und Daten

Datenpfad

8, 16, 32 bit (letzteres erfordert Leiterplatten doppelter Größe mit einem zweiten Steckverbinder)

Adressenpfad

16, 24, 32 bit

zulässige Leitungslänge

etwa 48 cm

zulässige Anzahl von Mastereinrichtungen

begrenzt durch elektrische Belastbarkeit des Bussystems (es werden 20 Lasteinheiten getrieben)

Prinzip der Masterauswahl

Es sind drei Methoden verfügbar:

- Direkte Zuordnung durch eine zentrale Steuerung auf der Basis fest definierter Prioritäten, wofür vier Anforderungsleitungen vorgesehen sind
- zyklische Zuordnung durch zentrale Steuerung (engl. round robin)
- Daisy-Chain-Prinzip.

Interruptstruktur

sieben Interruptanforderungssignale, ein Daisy-Chain-Signal und ein Interruptbestätigungssignal

Besonderheiten

Für das Bussystem ist eine Read-Modify-Write-Folge definiert, bei der zwischen Lese- und Schreibzugriff der Bus nicht erneut vermittelt wird. Dies gestattet es, Verwaltungsabläufe der Art Test and Set (s. Abschnitt 1.5.) auszuführen. Weiterhin sind Leitungen für diagnostische Signale und für das Rücksetzen des Systems vorgesehen.

2.3. Realisierungsbeispiel

In diesem Abschnitt wird ein Bussystem beschrieben, das für Multimikrorechnersysteme auf Basis des U 880 oder ähnlicher Mikroprozessorfamilien entwickelt wurde [8].

2.3.1. Bussystem

Das Bussystem ist für die asynchrone parallele Übertragung von Adressen und Daten vorgesehen. Die Bussteuerung erfolgt durch eine zentrale Steuereinrichtung (Funktionseinheit BUS CONTROL). Das Leitungssystem und die Signalfolgen wurden unter besonderer Berücksichtigung folgender Aspekte definiert:

- Realisierung der Steuer- und Koppelschaltungen mit SSI- und MSI-IS
- einfache Kopplung an U-880-Strukturen bei Erweiterung des Adressenraumes über 64 Kbyte hinaus
- Möglichkeiten zur Implementierung komplexer Software-Organisationsformen
- Vorkehrungen für Fehlerbehandlung und Diagnose
- physisch kompakter Aufbau des Systems, s. Bild 5.

2.3.2. Übersicht über die Busleitungen

Anfangsrücksetzen	RESET
Masterauswahl	HOLD
	REQUEST 4-1
	PRIORITY 1,0
	SELECT
Ablaufsteuerung für Buszyklen	BUSY
	REPLY
	RELEASE
	AUXILIARY ACKNOWLEDGE
	PULSE
	ACKNOWLEDGE
Begleitsignale	READ
	WRITE
	INTERRUPT
Zustandssignale des Slave	COMPARE MATCH
	SLAVE ERROR
allgemeine Steuer- und Zustandssignale	BURST MODE
	ERROR
	NON EXECUTIVE STATE
Datensignale	DATA 7-0
Adressensignale	ADRS 19-0
selektives Rücksetzen	SELECTIVE RESET
Prüftakt	EXTERNAL INJECT

2.3.3. Koppelbaustufen

Es werden weitgehend Tristateverbindungen eingesetzt. Einige Signale werden von TTL-Schaltkreisen geliefert. Die Bilder 30 bis 33 geben einen Überblick über die verschiedenen Ausführungen der Buskopplung. Für die Wahl dieser Ausführungen waren folgende Gesichtspunkte wesentlich:

- Für alle bidirektionalen Leitungen sind Tristate-Baustufen vorgesehen. Der Vorteil besteht darin, mit einem Schaltkreis (z. B. DS 8216) bis zu vier Leitungen sowohl eingangs- als auch ausgangsseitig auf den Bus koppeln zu können. Ein prinzipieller Nachteil besteht in der Notwendigkeit, zu gewährleisten, daß zu keinem Zeitpunkt zwei Treiber gleichzeitig auf eine Busleitung wirken. Weiterhin hat es sich gezeigt, daß verschiedene Schaltkreise Störimpulse abgeben, wenn sie durch Umsteuern des Chip-Enable-Eingangs aktiviert bzw. deaktiviert werden (Bild 34). Diese Nachteile treten beispielsweise bei Open-Collector-IS nicht auf. Die Ankopplung eines bidirektionalen Signals erfordert allerdings zwei Gatter, einen Pull-up-Widerstand je Busleitung und einen je Einrichtung (Bild 35). Da von den 52 Busleitungen 38 bidirektional sind und sich bei dieser Alternative die Anzahl der Koppelschaltkreise in jeder Funktionseinheit verdoppelt, erweist es sich als günstiger, die Tristate-Kopplung durchgehend anzuwenden und die Funktionssicherheit dadurch zu gewährleisten, daß spezielle Steuerungsmaßnahmen eingeführt werden, um Störimpulse auszublenden und um zu verhindern, daß zwei Schaltkreise gleichzeitig eine Leitung treiben.

Bild 30: Buskopplung mit TTL-Schaltkreisen

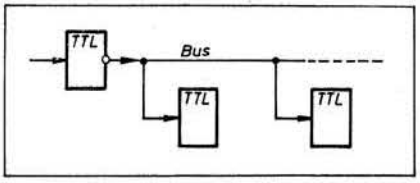
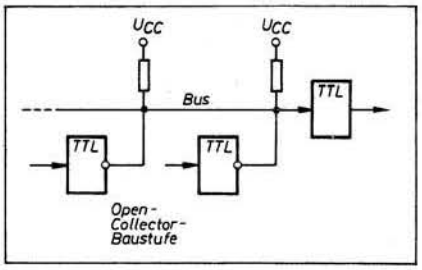


Bild 31: Buskopplung mit Open-Collector-Schaltkreisen. Die Busleitung kann gleichzeitig von mehreren Quellen aus erregt werden



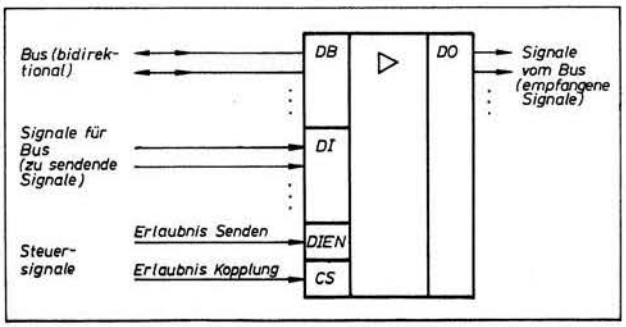


Bild 32: Buskopplung für bidirektionale Steuersignale mit Tristateschaltungen

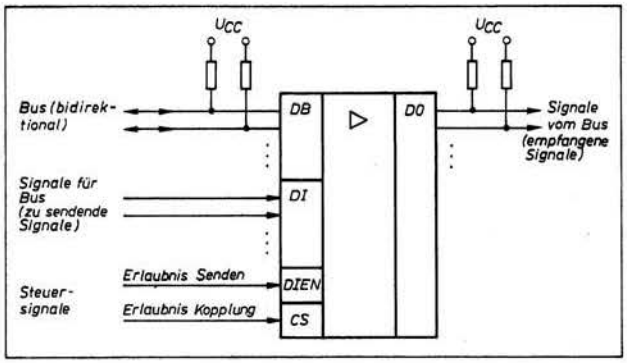


Bild 33: Buskopplung für Daten- und Adressensignale mit Tristateschaltungen

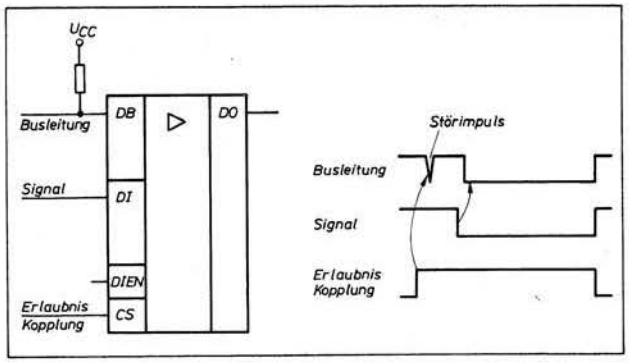


Bild 34: Mögliche Störimpulse beim Aktivieren des Chip-Enable-Einganges eines Tristate-Buskoppelbausteins

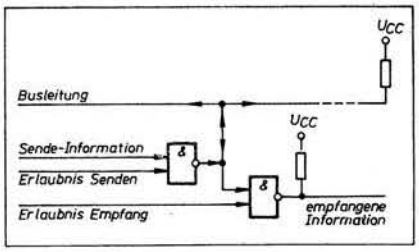


Bild 35: Beschaltung einer bidirektionalen Busleitung mit Open-Collector-Schaltungen ▶

- Für die Signale, die von der zentralen Bussteuerung erregt und von den anderen Einrichtungen empfangen werden, sind TTL-Treiber-Baustufen mit hinreichender Belastbarkeit vorgesehen. Die Rücksetzleitungen (RESET, SELECTIVE RESET) sind als aktiv L definiert, da die Störsicherheit bei dieser Pegelzuordnung besser ist. (Dies ist wesentlich, da Rücksetzsignale zu jeder Zeit wirksam sind und ein unberechtigtes Rücksetzen einen totalen Systemausfall hervorrufen würde.)
  Die anderen Signale sind als aktiv H definiert, um Entkopplungsschaltungen in den Funktionseinheiten zu ersparen. (Die Signale sind direkt an die entsprechenden Steuerschaltungen angeschlossen.) Die erforderliche Störsicherheit wird durch die
  - Anordnung dieser Leitungen in den Buskabeln
  - Tatsache, daß die Busleitungen kurz sind
  - Definition der Signalfolgen, so daß Störungen entweder nicht entstehen oder nicht zu Fehlfunktionen führen,

  gewährleistet.
- Für Signale, die von mehreren Funktionseinheiten gleichzeitig erregt werden können, sind Open-Collector-TTL-Baustufen vorgesehen. Diese Signale sind aktiv L definiert.

2.3.4. Bussignale

ADRS 19-0

Es sind 20 bidirektionale Adressenleitungen vorgesehen (Kopplung gemäß Bild 34), so daß ein Adressenraum von 1 Mbyte verfügbar ist. Dieser ist über das gesamte System verteilt. Im allgemeinen werden einige der höchstwertigen Adressenbits zur Auswahl des Slave verwendet. Die übliche Aufteilung ist im Bild 36 dargestellt.
Die Auslegung des Systems für maximal 16 Slaveeinrichtungen, wobei in jeder maximal 64 Kbyte über den Bus zugänglich sein können, korrespondiert mit den aktuellen technischen Möglichkeiten. (Da der Bus elektrisch kurz sein muß, hat es keinen Sinn, Vorkehrungen für eine größere Zahl von Funktionseinheiten zu treffen, und eine Kapazität von 64 Kbyte ist mit den gegebenen Speicherschaltkreisen nur auf wenigen Typen von Funktionseinheiten realisierbar.) Für Funktionseinheiten, bei denen mehr als 64 Kbyte über den Bus erreichbar sein sollen, müssen einige der Adressenbits 19-16 zur internen Adressierung mitbenutzt werden. Dementsprechend verringert sich die Anzahl der anschließbaren Slaveeinrichtungen. Die Adressensignale werden auf den Bus gelegt, wenn die betreffende Mastereinrichtung den Bus zugesprochen bekommen hat. Die niederen 16 Signale werden von der Slaveeinrichtung empfangen und die höchsten vier von allen Einrichtungen, die als Slave adressierbar sind (mit Ausnahme des aktuellen Masters).
Ist die Leitung INTERRUPT aktiv, so ist die Belegung der niederen 16 Adressensignale undefiniert, der Slave darf diese nicht auswerten.

DATA 7-0

Es sind acht bidirektionale Datenleitungen für die parallele Übertragung eines Bytes vorgesehen und gemäß Bild 33 an den Bus gekoppelt. Der Bus wird dabei vom Master bei Schreibzugriffen und vom Slave bei Lesezugriffen belegt. Die Busbelegung wird vom Master bei Lesezugriffen und vom Slave bei Schreibzugriffen empfangen.

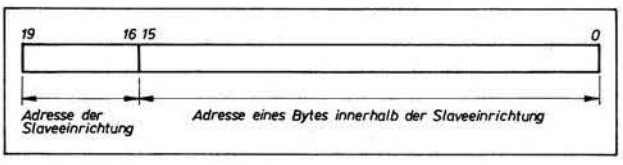


Bild 36: Aufteilung der 20-bit-Busadresse

REQUEST 4-1

Es sind vier Anforderungsleitungen vorgesehen, wobei jeder eine Priorität zugeordnet ist. REQUEST 1 hat die niedrigste, REQUEST 4 die höchste Priorität. Die Signale sind gemäß Bild 31 angekoppelt.

Eine Funktionseinheit, die Master werden will, erregt eine dieser Leitungen. Die Prioritätszuordnung kann durch Verdrahtung, durch einstellbare Schalter oder durch programmierbare Steuerschaltungen realisiert sein. Die Anforderungssignale werden von der Bussteuerung ausgewertet.

HOLD

Auf dieser Leitung wird ein Steuerimpuls für die Masterauswahl von der Bussteuerung an alle Einrichtungen geliefert, die Master werden können. Das Signal ist als aktiv H definiert und gemäß Bild 30 angekoppelt.

PRIORITY 1,0

Diese Leitungen führen einen Prioritätskode für die Masterauswahl von der Bussteuerung zu allen Einrichtungen, die Master werden können. Die Signale sind als aktiv H definiert und gemäß Bild 30 angekoppelt. Die Belegung der Leitungen entspricht jeweils einer bestimmten Priorität:

00 ≙ REQUEST 1
01 ≙ REQUEST 2
10 ≙ REQUEST 3
11 ≙ REQUEST 4

SELECT

SELECT ist die Auswahlleitung für die Masterauswahl. Sie ist als einzige Leitung des Bussystems nach dem Daisy-Chain-Prinzip durch alle Funktionseinheiten geführt, die Master werden können. Das Signal ist als aktiv H definiert und gemäß Bild 30 angekoppelt. In jeder entsprechenden Einrichtung gibt es eine ankommende und eine abgehende SELECT-Leitung.

BUSY

Dieses Signal kennzeichnet den Besetztzustand des Busses, d. h., es wird gerade ein Buszyklus ausgeführt. Es ist als aktiv L definiert und gemäß Bild 32 angekoppelt. Die Leitung wird vom aktuellen Master aktiviert und von allen Einrichtungen empfangen, die Slave werden können.

REPLY, RELEASE

Beide Leitungen sind gemäß Bild 32 an den Bus angekoppelt (aktiv L). Sie werden vom aktuellen Slave erregt, um das Ende des Buszyklusses zu signalisieren, und vom aktuellen Master sowie von der Bussteuerung ausgewertet. Es wird jeweils nur eine Leitung aktiviert:

REPLY ist die Antwortleitung. Damit signalisiert der Slave die korrekte Ausführung des Zugriffs im Buszyklus. (Beim Schreiben wurden die Daten vom Bus übernommen; beim Lesen sind die Busdaten gültig.)

RELEASE ist die Freigabeleitung. Damit signalisiert der Slave, daß er den Zugriff nicht im aktuellen Buszyklus ausführen kann, so daß ein erneuter Buszyklus erforderlich wird.

Das Prinzip, einen Buszugriff, der in einer definierten Zeit nicht erfolgreich ausgeführt werden kann, durch ein entsprechendes Steuersignal abzubrechen und später neu zu versuchen, hat gegenüber der üblichen Waitsteuerung den Vorteil, daß der Bus sehr schnell für die Kommunikation anderer Einrichtungen frei wird.

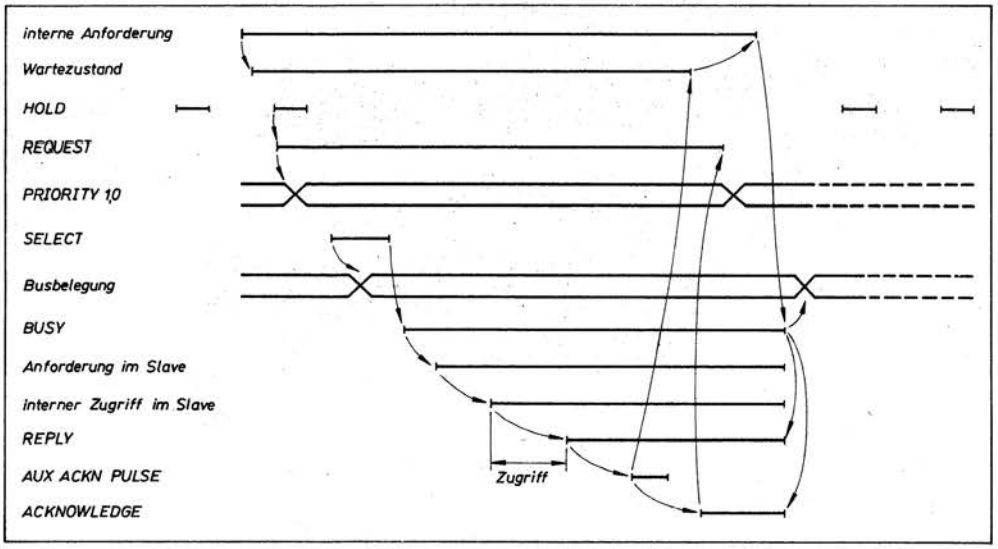


Bild 37: Impulsdiagramm eines normalen Buszyklus

Bild 38: Schematische Darstellung der wesentlichen Abläufe in Bussteuerung, Master und Slave bei normalem Buszyklus ▼

AUXILIARY ACKNOWLEDGE PULSE (AUX ACKN PULSE), ACKNOWLEDGE

Beide Signale werden von der Bussteuerung erzeugt, um allen Einrichtungen zu signalisieren, daß der aktuelle Buszyklus beendet werden kann. Sie sind als aktiv H definiert und gemäß Bild 30 angekoppelt.

AUX ACKN PULSE ist ein impulsförmiges Signal, das vor ACKNOWLEDGE abgegeben wird und von den internen Steuerschaltungen in den Einrichtungen ausgewertet wird. Damit wird der Buszyklus intern beendet. ACKNOWLEDGE selbst gibt die Erlaubnis zum Deaktivieren der Bussignale.

Der Zweck derartiger Erlaubnissignale besteht im wesentlichen darin, der Bussteuerung eine Kontrolle und Beobachtung der Buszyklen zu ermöglichen. Durch das zeitlich vorverlegte Hilfssignal AUX ACKN PULSE wird verhindert, daß Störungen, die durch das gleichzeitige Abschalten vieler Busleitungen entstehen, sich auf die Steuerschaltungen in den Funktionseinheiten auswirken.

READ, WRITE, INTERRUPT

Diese Begleitsignalleitungen sind gemäß Bild 33 an den Bus angekoppelt (aktiv L). Sie werden vom Master erregt und vom Slave ausgewertet. READ veranlaßt die Ausführung eines Lesezugriffs entsprechend der übertragenen Adresse. WRITE veranlaßt sinngemäß die Ausführung eines Schreibzugriffs.

Ist WRITE zusammen mit INTERRUPT aktiv, so wird eine Unterbrechung in der Slaveeinrichtung ausgelöst. Dabei wird das Datenbyte als Interruptvektor benutzt. Die niederen 16 Adressensignale sind ungültig (undefiniert bzw. im hochohmigen Zustand), wenn INTERRUPT aktiv ist.

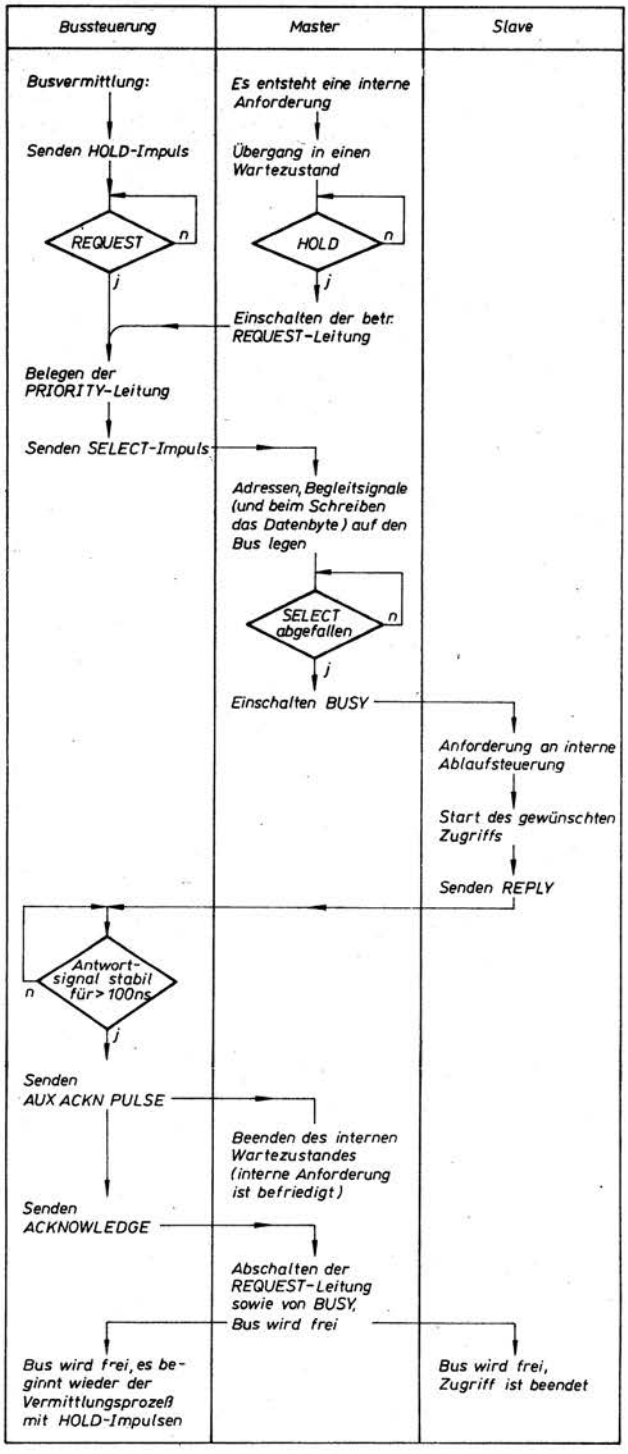
COMPARE MATCH, SLAVE ERROR

Diese Leitungen dienen zur Zustandssignalisierung des Slave. Sie sind gemäß Bild 32 angekoppelt, werden vom Slave erregt und von der Bussteuerung ausgewertet. COMPARE MATCH ist von Bedeutung, wenn der Slave Schaltmittel für Vergleichsstopps (Adressenvergleich bei Speicherzugriffen für Wartungszwecke) enthält. COMPARE MATCH signalisiert dann, daß während eines Slavezugriffs eine solche Vergleichsstoppbedingung aufgetreten ist.

SLAVE ERROR ist von Bedeutung, wenn die Slaveeinrichtung über Schaltmittel zur Fehlererkennung (z. B. Paritätsprüfung) verfügt. SLAVE ERROR signalisiert dann, daß während eines Slavezugriffs eine solche Fehlerbedingung aufgetreten ist.

RESET, SELECTIVE RESET

Diese beiden Rücksetzleitungen werden von der Bussteuerung erregt. Sie sind gemäß Bild 30 angeschlossen und als aktiv L definiert. RESET veranlaßt ein allgemeines Rücksetzen aller Einrichtungen (mit Ausnahme jener, die kein initiales Rücksetzen erfordern). SELECTIVE RESET veranlaßt im Rahmen einer selektiven Rücksetzfolge das Rücksetzen in einer Einrichtung, die durch die höchsten vier Adressenbits identifiziert wird. Im allgemeinen unterscheiden sich



beide Rücksetzoperationen in den einzelnen Einrichtungen, und zwar sowohl hinsichtlich der Hardware, die zurückgesetzt wird, als auch in den nachfolgenden Softwareabläufen.

BURST MODE

Diese Steuerleitung ist gemäß Bild 32 an den Bus gekoppelt (aktiv L) und wird vom Master erregt. Sie wird vom Slave und von der Bussteuerung ausgewertet. Das Signal veranlaßt, daß nach der Auswahl des Masters dieser so lange die Kontrolle über den Bus behält, bis er BURST MODE wieder ausschaltet (in diesem Zeitintervall wird also der Bus nicht neu vermittelt). Der einmal ausgewählte Master kann auf beliebig viele Slaveeinrichtungen zurückgreifen.

NON EXECUTIVE STATE (NES)

Diese Leitung wird von der Bussteuerung erregt und von allen Einrichtungen ausgewertet, für die dieser besondere Zustand von Bedeutung ist. Es handelt sich darum, in den betreffenden Einrichtungen einen Zustand herbeizuführen, in dem die Bussteuerung ungestörte Zugriffe zwecks Fehlerbehandlung, Testung, Initialisierung usw. ausführen kann. Dazu müssen Speicherschutzeinrichtungen und andere Kontrollschaltungen außer Betrieb gesetzt werden, Mikrorechner müssen in einen Wartezustand versetzt werden usw. Die Leitung NES wirkt nicht selektiv, sondern auf alle Einrichtungen gleichzeitig. Sie ist gemäß Bild 30 angeschlossen (aktiv H).

ERROR

Dies ist die Fehlersignalleitung des Bussystems. Sie ist gemäß Bild 31 (aktiv L) an alle Einrichtungen angeschlossen, die Fehlerkontrollschaltungen (z. B. Paritätsprüfung, Speicherschutz o. ä.) enthalten, und wird von der Bussteuerung ausgewertet. Die Leitung wird asynchron (ohne Beziehung zu den Buszyklen) erregt.

EXTERNAL INJECT

Dies ist ein Prüftaktsignal, das von der Bussteuerung für alle Einrichtungen geliefert wird (Anschluß gemäß Bild 30). Üblicherweise hat jede Einrichtung ihre eigene Takterzeugung, da nur so eine hohe Leistungsfähigkeit zu erreichen ist. Dadurch ist aber das Signalspiel am Bus völlig asynchron. Dies erschwert die Anwendung bestimmter Fehlersuchverfahren (namentlich die Signaturanalyse). Für diese Zwecke wird ein Prüftakt geliefert, der zeitstarr mit dem Takt der eigentlichen Bussteuerschaltung verkoppelt ist. Die einzelnen Einrichtungen enthalten eine Umschaltmöglichkeit (in Form eines DIL-Schalters auf der Leiterplatte), um für Fehlersuchzwecke den Prüftakt statt des internen Taktes zu verwenden.

2.3.5. Beschreibung der Bussteuerfolgen

Bild 37 zeigt das Impulsdiagramm eines normalen Buszyklus. Zusätzlich ist im Bild 38 dargestellt, welche wesentlichen Operationen in den beteiligten Einrichtungen (Bussteuerung, Master, Slave) ablaufen.

Eine Einrichtung will Master werden, wenn intern eine Anforderung nach einem Buszugriff entsteht, beispielsweise dadurch, daß ein Mikrorechner einen Befehl ausführt, der einen derartigen Zugriff notwendig macht. Eine derartige Anforderung veranlaßt zunächst einen internen Wartezustand.

Ist der Bus nicht belegt, so sendet die Bussteuerung zyklisch Impulse auf der HOLD-Leitung aus. Eine Einrichtung, die Master werden will, belegt die entsprechende REQUEST-Leitung, wenn ein HOLD-Impuls eintritt. Eine gewisse Zeit nach diesem wertet die Bussteuerung die Belegung der REQUEST-Leitungen aus. Ist keine dieser Leitungen erregt, wird wieder ein HOLD-Impuls abgegeben. Ist wenigstens eine der REQUEST-Leitungen erregt, so stellt die Bussteuerung die PRIORITY-Leitungen gemäß der höchstpriorisierten erregten REQUEST-Leitung ein. Daraufhin wird ein Impuls über die Auswahlleitung SELECT abgegeben. Die Einrichtungen, die keine Master-Anforderung oder eine mit niedriger Priorität gestellt haben, leiten den SELECT-Impuls zur jeweils benachbarten Einrichtung weiter. Die erste Einrichtung, bei der die Priorität der REQUEST-Leitung mit der Belegung der PRIORITY-Leitungen übereinstimmt, leitet den Impuls nicht weiter, sondern veranlaßt, daß nach der Vorderflanke dieses Impulses die Adresse, die erforderlichen Begleitsignale und beim Schreiben das Datenbyte auf den Bus gelegt werden.

Die Rückflanke des SELECT-Impulses veranlaßt, daß die BUSY-Leitung erregt wird. Damit können die anderen Einrichtungen am Bus erkennen, ob sie als Slave ausgewählt wurden.

Hat eine Einrichtung die Auswahl als Slave erkannt, so entsteht intern eine Slaveanforderung (SLAVE REQUEST), die zu einem entsprechenden Ablauf führt (Lesen eines Datenbytes, Schreiben eines Datenbytes, Auslösen eines Interrupts). Am Ende des jeweiligen internen Ablaufs wird die Leitung REPLY erregt. Die Bussteuerung hat daraufhin die Möglichkeit, die Busbelegung zu analysieren (z. B. auf Fehlerbedingungen).

Ist dies geschehen, so wird zunächst das Signal AUX ACKN PULSE und anschließend ACKNOWLEDGE abgegeben. AUX ACKN PULSE veranlaßt, daß der interne Wartezustand in der Mastereinrichtung aufgehoben wird. ACKNOWLEDGE veranlaßt das Abschalten der Bussignale des Masters und aller REQUEST-Leitungen. Mit dem Abfall von BUSY schaltet der Slave seine Bussignale ab. In der Bussteuerung wird ACKNOWLEDGE ausgeschaltet. Nach einem gewissen Zeitintervall beginnt eine neue Busvermittlung (HOLD-Impuls).

Bild 39 zeigt den Ablauf eines Buszyklus, wenn der Slave nicht in der Lage ist, den gewünschten Zugriff sofort auszuführen. In diesem Fall erregt er statt der Antwortleitung REPLY die Freigabeleitung RELEASE. Dadurch wird der Buszyklus mit Hilfe der Signalfolge AUX ACKN PULSE – ACKNOWLEDGE – Abfall von BUSY beendet, die interne Anforderung in der Mastereinrichtung bleibt aber bestehen. Somit beteiligt sich diese Einrichtung wieder an der Busvermittlung. Es werden so lange Buszyklen angefordert, bis der Slave den gewünschten Zugriff ausgeführt hat, d. h.,

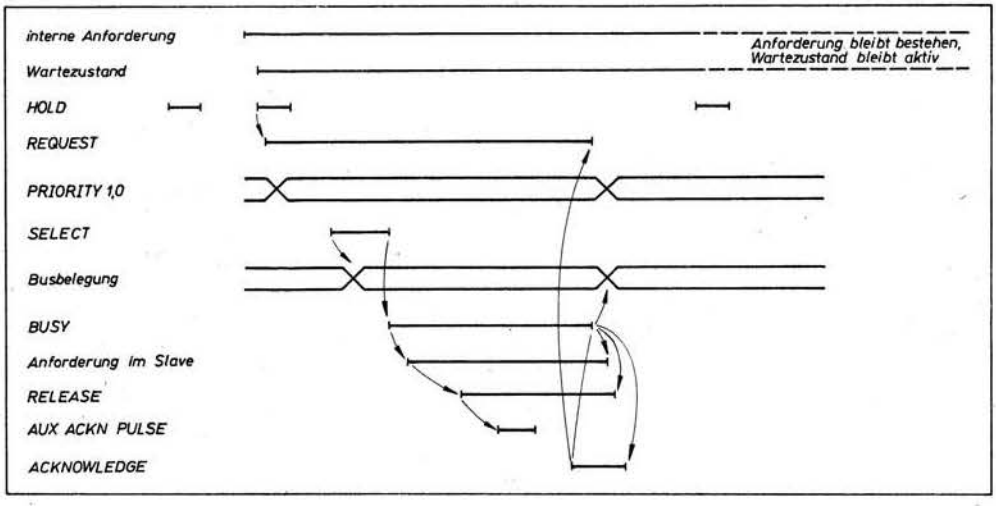


Bild 39: Impulsdiagramm eines Buszyklus, der vom Slave mit RELEASE beantwortet wird

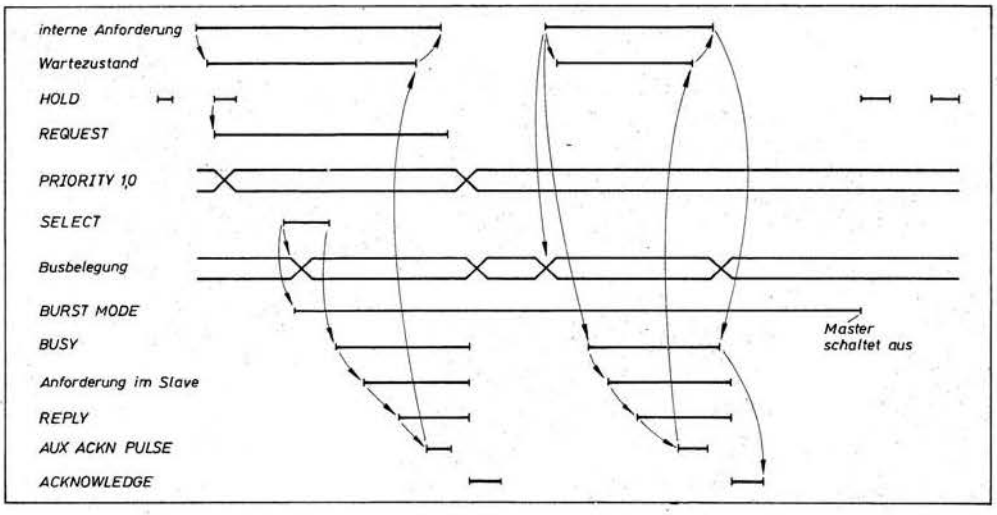


Bild 40: Impulsdiagramm einer Informationsübertragung im Burstmodus

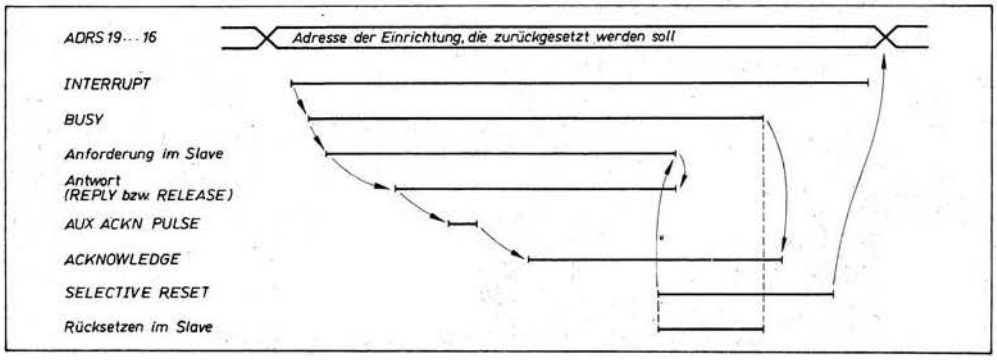
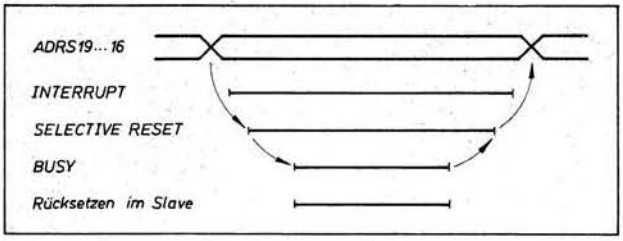


Bild 41: Impulsdiagramm des selektiven Rücksetzens für Einrichtungen, bei denen vor dem Rücksetzen ein Slavezugriff gestartet werden muß

Bild 42: Impulsdiagramm des selektiven Rücksetzens für Einrichtungen, die dafür keinen Slavezugriff verlangen ▼

bis der Zyklus mit REPLY beendet wurde. In der Zwischenzeit können andere Funktionseinheiten (mit höherer Priorität) den Bus benutzen.
Bild 40 zeigt den Ablauf einer Datenübertragung im Burstmodus. Darin behält eine Mastereinrichtung, nachdem ihr der Bus zugesprochen wurde, so lange die Kontrolle über den Bus, wie sie dies wünscht. Der Master wird auf die übliche Weise ausgewählt. Die Vorderflanke von SELECT veranlaßt jedoch zusätzlich die Erregung der Leitung BURST MODE. Damit wird in der Bussteuerung die Masterauswahl verhindert, d. h., nach Ende eines Buszyklus gibt es keine neuen HOLD-SELECT-Folgen. Ist ein Buszyklus beendet, kann der Master einen neuen starten, indem er den Bus mit Adressen und Begleitsignalen (und beim Schreiben mit dem Datenbyte) belegt und nach einer gewissen zeitlichen Verzögerung die BUSY-Leitung aktiviert. Die einzelnen Buszyklen werden auf übliche Weise beendet. Der Burstmodus wird verlassen (die Busvermittlung wird wieder wirksam), wenn der Master BURST MODE ausschaltet.
Die Abläufe des selektiven Rücksetzens ermöglichen es, von der Bussteuerung aus in einer ausgewählten Einrichtung ein Hardwarerücksetzen zu veranlassen.
Es gibt zwei Arten von Abläufen, die in den Bildern 41 und 42 dargestellt sind. Im Ablauf gemäß Bild 41 startet die Bussteuerung zunächst einen normalen Buszyklus. Die betreffende Einrichtung wird als Slave adressiert. Dabei sind weder READ noch WRITE aktiv, so daß im Slave keine Zugriffsoperation ausgelöst wird, und INTERRUPT ist aktiv, um die Auswertung der (undefinierten) niederen 16 Adressensignale zu verhindern. Nach einer gewissen Zeit wird SELECTIVE RESET erregt. Dabei fragt die Bussteuerung ab, ob REPLY oder RELEASE aktiv waren. Wurde vom Slave weder REPLY noch RELEASE erregt, so hält dies den Rücksetzablauf nicht auf, veranlaßt aber anschließend eine Fehlerreaktion in der Bussteuerung.
Im Ablauf gemäß Bild 42 wird SELECTIVE RESET von vornherein aktiviert.
Der zuerst beschriebene Ablauf ist zum Rücksetzen derjenigen Einrichtung vorgesehen, deren Speicher mit dynamischen RAMs realisiert sind und bei denen es erforderlich ist, den RAM-Inhalt über das Rücksetzen hinaus zu erhalten. Namentlich bei Mikrorechneranordnungen soll das selektive Rücksetzen bestimmte Softwareabläufe auslösen. Dazu müssen die Speicherinhalte erhalten bleiben. Dynamische RAMs haben nun die Eigenschaft, ihren Inhalt zu verfälschen, wenn Zugriffe undefiniert abgebrochen werden. Dieses undefinierte Abbrechen läßt sich vermeiden, wenn vor dem Rücksetzen ein definierter Zugriff ausgeführt wird (deshalb wird zunächst ein normaler Slavezugriff gestartet). Der zweite Ablauf ist für Einrichtungen vorgesehen, bei denen derartige Effekte nicht berücksichtigt werden müssen.



Um das gesamte System vollständig zurückzusetzen, erregt die Bussteuerung kurzzeitig die Leitung RESET. Die Leitungen COMPARE MATCH, SLAVE ERROR, ERROR, NES haben keinen Einfluß auf den Ablauf der einzelnen Buszyklen.
COMPARE MATCH bzw. SLAVE ERROR werden in einem Buszyklus vom Slave erregt, wenn die entsprechenden Bedingungen (Vergleichsstopp oder Fehler) auftreten. Sie veranlassen, daß der Bus angehalten wird (nach dem aktuellen Buszyklus wird der Bus nicht erneut vermittelt) und daß die Leitung NES eingeschaltet wird. Dadurch erhält die Bussteuerung die Möglichkeit, die signalisierte Bedingung zu analysieren und auszuwerten.
ERROR wird im Fehlerfall von der betreffenden Einrichtung erregt. Dies bewirkt, daß die Bussteuerung NES einschaltet (damit wird die Arbeit in den anderen Einrichtungen angehalten, um die Fehlersituation möglichst wenig zu verändern) und die Fehlermeldung näher analysiert. Um dies zu ermöglichen, muß jede Einrichtung, die an die ERROR-Leitung angeschlossen ist, es ermöglichen, Fehlerinformation mit Hilfe von Slavezugriffen abzufragen (beispielsweise durch ein abfragbares Fehlerregister).

2.3.6. Steuerschaltungen in den Funktionseinheiten

Die Steuerschaltungen zum Stellen von Masteranforderungen und zur Auswahl als Slave können auf einfache Weise realisiert werden. Um dies zu demonstrieren, zeigt Bild 43 eine elementare Masteranforderungsschaltung (ohne Vorkehrungen für den Burstmodus).

Flip-Flop A bewirkt, daß die interne Anforderung mit der Vorderflanke von HOLD als REQUEST-Signal auf dem Bus wirksam wird. Die Vergleichsschaltung B vergleicht ständig die Belegung der PRIORITY-Leitungen mit dem intern eingestellten Prioritätskode. Trifft am Flip-Flop C ein SELECT-Impuls ein und entspricht die Belegung der PRIORITY-Leitungen dem internen Prioritätskode, wird das Steuersignal MASTER BUSY aktiv. Dies veranlaßt direkt das Beaufschlagen des Bussystems mit Adressen, Begleitsignalen usw. Fällt SELECT ab, so wird die BUSY-Leitung aktiviert. Damit wird die Kompensation des Schrägversatzes (engl. deskewing) der Busbelegung gegenüber BUSY auf einfachste Weise erreicht. (BUSY darf erst dann erregt werden, wenn die Busbelegung eingeschwungen ist.)

Stimmt die Belegung der PRIORITY-Leitungen nicht mit der internen Priorität überein oder wurde überhaupt keine Busanforderung gestellt, wird der SELECT-Impuls zur benachbarten Einrichtung weitergegeben. Der Buszyklus wird beendet (MASTER BUSY wird ausgeschaltet, damit wird der Bus freigegeben, und BUSY fällt ab), wenn die interne Busanforderung erfüllt wurde (als Folge von REPLY) oder wenn RELEASE empfangen wurde.

Bild 44 illustriert die Erkennung der Slaveauswahl in den Funktionseinheiten. Der Vergleicher A vergleicht ständig die Belegung der höchstwertigen Adressensignale mit der fest eingestellten Slaveadresse der Einrichtung. Wird bei Gleichheit das BUSY-Signal empfangen, so wird eine interne Anforderung für einen Slavezugriff (SLAVE REQUEST) wirksam. Weiterhin ist dargestellt, wie das Signal SELECTIVE RESET auf einfache Weise ausgewertet werden kann.

Weitere Details dieser Steuerschaltungen, namentlich im Zusammenhang mit Mikrorechneranordnungen, sollen im folgenden Kapitel 3. erläutert werden.

2.3.7. Zeitliche Bedingungen

Die folgenden Angaben zeitlicher Anforderungen sollen einen Überblick über die Leistungsfähigkeit des Bussystems und die Anforderungen an die entsprechenden Schaltmittel geben.

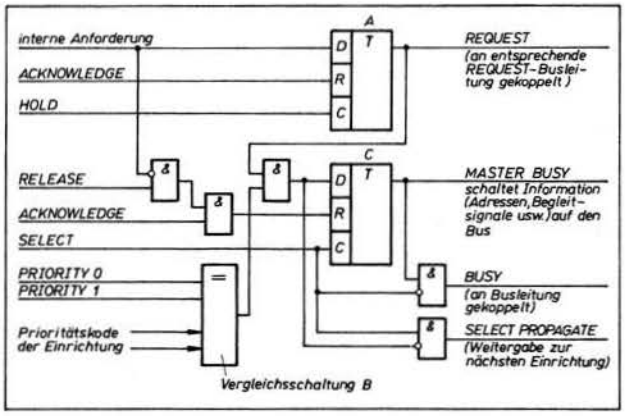


Bild 43: Einfache Masteranforderungsschaltung in einer Funktionseinheit

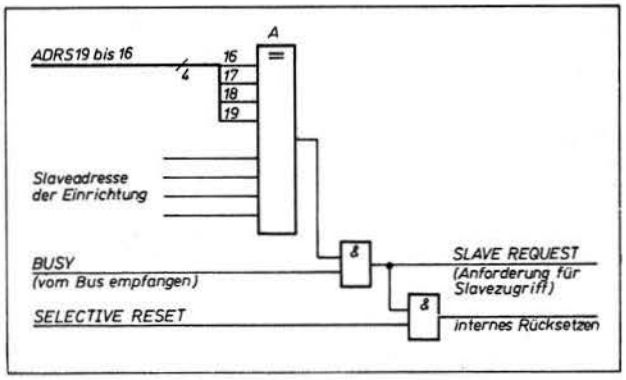


Bild 44: Schaltung zur Erkennung der Auswahl als Slave in einer Funktionseinheit

- Abstand zwischen den Buszyklen
  Zwischen dem Abfall von BUSY und dem Einschalten von HOLD (bzw. BUSY im Burstmodus) muß der Abstand mindestens 200 ns betragen. In dieser Zeit müssen alle Einrichtungen, die am letzten Zyklus beteiligt waren, den Bus freigeben. Bei kontinuierlicher Arbeit des Bussystems (es ist weder BURST MODE eingeschaltet noch liegen besondere Bedingungen vor) darf der Abstand zwischen dem Abfall von BUSY und dem Einschalten von HOLD maximal 400 ns betragen.
- Busbelegung (Adressen, Daten, Begleitsignale) gegenüber BUSY
  Die Busbelegung mit Ausnahme der vier höchstwertigen Adressensignale muß mit dem Einschalten von BUSY gültig sein. Die genannten Adressensignale müssen wenigstens 100 ns vor Einschalten von BUSY gültig sein, um die Slaveauswahl zu gewährleisten. Für die anderen Signale muß die Slaveeinrichtung einen eventuellen internen Versatz gegenüber BUSY selbst kompensieren. Bei Leseoperationen müssen die Datenleitungen wenigstens 100 ns vor dem Einschalten von REPLY gültig sein. Die Gültigkeit der Busbelegung muß bis zum Ausschalten der internen Anforderung bzw. bis zum Einschalten von ACKNOWLEDGE (je nachdem, was eher wirksam wird) aufrechterhalten werden.
- Dauer von impulsförmigen Bussignalen

HOLD	100...200 ns
SELECT	200...300 ns
SELECTIVE RESET	2...20 µs
RESET	2 µs...2 ms
AUX ACKN PULSE	50...100 ns

- Mindestdauer von Antwortsignalen
  REPLY bzw. RELEASE müssen wenigstens 100 ns aktiv sein, um als Antwort des Slave akzeptiert zu werden. Diese Bewertungszeit sollte 200 ns nicht übersteigen.
- Abstand zwischen HOLD und PRIORITY 1,0
  Der Abstand zwischen dem Abfall von HOLD und dem Einstellen der aktuellen Belegung der PRIORITY-Leitungen beträgt maximal 100 ns.
- Verzögerung von HOLD zu REQUEST 4 bis 1
  Spätestens 100 ns nach dem Einschalten von HOLD müssen die REQUEST-Signale an der Bussteuerung eintreffen.
- Abstand zwischen PRIORITY 1,0 und SELECT
  Der Abstand zwischen dem Einstellen der aktuellen PRIORITY-Belegung und dem Erregen von SELECT beträgt zwischen 100 ns und 200 ns.

Diese Bedingungen wurden so definiert, daß sie einerseits mit herkömmlichen Schaltmitteln (TTL-Schaltkreisen, U-880-Mikroprozessoren) zu erfüllen sind und andererseits nicht zu einer übermäßigen Leistungsminderung führen.

2.3.8. Bussteuerung

Die Funktionseinheit, die als zentrale Bussteuerung (BUS CONTROL) eingesetzt ist, besteht aus einer hart verdrahteten Schaltung zur Ablaufsteuerung und Überwachung der Buszyklen und aus einer Mikrorechneranordnung (Bild 45). Diese ist eine elementare U-880-Konfiguration mit folgenden Ausstattungsmerkmalen: 10 Kbyte PROM, 1 Kbyte RAM, 4 PIOs, 1 CTC und spezielle Ankopplungsschaltungen für die Steuermittel des Bussystems, darunter sieben Auffangregister, in denen die Belegung des jeweils letzten Buszyklus gehalten wird und die vom Mikrorechner gelesen werden können.
Hinsichtlich des Bussystems bestehen die wesentlichen Aufgaben der Mikrorechneranordnung in der Initialisierung nach dem Einschalten, der Testung (durch Abarbeitung spezieller Testprogramme) und der Behandlung von Fehlern, die von Überwachungsschaltungen gemeldet werden.

Während des normalen Betriebes wird der Mikrorechner praktisch nicht zur Steuerung des Bussystems benötigt. Er kann deshalb im Rahmen des Systems andere Aufgaben übernehmen. Er ist selbst eine Einrichtung am Bus, wenngleich mit einigen Besonderheiten. Er kann als Master zu allen anderen Einrichtungen zugreifen und hat dafür die absolut höchste Priorität (REQUEST 4; erste Position am Bus). Als Slave ist er unter der festen Adresse FH erreichbar, aber nur zu dem Zweck, Interrupts zu empfangen. Somit sind im Rahmen des Gesamtsystems nur Anwendungsfälle der Art möglich, bei der Aufträge von anderen Einrichtungen (durch Interrupt signalisiert) entgegengenommen und ausgeführt werden.

Besondere Zustände bzw. Fehlersignale aus den Bussteuerschaltungen werden auf folgende Weise an den Mikrorechner übermittelt:

- Fehlersignale während des normalen Betriebs, wenn der Mikrorechner den Bus nicht selbst benutzt: Interrupt über Port A der ersten PIO
- Fehlersignale des Bussystems, wenn der Mikrorechner selbst als Master den Bus belegt hat: Rücksetzen der CPU, worauf die Software zu einer speziellen Behandlungsroutine verzweigt
- Empfang von RELEASE, wenn der Mikrorechner selbst als Master den Bus belegt hat: NMI.

Der Mikrorechner kann in der Steuerhardware selbst Wirkungen auslösen. Dazu können mit E-A-Befehlen einzelne ACTION-Signale erzeugt werden (über Dekoder, die an den Datenbus angeschlossen sind). Die Tafel 2 zeigt das zugrundeliegende Datenformat und die Steuerwirkungen, die damit auslösbar sind. Die Steuer- und Überwachungsschaltungen für das Bussystem sind in den Bildern 46 bis 51 überblicksmäßig dargestellt.

Tafel 2: Übersicht über die ACTION-Kodes zur Auslösung von Steuerwirkungen in der speziellen Bussteuerhardware

7 … 4	3 … 0
EMIT-Feld	ACTION-Kode

ACTION-Kode (hex.)	Funktion	Belegung des EMIT-Feldes
0	NOP	—
1	Ablaufsteuerung einschalten	—
2	Ablaufsteuerung ausschalten	—
3	NES einschalten	—
4	NES ausschalten	—
5	Sicherungsregister laden	—
6	Ablaufsteuerung zurücksetzen	
7	EMIT STATUS (Laden des Statusregisters)	7 6 5 4 [X] — ENABLE WATCHDOG, REGISTRATION MODE, ERROR MASK
8	Steuerung der Rücksetzabläufe	7 6 5 4 — RESET DVC, ADRS TO BUS, SET BUSY, SELECTIVE RESET, RESET
9 … F	anderweitige Wirkungen bzw. nicht belegt	

Das wesentlichste Mittel der Fehlererkennung ist die zeitliche Überwachung der Buszyklen. Ein normaler Buszyklus wird vom Anstieg des REQUEST-Signals bis zum Ende des Buszyklus (END OF CYCLE) überwacht, ein Zyklus im Burstmodus vom Anstieg des BUSY-Signals bis END OF CYCLE. Die Zeitbegrenzung ist auf etwa 20 μs eingestellt. Da die Steuerschaltungen so ausgelegt sind, daß nur zulässige Signalfolgen zum Ende des Zyklus führen, äußern sich Fehler in der Regel in einem „Hängenbleiben", wodurch TIMEOUT signalisiert wird. Dies veranlaßt in der Hardware eine Übernahme der (fehlerhaften) Busbelegung in die Auffangregister, so daß eine programmtechnische Fehleranalyse möglich ist.

Weitere Fehlersignale sind

ASYNC ERROR

Die Fehlerleitung ERROR des Bussystems wird als asynchroner Fehler ausgewertet, wenn das entsprechende Bit (ERROR MASK) im Statusregister gesetzt ist.

REQUEST STUCK

Es wird überwacht, ob alle REQUEST-Leitungen inaktiv

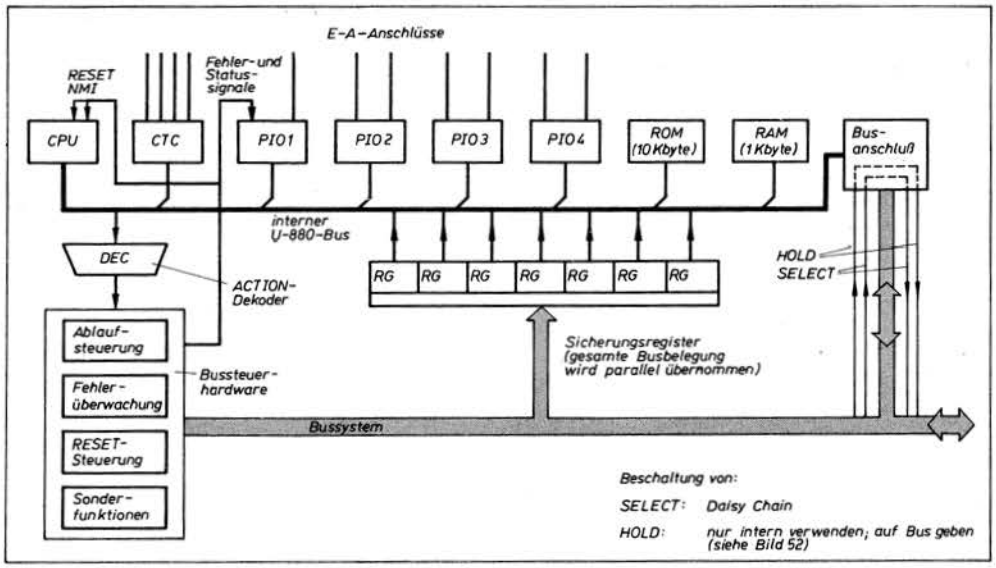


Bild 45: Prinzipschaltung der zentralen Bussteuerung, die aus einer Mikrorechneranordnung und spezieller Steuerhardware besteht

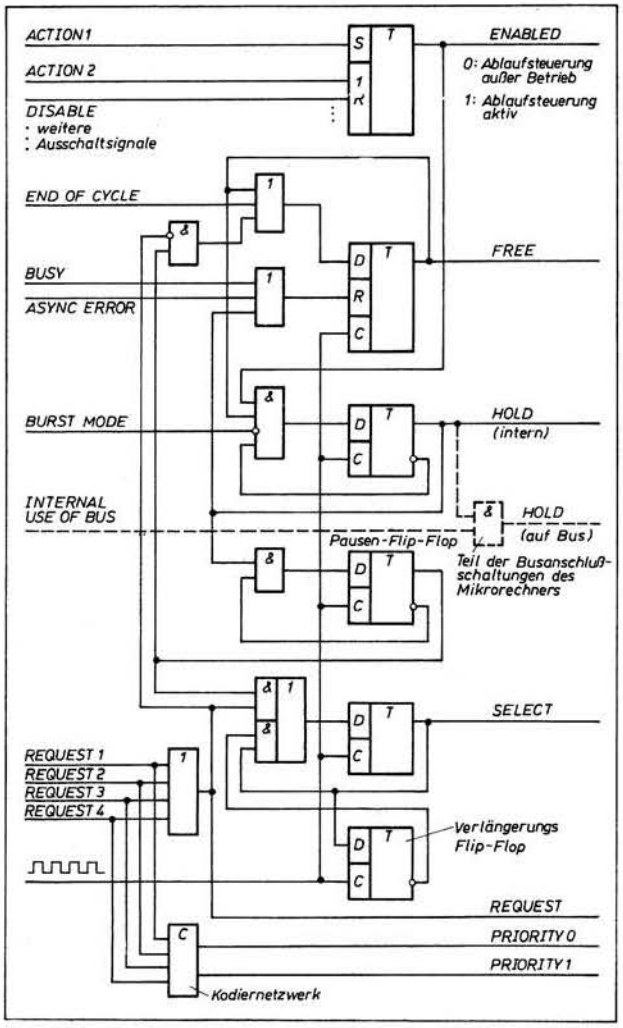


Bild 46: Teil der Ablaufsteuerung für die Masterauswahl

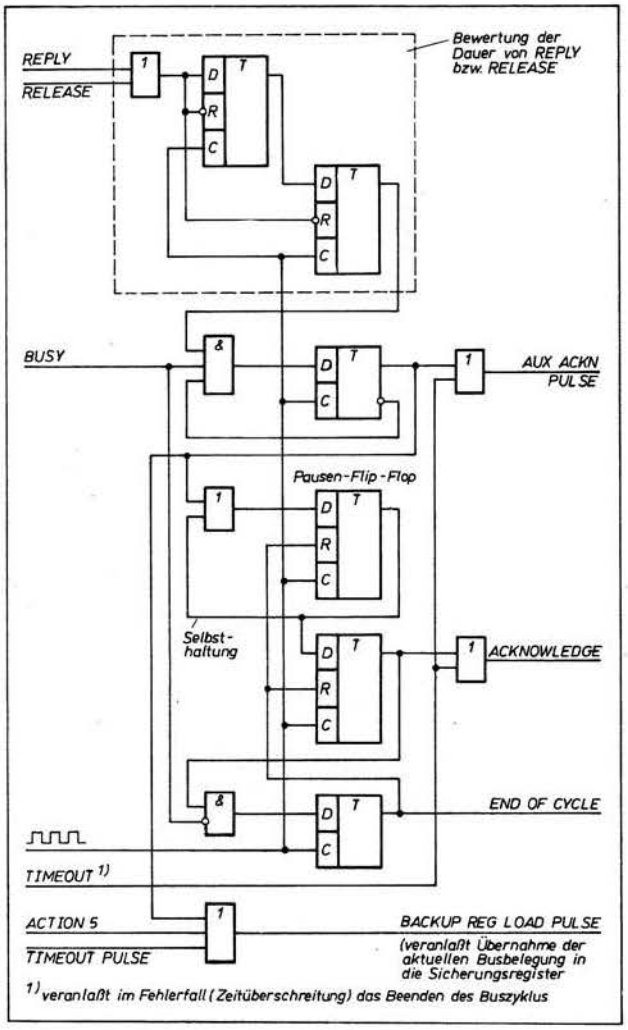


Bild 47: Teil der Ablaufsteuerung für das Beenden der Buszyklen

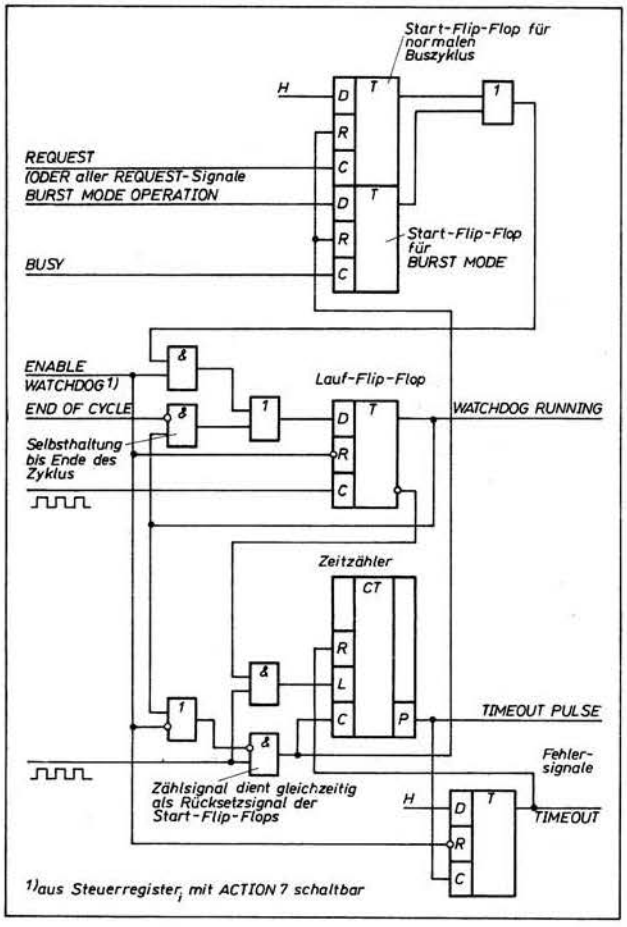


Bild 48: Kontrollschaltung zur Überwachung der Dauer der Buszyklen

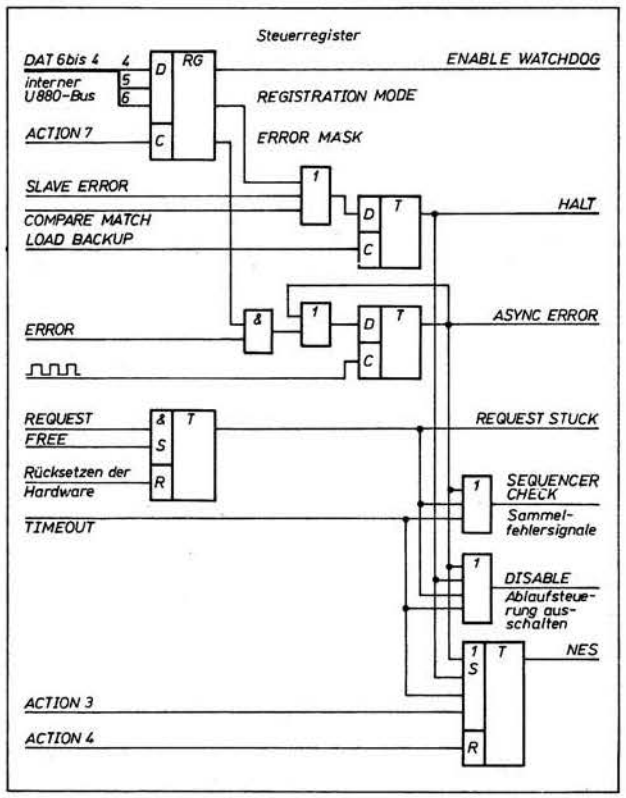


Bild 49: Bildung der Fehler- und Zustandssignale der Ablaufsteuerung sowie des Bussignals NES

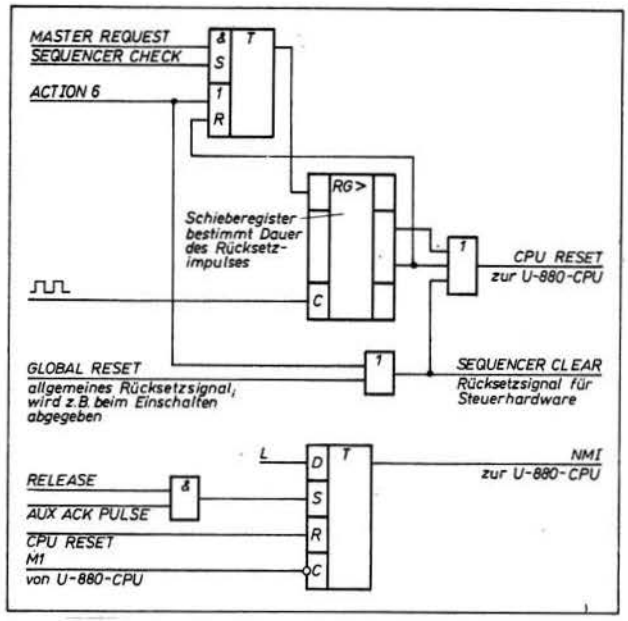


Bild 50: Bildung der Rücksetz- und NMI-Signale

sind, wenn kein Buszyklus ausgeführt wird (FREE ist aktiv).

Das HALT-Signal zeigt an, daß der Bus nach dem aktuellen Zyklus angehalten wurde (es wird keine Vermittlung mehr durchgeführt). Ursachen dafür sind:

- Slave hat Fehler durch SLAVE ERROR signalisiert
- Slave hat Vergleichsstoppbedingungen durch COMPARE MATCH signalisiert
- im Statusregister ist REGISTRATION MODE eingeschaltet.

Die letztere Bedingung ermöglicht ein programmtechnisches Beobachten der Buszyklen. Nach jedem Zyklus wird der Bus angehalten, so daß die Busbelegung (über die Auffangregister) von Programmen untersucht werden kann. Die Datenrate des Bussystems reduziert sich dadurch allerdings beträchtlich.
Die multivalente Nutzung des Mikrorechners (er kann den Bus selbst als Master benutzen und auch von anderen Einrichtungen beauftragt werden, bestimmte Aufgaben auszuführen) bringt einige Probleme mit sich, die mit den Geschwindigkeitsverhältnissen zwischen Befehlsausführung und Buszyklen und mit dem Unterbrechungsverhalten der U-880-CPU zusammenhängen. U-880-Befehle sind in ihrem Ablauf nicht unterbrechbar (mit Ausnahme von LDIR usw.), und die Buszyklen sind schneller als die Befehlsausführung. Wird beispielsweise ein Bitmodifikationsbefehl [etwa SET 3, (HL)] so ausgeführt, daß das zu modifizierende Bit in einer Slaveeinrichtung liegt (die Adresse in HL zeigt auf eine Position in einer Slaveeinrichtung), so enthält dieser Befehl zwei Buszyklen (Byte holen; das modifizierte Byte zurückschreiben). Zwischen diesen beiden Zyklen können anderweitige Zugriffe über den Bus ausgeführt werden (von anderen Einrichtungen). Veranlaßt einer dieser Zugriffe nun eine Unterbrechungsanforderung für die CPU der Bussteuerung, so wird die Bussteuerhardware angehalten (damit der Inhalt der Auffangregister für die spätere Analyse erhalten bleibt), die Unterbrechung wird aber nicht wirksam (da die CPU noch den Bitmodifikationsbefehl abarbeitet). Der zweite Operandenzugriff (Schreibzugriff) dieses Befehls findet somit eine angehaltene Bussteuerung vor, so daß das gesamte System hängenbleibt. Deshalb muß die Bussteuerung in den Zeitabschnitten, in denen sie den Bus selbst benutzen will, verhindern, daß andere Einrichtungen den Bus belegen. Dazu ist in einem Steuerregister die Bitposition INTERNAL USE OF BUS vorgesehen, die programmtechnisch ein- und ausgeschaltet werden kann. Ist INTERNAL USE OF BUS aktiv, so wird die Weitergabe der HOLD-Impulse verhindert. Damit werden Gesuche anderer Einrichtungen nicht wirksam.

Weiterhin wird in diesen Zeitabschnitten die Leitung NES erregt. Das zeitliche Verhalten ist im Bild 52 dargestellt. INTERNAL USE OF BUS ist ein Signal, das softwareseitig ein- und ausgeschaltet wird. Ist das Signal aktiv, werden die HOLD-Impulse nur intern ausgewertet und gelangen nicht zu den anderen Einrichtungen, so daß für jene keine Busvermittlung stattfindet. Während des Umschaltens der Betriebsart wird die Ablaufsteuerung angehalten (ACTION 2) und jeweils anschließend wieder gestartet (ACTION 1).

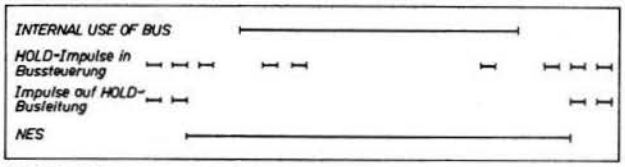


Bild 52: Schema der Umschaltung des Bussystems, wenn die Bussteuerung selbst als Master aktiv werden will

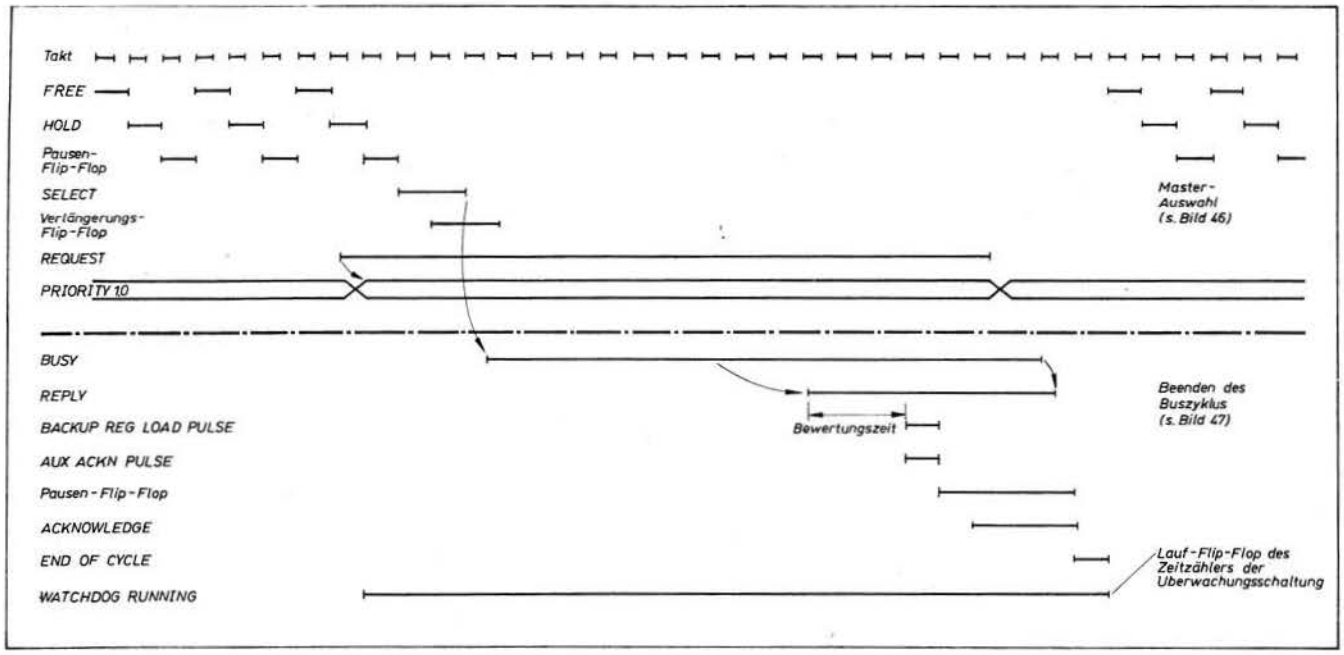


Bild 51: Impulsdiagramm der Ablaufsteuerung für einen normalen Buszyklus

2.3.9. Diskussion

Wesentliche Kriterien bei der Definition des Bussystems waren Funktionssicherheit, geringer Aufwand und angemessene Datenrate.
Um eine hohe Funktionssicherheit zu gewährleisten, wurden die zeitlichen Anforderungen so festgelegt, daß die Schaltkreise auch unter Worst-Case-Bedingungen die Spezifikationen noch erfüllen können. Weiterhin wurden die Steuerfolgen so definiert, daß Wettlauferscheinungen prinzipiell ausgeschlossen sind und daß die gegenseitigen Störbeeinflussungen minimal bleiben. Diese Aspekte betreffen im wesentlichen

- die Einführung eines HOLD-Signals zur Synchronisation des Einschaltens der REQUEST-Leitungen (Gesuche werden nur in dem definierten Zeitintervall nach dem HOLD-Impuls ausgewertet; da dabei keine neuen Gesuche entstehen können, gibt es keine Hazards bei der Prioritätsfestlegung)
- das Mindestzeitintervall, in dem REPLY bzw. RELEASE aktiv sein müssen, bevor AUX ACKN PULSE abgegeben wird (zur Eliminierung der Wirkung von Störimpulsen auf diesen Leitungen)
- die Einführung des AUX-ACKN-PULSE-Signals (damit wird in den Einrichtungen der interne Abschluß des Buszugriffs vom Abschalten der Busleitungen zeitlich definiert getrennt, so daß dabei auftretende Störimpulse wirkungslos bleiben).

Um den Aufwand möglichst gering zu halten, wurden die Steuerfolgen so festgelegt, daß die Buskopplung in den Einrichtungen mit möglichst einfachen Mitteln erfolgen kann, namentlich dann, wenn nicht alle Sonderfunktionen des Bussystems benötigt werden (die Realisierung der Schaltungen gemäß den Bildern 43, 44 erfordert weniger als zehn Schaltkreise).
Das Prinzip der Masterauswahl ist praktisch eine Kombination des Daisy-Chain-Prinzips mit dem der unabhängigen Anforderungen. Es gestattet, das Bussystem mit Flachbandleitungen zu realisieren. Dabei gibt es keine individuellen Verbindungen, sondern nur eine Daisy-Chain-Leitung. Für diese können gemäß Bild 53 die beiden außenliegenden Adern in einer Flachbandleitung benutzt werden. Damit wechselt bei jeder Einrichtung die physische Lage der Anschlüsse für die ankommende und die abgehende SELECT-Leitung zwischen Oben und Unten.
Es ist aber unproblematisch, auf den Leiterplatten Umschaltmöglichkeiten, etwa DIL-Schalter oder Drahtbrücken, vorzusehen, damit auf jeder Funktionseinheit die korrekte Verbindung entsprechend der Position am Bus eingestellt werden kann. Die zwischen zwei Einrichtungen nicht benötigte Leitung muß gemäß Bild 53 aufgetrennt werden. Umfaßt das System nur vier Mastereinrichtungen (einschließlich der Bussteuerung), kann auf das Auftrennen verzichtet werden (da jeder Einrichtung eine Priorität zugeordnet werden kann, wird die Daisy-Chain-Auswahl nicht benötigt).
Das Bussystem wurde weiterhin darauf ausgelegt, daß zur Testung, zur Fehlerbehandlung sowie für Sonderfunktionen die zentrale Bussteuerung mit einem Mikrorechner ausgerüstet werden kann. Dies ergibt eine praktisch unbegrenzte Flexibilität bei akzeptablen Kosten. Da eine ausgiebige programmtechnische Testung des Systems möglich ist, wurde auf weitergehende Kontrollmaßnahmen (z. B. Paritätsprüfung der Daten- und Adressenleitungen) verzichtet. Die Datenrate des Bussystems liegt bei etwa 300 Kbyte/s, kommunizieren spezielle Hardwareeinrichtungen ohne Mikrorechner miteinander, so kann sie 600 Kbyte/s überschreiten. Dies ist dem Einsatzfall angemessen. Üblicherweise werden Befehle aus dem lokalen Speicher eines Mikrorechners abgearbeitet. Buszugriffe werden in den meisten Fällen zum Lesen bzw. Schreiben von Operanden ausgeführt. Die schnellste sinnvolle längere Folge von Operandenzugriffen wird beim U 880 mit Blocktransportbefehlen erreicht. Die Mindestzeit je Byte beträgt dabei 21 Maschinenzyklen (das sind bei normaler Arbeitsgeschwindigkeit etwa 8 μs). Damit hat es kaum Sinn, eine übermäßig hohe Datenrate für das Bussystem anzustreben, zumal in der Regel nur etwa 20...30 % der Operandenzugriffe eines Mikrorechners Buszugriffe sind. Für Systeme mit prinzipiell anderen Verhältnissen müssen diese Überlegungen selbstverständlich sinngemäß modifiziert werden.

2.4. Weiterführende Betrachtungen

2.4.1. Modifikationen des Beispiels

Andere Randbedingungen führen dazu, daß andere Lösungen als günstiger erscheinen. Dies soll anhand des vorstehend diskutierten Realisierungsbeispiels erläutert werden. So wäre beim Aufbau des Gesamtsystems in Paneelbauweise das Vermittlungsprinzip der unabhängigen Anforderungen vorzuziehen, sofern dafür genügend Steckverbinderkontakte verfügbar sind. Damit reduziert sich (bei sonst unveränderten Gegebenheiten) die Zeit für die Masterauswahl von etwa 600 ns auf etwa 200 ns.
Für höhere Anforderungen an den Datendurchsatz können für die Steuersignale statt einfacher Koppelschaltungen aufwendigere (z. B. Open-Collector-Baustufen) eingesetzt werden. Damit könnte die Signalisierung des AUX ACKN PULSE sowie die Forderung nach einer Mindestdauer des

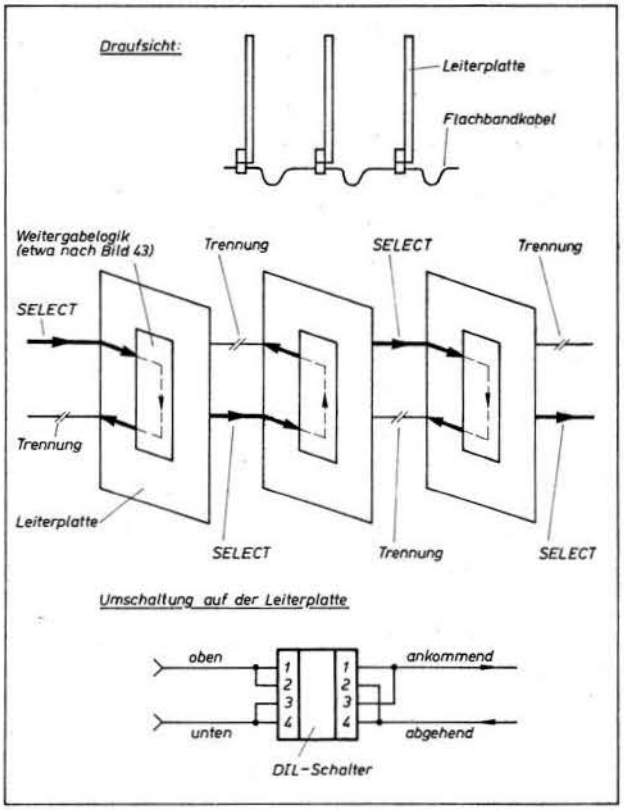


Bild 53: Prinzip der konstruktiven Ausführung der Busverbindungen mit Flachbandkabeln

Antwortsignals entfallen. Dies würde den Buszyklus um weitere 200...300 ns verkürzen.

2.4.2. Elektrische Gesichtspunkte

Für weiter ausgedehnte Bussysteme sollten im Interesse einer hohen Störsicherheit folgende Maßnahmen erwogen werden:

- Alle Steuerleitungen als aktiv L definieren und mit Pull-up-Widerständen versehen!
- In der Leitungsführung zwischen jeder Signalleitung zwei Masseleitungen anordnen!
- Gewährleisten, daß in physisch benachbarten Leitungen die Richtung des Signalflusses gleich ist, um Störungen gering zu halten!
- Die Flankensteilheiten der Bussignale so gering wie möglich halten (Verschleifen der Flanken)! Diese Maßnahme trägt nicht nur zur Verminderung der internen Störungen, sondern auch zur Verringerung der Funkstörstrahlungen bei.

2.4.3. Diagnostische Maßnahmen

Es hat sich bewährt, in einigermaßen komplexen Systemen einen Mikrorechner für Aufgaben der Initialisierung, Diagnose, Testung, Fehlerbehandlung usw. vorzusehen. Der Versuch, diesen Mikrorechner (der bei normal funktionierendem System praktisch unbeschäftigt ist) für Verarbeitungsaufgaben auszunutzen, führt zwar zu einer kostengünstigen Hardware, bringt aber in der Entwicklungsphase schaltungstechnische und softwaremäßige Probleme mit sich. Für Systeme, bei denen die Hardwarekosten nicht die entscheidende Rolle spielen, sollte diese multivalente Nutzung unterbleiben, auch wenn dadurch ein weiterer Mikrorechner benötigt wird. Stehen z. B. Einchip-Mikrorechner zur Verfügung, kann der Diagnosemikrorechner auch physisch klein gehalten werden. Auf dieser Basis lohnt sich der Einsatz auch in relativ kleinen Systemen, wie z. B. Bürocomputern oder intelligenten Terminals, bei denen ein Zwang zur Minimierung der Hardware- und Fertigungskosten besteht [9].

Zur Fehlerkontrolle hat sich die Kombination aus Testroutinen und relativ globalen Kontrollmaßnahmen allgemein durchgesetzt. Testroutinen dienen zum Nachweis der Funktionsfähigkeit und zur Fehlerlokalisierung. Die Kontrollmaßnahmen signalisieren Fehlersituationen während des Betriebs. Eine Zeitkontrolle wird dabei zumeist als ausreichend angesehen.

Weitergehende Überwachungsmaßnahmen wie Paritätskontrolle von Daten- und Adressenleitungen, Überwachung der Signalpegel sowie der Ströme in den Busleitungen werden nur in seltenen Fällen angewandt. Bei relativ geringer Anzahl von Komponenten im Gesamtsystem, übersichtlichem Aufbau und Betrieb unterhalb der Grenzwerte des Bauelementespektrums kann darauf verzichtet werden. Die Schaffung von Testroutinen zur Überprüfung aller Busleitungen bietet keine prinzipiellen Schwierigkeiten.

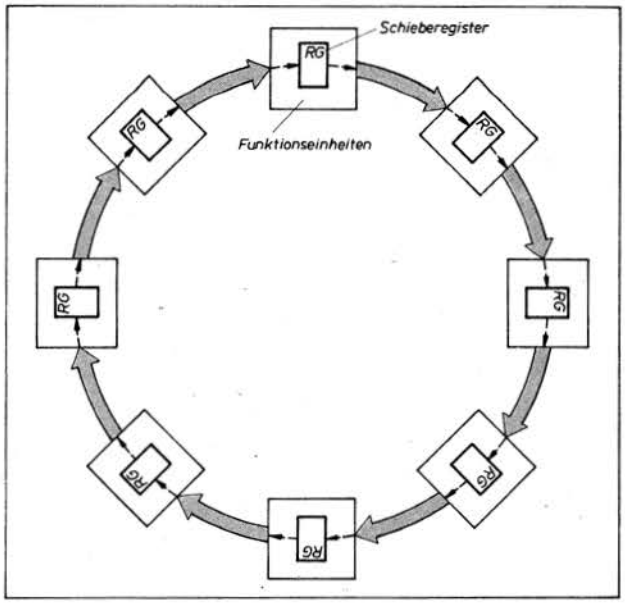


Bild 54: Schematische Darstellung eines Bussystems, das durch eine ringförmige Zusammenschaltung von Schieberegistern realisiert ist

Ein problematischer Aspekt von Tristate- bzw. Open-Collector-Bussystemen ist die Fehlerzuordnung, wenn ein Teilnehmer Busleitungen unberechtigterweise belegt (etwa auf Grund eines defekten Treiberbausteins). Um den Fehler der jeweiligen Funktionseinheit zuordnen zu können, sollte ein Satz von Testroutinen vorgesehen werden, der es gestattet, die Buskommunikation zu prüfen, wenn verdächtige Einrichtungen vom Bus entfernt wurden. Diejenige, nach deren Entfernung die Buskommunikation fehlerfrei abläuft, ist mit hoher Wahrscheinlichkeit fehlerbehaftet.

Neben Mitteln zum Überwachen und Testen der Hardware sind solche für die Fehlersuche in der Software (Debugging) wesentlich. Maßnahmen dafür, die sich bei umfangreichen Systemen auch in eingebauter Form (im Gegensatz zu extern anschließbaren Logikanalysatoren o. ä.) lohnen, sind Ablaufspeicher für eine gewisse Anzahl von Buszyklen (16 bis 1 024 Busbelegungen) und hardwaremäßige Busvergleicher, um bestimmte Belegungen nachweisen zu können, den Bus bei Auftreten bestimmter Bedingungen anhalten zu können usw. Dazu sollte das Bussystem die Identifikation des aktuellen Masters gestatten. Dies ist beim Vermittlungsprinzip der unabhängigen Anforderungen eine triviale Angelegenheit. Ansonsten müssen zusätzliche Busleitungen MASTER IDENT eingeführt werden.

Die ausschließlich softwaremäßige Busbeobachtung ist zwar sehr flexibel bei minimalem Hardwareaufwand, aber

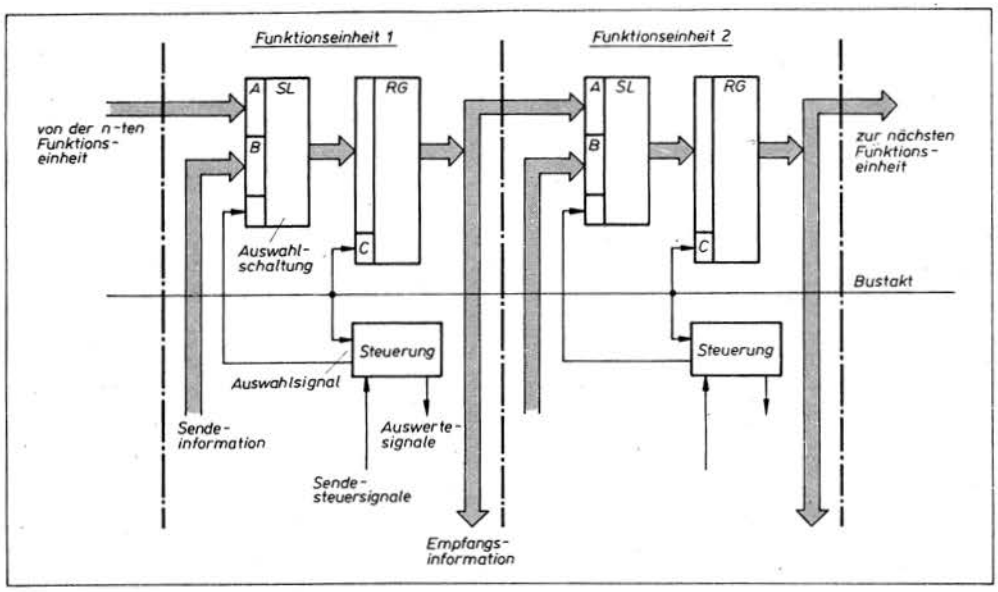


Bild 55: Anschaltung einer Funktionseinheit nach Bild 54

nicht anwendbar, wenn Probleme untersucht werden sollen, die beim komplexen Zusammenwirken der Funktionseinheiten im realen Betrieb auftreten.

2.4.4. Ansätze für zukünftige Lösungen

Für die Konzipierung zukünftiger Bussysteme müssen Möglichkeiten und Trends der Schaltkreisentwicklung sowie die Anforderungen berücksichtigt werden, die sich aus der Komplexität der Software und den Gegebenheiten der Diagnose, Testung und Wartung derartiger Systeme ergeben. Die Möglichkeit, umfangreiche Hardware als LSI-IS kostengünstig zu produzieren, darf nicht lediglich zur bloßen Leistungserhöhung bzw. Reduktion der Fertigungskosten benutzt werden, sondern muß auch dafür eingesetzt werden, Probleme der Software, des Testens, der Wartung usw. beherrschbarer zu machen. Einige Aspekte dazu werden im folgenden angeführt.

- Hinreichende Ausstattung der einzelnen Mikrorechner mit Speichern und E-A-Anschlüssen (und erforderlichenfalls mit speziellen Koprozessoren), so daß einzelne Problemprogramme keine Betriebsmittel (etwa Arbeitsspeicher) außerhalb des Mikrorechners benötigen
- Standardisierung der Buskommunikation nicht nur auf physischer Ebene, sondern auch auf der Ebene des Austauschs von Datenblöcken (definierte Protokolle des Datenaustausches statt wahlfreier Zugriffe zu einzelnen Bytes). Damit kommt der Übertragung von Datenpaketen eine wesentliche Bedeutung zu.
- Verringerung der Anzahl der Busleitungen und Anordnung intelligenter Anschlußschaltungen für das Bussystem.
- Vermeidung von Wired-OR- bzw. Tristate-Verbindungen, um die Diagnose zu verbessern.

Eine bekannte Möglichkeit dazu besteht darin, das Bussystem als eine Kette von Schieberegistern zu realisieren (Bild 54).
Das Bussystem ist als eine Ringstruktur aus Schieberegistern aufgebaut, wobei für jede Funktionseinheit ein Schieberegister vorgesehen ist. Diese Register enthalten

- eine Quellidentifikation
- eine Zielidentifikation
- Adressen
- Daten
- Steuerinformation.

Eine Einrichtung kann Information in ihr Register eintragen, wenn das Register als frei deklariert ist bzw. wenn es die Identifikation der Einrichtung als Quellidentifikation enthält. Informationen können dem Register entnommen werden, wenn es die entsprechende Zielidentifikation enthält. Danach kann die entnehmende Einrichtung den Freizustand einstellen. Die Information durchläuft in entsprechenden Bustakten alle Schieberegister.
Eine derartige Anordnung hat sehr gute Diagnoseeigenschaften. Diese können noch weiter verbessert werden, wenn die einzelnen Register gewissermaßen quer ebenfalls als Schieberegister ausgebildet sind (engl. level scan sensitive design; s. dazu [10]). Damit kann die Busankopplung jeder Einrichtung durch eine zentrale Diagnoseeinheit (üblicherweise ein separater Mikrorechner) weitgehend geprüft werden. Die Busregister können vom Diagnoserechner einzeln bitseriell geladen und gelesen werden. Die Probleme der Fehlerlokalisierung, die bei üblichen Bussystemen die Servicearbeiten beträchtlich erschweren, treten praktisch nicht auf (Bild 56).

Das Prinzip hat allerdings auch einige Nachteile:

- Für eine gegebene Anzahl von Busleitungen ist je Einrichtung etwa die doppelte Anzahl von Kontakten erforderlich.
- Die Leitungsführung ist starr, da die ringförmige Zusammenschaltung stets gewährleistet sein muß. Ein einfaches Zufügen oder Entfernen von Einrichtungen in einer gegebenen Installation ist nicht möglich.
- Der Bus darf wegen der Toleranzprobleme des Schiebetaktes physisch nicht übermäßig ausgedehnt sein.
- Die Zeit zwischen Absenden der Daten und deren Auswertung wächst mit der Entfernung zwischen sendender und empfangender Einrichtung, bei paketweiser Datenübertragung wird dieser Nachteil jedoch unerheblich.
- Bestimmte Funktionen wie kurzzeitiges Ausschließen anderer Einrichtungen von der Busbenutzung, Steuerung der Prioritätszuordnung usw., die bei üblichen Bussystemen keinen sonderlichen Aufwand erfordern, werden problematisch.

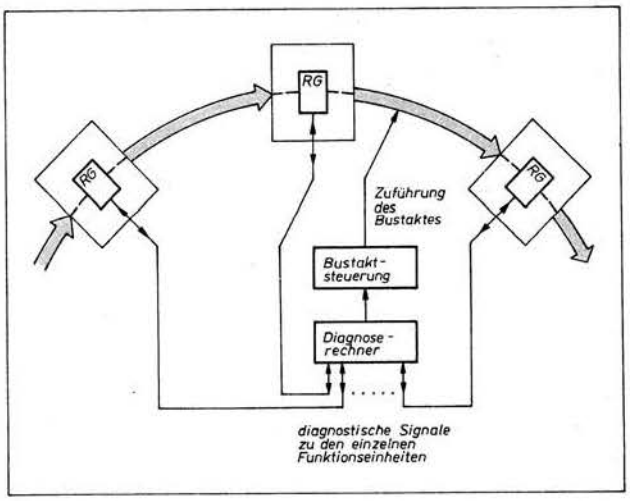


Bild 56: Verbesserung der Struktur nach Bild 54 hinsichtlich der Diagnoseeigenschaften

Eine weitere Entwicklung besteht darin, Mikrorechner für den Anschluß an einen Paketbus auszulegen, wobei die physische Struktur des Bussystems den konkreten Leistungsanforderungen angepaßt werden kann. So kann in einem kleinen System eine byteparallele Datenübertragung vorgesehen sein, während für ein Hochleistungssystem die Busstruktur so gewählt werden kann, daß mehrere Bytes eines Paketes parallel übertragen werden [12].

Literatur

[5] Thurber, K.; Jensen, E.: A systematic approach to the design of digital bussing structures. Fall Joint Computer Conference 1972. Proceedings Vol. 41, Part II, S. 719–740

[6] A mix of standard and proprietary buses marks the latest µC systems. Electronic Design, New York 21 (1983) 17. 3., S. 101–113

[7] Buses form the backbone of computer systems. Electronic Design, New York 30 (1982) 23. 12., S. 117–132

[8] WP G 06 F/227 967 4: Bussystem zur Verbindung von logischen Funktionsmoduln

[9] Edwards, R. C.: SOS Technology Yields Low-Cost HP 3000 Computer Systems. Hewlett-Packard Journal, Palo Alto 30 (1979) 9, S. 3–8

[10] Eichelberger, E. B.; Williams, T. W.: A Logic Design Structure for LSI Testability. 14th Annual Design Automation Conference, New Orleans (1977) Juni, S. 462–467

[11] Mucha: The Complexity Problem in Design, Verification and Testing. Digital Technology, Status and Trends (Fachberichte und Referate, Band 12). München, Wien: Oldenbourg Verlag 1981

[12] Rottner, J.; Lattin, W. W.: Ada determines architecture of 32-bit microprocessor. Electronics, New York 54 (1981) 4, S. 119–126

3. Mikrorechner

3.1. Prinzipien

Die Mikrorechnerbaugruppen sind die wesentlichen Funktionseinheiten in Multimikrorechnersystemen. Durch ihren Einsatz sollen die Aufwendungen für zusätzliche bzw. spezielle Hardwareeinrichtungen so gering wie möglich gehalten werden, so daß eine Kombination von mehreren Mikrorechnern mit speziellen Schaltungen vergleichsweise geringen Aufwandes ökonomische Vorteile gegenüber konventionellen Lösungen hat (s. a. Kapitel 1.).
Dies bedingt besondere Sorgfalt bei der Auslegung der Mikrorechnerbaugruppen. Es ist klar, daß die gewünschten ökonomischen Effekte nur bei deren weitgehender Standardisierung eintreten können und nicht dadurch, daß für jeden Einsatzfall statt spezieller Schaltungen konventioneller Auslegung spezielle Mikrorechneranordnungen geschaffen werden. Standardisierung bringt hier nicht nur die traditionell bekannten Vorteile in die Produktion der Hardware und nachfolgend in Wartung, Service usw., sondern ist auch von entscheidender Bedeutung für die Beherrschbarkeit der

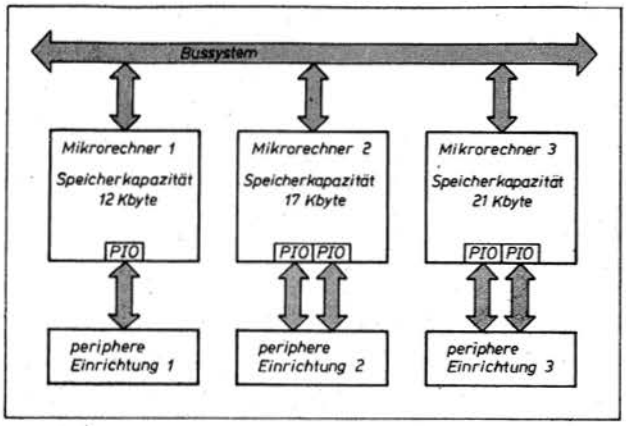


Bild 57: Ausschnitt aus Multimikrorechnersystem mit spezialisierten Mikrorechnern

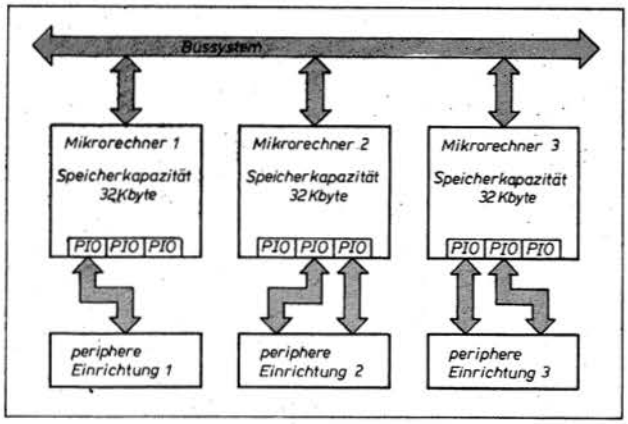


Bild 58: Ausschnitt aus Multimikrorechnersystem mit standardisierten Mikrorechnern

Softwareentwicklung hinsichtlich der Kosten und des Zeitbedarfes. Dies ist bei Systemen mit umfangreicher und komplizierter Software so wesentlich, daß lokale Hardwaremehraufwendungen zumeist akzeptiert werden können.

Bild 57 zeigt eine Anordnung aus drei Mikrorechnern sowie die Anforderungen an Speicherkapazität und E-A-Anschlußmöglichkeiten. Zugrundegelegt wurde das U-880-Mikroprozessorsystem. Dagegen ist im Bild 58 eine Möglichkeit dargestellt, diese Anordnung mit drei standardisierten Mikrorechnern zu realisieren.

Bild 57 soll den Fall veranschaulichen, daß die Mikrorechnerhardware jeweils gemäß den lokalen Anwendungserfordernissen ausgelegt wird. Derartige Zielstellungen führen in der Praxis häufig dazu, daß die einzelnen Schaltungskomplexe lokal optimiert werden. Ein typisches Beispiel dafür ist die Aufwandsoptimierung der Adressendekodierung (Bild 59). Der dadurch mögliche Programmiertrick funktioniert auf Mikrorechnern mit anderer Adressendekodierung nicht! Dies hat zur Folge, daß für die einzelnen Mikrorechnertypen die Software gesondert erstellt werden muß.

Die Lösung nach Bild 58 hat demgegenüber offensichtliche Vorteile:

- Die Software ist portabel; funktionell gleichartige Programme brauchen nur einmal entwickelt zu werden.
- Es entfällt die Hardwareentwicklung von zwei Mikrorechnerbaugruppen (mit all ihren Konsequenzen hinsichtlich der Überleitung in die Produktion usw.).
- Für den Fall, daß die funktionsbestimmenden Programme in RAMs untergebracht sind, ist es möglich, in einem System die Mikrorechnerleiterplatten untereinander zu tauschen. Dies erleichtert die Fehlerlokalisierung und vermindert den Bedarf an Ersatzleiterplatten.
- Die auf Grund der Standardisierung vorhandenen Überschüsse an Speicherkapazität und E-A-Anschlußmöglichkeiten gestatten unproblematisch Änderungen bzw. Erweiterungen, was bei einer knapp bemessenen Hardware nicht ohne weiteres möglich wäre.

Allerdings zeigt Bild 58 auch, daß der Mehraufwand aus Standardisierungsgründen für einzelne Mikrorechner recht hoch sein kann. Um diesen Nachteil zu mildern, die Vorteile der Standardisierung aber nicht gänzlich aufzugeben, bieten sich folgende Alternativen an:

- Selektive Bestückung der Baugruppen mit Speicher- und E-A-Schaltkreisen gemäß den jeweiligen konkreten Bedürfnissen
- Schaffung einer geringen Zahl standardisierter Baugruppen mit jeweils charakteristischer Auslegung für eine gewisse Klasse von Einsatzfällen
- interne Modularisierung des einzelnen Mikrorechners, was praktisch der selektiven Bestückung entspricht, aber eine geeignete konstruktive Auslegung erfordert (s. a. Abschnitt 1.3.).

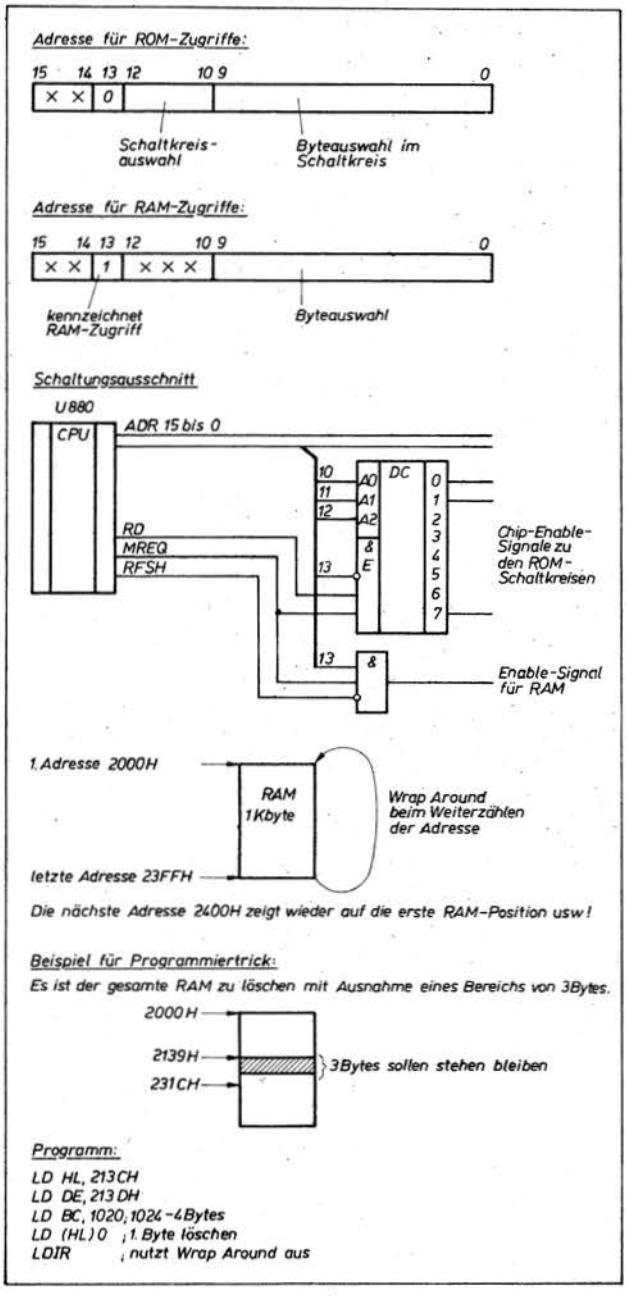


Bild 59: Beispiel für lokale Optimierung der Adressendekodierung

- Schaffung einer einzigen, relativ knapp ausgelegten Mikrorechnerbaugruppe mit der Möglichkeit, viele davon einzusetzen. Diese Lösung kann wirtschaftlich und leistungsfähig sein, ist aber nur in bestimmten Fällen anwendbar. Ist beispielsweise der Mikrorechner nur mit zwei PIOs ausgerüstet, während die zu steuernde Einrichtung vier benötigt, so müssen zwei Mikrorechner vorgesehen werden (Bild 60). Für die Koordinierung der E-A-Steuerung zwischen beiden Mikrorechnern ist ein recht kompliziertes Programm erforderlich, dessen Laufzeit die Steuerabläufe so verlangsamen kann, daß die gewünschte Funktion nicht realisierbar ist.

Um Mikrorechnerbaugruppen richtig auszulegen, müssen drei Punkte genauer untersucht werden.

3.1.1. Wirkungsmöglichkeiten auf andere Funktionseinheiten

Ein Mikrorechner kann nur dadurch auf andere Funktionseinheiten einwirken, daß er entsprechende Befehle ausführt, die lediglich Zugriffe zu Speicher- bzw. E-A-Adressen veranlassen. Manche Mikroprozessoren erzeugen keine speziellen E-A-Adressen, sondern behandeln Ein- und Ausgabeoperationen wie Speicherzugriffe (Memory Mapped I/O). Somit sind Zugriffe zu anderen Funktionseinheiten nur möglich, indem Operationen in besonderen Adressenbereichen ausgeführt werden. Der Wunsch des Mikrorechners, eine solche Operation auszuführen, ist durch zusätzliche Schaltungen (namentlich durch Dekodieren der betreffenden Adressenbits) auf einfache Weise erkennbar. So kann beispielsweise eine Masteranforderung für das gemeinsame Bussystem gestellt werden. Weiterhin lassen sich zusätzliche Funktionen durch spezielle Steuersignale auslösen, die von E-A-IS bzw. programmtechnisch ladbaren Steuerregistern geliefert werden (Bild 61).

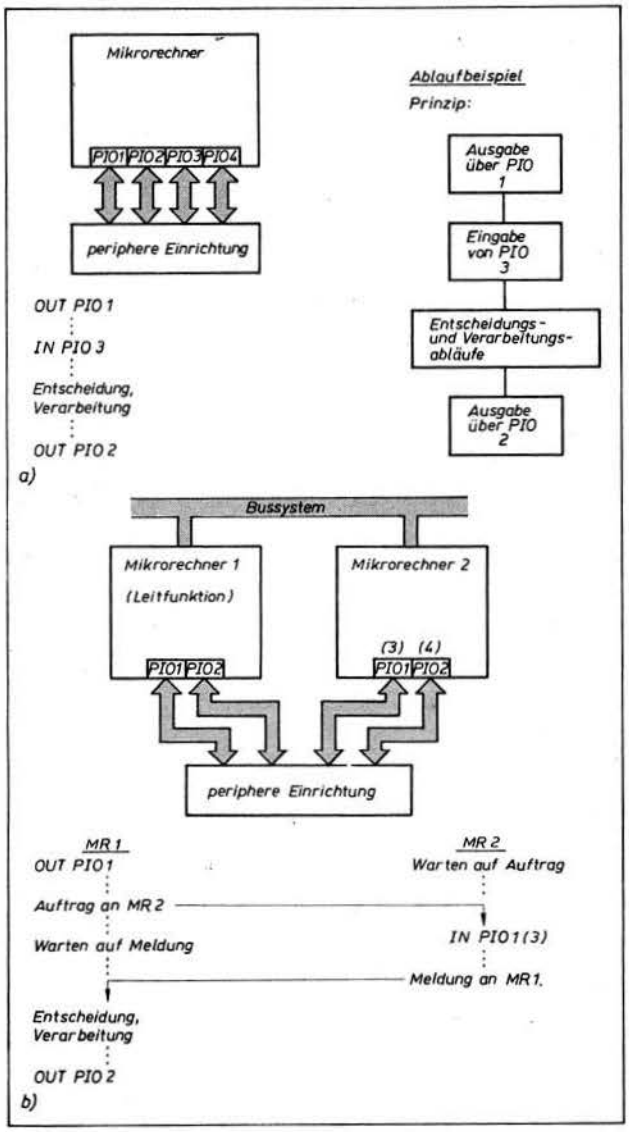


Bild 60: Steuerung einer umfangreichen peripheren Einrichtung. a) Durch einen Mikrorechner mit ausreichender E-A-Ausstattung; b) auf zwei Mikrorechner verteilt

3.1.2. Wirkungsmöglichkeiten auf die Mikrorechneranordnung

Auf die Abläufe in einer Anordnung aus CPU, E-A-Schaltkreisen, ROM, RAM usw. kann man von außen prinzipiell auf folgende Weisen einwirken:

- Rücksetzen

Ein externes Signal ermöglicht, die CPU und wenigstens einige der E-A-IS in einen Grundzustand zu versetzen. Nachdem das Signal inaktiv geworden ist, beginnt die Befehlsabarbeitung von einer festen Adresse (0 beim U 880). Man beachte, daß in der CPU einige Register zurückgesetzt werden und einige nicht und daß die E-A-Schaltkreise neu initialisiert werden müssen.

- Nichtmaskierbarer Interrupt (NMI)

Ein externes Signal veranlaßt, daß das gerade laufende Programm unterbrochen wird (mit Rettung der nächsten Befehlsadresse) und daß eine Routine von einer festen Adresse aus gestartet wird (66H beim U 880). Durch einen Rückkehrbefehl (RETN beim U 880) kann das unterbrochene Programm fortgesetzt werden; es ist aber auch möglich, zu einem anderen Programm zu verzweigen. (Die Probleme der Rettung von Registerinhalten, der Stackverwaltung usw. sollen hier nicht näher untersucht werden.) Beim U 880 gibt es keine automatischen Sperren für NMI, es kann also ein NMI auch während einer laufenden NMI-Behandlung wirksam werden (Bild 62).

- Normaler Interrupt

Wie beim NMI wird das laufende Programm unterbrochen, und es ist die Möglichkeit gegeben, es nach der Unterbrechungsbehandlung fortzusetzen. Die wesentlichen Unterschiede zum NMI sind:

- Zur Ermittlung der Startadresse der Interruptroutine sind verschiedene Verfahren programmierbar. Ein effektives Verfahren besteht darin, daß dem Interruptsignal ein Datenbyte zugeordnet wird, das die Startadresse über eine Tabelle identifiziert (Interrupt Mode 2 beim U 880, s. Bild 63).
- Der Interrupt ist maskierbar (beim U 880 Befehl DI zum Verhindern. EI zum Erlauben von Unterbrechungen).
- In einer laufenden Interruptbehandlung werden keine weiteren Interrupts akzeptiert, sofern dies nicht ausdrücklich (mit EI-Befehlen) erlaubt wird.

- Verändern von Speicherinhalten

Besteht die Möglichkeit, Information von außen in RAM-Bereiche zu schreiben (etwa wenn der RAM als Dual Port Memory gemäß Bild 19 sowohl von der CPU als auch über den gemeinsamen Bus zugänglich ist), lassen sich in Verbindung mit entsprechender Software Modifikationen des internen Ablaufs veranlassen (Bild 64).

Bild 61: Auslösung von Masteranforderungen und zusätzlichen Funktionen durch einen Mikrorechner

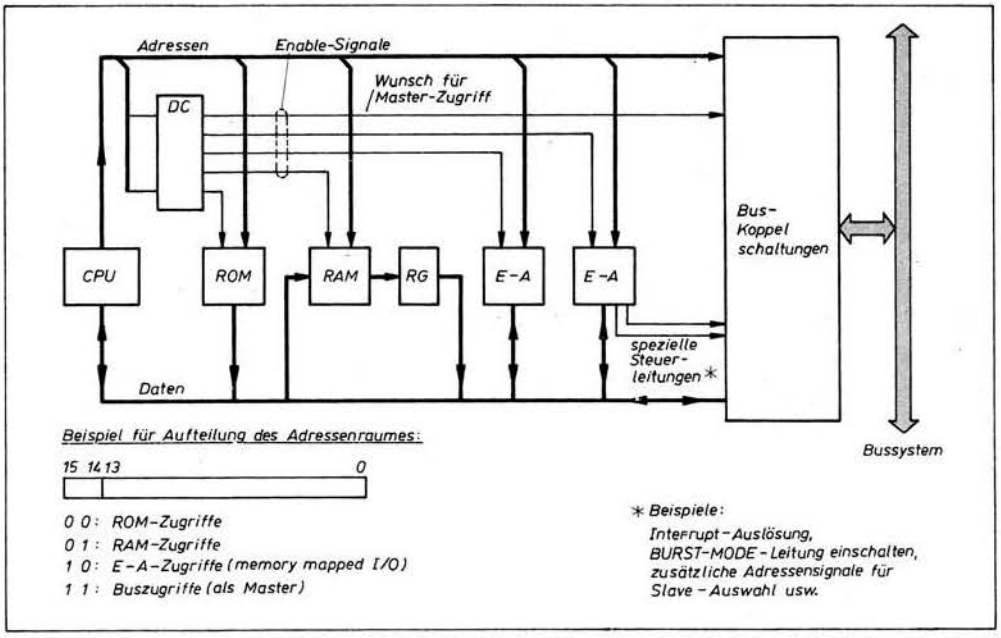
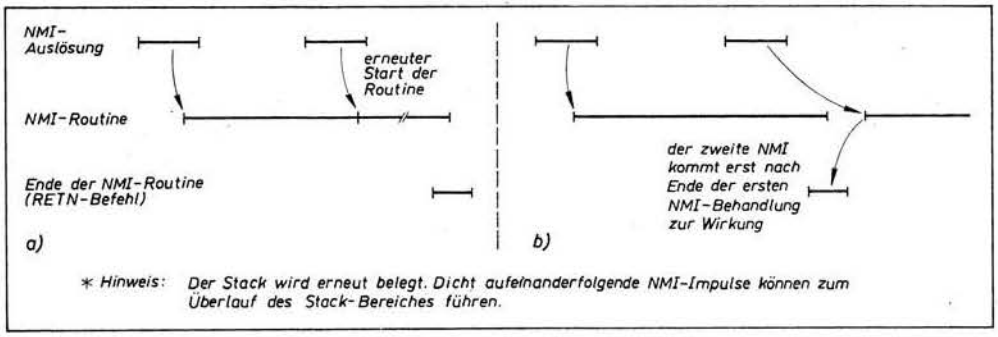


Bild 62: Behandlung von NMI beim U 880. a) Tatsächliches Verhalten; b) gelegentlich gewünschtes Verhalten



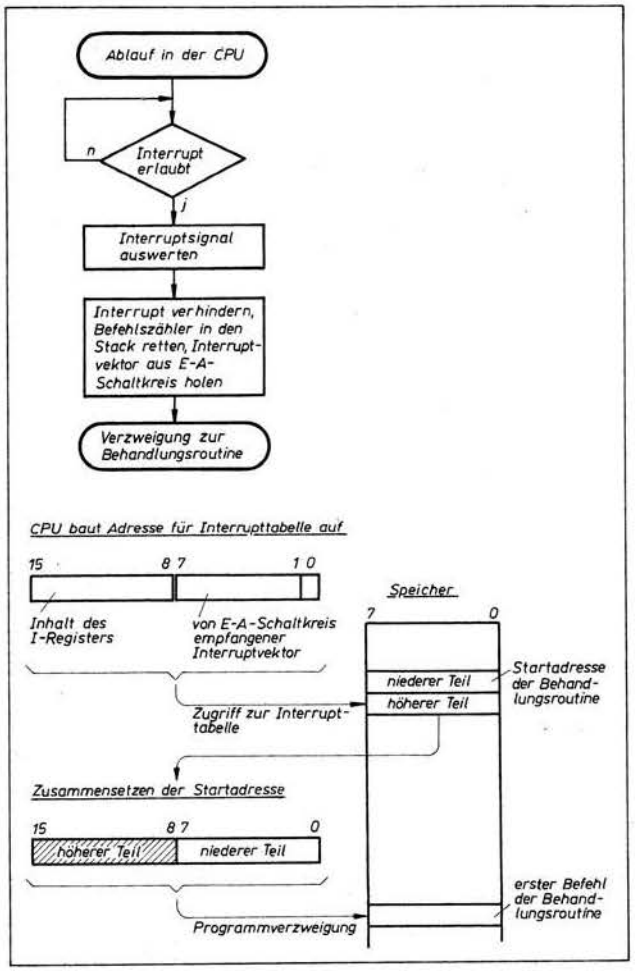


Bild 63: Interruptbehandlung IM2 beim U 880

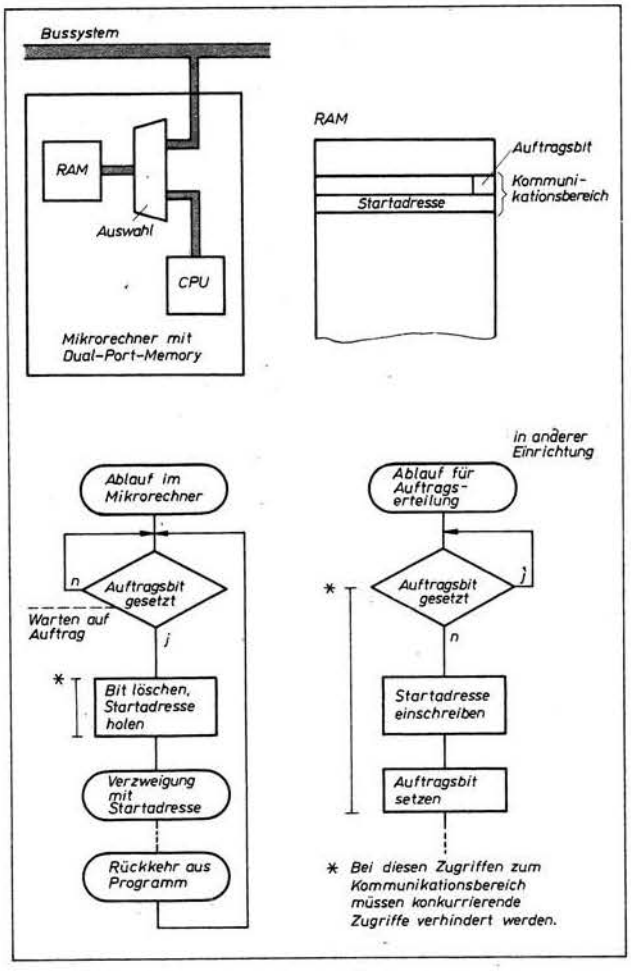


Bild 64: Ablaufmodifikation durch Verändern von RAM-Inhalten

Tafel 3: Möglichkeiten des Einflusses auf einen Mikrorechner

Prinzip	Wirkung	Auswirkung auf Programmablauf	Programmfortsetzung	unterdrückbar	Eignung für
Rücksetzen	Programmstart von fester Adresse (0); Rücksetzen der E-A-Schaltkreise	völliger Neubeginn erforderlich	nein	nein*)	Initialisierung, Fehlersignalisierung
Nichtmaskierbarer Interrupt (NMI)	Rettung des aktuellen Befehlszählers in den Stack, Programmstart von fester Adresse	beliebig programmierbar, Rückkehr zum unterbrochenen Programm möglich	möglich, wenn entsprechend programmiert	nein*)	Signalisierung von Fehlern und anderen außergewöhnlichen Zuständen
Interrupt	Rettung des aktuellen Befehlszählers in den Stack, Start einer spezifischen Behandlungsroutine	beliebig programmierbar, Rückkehr zum unterbrochenen Programm möglich	möglich, wenn entsprechend programmiert	ja	Signalisierung von Aufträgen
Verändern von Speicherinhalten	Verzweigung im Rahmen der Softwarekonventionen des Systems	beliebig programmierbar, die Behandlung obliegt ausschließlich der Software	möglich, wenn entsprechend programmiert	ja	Signalisierung von Aufträgen
WAIT-Leitung erregen	der aktuelle Maschinenzyklus der CPU wird so lange verlängert, bis WAIT inaktiv wird	nur Verlangsamung	ja	mit entsprechender Hardware möglich	Anpassung an Dauer der Zugriffe
BUS-REQUEST-Leitung erregen	der folgende Maschinenzyklus wird nicht ausgeführt, der interne Bus wird von der CPU freigegeben	nur Verlangsamung	ja	mit entsprechender Hardware möglich	„Freischalten" des internen Busses (z. B. für DMA)
CPU-Takte stoppen bzw. verlangsamen	der aktuelle Taktzyklus dauert entsprechend länger	nur Verlangsamung	ja	mit entsprechender Hardware möglich	Anpassung an Dauer der Zugriffe

*) Zusatzhardware zum Unterdrücken ist aufwendig und nicht besonders sinnvoll

● Erregen der WAIT-Leitung

Damit wird ein interner Wartezustand in der CPU ausgelöst. Der aktuelle Maschinenzyklus wird so lange angehalten, wie WAIT aktiv ist. Damit läßt sich die CPU an langsamere Abläufe (z. B. Buszugriffe) anpassen.

● Erregen der BUS-REQUEST-Leitung (BUSRQ)

Damit wird die Befehlsabarbeitung in der CPU angehalten, und deren Bussystem wird in den hochohmigen Zustand versetzt, so daß es von anderen Einrichtungen benutzt werden kann (z. B. von einem DMA-Schaltkreis). Die BUSRQ-Leitung wird allerdings nur am Ende eines Maschinenzyklus ausgewertet. Der hochohmige Zustand wird von der CPU durch Aktivieren des Signals BUSAK angezeigt.

● Stoppen bzw. Verlangsamen des CPU-Taktes

Bei manchen Mikroprozessoren kann damit die Befehlsausführung angehalten bzw. verlangsamt werden. (Beim U 880 darf auf Grund der internen dynamischen Arbeitsweise die Taktfrequenz einen Minimalwert von 85 kHz nicht unterschreiten.)

Man beachte, daß die drei zuletzt angeführten Möglichkeiten den Befehlsablauf in der CPU lediglich verlangsamen, aber nicht modifizieren können. Tafel 3 gibt einen Überblick über die wesentlichen Eigenschaften der genannten Verfahren.

Da stets mehrere Maßnahmen kombiniert vorgesehen werden müssen (z. B. Rücksetzen zur Initialisierung, Interrupts zur Übergabe von Aufträgen, WAIT zur Anpassung an die Dauer der Buszyklen), ist es erforderlich, die wechselseitigen Abhängigkeiten zu betrachten. Diese Untersuchungen müssen für jede Mikroprozessorfamilie, deren Einsatz erwogen wird, im Detail ausgeführt werden. Die folgenden Betrachtungen beziehen sich auf den U 880. Tafel 4 zeigt die relative Dominanz der einzelnen ablaufmodifizierenden sowie der zeitverzögernden Maßnahmen.

Schaltungstechnisch sind alle Maßnahmen relativ einfach zu verwirklichen; Interrupts sollten allerdings nur über die entsprechenden E-A-Schaltkreise ausgelöst werden, insbesondere dann, wenn eine Behandlung gemäß Bild 63 gewünscht wird. (Die Realisierung des erforderlichen Verhaltens mit TTL-Schaltkreisen verlangt einen beachtlichen Aufwand.) Rücksetzen und NMI sind wegen ihrer „unkontrollierbaren" Wirkung für die Steuerung von Ablaufmodifikationen im Rahmen einer komplexen Softwareorganisation (etwa für das Einbringen von Aufträgen anderer Funktionseinheiten) nicht geeignet. Sie sind aber vorteilhaft für die Initialisierung, zur Steuerung von Testabläufen, zu Diagnosezwecken, zur Fehlerbehandlung usw. einsetzbar. Für die Interaktion verschiedener Funktionseinheiten auf Softwareebene eignet sich eine Kombination von Interrupts und programmierten Abfragen. Die WAIT-Steuerung ist von vornherein für die Anpassung der Maschinenzyklen an langsamere Abläufe (länger dauernde E-A-Zugriffe, Speicher mit größerer Zugriffszeit, Zugriffe über das gemeinsame Bussystem usw.) vorgesehen. Das Verlangsamen bzw. Ausblenden von Taktzyklen kann bei Mikroprozessoren, die dies gestatten, etwas effektiver als WAIT sein (Bild 65), ist aber schwieriger zu realisieren. Die Speicherzugriffszeit und die Taktzykluszeiten sind in beiden Fällen gleich. Manche Kombinationen von Verfahren können zur gegenseitigen Behinderung der Funktionseinheiten führen und erweisen sich deshalb als völlig ungeeignet.

Beispiel 1:

Wird bei Masterzugriffen ein WAIT-Zustand erzeugt, um den Maschinenzyklus bis zur Ausführung des Buszyklus zu verzögern, und ist BUSRQ dazu vorgesehen, bei Slavezugriffen über den Bus die Speicher des Mikrorechners von außen erreichen zu können (Bild 66), so kann folgende Situation auftreten:

Tafel 4: Gegenseitige Dominanz der einzelnen Maßnahmen

Dominanzebene	Ablaufmodifikation Prinzip	Ablaufmodifikation Wirksamkeit	Zeitverzögerung Prinzip	Zeitverzögerung Wirksamkeit
1	Rücksetzen	stets	Taktstopp bzw. Taktverlangsamung	stets
2	NMI	zwischen zwei Befehlen	WAIT	innerhalb eines Maschinenzyklus
3	Interrupt	zwischen zwei Befehlen, wenn erlaubt	REQUEST BUS	zwischen zwei Befehlen
4	Verändern von Speicherinhalten (+ Abfrage)	softwaregesteuert (bei Durchlauf der Abfrageschleife)	—	—

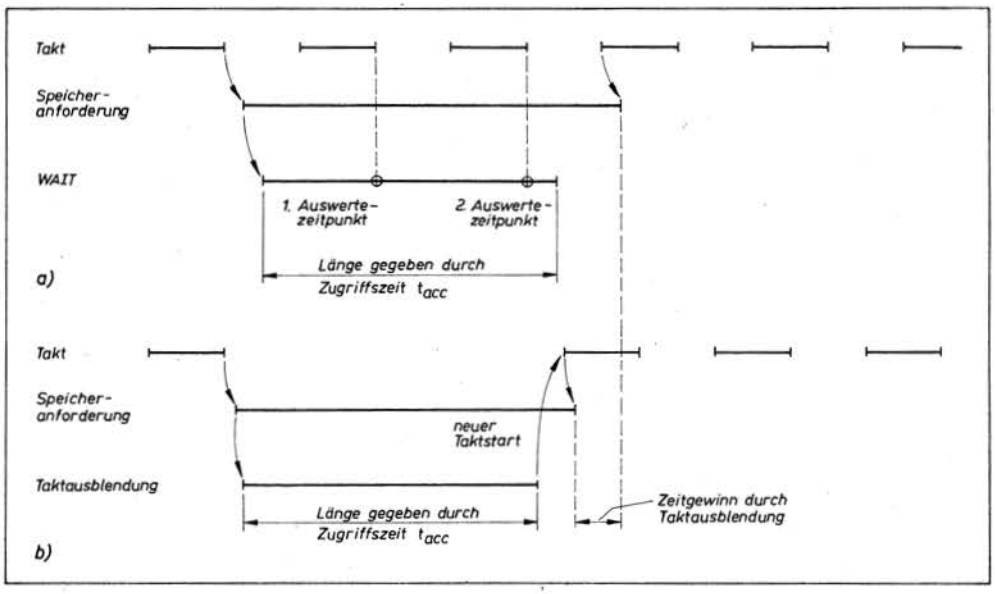


Bild 65: Maschinenzyklusdauer.
a) WAIT-Steuerung;
b) Taktausblendung

Der Mikrorechner verlangt einen Buszugriff zu einer Einrichtung, die diesen nicht sofort ausführen kann. Damit verbleibt der Mikrorechner im WAIT-Zustand. Die andere Einrichtung will auf den Speicher des Mikrorechners zugreifen und beantragt deshalb über BUSRQ die Freigabe des internen Mikrorechnerbussystems. Dies kann aber nicht erfolgen (BUSAK wird nicht signalisiert), da der aktuelle Befehl im Mikrorechner wegen WAIT nicht beendet werden kann. Im Normalfall wird die Buskommunikation beträchtlich verlangsamt. Bei bestimmten Operationen der Art Test and Set (s. Abschnitt 1.5.) kann es zur totalen Behinderung kommen, etwa wenn die andere Einrichtung Slavezugriffe erst dann wieder zuläßt, wenn sie bestimmte Zugriffe zu dem betrachteten Mikrorechner ausgeführt hat. Solange also WAIT aktiv ist, kann BUS REQUEST nicht wirksam werden, d. h., BUSAK wird nicht aktiv, ein Slavezugriff ist nicht möglich.

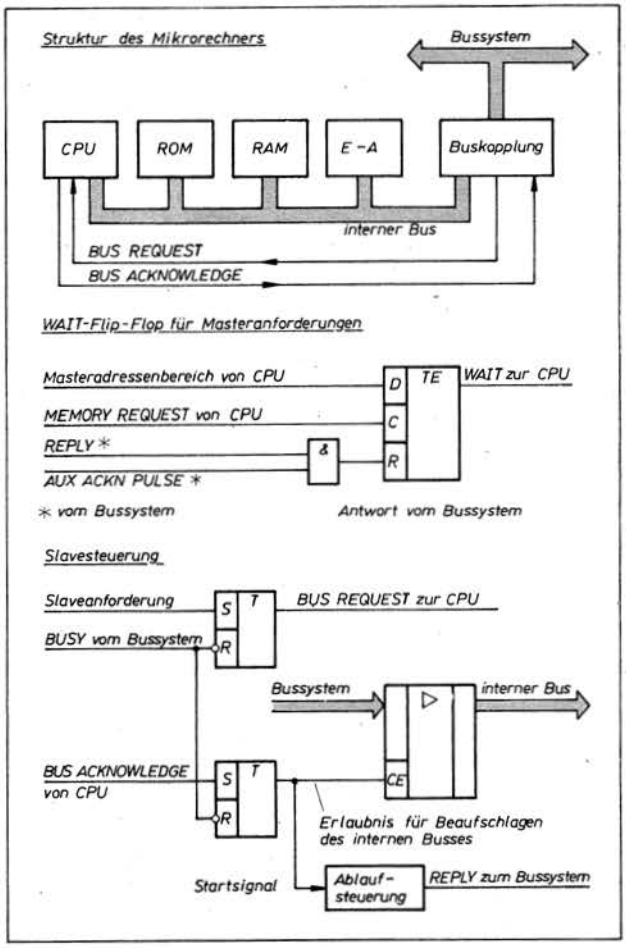


Bild 66: Benutzung von BUS REQUEST bei Slavezugriffen

Weitere Konsequenzen folgen aus der Tatsache, daß bei vielen Mikroprozessoren (auch beim U 880) der einzelne Maschinenzyklus zwar beliebig verlängert, aber nur durch Rücksetzen abgebrochen werden kann. Eine Ausnahme bildet z. B. der MC 68 000: Ein Maschinenzyklus kann nicht nur regulär beendet, sondern auch mit BUS ERROR abgebrochen werden [13]. Die CPU kann durch diese Bedingung entweder zu einer Wiederholung des Buszyklus (im Fehlerfall) oder auch zu einem Trap (Verzweigung zu einer Softwareroutine mit Rückkehrmöglichkeit) veranlaßt werden.
Bei Mikroprozessoren, die derartige Möglichkeiten nicht bieten, lassen sich Prinzipien, die von größeren Rechnern her bekannt und an sich sehr nützlich sind, nur mit sehr hohem Zusatzaufwand realisieren.

Beispiel 2:

Bei hohen Anforderungen an die Zuverlässigkeit können Kontrollschaltungen vorgesehen werden. Ein Beispiel ist die Paritätskontrolle an Speichern. Eine übliche Art der Behandlung eines Paritätsfehlers ist die Wiederholung des Zugriffs, bei dem der Fehler erkannt wurde. Man kann aber einen Mikroprozessor wie den U 880 nur im Wartezustand halten oder zurücksetzen. Damit ist durch den Mikrorechner selbst eine Fehlerbehandlung mit sinnvoller Fortsetzung der Arbeit nicht möglich. Es ergeben sich drei Alternativen.

● Der Mikrorechner wird im WAIT-Zustand belassen (der Fehler hat sich noch nicht auf den Programmablauf ausgewirkt). Für die Fehlerbehandlung wird ein anderer Mikrorechner benutzt. Dieser kann den Speicherinhalt analysieren, feststellen, ob es sich um einen flüchtigen oder um einen latenten Fehler handelt, und dementsprechend entscheiden, ob der WAIT-Zustand aufgehoben werden kann oder ob eine neue Initialisierung (durch Rücksetzen) erforderlich ist. Dies bedingt die Anordnung eines besonderen Mikrorechners für die Fehlerbehandlung, eine entsprechende Auslegung des Bussystems und Schaltungsmaßnahmen im Mikrorechner selbst. Im wesentlichen handelt es sich darum, zu gewährleisten, daß die betreffende CPU im WAIT-Zustand gehalten wird, der jeweils angesprochene Speicher aber anderweitig (vom behandelnden Mikrorechner aus) adressiert werden kann und zur Fortsetzung der Arbeit wieder auf die betreffende CPU umgeschaltet werden kann. Da dies sowohl bei lokalen Zugriffen innerhalb des Mikrorechners als auch bei Buszugriffen möglich sein muß, sind dazu recht aufwendige Schaltungen erforderlich.

● Der aktuelle Befehl wird beendet, und es wird NMI ausgelöst, wodurch eine Fehlerroutine gestartet wird. Damit steht relativ viel Information zur Verfügung, um die Fehlersituation zu charakterisieren (Registerinhalte, aktueller Befehlszähler, aktueller Stackpointer), durch das Beenden des Befehls können aber Folgefehler verursacht worden sein (durch Verfälschen anderer Speicherzellen oder von Registerinhalten).

● Es wird sofort ein Rücksetzsignal ausgelöst, die Fehlerbehandlung beschränkt sich auf das Registrieren des Fehlers. Die weitere Ausbreitung von Fehlern wird dadurch unterbunden, es geht allerdings Information über die aktuelle Fehlersituation verloren, es sei denn, für das Retten dieser Information gibt es spezielle Schaltmittel (etwa die Auffangregister in der Bussteuereinrichtung gemäß Bild 45).

Beispiel 3:

Eine Auslegung des Systems, bei der bei jedem Speicherzugriff festgestellt wird, ob sich das adressierte Informationsobjekt (Programmstück, Datenbereich) im Speicher befindet oder ob es erst von einer externen Speichereinrichtung (z. B. Floppy Disc) geholt werden muß, stößt auf dieselben Schwierigkeiten wie die Fehlerbehandlung. Es ist keine andere Maßnahme möglich, als die CPU solange im WAIT-Zustand zu halten, bis die Information verfügbar ist (d. h., in den Speicher geladen wurde).

3.1.3. Interne Funktionen des Mikrorechners

Mikrorechner können durch Programme Speicherinhalte modifizieren und externe Anschlüsse abfragen bzw. steuern. Je nachdem, ob die Funktion eines Programmes darin besteht, Daten aus Speichern zu übernehmen, zu verarbeiten und wieder zu speichern, oder darin, auf externe Signale zu reagieren bzw. diese anzusteuern, werden die Aufgaben in Verarbeitungs- und E-A-Steueraufgaben unterteilt.

Für die Auslegung ist es wesentlich, zu bestimmen, welche Aufgaben intern (ohne Zugriff zu anderen Funktionseinheiten) und welche mit Hilfe derartiger Zugriffe ausgeführt werden sollen. Ist es vorgesehen, spezielle E-A-Schaltkreise wie CTC, PIO, SIO oder andere Ergänzungsschaltkreise des Mikroprozessorsystems (z. B. Arithmetikprozessoren) einzusetzen, so ist es zweckmäßig, diese gemäß den Zusammenschaltungsregeln des Mikroprozessorsystems direkt an den internen Bus anzuschließen. Der Anschluß an den Systembus bringt erhebliche Probleme mit sich, da diese Schaltkreise nur dann befriedigend arbeiten, wenn der Signalaustausch mit der CPU genau den Konventionen des Mikroprozessorsystems entspricht, deren Einhaltung im Rahmen eines Multimasterbussystems besonders hinsichtlich der zeitlichen Bedingungen schwierig ist.

E-A-Komplexe, die derartige Schaltungen nicht benutzen, können wahlweise an den internen Bus des Mikrorechners oder an das gemeinsame Bussystem angeschlossen werden. Beim Anschluß an den internen Bus hat man noch die Wahl, die Zugriffe als E-A- oder als Speicherzugriffe zu organisieren. Die Konsequenzen dieser Varianten sind in der Tafel 5 gegenübergestellt.

Speichermittel können innerhalb oder außerhalb des Mikrorechners vorgesehen werden. Speicherzugriffe lassen sich folgendermaßen einteilen:

● Befehlszugriffe (nur Lesen)

● Datenzugriffe (Lesen, Schreiben)
Damit sind jene Zugriffe gemeint, die von Befehlen initiiert werden, um Verarbeitungsaufgaben auszuführen

Tafel 5: Anschlußmöglichkeiten von E-A-Einrichtungen, die nicht durch LSI-IS gesteuert werden können

Aspekte	Anschluß am internen Bus	am Systembus
technisch-konstruktive Realisierung	ungünstig bei kompaktem Aufbau: jeder E-A-Komplex bedingt einen neuen Typ der Mikrorechnerleiterplatten; günstig bei modularem Aufbau	günstig hinsichtlich Standardisierung, aber in der Regel aufwendiger (wegen der umfangreicheren Buskoppelschaltungen) als der Anschluß an den internen Bus
Leistungsfähigkeit	durch Mikrorechner bestimmt; bei Anschluß über E-A-Zugriffe in der Regel etwas geringer als bei Anschluß über Speicherzugriffe, da nur die E-A-Befehle benutzt werden können statt der vielseitigeren speicherbezogenen Befehle	durch die Auslegung der Schaltung selbst bestimmt; bei äquivalenter Auslegung im Mittel etwas geringer als beim internen Anschluß, da die Systembuszugriffe wegen der Masterauswahl länger dauern als interne Zugriffe
Benutzbarkeit durch mehrere Mikrorechner	problematisch, erfordert besondere Softwareorganisation (Übergabe von Aufträgen an den Mikrorechner, der die E-A-Einrichtung betreibt)	an sich unproblematisch; Ressourcenverwaltung kann erforderlich sein (s. Abschnitt 1.6.)
Aufwand	geringer als bei Systembusanschluß	

● Zugriffe zur Auftragssteuerung, Koordinierung usw. (Lesen, Schreiben)
Im Gegensatz zu den normalen Datenzugriffen können derartige Zugriffe auch von anderen Einrichtungen (die den betreffenden Speicher als Slave adressieren) ausgeführt werden. Dazu sind gelegentlich besondere Betriebszustände erforderlich (etwa die Blockierung anderweitiger konkurrierender Zugriffe, die Erregung bestimmter Steuerleitungen o. ä.).

Die Probleme, die die Anordnung der Speicher mit sich bringt, sollen zunächst anhand der beiden extremen Auslegungen diskutiert werden.

● Die Mikrorechner enthalten keine lokalen Speicher. Damit ist jeder Speicherzugriff über das Bussystem zu führen. Es ist ersichtlich, daß deutliche Leistungsminderungen nur dann vermieden werden können, wenn das Bussystem auf extrem kurze Vermittlungs- und Übertragungszeiten und die Speicheranordnungen auf geringe Zugriffszeiten hin (und somit entsprechend aufwendig) ausgelegt sind.
Eine derartige Ausführung kann Vorteile haben, wenn große Speicherkapazitäten erforderlich sind, die mit dynamischen MOS-Speichern realisiert werden sollen. Da deren Ansteuerung einen relativ hohen Grundaufwand erfordert, sind große Speicher kostengünstiger als kleine. Weiterhin sind beim Einsatz dynamischer Speicher-IS großer Speicherkapazität (z. B. 64 Kbit) Fehlerkorrekturmaßnahmen sinnvoll. Auch hier tritt der Aufwand für die Steuer- und Kodier- bzw. Korrekturschaltungen nur einmal auf. Noch wesentlicher ist aber, daß Fehlerkorrekturkodes mit zunehmender Aufrufbreite wirtschaftlicher werden. Die Verhältnisse für die Korrektur von 1-bit-Fehlern sind in der Tafel 6 dargestellt.

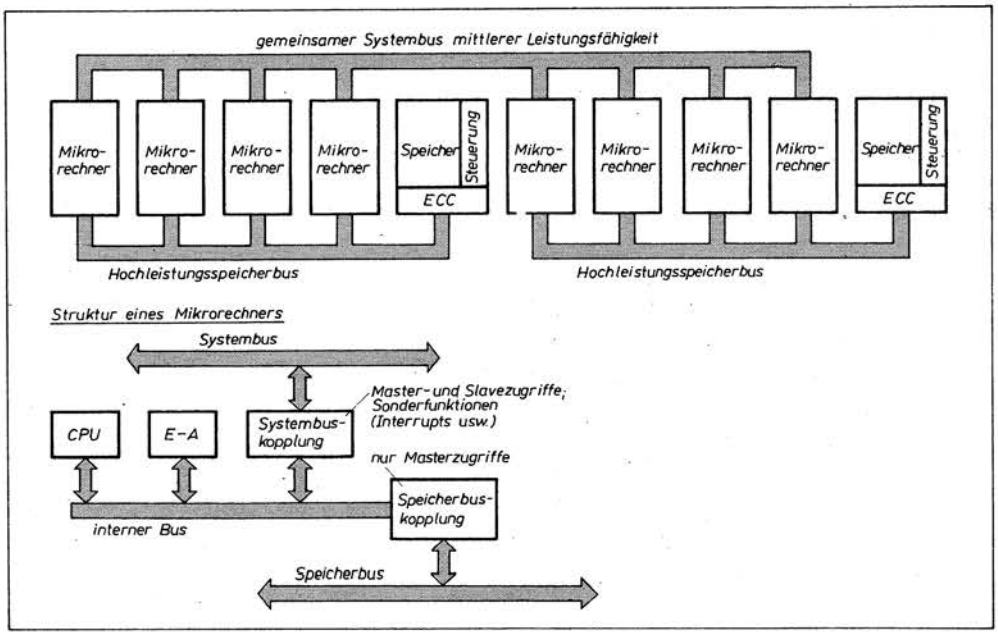


Bild 67: Struktur eines Multimikrorechnersystems mit vollständig zentralisierten Speichern

Tafel 6: Anzahl der Zusatzbits zur Korrektur von 1-bit-Fehlern

Aufrufbreite des Speichers in bit	minimale Anzahl von Zusatzbits
8	4
16	5
32	6
64	7

Für bestimmte Anwendungen können derartige Konfigurationen durchaus von Interesse sein. Die Problematik der Datenrate des Busses kann durch Einsatz mehrerer Bussysteme beherrscht werden [6]. Bild 67 zeigt eine Struktur, bei der jeweils vier Mikrorechner an einen gemeinsamen Speicher angeschlossen sind.

Weiterhin dürften sich derartige Strukturen vorteilhaft realisieren lassen, wenn Mikroprozessorfamilien eingesetzt werden, bei denen der Datenaustausch in Paketform organisiert ist [2] [12]. Ist das Mikroprozessorsystem beispielsweise für die Übertragung von Paketen mit einer Länge von 16 byte eingerichtet, kann ein zentraler Speicher aus zwei Moduln von je 8 byte Aufrufbreite, die überlappend arbeiten, vorgesehen werden (Bild 68).

● Die Mikrorechner enthalten nur lokale Speicher. Alle Programme stehen in eigenen Speichern, und der Datenaustausch mit anderen Einrichtungen beschränkt sich auf Steuerungszugriffe und auf den Austausch von Datenblöcken. Dadurch werden die Anforderungen an das Bussystem gegenüber der ersten Auslegung beträchtlich vermindert, ohne daß dies zu einem Abfall der Verarbeitungsleistung der einzelnen Mikrorechner führt. Diese Lösung ist für mittlere Speichergrößen sehr effektiv, namentlich dann, wenn ROMs und statische RAMs zum Einsatz kommen, deren Ansteuerung relativ einfach ist.

● Die Mikrorechner haben lokale Speicher, benutzen aber auch Speichereinrichtungen in anderen Funktionseinheiten. Dies stellt eine Ausweichmöglichkeit für den Fall dar, daß weder lokal genügend Speicherkapazität bereitgestellt werden kann noch daß der Aufwand für große zentrale Speicher und die zugehörigen Hochgeschwindigkeits-Bussysteme akzeptabel ist. Bei dieser Variante hat sich eine Organisationsform bewährt, bei der für die Ausführung von Programmen und für das Halten der wesentlichen Datenbereiche der lokale Speicher benutzt wird. Entsprechende Betriebssoftware veranlaßt, daß die benötigten Programme, Datenblöcke usw. vor der Ausführung in den lokalen Speicher transportiert werden, sofern sie dort nicht schon präsent sind. Das Prinzip ist im Bild 69 dargestellt. Die Variante a) ist ungünstig, da für jeden Befehl ein Buszugriff erforderlich ist. Der Transport in den lokalen Speicher läuft so ab, daß das gesamte Programm aus dem Speicher einer anderen Funktionseinheit in den des Mikrorechners verlagert wird, seine Abarbeitung erfolgt dann durch lokale Zugriffe zum Speicher des Mikrorechners.

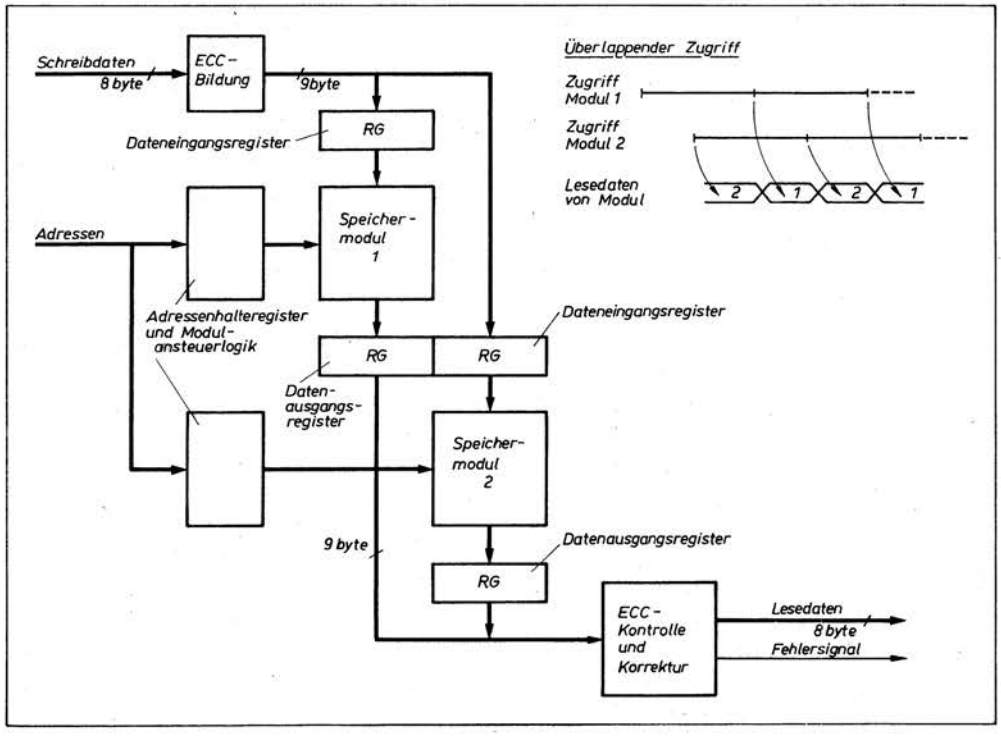


Bild 68: Struktur eines zentralisierten Speichers, der aus zwei überlappend arbeitenden Moduln aufgebaut ist

Der Vorteil liegt im wesentlichen darin, daß nach dem Transport, der beim U 880 mit Blocktransportbefehlen mit der maximal möglichen Geschwindigkeit abläuft, überwiegend lokale Zugriffe ausgeführt werden, so daß der Bus nicht übermäßig belegt wird und die Verarbeitungsleistung des Mikrorechners nicht sinkt.

3.2. Diskussion spezieller Probleme

3.2.1. Wahl der Speicherbestückung (ROM, RAM)

Für jeden Mikrorechner muß eine gewisse RAM-Kapazität verfügbar sein. Diese wird bestimmt durch

- den Mindestbedarf für die Steuerinformation und für den Stack. Dafür reichen normalerweise 256 byte bis 1 Kbyte aus, sofern kein anspruchsvolles Echtzeitbetriebssystem implementiert werden soll. In diesem Fall muß der genannte Größenbereich für jeden Prozeß (Task, Partition) veranschlagt werden
- den Bedarf für Datenbereiche, die gelesen und beschrieben werden müssen
- den Bedarf für Programme bzw. Dateien, die von externen Datenträgern geladen werden müssen (transiente Programme oder Dateien).

Für Programme, die ständig verfügbar sein müssen, und für Konstanten können ROM-Speicherschaltkreise vorgesehen werden. Diese ermöglichen eine einfache Realisierung des Mikrorechners, denn die Ansteuerung ist unproblematisch, und sie haben den Vorteil, daß die Information sofort mit dem Einschalten der Betriebsspannung präsent ist und auch nicht durch Fehler in der Software zerstört werden kann.

Die Orientierung auf den vorrangigen Einsatz von PROMs führt zu folgenden Konsequenzen:

- Softwareänderungen werden schwierig; Eingriffe zu Erprobungszwecken (z. B. das Ändern einiger Bytes während des Betriebes) sind nicht möglich.
- In einem Multimikrorechnersystem werden an sich standardisierte Funktionseinheiten durch unterschiedliche ROM-Inhalte zu spezifischen Funktionseinheiten, die nicht mehr untereinander getauscht werden können, es sei denn, man sieht für alle ROMs Steckfassungen vor. Dies wirkte sich jedoch ungünstig auf Kosten und Zuverlässigkeit aus.

Diese Nachteile lassen sich vermeiden, indem die Speicher der Mikrorechner prinzipiell mit RAM-Schaltkreisen realisiert werden, wobei nur eine minimale ROM-Ausstattung für Initialisierung, Testung usw. vorgesehen ist. Damit werden Maßnahmen für das initiale Programmladen erforderlich. Üblicherweise werden dazu externe Speicher wie Floppy Disc oder Magnetbandkassetten verwendet.
Für bestimmte Einsatzgebiete, namentlich dort, wo diese Speichereinrichtungen nicht robust genug sind, kann auch eine ROM-Anordnung verwendet werden (Bild 70). Nach dem Einschalten werden die RAMs der einzelnen Mikrorechner aus der zentralen ROM-Anordnung geladen. Dabei gibt es zwei Alternativen: Entweder enthält die ROM-Anordnung die Steuerung und greift als Master zu den Mikrorechnern zu, oder die Mikrorechner enthalten eine gewisse ROM-Kapazität und steuern damit den Ladeprozeß selbst.

Die ROM-Anordnung wird dabei als Slave adressiert. Gegenüber der unmittelbaren Anordnung von ROMs in den einzelnen Mikrorechnern ist dies natürlich ein höherer Aufwand, der aber folgende Vorteile bietet:

- Die Mikrorechner selbst können standardisierte Funktionseinheiten sein, die erst durch das Programmladen für den jeweiligen Einsatzfall personalisiert werden. Damit sind die Mikrorechner untereinander austauschbar.
- Programme, die als identische Kopien in mehreren Mikrorechnern benötigt werden, belegen nur einmal Speicherplatz im ROM.
- Das Einbringen von Softwareänderungen betrifft nur eine einzige Stelle und ist demzufolge organisatorisch einfacher beherrschbar.
- Eingriffe zu Erprobungszwecken werden möglich.

Die dargestellte Lösung ermöglicht eine weitere Verbesserung des Änderungsprozesses. Bei komplexen Systemen

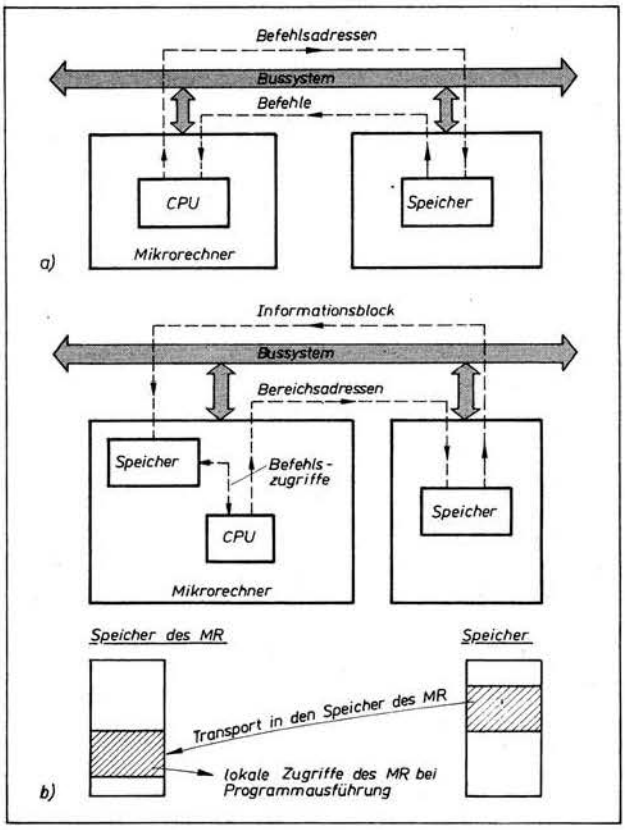


Bild 69: Benutzung von Speichern in anderen Funktionseinheiten. a) Befehlsausführung direkt über Bussystem; b) Transport in lokalen Speicher

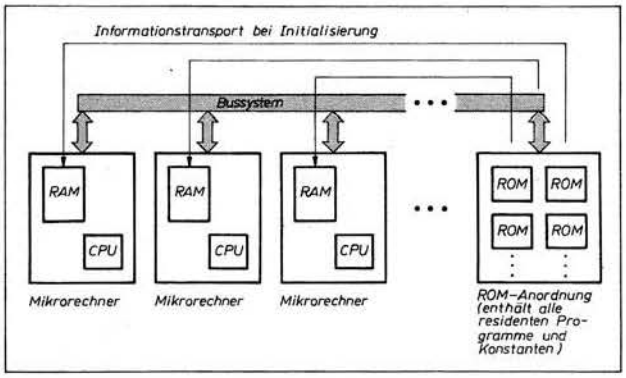


Bild 70: Multimikrorechnersystem mit zentralem ROM für das Anfangsprogrammladen

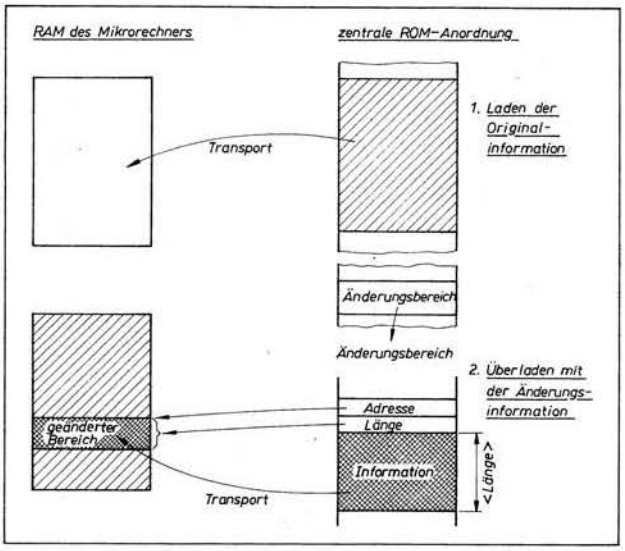


Bild 71: Einbringen von Änderungen aus einem speziellen Änderungsbereich des zentralen ROM

muß selbst nach umfassender Erprobung noch nach der Auslieferung an die Anwender mit Softwarefehlern gerechnet werden. Derartige Änderungen betreffen meist nur wenige Bytes, die jedoch im gesamten Speicherraum verteilt sein können, so daß auch recht triviale Änderungen Ursache dafür sind, daß eine größere Zahl von ROMs gewechselt werden muß.
Abhilfe läßt sich dadurch schaffen, daß ein spezieller ROM für alle Änderungen reserviert wird. Beim Anfangsprogrammladen wird zunächst die ursprüngliche Information in die Mikrorechner transportiert. Danach wird der besagte spezielle ROM abgefragt, der die Änderungen beispielsweise in der Form „Adresse – Länge – neuer Inhalt" enthält. Mit dieser Änderungsinformation werden die Speicher in den Mikrorechnern selektiv überschrieben (Bild 71).

3.2.2. Segmentierung des Adressenraumes

Übliche Mikroprozessoren arbeiten in der Regel mit absoluten Adressen definierter Länge (z. B. 16 bit beim U 880). Der damit verfügbare Adressenraum ($2^{16} = 64$ Kbytes beim U 880) erweist sich in komplexen Systemen häufig als zu klein. Damit muß die Adresse, die die CPU bei ihren Speicherzugriffen liefert, in eine entsprechend längere Adresse umgesetzt werden.
Eine bekannte Möglichkeit, die bei mehreren Minicomputer-Familien und 16-bit-Mikroprozessoren benutzt wird [13], besteht darin, zu der eigentlichen Adresse Modussignale aus der CPU hinzuzufügen, die die Art des Zugriffs kennzeichnen. Es kann z. B. unterschieden werden zwischen

- Kodezugriffen (Befehlslesen)
- Zugriffen zu Stackbereichen
- Zugriffen zu Daten.

Weiterhin verfügen einige CPUs über zwei Verarbeitungszustände (Normalzustand, Systemzustand). Diese Zustandsinformation kann ebenfalls zur Adresse hinzugefügt werden. Eine hypothetische Lösung ist im Bild 72 dargestellt.
Dieses einfache Schema ist in der Praxis nicht flexibel genug. Die eingeführten Lösungen sind dahingehend verbessert, daß für jeden Teiladressenraum ein Segmentregister

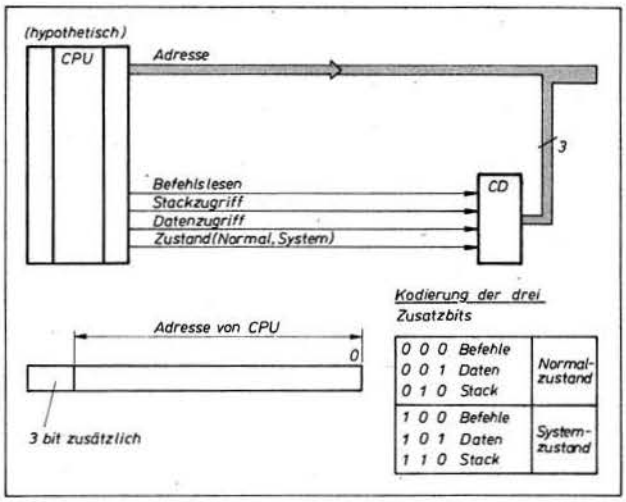


Bild 72: Erweiterung einer CPU-Adresse durch Zustandssignale der CPU

vorgesehen ist, dessen Inhalt entweder zur Adresse addiert (Bild 73) oder dieser vorangestellt wird (Bild 74).

Die Lösung gemäß Bild 73 (Addition der Segmentbasisadresse) erlaubt, daß sich die Segmente überlappen, erfordert aber längere Segmentregister und den Aufwand (hinsichtlich Logik und Laufzeit) für die Addition. Beim einfachen Voranstellen der Segmentadresse nach Bild 74 ist der Aufwand geringer, und es treten keine Zeitverluste auf; die Segmente können sich aber nicht überlappen. Die Tafel 7 gibt eine Übersicht über existierende Systeme [13].

Die dargestellten Prinzipien sind nur dann ohne weiteres anwendbar, wenn sie durch das Mikroprozessorsystem selbst realisiert sind (in der CPU oder mit Hilfe spezieller Speicherverwaltungs-IS) bzw. wenn die CPU wenigstens die entsprechenden Zustandssignale bei jedem Speicherzugriff mitliefert. Mikroprozessoren der Art des U 880 tun dies nicht.

Wird fortgesetzt

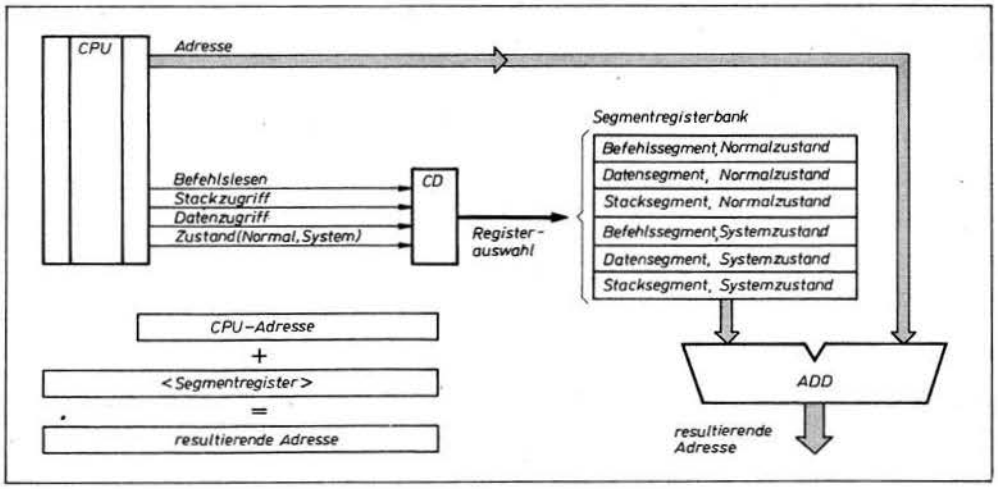


Bild 73: Segmentierung durch Adressenaddition

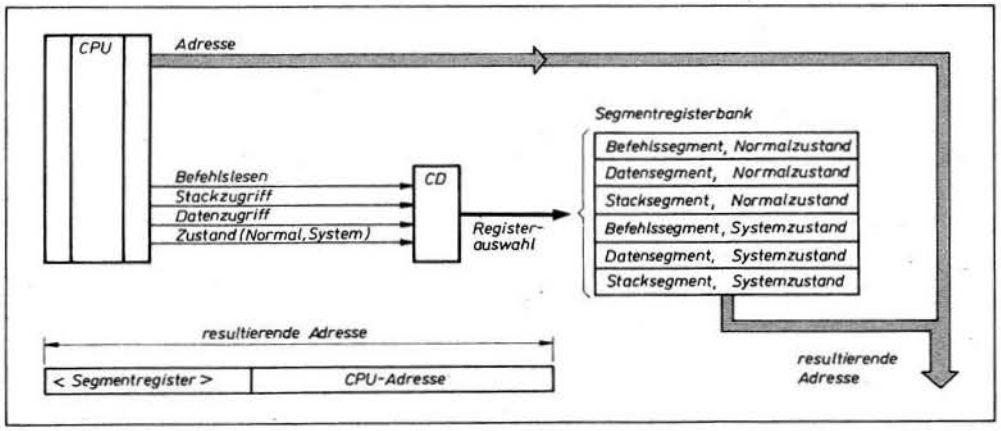


Bild 74: Segmentierung durch Voranstellen der Segmentadresse

Die Realisierung einer der vorgestellten Lösungen würde erfordern, daß alle Speicherzugriffe von externer Hardware kontrolliert werden, um zwischen Befehls-, Daten- und Stackzugriffen zu unterscheiden. Des weiteren wären bestimmte Programmierregeln einzuhalten. Eine hypothetische Lösung veranschaulicht Bild 75.
Der beachtliche Zusatzaufwand ist für die meisten Einsatzfälle nicht tragbar. Man erhält einfachere Lösungen durch eine direkte Abbildung der 16-bit-Adresse auf die gewünschte längere Adresse. Dies bedingt jedoch eine Softwareunterstützung, denn man kann einen 64-Kbyte-Adressenraum nicht überschreiten, ohne dafür eine Softwareaktivität vorzusehen. Eine weitere Konsequenz besteht darin, daß man für das einzelne Programm mit 64 Kbyte für Befehle , Daten, Stackbereiche usw. auskommen muß. Änderungen oder Überschreitungen dieses Bereiches müssen jeweils explizit programmiert werden. Bild 76 zeigt das Prinzip einer Lösung.
Es werden einige der höchstwertigen Adressenbits durch Bits substituiert, die von zusätzlichen Schaltmitteln (RAMs, Register o. ä.) geliefert werden. Durch Umladen der Register bzw. RAM-Zellen (beispielsweise mit E-A-Befehlen) kann man auf andere physische Adressenbereiche zugreifen. Ähnlich wie bei der zuvor diskutierten Segmentierung läßt sich die resultierende Adresse auch durch Addition der niederen Adressenbits zur Segment-Basisadresse ermitteln. Durch Rücksetzen (nach dem Einschalten usw.) wird die Segmentierung verhindert, d. h., es wird ständig Segment 0 adressiert. Erst durch einen speziellen E-A-Befehl wird die Segmentierung eingeschaltet.
Der zusätzliche Aufwand ermöglicht das Überlappen der Segmente und eine bessere Ausnutzung des verfügbaren Speicherraumes.
Es verbleibt die Entscheidung, wie die CPU-Adresse auf den Speicherraum abgebildet werden soll. Man hat die Wahl zwischen vielen kleinen und wenigen großen Segmenten. Weiterhin kann der gesamte CPU-Adressenraum segmentiert werden oder auch nur ein Teil davon. Bild 77 und Tafel 8 veranschaulichen einige Alternativen.

Tafel 7: Übersicht über die Prinzipien der Segmentierung bei einigen Mikroprozessorfamilien

Typ	Prinzipien der Segmentierung
Intel 8086	vier Segmente zu 64 Kbyte Es wird eine 20-bit-Adresse erzeugt; die Segmente können sich überlappen. Segmentaufteilung: Kode (Befehle) Stack Daten externe Daten
Zilog Z 8000	unsegmentierte Version: 64 Kbyte werden direkt adressiert segmentierte Version: 7-bit-Segmentadresse, die einem Offset von 8 bit bzw. 16 bit vorangestellt wird (damit sind maximal 128 nichtüberlappende Segmente von je maximal 64 Kbyte adressierbar, d. h. insgesamt 8 Mbyte). Es sind maximal zwölf Speicherverwaltungseinheiten (MMU) anschließbar, von denen jede bis zu 64 Segmente verwalten kann. Je ein Paar dieser MMUs gehört zu einem virtuellen Adressenbereich. Bereichsauswahl durch Modus (normal, System) und Zugriffsart (Kode, Daten, Stack).
Motorola MC 68 000	Der Mikroprozessor liefert direkt eine 24-bit-Adresse. Der große Adressenraum von 16 Mbyte erübrigt in vielen Fällen eine Segmentierung. Für Speicherverwaltungsschaltkreise ist ein Prinzip ähnlich zu Z 8000 vorgesehen.
DEC PDP 11/70 (Minicomputer)	Über Seitenregister sind vier Segmente adressierbar: ● Supervisorzustand, Befehle ● Supervisorzustand, Daten ● Userzustand, Befehle ● Userzustand, Daten

3.2.3. Zusätzliche Wirkungen

Im Rahmen von Multimikrorechnersystemen kann es erforderlich sein, daß manche Zugriffe, die von einem Mikrorechner ausgeführt werden, unter besonderen Umständen ablaufen müssen oder besondere Wirkungen anstelle des üblichen Lesens bzw. Beschreibens von Speicherpositionen veranlassen müssen.
Ein Beispiel ist der bereits mehrfach erwähnte Test-and-Set-Ablauf, bei dem innerhalb eines bestimmten Zeitintervalls konkurrierende Zugriffe unterdrückt werden. Außerdem werden über das gemeinsame Bussystem in anderen Funk-

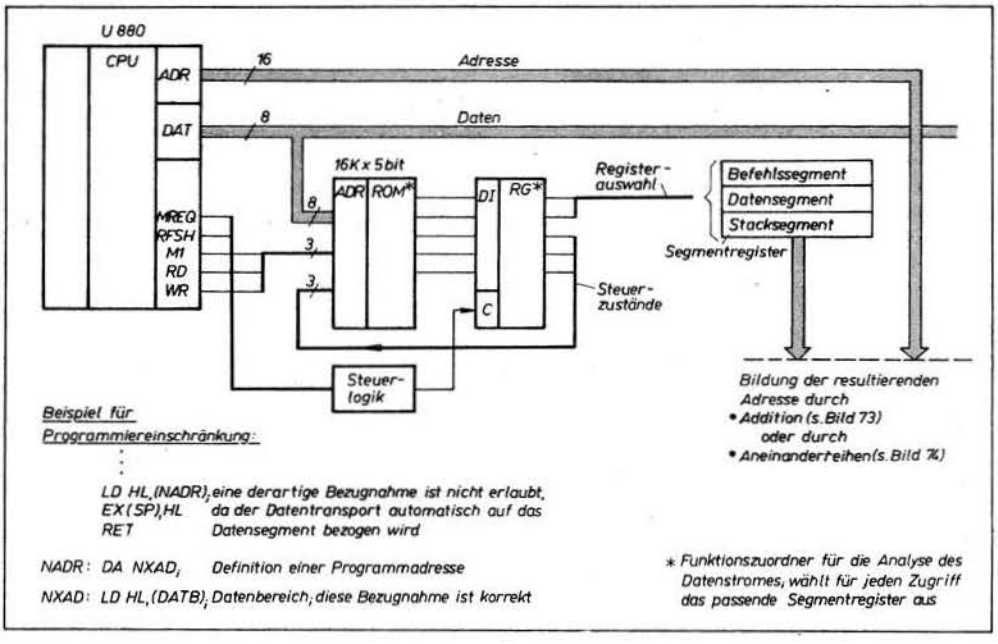


Bild 75: Hypothetische Lösung für eine Segmentierung beim U 880. Der ROM bildet in Verbindung mit dem nachgeschalteten Register einen Steuerautomaten zur Analyse des Datenstromes

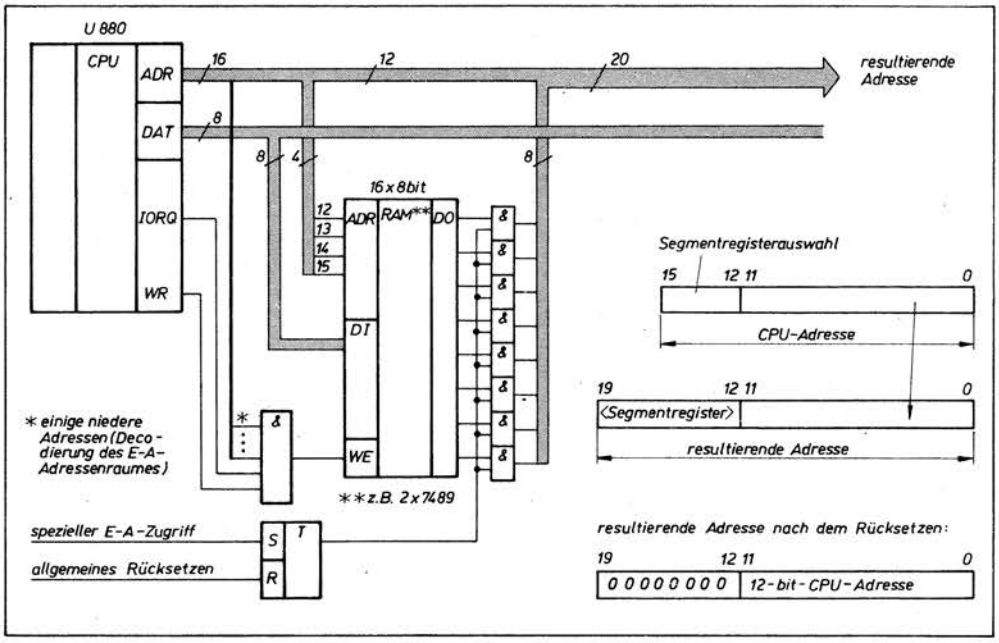


Bild 76: Softwaregesteuerte Segmentierung beim U 880 (Lösungsvorschlag)

tionseinheiten Interrupts ausgelöst. Aus diesem Grunde ist im Bussystem nach Abschnitt 2. dafür die Busleitung INTERRUPT vorgesehen.
Derartige Steuerwirkungen, für die die CPU des Mikrorechners keine Steuersignale abgibt, lassen sich ähnlich der beschriebenen Erweiterung des Adressenraumes durch zusätzliche Steuerregister hervorrufen. Das läßt sich auf einfache Weise realisieren, indem die Speichermittel für die Adressenerweiterung um zusätzliche Bitpositionen ergänzt werden. Faktisch läuft das auf eine nochmalige Erweiterung des Adressenraumes hinaus, wobei Zugriffe zu bestimmten Teilen mit Sonderwirkungen verbunden sind. Beispielsweise kann in der Anordnung nach Bild 76 der RAM um zwei Bits erweitert werden. Der erste veranlaßt im Rahmen des beschriebenen Bussystems die Erregung von INTERRUPT und das zweite die Unterdrückung fremder Zugriffe (durch Ansteuerung von BURST MODE).
Diese Bits sind vor die jeweiligen Zugriffe mit E-A-Befehlen zu stellen und anschließend wieder auszuschalten. Bild 78 illustriert die Erweiterung gegenüber Bild 76 und veranschaulicht den Ablauf eines Test-and-Set-Zugriffs.

3.2.4. Interrupts

Die Mikrorechner müssen Interrupts nicht nur signalisieren, sondern auch empfangen können. Für die Behandlung einer Unterbrechung ist einfache Signalisierung ihres Auftretens nicht ausreichend. Vielmehr muß zusätzliche Information über die gewünschte Reaktion der CPU bereitgestellt werden. Interne Interrupts, die von lokalen E-A- bzw. anderen Zusatzschaltkreisen erzeugt werden, sind in diesem Zusammenhang unkritisch, da diese Identifizierung im Rahmen der Konventionen zur Interruptbehandlung automatisch erfolgt (s. a. Bild 62).
Bei Interrupts, die von anderen Einrichtungen über den Systembus geliefert werden, ist dies nicht von vornherein gegeben.
Man kann die Schwierigkeiten, die die Realisierung einer speziellen Interruptschaltung bereitet, dadurch umgehen, daß ein Port eines geeigneten E-A-Schaltkreises zur Annahme derartiger Interrupts reserviert wird. Gemäß Bild 79 kann das ein Port einer PIO sein, der mit der Interruptleitung und dem Datenpfad des Bussystems verbunden ist.
Die Identifikation des Interrupts ist zweistufig organisiert: Die PIO liefert einen Interruptvektor, der signalisiert, daß der Interrupt über das Bussystem ausgelöst wurde und somit die Verzweigung zur entsprechenden Behandlungsroutine veranlaßt. Diese holt das Datenbyte, das in die PIO vom Bussystem her eingetragen wurde, und veranlaßt damit über eine zweite Tabelle den Aufruf der Routine, die die eigentliche Interruptreaktion steuert (Bild 80).
Die Konventionen der Interruptsignalisierung können auch anders definiert werden. Beispielsweise ist es ausreichend, über einen E-A-Schaltkreis nur das Eintreffen des Interrupts zu signalisieren und die zugehörigen Parameter zuvor in definierten Speicherzellen abzulegen. Es sind auch Kombinationen möglich, beispielsweise, indem gemäß Bild 80 ein Identifikationsbyte als Datenbyte während des Interruptzyklus übertragen wird und die dadurch gestartete Interruptroutine zusätzliche Parameter aus definierten Speicherzellen entnimmt. Die Folge des Einschreibens der Parameterbytes und das Auslösen des Interrupts muß in einem besonderen Modus ablaufen, der anderweitige Zugriffe auf die betreffenden Speicherzellen ausschließt (BURST MODE beim Bussystem nach Abschnitt 2.).

3.2.5. Mikrorechnerkonfigurationen

In einem komplexen System werden für verschiedene Mikrorechner unterschiedliche Anforderungen an die Ausstattung

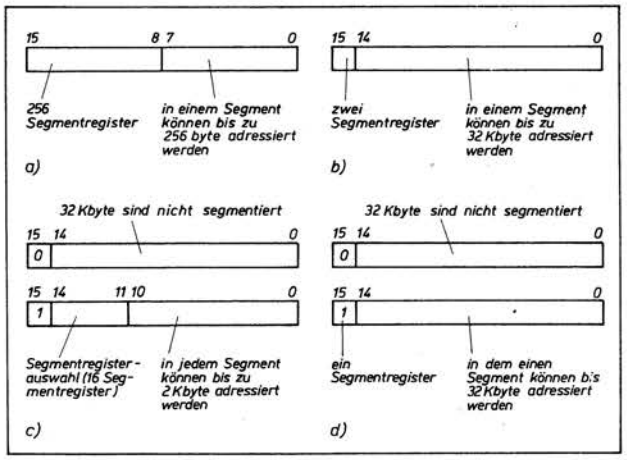
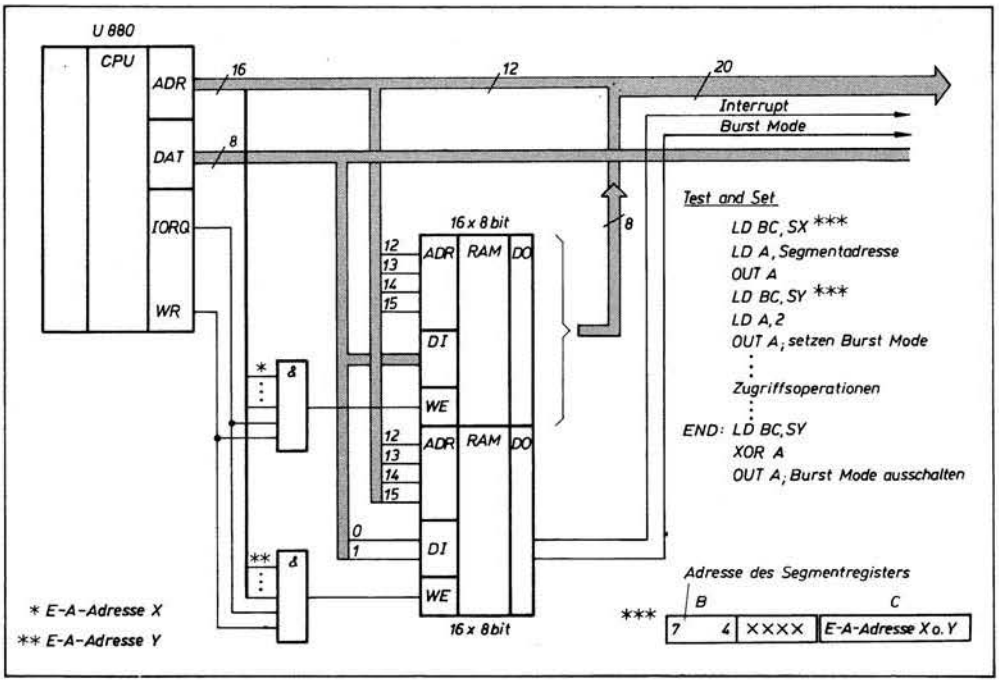


Bild 77: Varianten der Segmentierung des CPU-Adressenraumes

Tafel 8: Alternativen der Segmentierung

Bild	Kennzeichen	Vorteile	Nachteile
77a	viele kleine Segmente	sehr flexible Speicheraufteilung	Organisationsaufwand
77b	wenige große Segmente	direkter Zugriff auf große Speicherbereiche	oftmaliges Umladen der Segmentregister bei „verstreuter" Speicherung
77c	nur teilweise segmentiert, mittlere Zahl von Segmenten	organisatorisch einfacher beherrschbar als a)	weniger flexibel als nach Bild 77a. Die internen 32 Kbyte müssen stets lokal sein
77d	nur teilweise segmentiert, nur ein Segment	minimaler Hardwareaufwand	wie nach Bild 77b und Bild 77c

Bild 78: Beispiel für das Einführen zusätzlicher Steuerwirkungen beim U 880



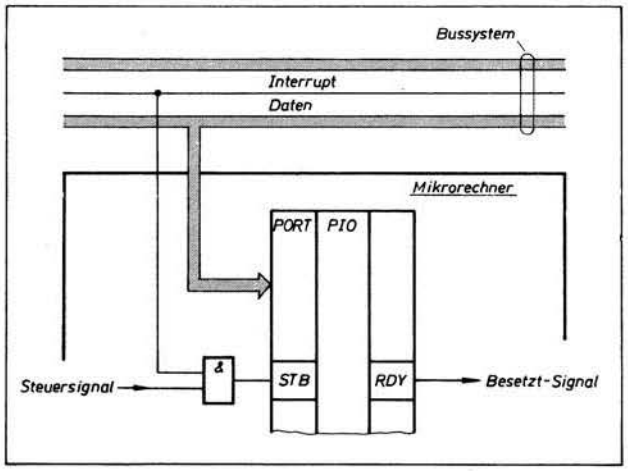


Bild 79: Einsatz einer PIO zum Empfangen von Businterruptsignalen

mit Speichern, E-A-Schaltkreisen usw. bestehen. Andererseits ist eine Standardisierung der Hardware unerläßlich. Die interne Modularisierung der Mikrorechner kann das Problem lösen, bedingt jedoch beachtliche Aufwendungen für Steckverbinder, Rückverdrahtungsplatinen, Steckeinheitenrahmen und dergleichen. Ist dieser Aufwand nicht tragbar, und entscheidet man sich deshalb für eine kompakte Realisierung (kompletter Mikrorechner auf nur einer Leiterplatte), so muß man sich auf wenige standardisierte Typen beschränken.

Ein einziger Typ mit extremer Auslegung (maximale Speicherkapazität, umfangreiche E-A-Ausstattung) ist in den meisten Fällen nicht auf einer Leiterplatte realisierbar. Die Alternative besteht darin, entweder einen Typ mit mittlerer Ausstattung hinsichtlich Speicherkapazität und E-A-Anschlußmöglichkeiten zu schaffen oder mehrere Typen zu entwickeln, die jeweils für bestimmte Klassen von Aufgaben optimiert sind.

Beispielsweise kann ein Typ als Verarbeitungsrechner mit

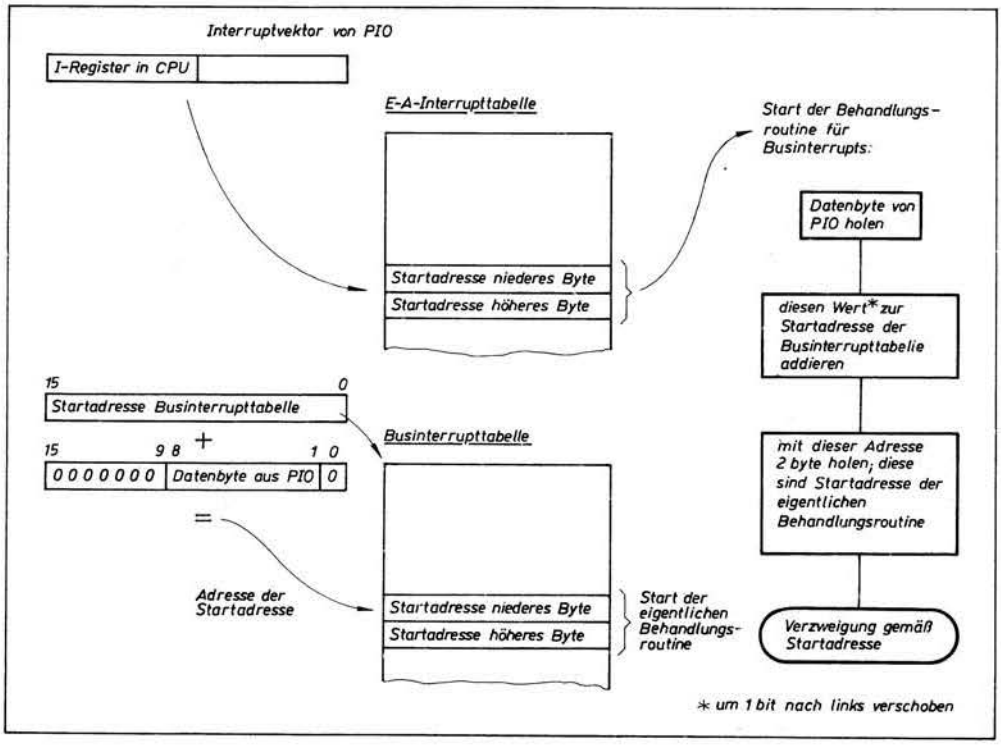


Bild 80: Ablauf der Behandlung von empfangenen Businterrupts im Mikrorechner

Tafel 9: Merkmale verschiedener Mikrorechnertypen

Rechnertyp	Speicherausstattung	E-A-Ausstattung	sonstige Ausstattung	Funktionsmerkmale am Bussystem	Beispiel für Ausstattung
Verarbeitungs-rechner	Maximalausstattung (vorwiegend RAM)	nur für interne Zwecke (Empfangen von Bus-interrupts, als Zeitgeber, zu Diagnosezwecken)	—	alle Funktionen ● Lesen, Schreiben ● Interrupts empfangen ● Interrupts senden ● Burst-Mode-Zugriffe ● extern bedingte Wartezustände	4-Kbyte-ROM 48-Kbyte-RAM 1 CTC 1 PIO
E-A-Rechner	mittlere Ausstattung (vorwiegend RAM)	mit universell einsetzbaren E-A-Schaltkreisen, so daß die E-A-Anschlußkontakte für viele Anwendungen sinnvoll belegt sind	—	nur eine Zugriffsart im Prinzip ausreichend; spezieller isolierter Modus, so daß bei zeitkritischen E-A-Abläufen kein Zeitverzug durch Buszugriffe eintreten kann	8-Kbyte-ROM 16-Kbyte-RAM 1 CTC 2 SIOs 4 PIOs
Arithmetikrechner	mittlere Ausstattung (ROM für Standardroutinen, RAM als Arbeitsspeicher)	keine bzw. wie Verarbeitungsrechner	spezielle Arithmetikschaltkreise	nur eine Zugriffsart (Master bzw. Slave) im Prinzip ausreichend; Möglichkeit zum Empfang von Interrupts wird empfohlen	16-Kbyte-ROM 16-Kbyte-RAM 1 PIO 1 Gleitkommaprozessor

umfangreicher Speicherausstattung, aber ohne E-A-Anschlüsse und einer als E-A-Rechner mit mittlerer Speicherkapazität und möglichst vielen E-A-Anschlüssen ausgelegt werden. Weiterhin wäre für Systeme, in denen viele arithmetische Aufgaben zu lösen sind, ein spezieller Arithmetikrechner denkbar, der statt der E-A-Schaltkreise spezielle Arithmetikschaltkreise enthält und über eine mittlere Speicherausstattung verfügt. Gemäß dem Prinzip, mikroprozessorspezifische LSI-Schaltkreise nur in ihrer natürlichen Zusammenschaltung einzusetzen, müssen zusätzliche Mikrorechnertypen entwickelt werden, wenn neue E-A- oder Spezialschaltkreise angewendet werden sollen. Verarbeitungsrechner müssen Vorkehrungen dafür besitzen, daß komplexe Softwareorganisationsformen implementiert werden können (Test and Set, Interrupt senden, Interrupts empfangen usw.). Die Anschlußschaltungen für den Systembus müssen deshalb für alle vorgesehenen Funktionen ausgelegt werden.

Für die anderen Mikrorechnertypen ist dies nicht erforderlich. Sie müssen lediglich in der Lage sein, Aufträge entgegenzunehmen und Resultate abzuliefern. Dazu ist theoretisch nur eine Zugriffsweise (Buszugriffe entweder nur als Master oder nur als Slave) ausreichend. Sind nur Masterzugriffe möglich, so müssen die Aufträge in einem Speicherbereich hinterlegt werden, der über das Bussystem zugänglich ist. Der E-A- bzw. Arithmetikrechner fragt zyklisch diesen Speicherbereich nach Aufträgen ab, holt von dort die Parameter und hinterlegt die Resultate bzw. die Fertigmeldungen. Sind nur Slavezugriffe möglich, so muß der Auftrag von der erteilenden Einrichtung im Speicher des E-A- oder Arithmetikrechners abgelegt werden. Die Resultate bzw. Fertigmeldungen werden von der erteilenden Einrichtung aus diesem Speicher abgeholt.

Da E-A-Rechner im wesentlichen dafür vorgesehen sind, mit lokalen Programmen Ein- und Ausgabeoperationen über ihre E-A-Schaltkreise zu steuern, muß verhindert werden können, daß externe Zugriffe über das gemeinsame Bussystem die Datenrate verringern. Dies wäre z. B. dann der Fall, wenn Buszugriffe längere Wait-Zustände hervorrufen würden. Die Maßnahmen dafür können entweder durch Hardware oder durch programmtechnisch einstellbare Funktionszustände vorgesehen werden.

Generell ist zu bemerken, daß eine Erweiterung der funktionellen Möglichkeiten hinsichtlich des Anschlusses am Bussystem und der Einstellung besonderer Funktionszustände wesentlich zur Erhöhung der Flexibilität der Mikrorechner beiträgt, so daß eine größere Vielfalt von Systemkonfigurationen und Softwareorganisationsformen realisierbar wird. Die Tafel 9 gibt einen Überblick über die wesentlichen Merkmale der einzelnen Rechnertypen.

Die Auslegung des Mikrorechnersystems kann damit begonnen werden, daß zunächst die unbedingt erforderlichen Grundschaltungen (Takterzeugung, CPU, Treiberschaltungen für den internen Bus, notwendige Anschlußschaltungen für den Systembus) vorgesehen werden. Beim Verarbeitungsrechner werden diese Schaltungen soweit ergänzt, bis alle notwendigen Funktionen ausführbar sind. Die verbleibende Leiterplattenfläche kann dann mit Speicherschaltkreisen und den zugehörigen Steuerschaltungen aufgefüllt werden.

Bei E-A- und Arithmetikrechnern werden zunächst die zusätzlichen LSI-Schaltkreise mit ihrer ergänzenden Beschaltung (Treiber, Adressendekoder usw.) vorgesehen. Die Anzahl der E-A-Schaltungen wird dabei in der Regel durch die Kontaktzahl der E-A-Steckverbinder begrenzt sein. In diesem Rahmen sollte eine sinnvolle Zusammenstellung von E-A-Schaltkreisen vorgesehen werden, z. B. eine CTC, zwei SIOs, vier PIOs. Anschließend ist für die Aufteilung der verbleibenden Leiterplattenfläche ein Kompromiß zwischen Funktionsvielfalt am Bussystem und Speicherausstattung zu finden.

In Verarbeitungs- und E-A-Rechnern sollte, wenn der im vorhergehenden Abschnitt diskutierte Aufwand akzeptabel erscheint, der lokale Speicher vorwiegend mit RAMs realisiert werden, um eine hohe Anpassungsfähigkeit zu gewährleisten. Bei Arithmetikrechnern kann der ROM-Anteil höher sein, da mathematische Standardroutinen in der Regel für viele Anwendungen nutzbar sein dürften.

Bild 81 gibt eine Übersicht über die Anteile der einzelnen Schaltungskomplexe an der Leiterplattenbelegung für die einzelnen Mikrorechnertypen.

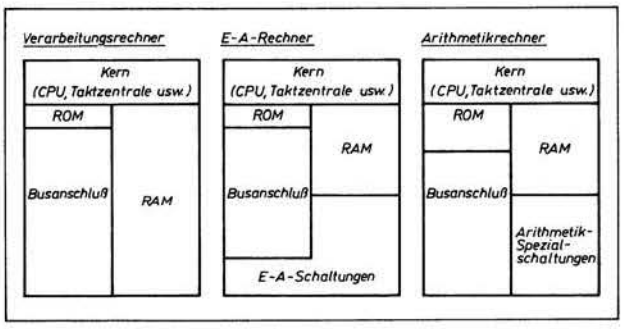


Bild 81: Übersicht über die Belegung der Leiterplatten der einzelnen Mikrorechnertypen

3.2.6. Architekturmerkmale

Die Softwareentwicklung hat bei komplexen Systemen einen sehr hohen Anteil an den gesamten Entwicklungsaufwendungen. Damit wird es sinnvoll, Maßnahmen zu treffen, die gewährleisten, daß diese Aufwendungen bei Weiterentwicklungen, Modernisierungen usw. nicht wieder in vollem Umfang erforderlich werden.
Dies läßt sich praktisch nur erreichen, indem die Schnittstellen zwischen Hardware und Software, also die charakteristischen Merkmale, die das System für den Programmierer hat, sorgfältig definiert und bei Erweiterungen bzw. Modernisierungen konsequent eingehalten werden. Dazu ist es erforderlich, sich von vornherein Klarheit zu verschaffen, welche Merkmale des Systems und seiner Funktionseinheiten Architekturcharakter haben, also unbedingt beibehalten werden müssen, welche Merkmale unbedenklich geändert werden dürfen und bei welchen Erweiterungsmöglichkeiten vorgesehen werden sollten. Die Tafel 10 gibt einen Überblick über die charakteristischen Merkmale.
Bei der Festlegung der Architekturmerkmale sind die praktischen Gegebenheiten der Softwareentwicklung von entscheidender Bedeutung. Es ist theoretisch vorstellbar, daß die gesamte Programmierung auf abstraktem Niveau in höheren Programmiersprachen vorgenommen wird. Man könnte dann die Gewährleistung der Lauffähigkeit auf einer anderen Hardware als Angelegenheit des Compilers (sowie des Laufzeitsystems der neuen Hardware) ansehen. Dies ist aus zwei Gründen derzeit nicht praktikabel:

- Entsprechend leistungsfähige Entwicklungshilfen stehen in der Regel nicht zur Verfügung, so daß auf Mikrorechnerentwicklungssysteme, Cross-Assembler und dergleichen zurückgegriffen werden muß.
- Viele der Aufgaben, die für Multimikrorechnersysteme programmiert werden sollen, bedingen eine der Hardware nahe Programmierung, und zwar zum einen, weil auf Hardwaredetails unbedingt Rücksicht genommen werden muß (vor allem für E-A-Steuerungsaufgaben), und zum anderen aus Effizienzgründen, da die üblichen Mikrorechner in der Regel nicht leistungsfähig genug sind, um trotz des Geschwindigkeitsverlustes, der durch die Compilierung bedingt ist (overhead), den Realzeitanforderungen gerecht zu werden.

Bei der Programmierung mit üblichen Entwicklungshilfen, die die Besonderheiten von Multimikrorechneranwendungen nicht berücksichtigen und dafür keine Unterstützung bereitstellen, muß die Programmierdisziplin durch Absprachen, manuelle Kontrollen usw. gewährleistet werden. Sind beispielsweise die Objektprogramme zur Ausführungszeit nicht verschiebbar (wie beim U 880 der Fall) bzw. ist es nicht möglich, gleichzeitig die Objektprogramme des gesamten Anwendungssystems zu erzeugen, so müssen die Speicherbereiche manuell aufgeteilt und den einzelnen Programmierern zugewiesen werden, die Kommunikation zwischen den Programmen muß abgesprochen werden usw.
Diese Vielfalt von Bedingungen und Konventionen läßt sich für die begrenzte Zeit eines Projektes durchaus beherrschen. Es ist aber unrealistisch anzunehmen, daß nach einer gewissen Zeit diese Software auf einfache Weise an eine neue Hardware mit geänderten Parametern angepaßt werden kann (vielmehr ist eine weitgehende Neubearbeitung einschließlich Testung und Erprobung zu veranschlagen). Aus dieser Situation heraus ergeben sich folgende Alternativen:

- Schaffung entsprechend leistungsfähiger Entwicklungshilfen
- Verzicht auf Nutzung einmal geschaffener Software in Neu- und Weiterentwicklungen (bzw. realistische Abschätzung des tatsächlichen Aufwandes der Nutzung)
- verbindliche Definition aller Parameter, die die Schnittstellen zwischen Hardware und Software bestimmen.

Diese zu definierenden Parameter sind im folgenden genannt:

- Befehlsliste der CPU

Dies bedingt eine gewisse Weitsicht bei der Auswahl der Mikroprozessorfamilie. Es müssen Typen gewählt werden, bei denen man sicher sein kann, daß sie auch nach längerer Zeit noch verfügbar sind. Kriterien dafür sind

— weite Verbreitung auf dem Weltmarkt
— Unterstützung durch die Hersteller (Entwicklungssysteme, Betriebssysteme, Programmiersprachen usw.)
— Lieferung durch mehrere Hersteller.

In modernisierten Mikrorechnern können schnellere oder funktionell verbesserte CPUs eingesetzt werden, aber keine grundsätzlich anderen.

- Belegung der Adressenräume für Speicher und E-A-Zugriffe

Bei Neuentwicklungen sollten die Adressenbereiche in gleicher Weise wie beim Vorgänger belegt sein (mit ROM, RAM, PIOs usw.). Bisher freie Bereiche kann man neu belegen (beispielsweise durch den Einsatz höher integrierter Speicherschaltkreise).

- Funktionen am Bussystem

Für neue Mikrorechner können zusätzliche Operationen vorgesehen werden. Sofern es nur um den Erhalt der Softwarekompatibilität geht, kann die physische Realisierung des Bussystems in weiten Grenzen geändert werden.
Neben der Definition der Hardwareparameter müssen einige Konventionen beim Programmieren eingehalten werden:

Tafel 10: Charakteristische Architekturmerkmale

Komplex	zu erhaltende Merkmale	Erweiterungsmöglichkeiten	unbedenklich änderbar*)
Gesamtsystem	Prinzipien der Kommunikation zwischen den Einrichtungen, Softwareorganisation	zusätzliche Funktionseinheiten, zusätzliche Softwarefunktionen	technische Realisierung, neue E-A-Einrichtungen, interne Arbeitsgeschwindigkeiten der Funktionseinheiten, Prinzipien für Wartung, Testung usw.
Bussystem	Zugriffsmöglichkeiten, Sonderfunktionen, Adressenraum	erhöhte Datenrate, schnellere Masterauswahl, zusätzliche Funktionen, Erweiterung des Adressenraumes	technische Realisierung, Beschleunigung von Abläufen
Mikrorechner	Befehlsliste, Belegung des Adressenraumes (ROM, RAM, E-A usw.), Sonderfunktionen, Funktionen am Bussystem	verbesserte Mikroprozessorschaltkreise (schneller, zusätzliche Befehle), zusätzliche E-A- und Speichermittel, zusätzliche Sonderfunktionen, zusätzliche Funktionen am Bussystem	technische Realisierung, neue Speicherschaltkreistypen usw.

*) Probleme der Hardware-Kompatibilität sind hier außer acht gelassen!

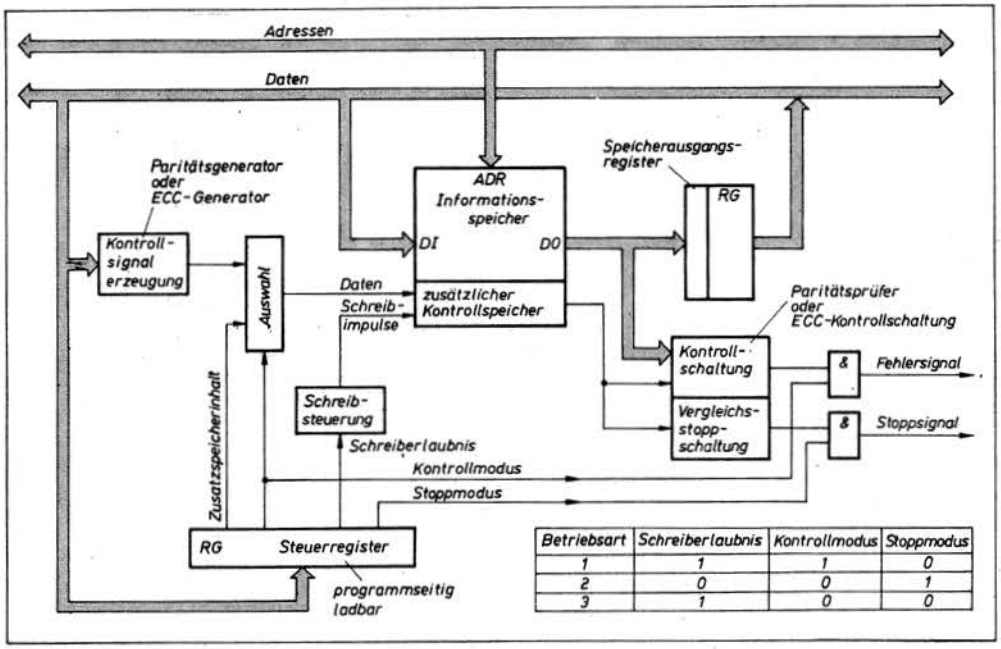


Bild 82: Speicheranordnung mit Zusatzspeicher für Kontrolle und Adressenvergleich

- Programmiertricks, die De-Facto-Eigenschaften der existierenden Hardware ausnutzen (etwa ein bestimmtes Verhalten beim Zugriff zu nicht belegten Adressen; s. a. Bild 59), sind grundsätzlich zu verbieten.
- Zeitgeberroutinen und Abläufe, die extrem zeitkritisch sind, müssen an einer einzigen Stelle lokalisiert werden (etwa als Komponenten des Betriebssystems), so daß spätere Änderungen unproblematisch werden.
- Die Anwendung unüblicher Programmiertechniken (etwa das Modifizieren von Programmen während des Ablaufs) ist sorgfältig zu dokumentieren.

3.2.7. Vorkehrungen für Wartung und Diagnose

In Analogie zum Bussystem müssen für die Fehlersuche bei Funktionsmängeln bzw. Funktionsunfähigkeit der Hardware und für die Suche und Analyse von Softwarefehlern Maßnahmen vorgesehen werden. Das Suchen von Hardwarefehlern basiert üblicherweise auf Testsoftware statt auf einer Vielzahl von Kontroll- und Überwachungsschaltungen. Bei extremen Anforderungen an die Verfügbarkeit werden redundante Systeme eingesetzt (Redundanz auf der Ebene der Funktionseinheiten).
Die Belange der Hardwaretestung müssen bei der Konzipierung der Mikrorechner von Anfang an berücksichtigt werden. Wesentliche Aspekte sind

- Speicherplatz für Testabläufe
- Vorkehrungen für das Auslösen der Testabläufe und für das Anzeigen der Testergebnisse
- Anordnung von Kontrollschaltungen, diagnostischen Einspeisungs- und Abfragemöglichkeiten usw. für jene Funktionskomplexe, die für die programmtechnische Testung nicht zugänglich sind
- allgemein prüffreundliche Auslegung der Hardware.

Damit ist zunächst das Go/No-Go-Testen der Funktionseinheiten möglich (Verifizierungstest).
Um ein ausgefallenes Bauelement zu lokalisieren, hat man die Wahl vorzusehen, daß die verdächtige Funktionseinheit auf einer entsprechenden automatischen Testeinrichtung geprüft wird, oder Testvorkehrungen zu schaffen, die mit erträglichem manuellem Aufwand das Lokalisieren der Bauelemente im Gerät selbst gestatten. Eine bekannte Methode dazu ist die Signaturanalyse [14].
Die erste Möglichkeit ist für die Fertigung der Funktionseinheiten gut geeignet, erfordert aber relativ hohe Investitionen. Für die Kundendienstorganisation bedeutet dieses Verfahren, daß die verdächtigen Funktionseinheiten stets getauscht und zum Hersteller (bzw. zu einem entsprechend ausgerüsteten Reparaturstützpunkt) eingeschickt werden müssen. Beide Methoden erfordern Softwareaufwendungen und die Einhaltung bestimmter Regeln bei der detaillierten Auslegung der Schaltungen.
Ob man in den Mikrorechnern Mittel zur Suche von Softwarefehlern fest anordnet (und somit an die Anwender ausliefert), ist abhängig von dem quantitativen Umfang und der Komplexität der Software, dem Umfang der Hardwarekonfiguration und den Betriebsbedingungen, wobei insbesondere zu erwartende Änderungen durch Benutzeranforderungen und Erwartungen über die Vollständigkeit der Austestung aller Funktionen im Entwicklungsprozeß von Bedeutung sind.
Ist die Software in Umfang und Komplexität überschaubar und kann deren Funktionsfähigkeit vollständig ausgetestet werden, so könnte man auf den festen Einbau derartiger Mittel verzichten. In den anderen Fällen (sehr komplexe Software, viele potentielle Änderungswünsche, Austestung der Funktionsvielfalt nicht vollständig garantiert) erscheint ein völliger Verzicht problematisch.
Die gewünschten Vorkehrungen laufen darauf hinaus, Abläufe beobachten und Speicherinhalte ändern zu können. Das Ändern von Speicherinhalten (RAM) erfordert lediglich Bedien- und Anzeigemittel (Tastenfeld, LED bzw. Bildschirm) sowie entsprechende Dienstprogramme. Maßnahmen zur Ablaufbeobachtung müssen hingegen den Bedingungen des Echtzeiteinsatzes gerecht werden, wenn sie überhaupt einen Sinn haben sollen. Dafür gilt die Forderung, daß die Beobachtung des Echtzeitablaufs diesen nicht beeinträchtigen darf, es sei denn, daß die Auslösung eines Stoppzustandes, eine befehlsweise Abarbeitung oder ähnliches gewünscht ist. Somit sind ausschließlich softwaregestützte Mittel wie das Setzen von Breakpoints nicht geeignet.
Eine Lösungsmöglichkeit besteht darin, an den Mikrorechnern Anschlußpunkte für Logikanalyzer vorzusehen. Allerdings kann man in einem Multimikrorechnersystem mit nur einem Logikanalysator jeweils nur einen Ausschnitt des gesamten Betriebsverhaltens betrachten.
Eine einfache Schaltungsanordnung, die in jedem Mikrorechner fest vorgesehen werden kann, ist in [15] beschrieben. Damit ist es möglich, Aussagen darüber zu erhalten, ob Zugriffe auf bestimmte Speicherzellen bzw. -bereiche ausgeführt wurden oder nicht. Gemäß Bild 82 wird dazu die Speicheranordnung, die überwacht werden soll, durch zusätzliche Speicherschaltkreise erweitert.
Die zusätzlichen Speicher (die stets mit RAMs realisiert sind) sind für zwei alternative Aufgaben vorgesehen. Während des normalen Betriebes (wenn keine Softwaretestung stattfinden soll) können sie zur Fehlerkontrolle der zugeordneten Informationsspeicher benutzt werden. Im einfachsten Fall handelt es sich dabei um eine Paritätsprüfung. Während der Phasen der Softwaretestung können sie als Adressenvergleicher benutzt werden. Wird bei einem Zugriff mit einer bestimmten Adresse in der korrespondierenden Position des zusätzlichen Speichers eine bestimmte Belegung vorgefunden, so führt dies dazu, daß eine Vergleichsbedingung signalisiert wird.

Die zusätzlichen Speicher werden parallel zu den eigentlichen Informationsspeichern adressiert. Für die Lese- und Schreibzugriffe zu den zusätzlichen Speichern gibt es drei alternative Betriebsarten:

- Betriebsart der Speicherinhaltsüberwachung

Dies ist der normale Betriebsfall, in dem der Inhalt des Informationsspeichers durch zusätzliche Kontrollbits überwacht wird. Dabei erfolgt das Schreiben oder Lesen des Zusatzspeichers völlig synchron mit den entsprechenden Zugriffen zum Informationsspeicher. Die Eingangssignale des Zusatzspeichers werden beispielsweise von einem Paritätsgenerator geliefert und die Ausgangssignale von einem Paritätsprüfer ausgewertet. Prinzipiell wäre mit entsprechendem Aufwand an zusätzlichen Speicherschaltkreisen und Kontrollschaltungen auch ein Fehlerkorrekturkode realisierbar.

- Betriebsart der Speicherzugriffsüberwachung (Adressenvergleichsstopp)

Die zusätzlichen Speicher werden nur gelesen (auch bei Schreibzugriffen zu den Informationsspeichern). Wird aus den Zusatzspeichern eine Bitkombination gelesen, die der gewünschten Stoppbedingung entspricht, so wird eine Stoppbedingung in der Hardware wirksam. Im einfachsten Fall liefern die zusätzlichen Speicher nur ein Bit, so daß eine 1 die Stoppbedingung repräsentiert. Ein Beispiel für die Wirkung der Stoppbedingung ist die Auslösung eines NMI, der eine Softwareroutine zur Auswertung, Anzeige usw. startet. Alternativ dazu wäre etwa das Aufrechterhalten eines Wait-Zustandes möglich. Die Konsequenzen sind in der Tafel 11 gegenübergestellt. Es ist ersichtlich, daß zur Suche von Softwarefehlern eine interne Behandlung durch die CPU (über NMI angeregt) Vorteile bietet, da nur auf diese Weise ein Beobachten der internen Register und des Befehlszählers möglich ist. Hingegen könnte das Anhalten des aktuellen Zugriffs von Vorteil sein, wenn man die Schaltung zur Suche von Hardwarefehlern einsetzen möchte.

Tafel 11: Alternativen für die Auswertung der Stoppbedingung

Wirkung in CPU	Wirkung der Stoppbedingung NMI	WAIT
Stopp wirkt in CPU	vor dem nächsten Befehl (der aktuelle Befehl wird zu Ende geführt, unabhängig davon, in welchem Zyklus die Stoppbedingung erkannt wurde)	im aktuellen Zyklus (die CPU liefert weiterhin die aktuelle Adresse)
CPU kann die Bedingung selbst behandeln	ja	nein
Möglichkeit zur Auswertung der CPU-internen Register (A, F, B, C..., SP, PC)'	ja	nein

- Betriebsart der Umsteuerung zwischen den beiden ersten Betriebsarten

Dabei werden mit den zusätzlichen Speichern nur Schreiboperationen synchron mit Schreibzugriffen zu den Informationsspeichern ausgeführt, aber keine Leseoperationen. Die Schreibinformation der Zusatzspeicher wird durch Inhalte von Steuerregistern bestimmt. Im einzelnen können Ausgangssignale des Paritäts- bzw. ECC-Generators, feste Werte 0 und feste Werte 1 eingeschrieben werden.

3.3. Beispiel eines Mikrorechners

Um die Probleme der Konzipierung von Mikrorechnerbaugruppen näher zu veranschaulichen, wird in diesem Abschnitt ein Mikrorechner vorgestellt, der für den Anschluß an das im Kapitel 2. beschriebene Bussystem vorgesehen und auf der Basis der U-880-Schaltkreisfamilie aufgebaut ist. Konstruktiv ist er als Einkartenrechner für den Einbau in Anordnungen gemäß Bild 5 ausgeführt.

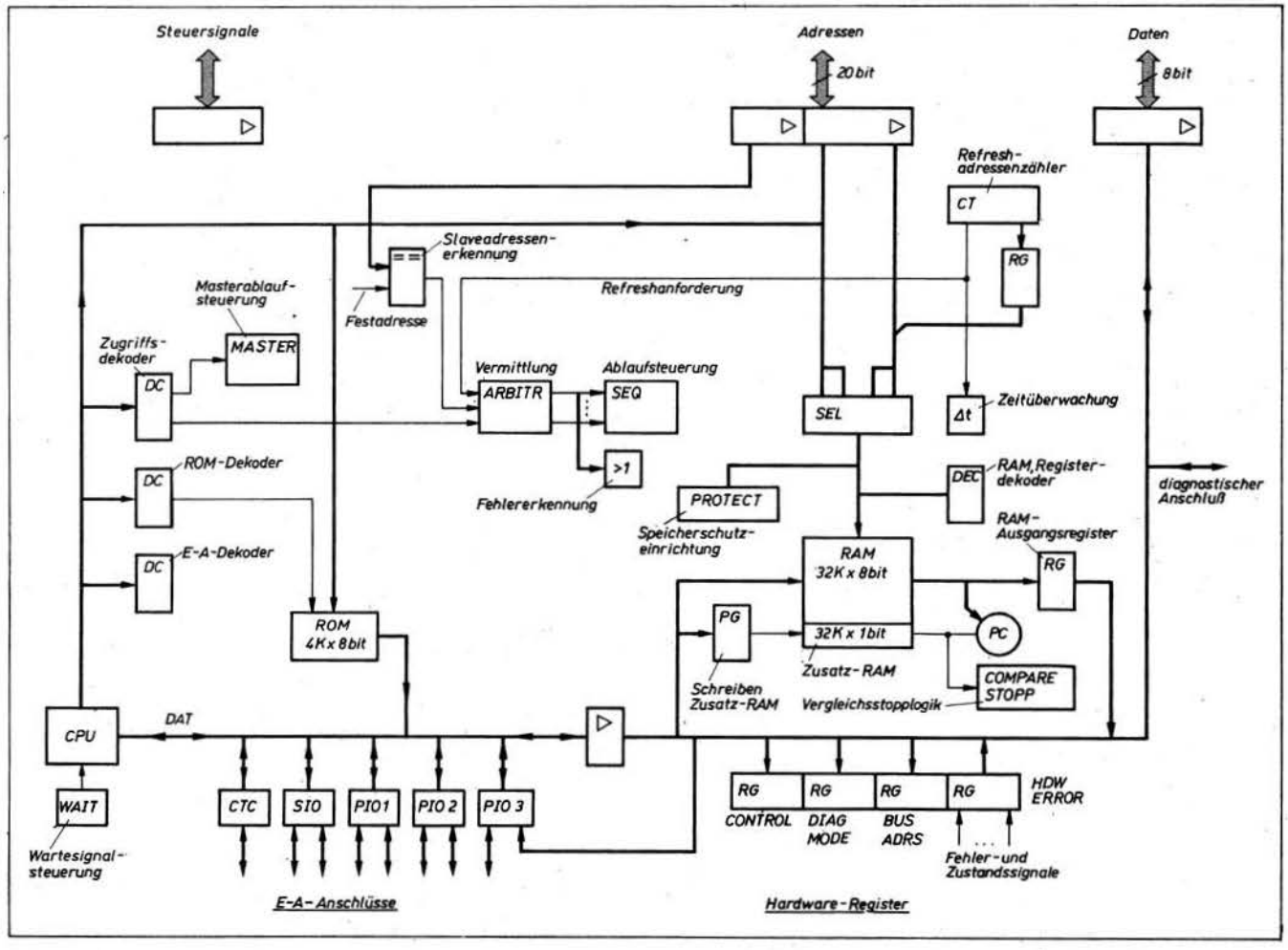


Bild 83: Blockschaltbild des als Beispiel gewählten Mikrorechners

3.3.1. Ausstattungsmerkmale

Der Mikrorechner, dessen Struktur im Bild 83 dargestellt ist [16], hat folgende Ausstattungsmerkmale:

CPU	U 880
E-A-Schaltkreise	1 CTC 1 SIO (Wait-Steuerung möglich) 3 PIOs (fünf Ports extern verfügbar)
Speicher	4 Kbyte ROM-Bereich 32 Kbyte RAM-Bereich
Sondermerkmale	RAM mit zusätzlichem 1-bit-Speicher für Paritätskontrolle bzw. Adressenvergleichsstopp, vier programmtechnisch zugängliche 1-byte-Register, spezieller Anschluß für Diagnosezwecke
Funktionen am Bussystem	Masterzugriffe mit 20-bit-Adressen, Anforderung von BURST MODE und Auslösung von Interrupts in anderen Einrichtungen möglich, Slavezugriffe auf RAM, zu Hardwareregistern und für Interruptauslösung, Signalisierung interner Fehler, interne Wartezustände auf Grund von Bussignalen (NES, BURST MODE), selektives Rücksetzen wird ausgewertet, spezielle Arbeitszustände (z. B. das Abweisen von Buszugriffen) sind programmtechnisch einstellbar

3.3.2. Aufteilung des Adressenraumes

Bild 84 zeigt, wie die von der CPU gelieferte 16-bit-Adresse für Speicher- und E-A-Zugriffe aufgeteilt ist.

Die Adressierung der E-A-Schaltkreise weist prinzipiell keine Unterschiede zu herkömmlichen U-880-Mikrorechner-Anordnungen (z. B. K 1520) auf.

Hingegen muß die Speicheradresse den Zugriff zu verschiedenen Einrichtungen ermöglichen, nämlich zum ROM, zum RAM, zu den speziellen Hardwareregistern und zu anderen Einrichtungen über das Bussystem (Masterzugriffe). Die entsprechenden Adressenformate sind in den Bildern 85 und 86 dargestellt.

Bei ROM-Zugriffen kann die CPU maximal 2 Kbyte (beginnend ab Adresse 0) adressieren. Zu welchem 2-Kbyte-Bereich der insgesamt installierten 4 Kbyte dabei zugegriffen wird, bestimmt ein Bit in einem Steuerregister (CONTROL REG Bit 0). Ein Wechsel zwischen beiden Bereichen ist durch programmtechnisches Umschalten dieses Bits möglich. Ab Adresse 8000H ist der gesamte RAM-Bereich (32 Kbyte) direkt zugänglich.

Der Bereich für Registerzugriffe beginnt ab Adresse 2000H. Es sind jedoch nur sechs Registerpositionen vorgesehen, von denen zwei nur gelesen und vier beschrieben werden können. Bei Zugriffen zu den Schreibregistern werden parallel Zugriffe zu definierten RAM-Plätzen ausgeführt, so daß im RAM automatisch Kopien der Registerinhalte mitgeführt werden. Bei Schreibzugriffen wird parallel in das Register und die RAM-Position geschrieben. Bei Lesezugriffen können hingegen die Hardwareregister keinen Beitrag liefern, so daß die Information lediglich aus dem RAM entnommen wird. Durch diese Maßnahme können alle inhaltsmodifizierenden Befehle auf die Registerinhalte angewendet werden (etwa SET, RES, INCR), und es sind Abfragen bzw. Vergleiche möglich (BIT, AND, CMP). Da aus Aufwandsgründen der Dekoder für die Registerzugriffe an die Auswahlschaltung für die RAM-Adressen angeschlossen wurde (es sind dynamische 16-Kbit-MOS-IS eingesetzt), ergeben sich für die vier Schreibregister recht krumme Adressenpositionen. Dieser Kompromiß zwischen

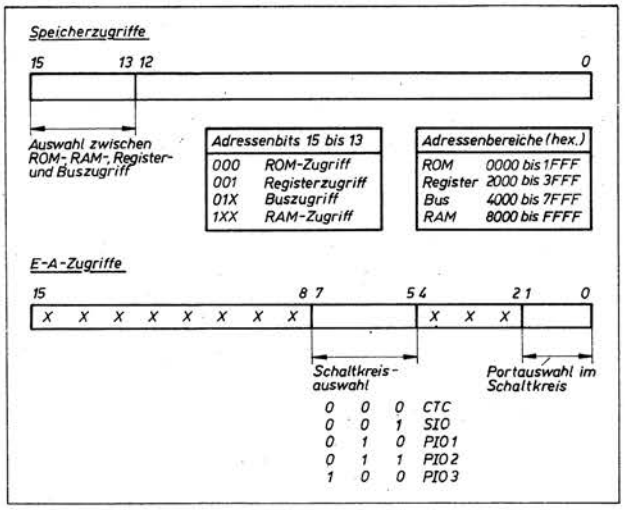


Bild 84: Aufteilung des Adressenraumes

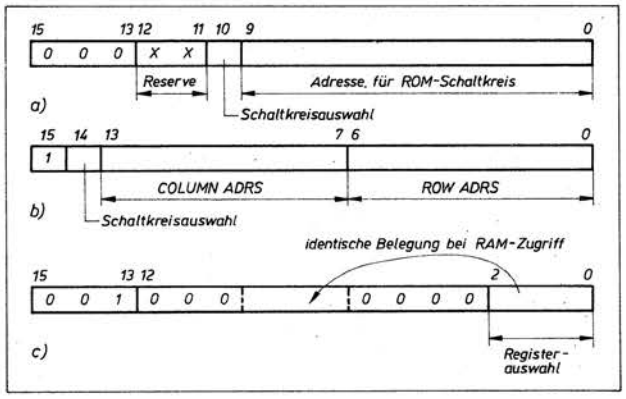


Bild 85: Adressenformate für interne Zugriffe. a) ROM-Zugriffe; b) RAM-Zugriffe; c) Registerzugriffe

Aufwand und Eleganz ist tragbar, da die niedrigsten 512 Bytes des RAM an sich für Steuerbereiche, Interrupttabellen und ähnliches vorgesehen sind, so daß es nicht schwerfällt, auf die Registerkopien Rücksicht zu nehmen. Eine elegante Einordnung der Registeradressen, etwa 2000H, 2001H usw., kostet etwa zwei IS-Gehäuse mehr. Für Masterzugriffe wird ein 16 Kbyte großer Bereich ab Adresse 4000H benutzt, das heißt, immer dann, wenn die CPU einen Zugriff zu diesem Adressenbereich ausführt, wird eine Masteranforderung für das Bussystem gestellt. Da für dieses eine 20-bit-Adresse benötigt wird, die CPU aber effektiv nur die niederen 14 Bits liefern kann, werden die fehlenden sechs Bits durch den Inhalt des Busadressenregisters ergänzt. Dieses enthält weiterhin noch Steuerbits für Sonderfunktionen im jeweiligen Buszugriff (Übertragung im Burst Mode, Senden von Interrupts).

Wird der Mikrorechner als Slave adressiert, so sind externe Zugriffe zu den Registern und zum RAM möglich. Die niederen 16 Bits müssen dazu gemäß den Bildern 85 und 86 aufbereitet werden, und die höchsten vier Bits der 20-bit-Busadresse müssen die Slaveadresse des Mikrorechners enthalten. Slavezugriffe zu anderen Bereichen des Mikrorechners werden von dessen Hardware als fehlerhaft erkannt.

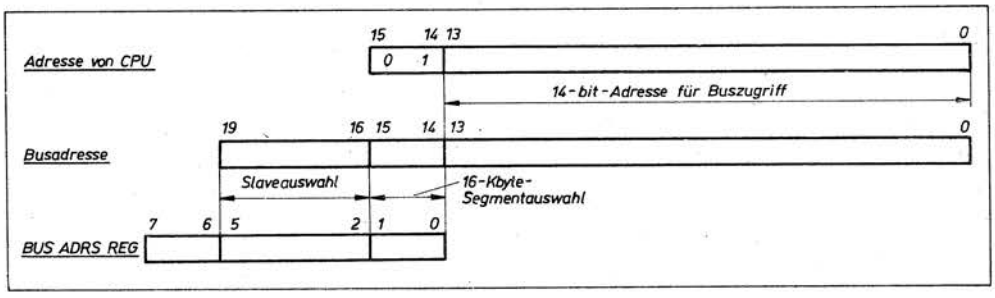


Bild 86: Adressenbildung bei Masterzugriffen

3.3.3. Registerbelegungen

Die Tafel 12 gibt einen Überblick über die vorgesehenen Registerpositionen. Die diagnostischen Register sind dabei in der Hardware nicht vorhanden. Vielmehr werden die betreffenden Zugriffe zum diagnostischen Anschluß geführt. Eine dort angeschlossene Diagnoseeinrichtung muß diese Signale auswerten. Da durch die Registerzugriffe Bytes in die Diagnoseeinrichtung transportiert und von dort gelesen werden können, besteht eine sehr große Freizügigkeit in der Definition der Zusammenarbeit des Mikrorechners mit externen Diagnosemitteln.

Die Belegung der Schreibregister zeigt Bild 87, die des Fehlerregisters (HDW ERROR) Bild 88.

3.3.3.1. CONTROL-Register

● INITIALIZED

Dieses Bit dient dazu, einen der beiden 2-Kbyte-Bereiche des ROM auszuwählen. Nach dem Rücksetzen des Mikrorechners durch das Bussignal RESET ist INITIALIZED = 0. Damit beginnt die Programmabarbeitung von Adresse 0 des ersten ROM-Bereiches. Es handelt sich dabei um Testroutinen. Wurden diese erfolgreich ausgeführt, so wird INITIALIZED eingeschaltet, so daß aus dem zweiten ROM-Bereich Anwendungs- und Dienstprogramme abgearbeitet werden können. Im besonderen wird durch selektives Rücksetzen INITIALIZED nicht zurückgesetzt, so daß nach erfolgreichem Eigentest des Mikrorechners ein selektives Rücksetzen einen Programmstart von Adresse 0 des zweiten ROM-Bereiches veranlaßt.

Um INITIALIZED einzuschalten, muß eine Routine im ersten ROM-Bereich ein entsprechendes Programmstück in den RAM laden und dorthin verzweigen.

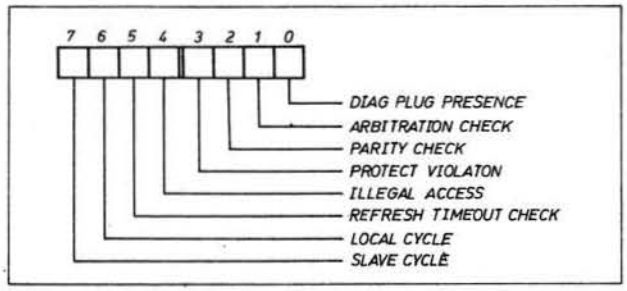


Bild 88: Belegung des Fehlerregisters (HDW ERROR)

● ENABLE PARITY

Die Paritätskontrolle am RAM wird zugelassen bzw. unterdrückt.

● ENABLE ERROR SIGNALIZATION

Das Signalisieren interner Fehlerbedingungen über die Busleitungen ERROR bzw. SLAVE ERROR wird zugelassen bzw. unterdrückt.

● ENABLE COMPARE STOP

Der Adressenvergleichsstopp (mit Hilfe des neunten RAM-Bits) wird zugelassen bzw. verhindert. Das Bit darf nicht gleichzeitig mit ENABLE PARITY gesetzt sein.

● STORAGE PROTECT BOUNDARY

Der RAM ist in aufeinanderfolgenden 2-Kbyte-Bereichen auf Schreibzugriffe überwachbar, so daß beim Schreiben in einen derart geschützten Bereich eine Fehlerbedingung signalisiert wird.
Das Feld enthält eine Kodierung für die obere Grenze des geschützten Bereiches.

Tafel 12: Hardwareregister des Mikrorechners

Adresse	Register	Zugriff	Verwendung
2000H	CONTROL	Schreiben in Hardware und RAM, Lesen der RAM-Kopie	Halten von Zustands- und Erlaubnissignalen sowie der Speicherschutzgrenze
2081H	DIAGNOSTIC MODE (DIAG MODE)		Steuerung und Einstellung der Vergleichsstoppbetriebsart, Steuerung der Buszugriffsverhinderung
2102H	BUS ADRS		enthält die höchsten sechs Bits der Busadresse sowie Bits für Interrupt und Burst Mode
2183H	DIAGNOSTIC OUTPUT (DIAG OUTPUT)		Liefern von Ausgabeinformation zum diagnostischen Anschluß
2206H	DIAGNOSTIC INPUT (DIAG INPUT)	nur Lesen	Holen von Eingabeinformation vom diagnostischen Anschluß
2207H	HDW ERROR		Fixieren von Hardwarefehlern und -zuständen für die programmtechnische Fehlerbehandlung

Register	Register-Adresse	RAM-Position	Bitposition 7	6	5	4	3	2	1	0
CONTROL	2000H	8000H	STORAGE PROTECT BOUNDARY				ENABLE COMPARE STOPP	ENABLE ERROR SIGNALIZATION	ENABLE PARITY	INITIALIZED
DIAG MODE	2081H	8081H	×	×	SUPPRESS EXTERNAL WAIT CONDS	REJECT SLAVE ACCESS	STOPP ON WRITE	STOPP ON READ	INJECT HIGH, FORCE EVEN PARITY	ENABLE INJECT
BUS ADRS	2102H	8102H	BURST MODE WANTED	INTERRUPT	Adresse des Slave am Bus				Adresse eines 16-Kbyte-Segmente	
DIAG OUTPUT	2183H	8183H	Ausgabeinformation für Diagnoseanschluß							

Bild 87: Belegung der vier Schreibregister

3.3.3.2. DIAG-MODE-Register

● ENABLE INJECT

Das Einschreiben in die zusätzliche Bitposition des RAM wird außerhalb der Paritätsprüfung erlaubt bzw. verhindert.

● INJECT HIGH/FORCE EVEN PARITY

Das Bit hat zwei verschiedene Wirkungen. Ist ENABLE INJECT aktiv, so bestimmt es die einzuschreibende Information für das zusätzliche RAM-Bit, d. h., die Belegung dieser Register-Position wird in das neunte Bit übernommen. Ist dagegen ENABLE PARITY aktiv, so veranlaßt es, daß der Paritätsgenerator auf gerade Parität eingestellt wird. Da für die Paritätskontrolle ungerade Parität vorgesehen ist, muß somit bei jedem Lesezugriff zum RAM ein Paritätsfehler signalisiert werden. Auf diese Weise sind die Schaltungen der Paritätskontrolle selbst prüfbar.

● STOP ON READ

Das Bit ist nur von Bedeutung, wenn der Vergleichsstoppmodus eingestellt ist. Es veranlaßt, daß die Stoppbedingung bei Lesezugriffen wirksam wird.

● STOP ON WRITE

Analog zu STOP ON READ veranlaßt dieses Bit, daß eine Stoppbedingung bei Schreibzugriffen wirksam wird. Beide Bits können auch gleichzeitig gesetzt sein. Ist keines von beiden eingeschaltet, so werden Vergleichsstoppbedingungen nie wirksam.

● REJECT SLAVE ACCESS

Ist dieses Bit gesetzt, so werden alle Slavezugriffe zum Mikrorechner abgewiesen, d. h., mit RELEASE beantwortet. In der Zeit, in der das Bit gesetzt ist, werden somit interne Abläufe nicht durch Zugriffe anderer Funktionseinheiten gestört.

● SUPPRESS EXTERNAL WAIT CONDS

Ist dieses Bit gesetzt, so werden Wartezustände, die durch die Busleitung NES bzw. durch Slavezugriffe im Burst Mode normalerweise entstehen, unterdrückt. Dies ist dann erforderlich, wenn ein Verzögern der Befehlsausführung zu inkorrekten Resultaten führt. Beispielsweise dürfen bei Zugriffen zu CTC-Schaltkreisen keine Wait-Zyklen eingefügt werden.
Ein anderes Beispiel ist die Steuerung des Datenaustausches mit einem Floppy-Disc-Laufwerk. Infolge der relativ hohen Datenrate würde bei längeren Wartezuständen ein Datenverlust auftreten.

3.3.3.3. BUS ADRS

● Adresse eines 16-Kbyte-Segmentes

Diese zwei Bits ergänzen die niederen 14 Bits der CPU-Adresse zur 16-bit-Adresse, so daß ein Zugriff zu einem 64-Kbyte-Adressenraum im Slave möglich ist.

● Adresse des Slave am Bus

Mit diesen vier Bits kann eine von 16 möglichen Slavefunktionseinheiten ausgewählt werden.

● INTERRUPT

Diese Leitung kennzeichnet, daß in der Slaveeinrichtung beim nächsten Zugriff ein Interrupt ausgelöst wird (d. h., in den Zugriffen, bei denen das Bit gesetzt ist, wird die Busleitung INTERRUPT erregt).

● BURST MODE WANTED

Solange dieses Bit gesetzt ist, werden Masterzugriffe im Burst Mode ausgeführt. In diesem Modus unterbleibt die Masterauswahl am Bussystem, so daß der Mikrorechner durch Anfordern von Burst Mode den Bus für sich monopolisieren kann.

3.3.3.4. HDW ERROR-Register

Dies ist das Fehlerregister des Mikrorechners. Es kann programmseitig nur gelesen werden. Die einzelnen Bits haben die folgenden Bedeutungen:

● DIAG PLUG PRESENCE

Dieses Bit zeigt an, ob am Diagnoseanschluß eine entsprechende Einrichtung angeschlossen ist.

● ARBITRATION CHECK

Speicher-Vermittlungsfehler: Zugriffe zum RAM sind erforderlich seitens der CPU, seitens des Bussystems (Slavezugriffe) und zum Auffrischen der Information (Refreshzugriffe). Zu einem Zeitpunkt kann nur ein Zugriff ausgeführt werden, so daß eine Vermittlungseinrichtung erforderlich ist. Es wird kontrolliert, daß diese tatsächlich jeweils nur einen bestimmten Zugriff veranlaßt (und nicht mehrere gleichzeitig).

● PARITY CHECK

Dies ist das Paritätsfehlersignal des RAM.

● PROTECT VIOLATION

Dieses Signal wird von der Speicherschutzeinrichtung erzeugt, wenn ein Schreibzugriff in einem geschützten Bereich ausgeführt wurde.

● ILLEGAL ACCESS

Diese Fehlerbedingung resultiert aus einer Plausibilitätskontrolle der Adressendekodierung und aus einer Überwachung der Registerzugriffe bei Slavezyklen.
Als Fehlerursachen gelten interne Zugriffe im Adressenbereich 4000H bis 7FFFH (bei Slavezugriffen ist dieser Bereich unzulässig, und bei CPU-seitigen Zugriffen dürfen keine internen Zugriffsoperationen ausgeführt werden) und Schreibzugriffe zu den Registern bei Slavezyklen, wenn NES nicht erregt ist. Damit soll ein unkontrolliertes Überschreiben der Register signalisiert werden. Anwendungsprogramme dürfen dies nicht.

● REFRESH TIMEOUT CHECK

Dieses Fehlersignal wird erzeugt, wenn der Abstand zwischen zwei Refreshzyklen zu groß geworden ist bzw. wenn gar keine Refreshzyklen mehr ausgeführt werden. In derartigen Fällen muß man mit einem Informationsverlust im RAM rechnen.

● LOCAL CYCLE, SLAVE CYCLE

Diese beiden Zustandssignale dienen zur Identifizierung des internen Zyklus, in dem die Fehlerbedingung in das Register übernommen wurde. Sie werden von den Fehlerbehandlungsroutinen ausgewertet.

3.3.4. Einzelheiten

Die Anordnung nach Bild 83 unterscheidet sich von üblichen Mikrorechnern dadurch, daß es in ihr mehrere unabhängige Funktionsabläufe gibt, von denen einige zeitweilig die gleiche Hardware benutzen. Da auch parallele Abläufe nacheinander beschrieben werden müssen, bereitet eine fortlaufende systematische Erläuterung einige Schwierigkeiten, die durch die Gliederung gemindert werden sollen.
Die Beschreibung betrifft nur Dinge, die neuartig, besonders problematisch oder als Schaltungsvorschlag von Interesse sind. Geläufige Einzelheiten, wie etwa das Zusammenwirken zwischen CPU und E-A-Schaltkreisen oder die Zugriffe zum ROM, werden nicht näher ausgeführt.

3.3.4.1. RAM-Komplex

Die eigentliche Speicheranordnung besteht aus 18 dynamischen 16-Kbit-nMOS-Speicherschaltkreisen (Bild 89). Für jede Bitposition sind zwei Schaltkreise vorgesehen (Wired-OR-Verbindung). Die Selektion erfolgt durch die Steuersignale RAS 0, RAS 1. Alle Zugriffe erfolgen byteparallel, wobei für die neunte Bitposition die Schreibimpuls- und Datenleitungen in besonderer Weise beschaltet sind.
Der RAM-Komplex ist an einen Teil des Datenbusses angeschlossen, der vom CPU-Datenbus durch Koppelschaltkreise isoliert ist (Bild 90).

Es gibt als Ursachen für RAM-Zugriffe Refreshzugriffe, Slavezugriffe vom Bussystem und interne Zugriffe von der CPU. Für das Auffrischen des RAM ist eine spezielle Hardware vorgesehen. Die Refreshzyklen der CPU sind aus zwei Gründen nicht nutzbar:

- Wait ist unwirksam, so daß sie nicht an die Zeitbedingungen des konkreten RAM-Zugriffs angepaßt werden können.
- Durch Wait-Zustände bei Befehls- bzw. Datenzugriffen können die Intervalle zwischen Refreshzyklen intolerabel verlängert werden (beispielsweise durch Non Executive State oder durch das Warten auf den Buszugriff bei Master-Anforderungen).

Bild 89: RAM-Array mit den Schaltmitteln des Datenpfades

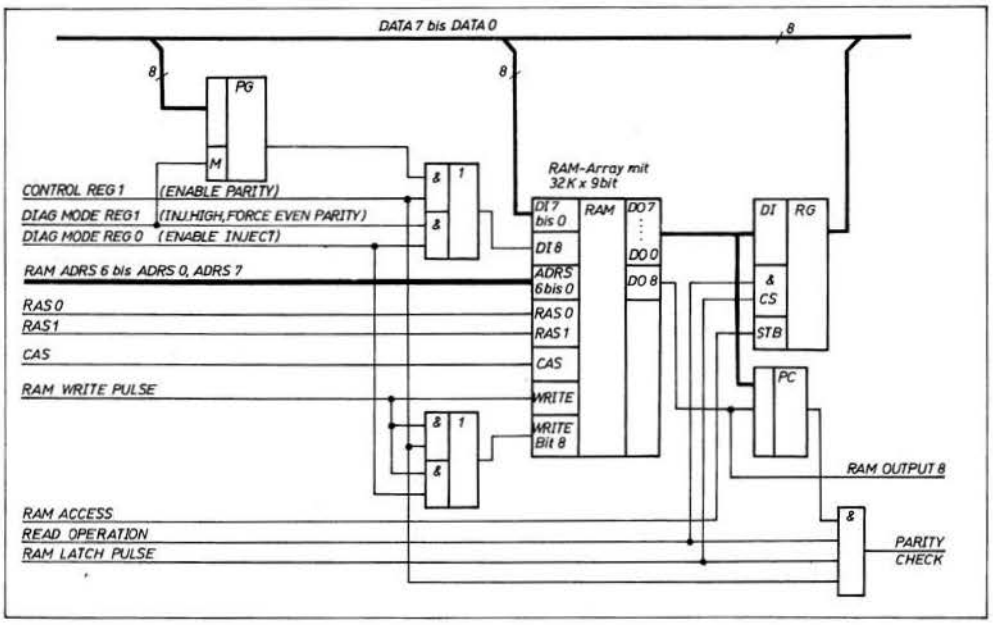


Bild 90: Datenbus

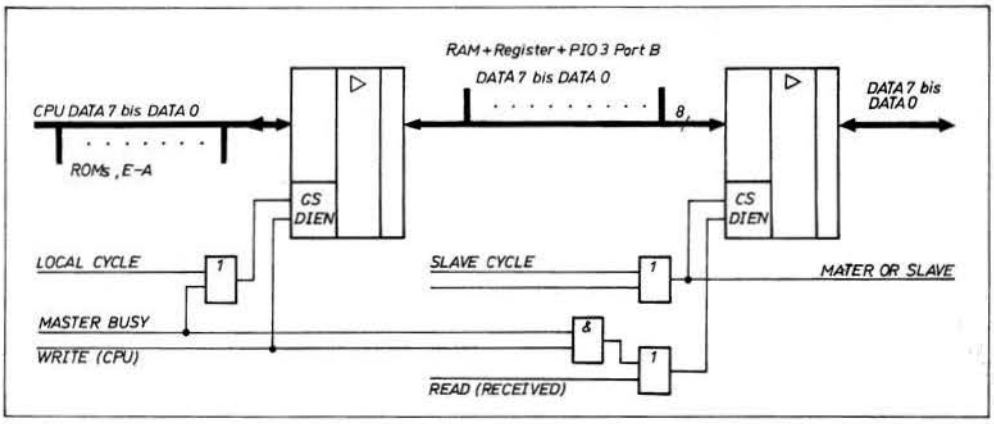
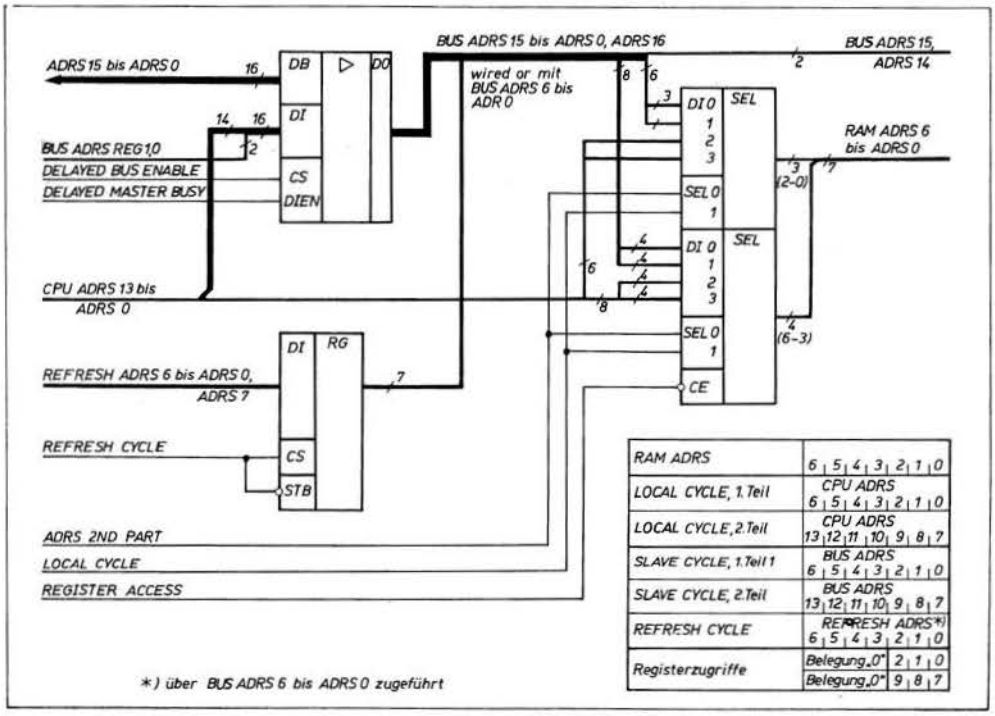


Bild 91: Bildung der RAM-Adressen



RAM ADRS	6 | 5 | 4 | 3 | 2 | 1 | 0
LOCAL CYCLE, 1. Teil	CPU ADRS 6 | 5 | 4 | 3 | 2 | 1 | 0
LOCAL CYCLE, 2. Teil	CPU ADRS 13 | 12 | 11 | 10 | 9 | 8 | 7
SLAVE CYCLE, 1. Teil 1	BUS ADRS 6 | 5 | 4 | 3 | 2 | 1 | 0
SLAVE CYCLE, 2. Teil	BUS ADRS 13 | 12 | 11 | 10 | 9 | 8 | 7
REFRESH CYCLE	REFRESH ADRS*) 6 | 5 | 4 | 3 | 2 | 1 | 0
Registerzugriffe	Belegung „0" 2 | 1 | 0
	Belegung „0" 9 | 8 | 7

Den drei Ursachen für RAM-Zugriffe entsprechen drei Quellen für RAM-Adressen (Bild 91). Über Multiplexer-IS wird die 7-bit-Adresse für die RAMs gebildet.
Während die CPU-Adresse diesen Schaltkreisen direkt zugeführt wird, werden die Busadresse der Slavezugriffe und die Refreshadresse über Tristatestufen zusammengefaßt.

Da zu einem Zeitpunkt nur ein RAM-Zugriff ausgeführt werden kann, ist eine Vermittlungsschaltung erforderlich (Bild 92). Für jede Art des Zugriffs gibt es ein Anforderungssignal, das in einem Anforderungsregister gespeichert wird. Aus den vorliegenden Anforderungen ermittelt ein Prioritätsnetzwerk die Anforderung mit der jeweils höchsten

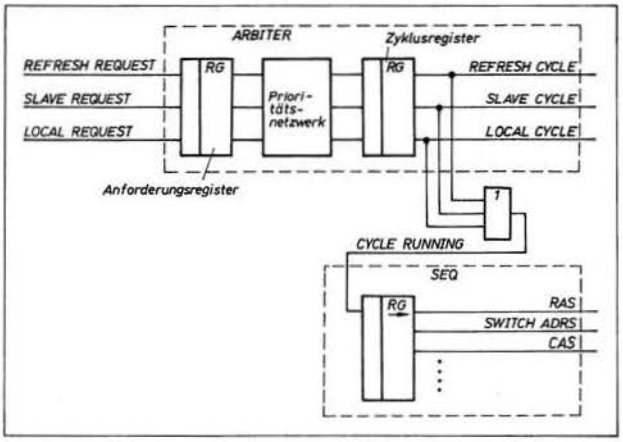


Bild 92: Prinzip der Vermittlung der Speicheranforderungen

Priorität. Dementsprechend wird das Zyklusregister geladen. Daraufhin wird eine Folgesteuerung aktiviert, die die Zugriffssteuersignale (wie RAS, CAS usw.) abgibt.
Bild 93 zeigt Details der Vermittlungsschaltung. Der Kern der Ablaufsteuerung ist im Bild 94 dargestellt.
Die einzelnen Zugriffsanforderungen haben folgende Prioritäten:

1. (höchste) Priorität: Refreshanforderungen
2. Priorität: Slaveanforderungen
3. Priorität: lokale Anforderungen (von der CPU).

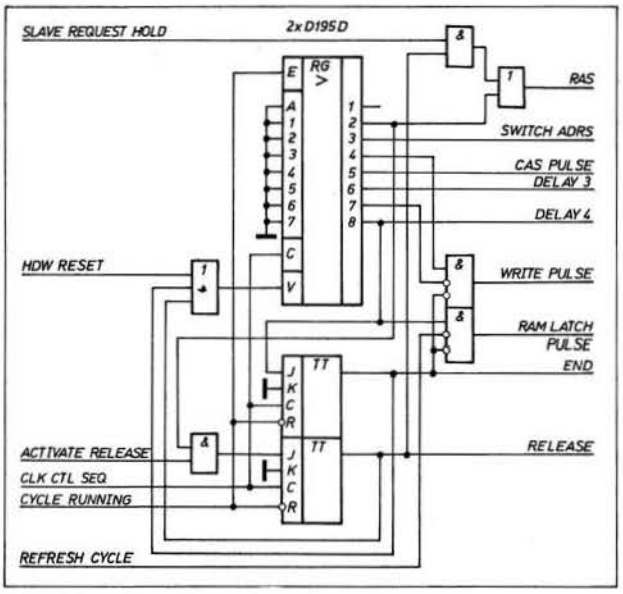


Bild 94: Ablaufsteuerschaltung

Bild 95 veranschaulicht den Ablauf eines Refreshzyklus. CLK CTL SEQ entspricht dem Grundtakt des Mikrorechners (etwa 13,8 MHz). Daraus wird durch 1:3-Teilung der Vermittlungstakt CLK RQ LATCH gebildet.

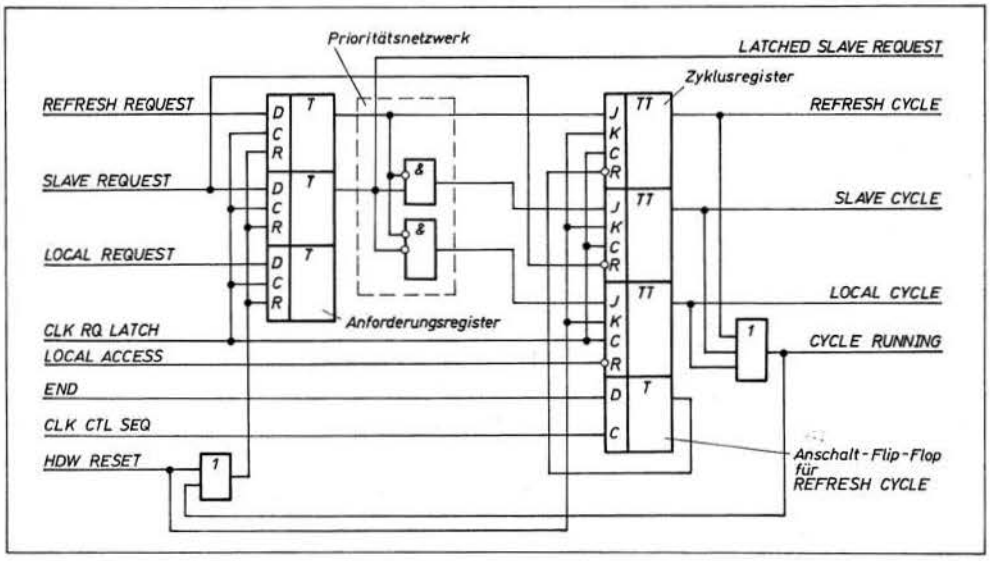


Bild 93: Vermittlungsschaltung

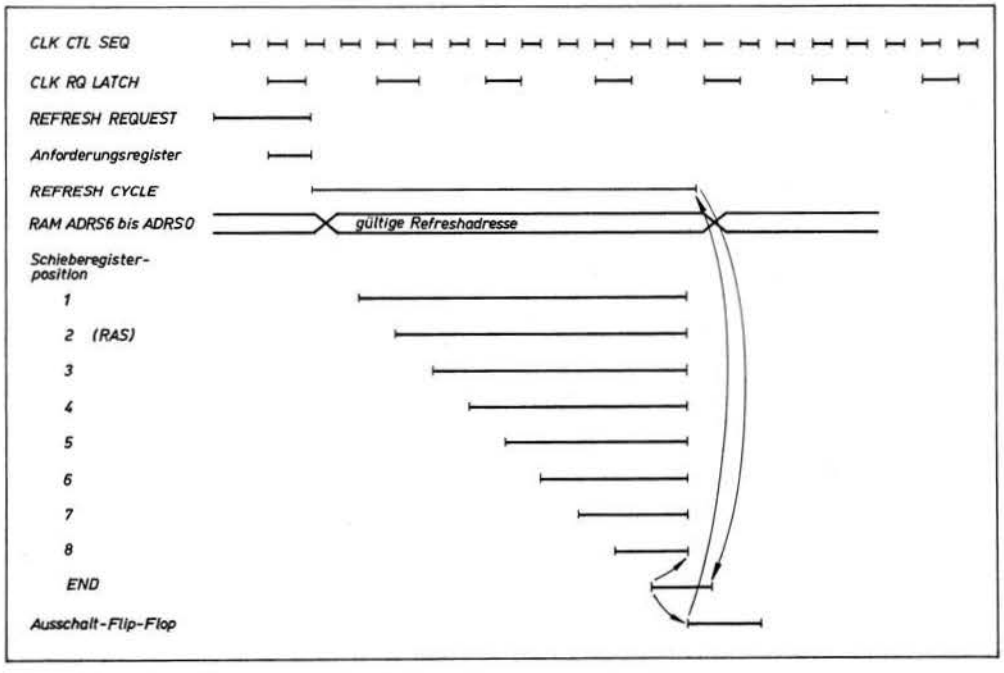


Bild 95: Ablauf eines Refreshzyklus

Die Refreshanforderung entsteht etwa alle 12 μs durch ein Signal des Refresh-Adressenzählers (Bild 96). Das Anforderungssignal wird mit der Vorderflanke von CLK RQ LATCH in das Anforderungsregister übernommen. Die Rückflanke dieses Taktsignals veranlaßt das Einschalten von REFRESH CYCLE und damit das Sperren des Anforderungsregisters für weitere Anforderungen (durch Aktivieren der Rücksetzeingänge). Mit REFRESH CYCLE wird die Refreshadresse auf den Tristatebus geschaltet (der Refreshadressenzähler kann unabhängig davon weiterzählen), und die Ablaufsteuerung wird aktiviert. Deren Schieberegister gibt nacheinander zeitversetzte Steuerimpulse ab, aus denen die Steuersignale für die RAMs durch einfache kombinatorische Verknüpfungen gebildet werden (Bilder 97, 98). Durch das END-Signal wird das Schieberegister zurückgesetzt. Im allgemeinen signalisiert END, daß die betreffende Anforderung erledigt wurde (Erregen von REPLY bei Slavezugriffen, Ausschalten des Wartezustandes bei lokalen Zugriffen). END bleibt solange aktiv, bis durch das Abschalten des Anforderungssignals das Zyklusregister zurückgesetzt wird. Bei

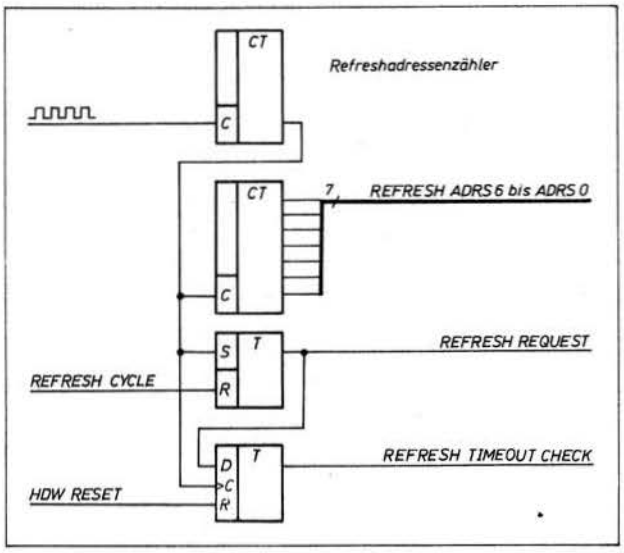


Bild 96: Bildung der Refreshadresse und Erkennung des Refreshzeitüberschreitungsfehlers

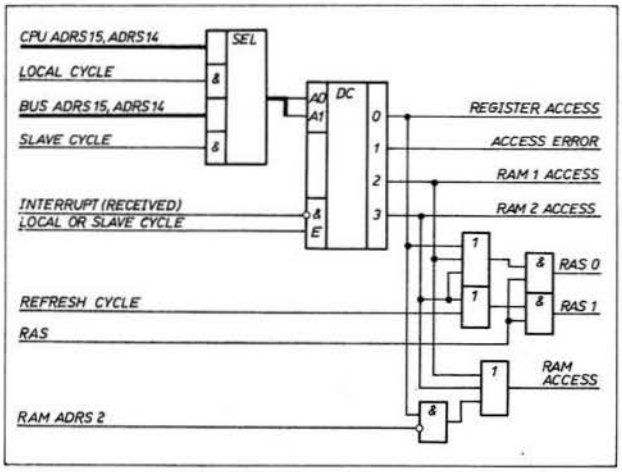


Bild 97: Bildung der Zugriffssignale

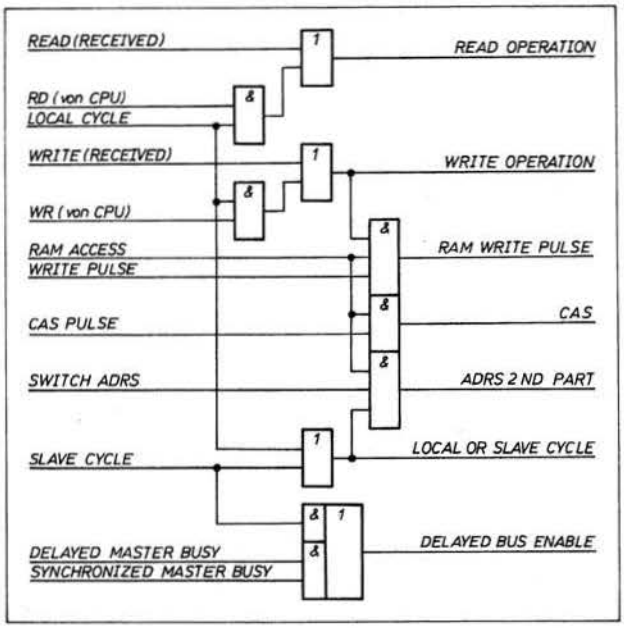


Bild 98: Bildung von Steuersignalen für Speicher- und Buszugriffe

Refreshzyklen veranlaßt END über ein Ausschalt-Flip-Flop (s. Bild 93) das Rücksetzen des Zyklusregisters.

Refreshzyklen haben folgende Besonderheiten gegenüber den anderen RAM-Zugriffen:

- Es werden beide RAS-Signale erregt; CAS wird nicht erregt (RAS-Only Refresh).
- Es erfolgt keine Adressenumschaltung.
- Der Datenbus wird nicht benutzt.

Damit können Zugriffe, die den RAM nicht benötigen, und Refreshzugriffe gleichzeitig ablaufen (beispielsweise mit Master-, E-A- und ROM-Zugriffen).
Bild 99 veranschaulicht den Ablauf eines RAM-Zugriffs. Dieser Ablauf gilt sinngemäß sowohl für lokale als auch für Slavezugriffe.
Registerzugriffe werden in grundsätzlich gleicher Weise wie RAM-Zugriffe ausgeführt. Bild 97 zeigte die Dekodierung der höchstwertigen Adressenbits. In Abhängigkeit vom Adressenbit 2 wird bei einem Registerzugriff ein simultaner RAM-Zugriff ausgeführt oder nicht. Die Bildung der Steuersignale für die Register ist im Bild 100 dargestellt. Es wird nur ein Dekoder benutzt, dessen Ausgangssignale je nach Zugriff entweder Schreibimpulse oder Leseauswahlsignale sind. Letztere werden nur erzeugt (vom Einschaltzeitpunkt des Steuersignals DELAY 3 bis zum Ende des Zyklus), wenn eines der Nur-Leseregister adressiert ist. Bei Lesezugriffen zu Schreibregistern wird die Kopie des Registerinhaltes aus dem RAM gelesen. Bild 91 zeigt, daß dabei die höherwertigen vier Bits der 7-bit-RAM-Adresse durch Abschalten der betreffenden Multiplexer auf 0 gehalten werden. Diese einfache Schaltungslösung führt zu den bereits erwähnten krummen Registeradressen.

Besonderheiten:

- ACCESS ERROR (Bild 97) signalisiert ein fehlerhaftes Arbeiten der Logik (Adressen für Masterzugriffe führen fälschlicherweise zur Erregung der Dekoderschaltung, bzw. es wird ein Slavezugriff mit einer Adresse im Bereich der Masterzugriffe des Mikrorechners ausgeführt).

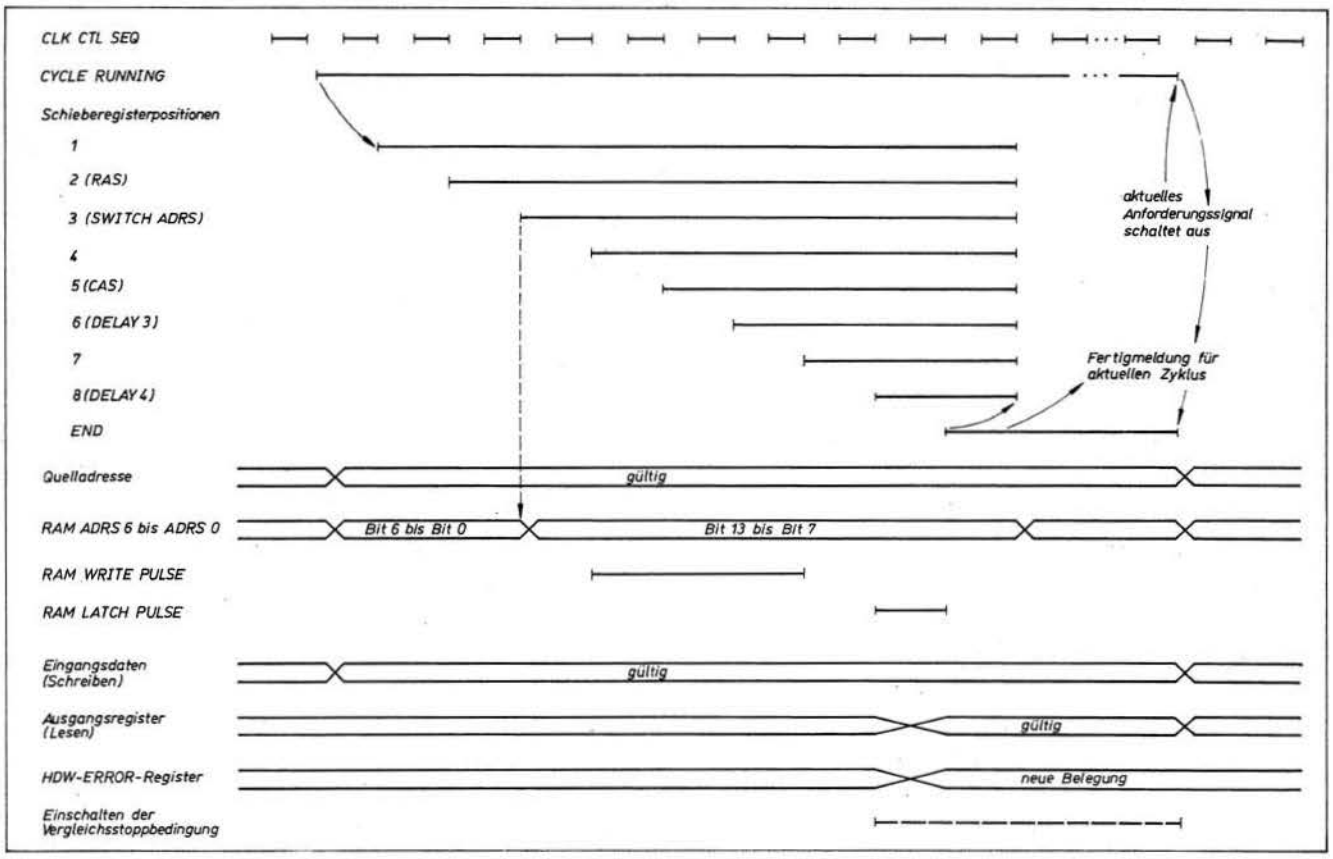


Bild 99: Allgemeiner Ablauf eines Speicherzugriffs

- Das Abschalten der Dekoder mit INTERRUPT (RECEIVED) bei Slavezugriffen ist erforderlich, um den Konventionen des Bussystems gerecht zu werden (die niederen 16 bit der Busadresse können bei Interruptauslösung undefiniert sein).
- Das Flip-Flop nach Bild 100 verhindert, daß Spikes auf der Leitung REGISTER ACCESS inkorrekte Registerschreibimpulse hervorrufen. Derartige Spikes können durch eine Wettlauferscheinung am Dekoder nach Bild 97 entstehen, wenn LOCAL OR SLAVE CYCLE den Dekoder schneller zuschaltet, als dessen Adresseneingänge umgesteuert sind.

3.3.4.2. Slavezugriffe

Bild 101 zeigt im Detail, daß eine Slaveanforderung entsteht, indem durch das BUSY-Signal ein Vergleich der vier höchstwertigen Adressenbits des Bussystems mit einer Slave-Adresse ausgewertet wird, die durch Schalter einstellbar ist. Das Anforderungssignal wird zunächst von einem Flip-Flop geliefert. Damit es im Anforderungsregister wirksam werden kann, muß es mindestens für einen Zyklus

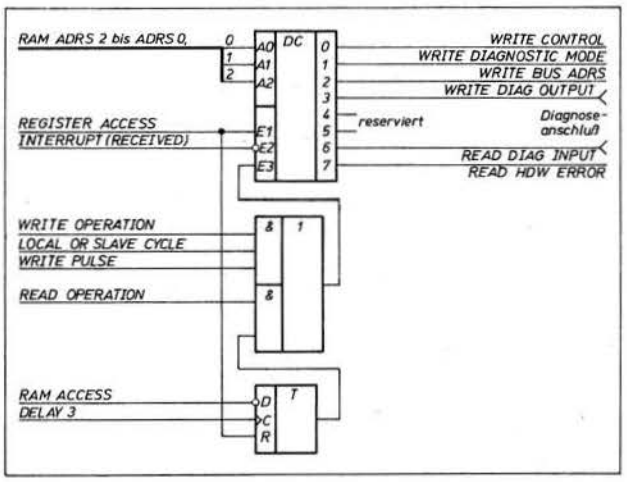


Bild 100: Dekodierung der Registerzugriffe

von CLK RQ LATCH ununterbrochen aktiv sein. Eventuelle Störimpulse, beispielsweise am DO-Ausgang des Koppelschaltkreises für BUSY (s. a. Bild 34), sind somit wirkungslos.

Nach dem Vermittlungsprozeß kennzeichnet SLAVE CYCLE die Ausführung des Zugriffs. Dazu illustriert Bild 102 die Anschaltung der Steuersignale an das Bussystem. Aus Bild 90 ist der Datenfluß ersichtlich:

CPU DATA sind entkoppelt, da sich sowohl LOCAL CYCLE und SLAVE CYCLE gegenseitig ausschließen (auf Grund der internen Vermittlung im Mikrorechner) als auch MASTER BUSY und SLAVE CYCLE (auf Grund der Mastervermittlung des Bussystems).

Die Datenleitungen des Bussystems sind angeschaltet, wobei die Richtung vom empfangenen Lesesignal bestimmt wird. Die niederen 16 Adressenbits werden gemäß Bild 91 vom Bussystem empfangen. Ein normaler Zugriff läuft dann so ab, wie im Bild 99 dargestellt. END ist die Quelle des REPLY-Signals, das dem Master die Ausführung des Buszugriffs anzeigt. Schaltet dieser BUSY aus, so fällt SLAVE REQUEST ab, wodurch der Zyklus beendet wird.

Das Bussystem ermöglicht das Abweisen von Buszyklen durch Erregen der Steuerleitung RELEASE. Der Mikrorechner benutzt diese Möglichkeit in zwei Fällen:

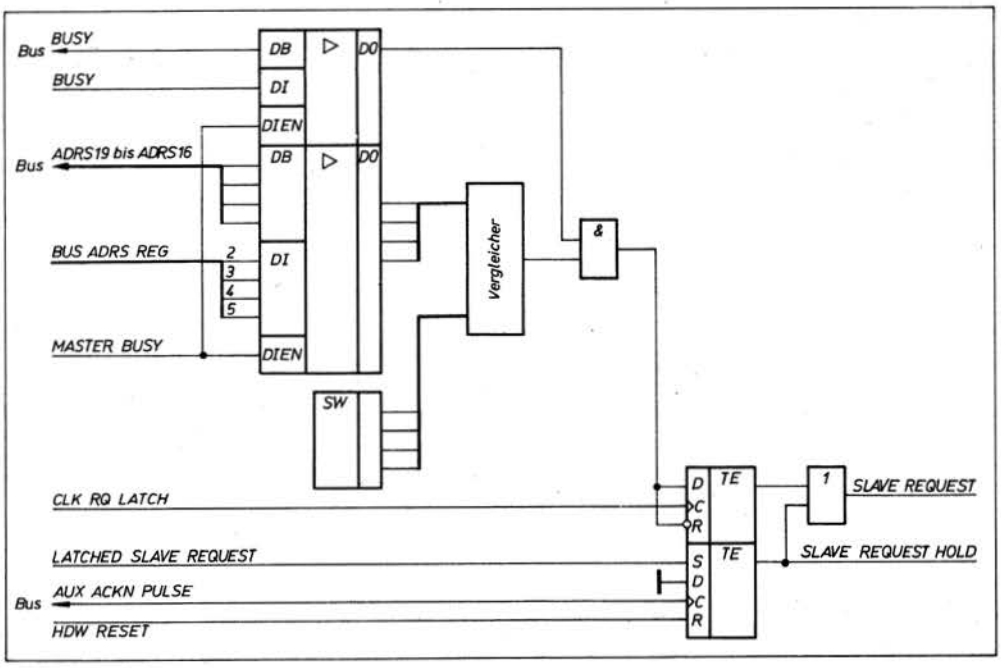
- Der Master will einen Interrupt signalisieren und die betreffende PIO ist dazu nicht bereit (das READY-Signal des betreffenden Ports ist inaktiv).
- Während des aktuellen Programmablaufes sollen externe Zugriffe ausgeschlossen werden (REJECT SLAVE ACCESS im DIAGMODE-Register ist eingeschaltet).

Bild 103 veranschaulicht die Bedingung für das Aktivieren von RELEASE zusammen mit den Schaltmitteln für die Interruptauslösung.

Bild 104 zeigt das Taktdiagramm eines Slavezyklus mit Abweisung durch RELEASE.

Bei diesem Ablauf veranlaßt das steuernde Schieberegister mit seinem zweiten Ausgangssignal das Einschalten des RELEASE-Flip-Flops (s. Bild 94). Das Schieberegister wird daraufhin zurückgesetzt. Somit werden die Steuersignale SWITCH ADRS . . . END nicht abgegeben. Es wird lediglich

Bild 101: Erkennung von Slaveanforderungen



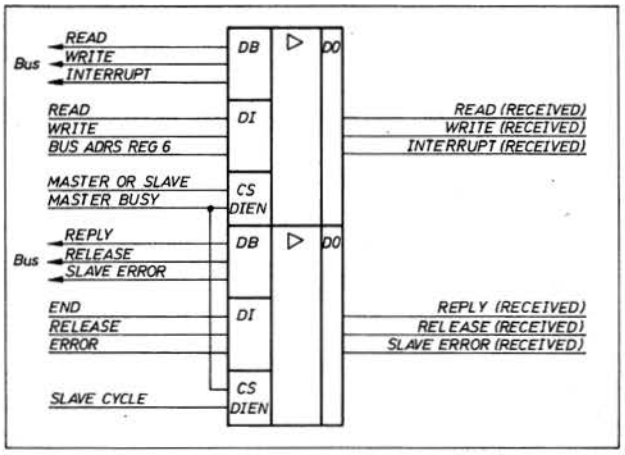


Bild 102: Anschaltung von Steuersignalen an den Systembus

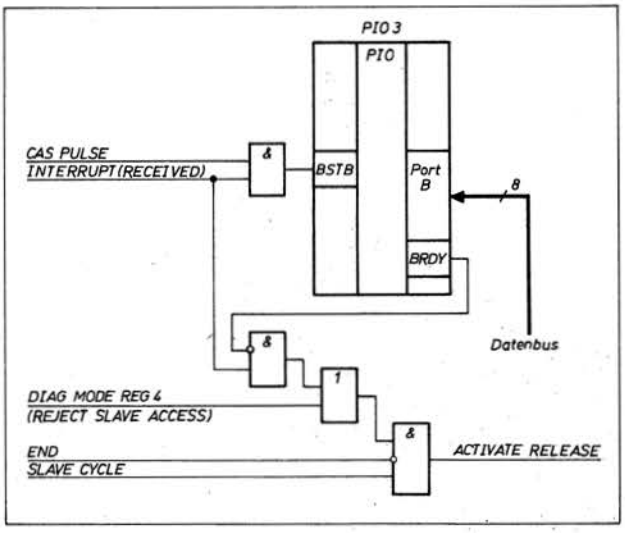


Bild 103: Empfang von Interrupts, Aktivierung von RELEASE

RAS erzeugt. Der RAS-Impuls gemäß dem zweiten Ausgangssignal des Schieberegisters wäre allein allerdings zu kurz gegenüber den Mindestforderungen für den Betrieb der RAM-Schaltkreise. Die Folge derart inkorrekter Impulse wäre ein Verfälschen von RAM-Inhalten. Um dies zu vermeiden, wird RAS durch SLAVE REQUEST HOLD (Bild 101) verlängert. Dieses Steuersignal dient weiterhin dazu, SLAVE REQUEST bis zum Ende des Buszyklus stabil zu halten (es wird unempfindlich gegen Störungen, die während des Buszyklus auf die BUSY-Leitung eingekoppelt werden könnten).

Der RAS-Impuls veranlaßt in den adressierten RAMs eine Art RAS-Only-Refresh; alle anderen Steuersignale werden durch das Rücksetzen des Schieberegisters nicht erzeugt. Diese Lösung wurde aus zeitlichen Gründen gewählt: Zu Beginn des Zyklus ist ein gewisses Zeitintervall für das Zuschalten der Buskoppelbaustufen und für die Entscheidung über die Art des Ablaufs erforderlich. Wird RAS erst nach der Entscheidung für die Ablaufmodifikation abgegeben, so bedingt dies eine längere Entscheidungszeit, so daß sich die Zugriffe verlängern würden.

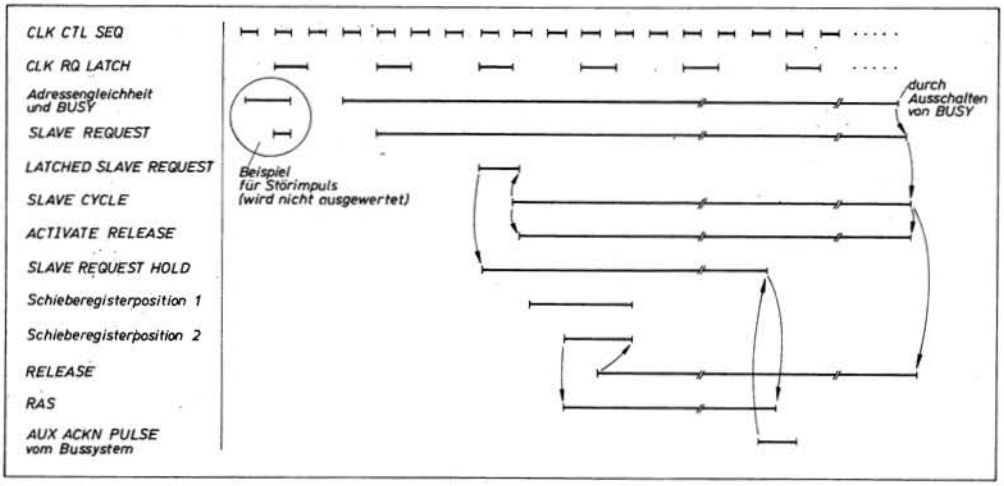


Bild 104: Ablauf eines Slavezugriffs mit RELEASE

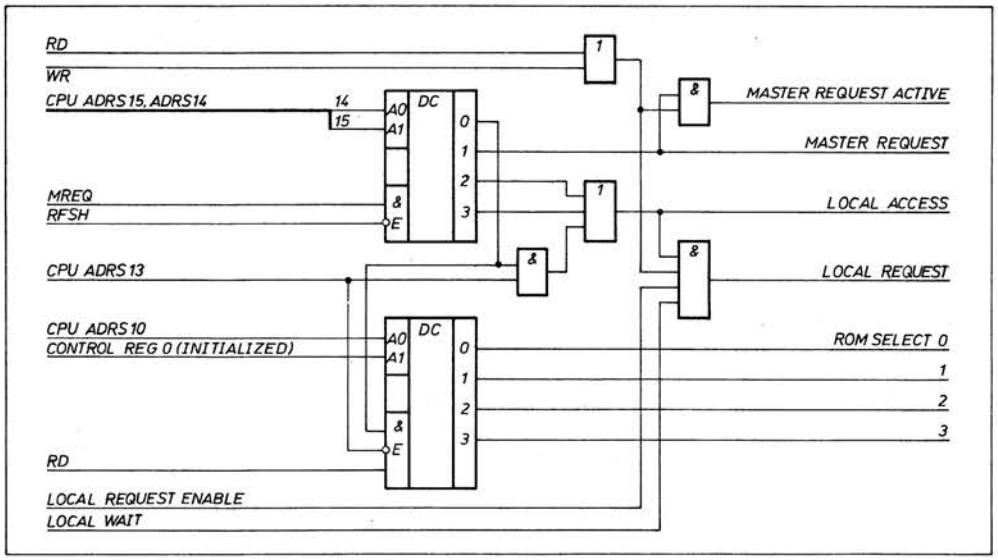


Bild 105: Dekodierung der CPU-Adressen zur Bildung von Anforderungssignalen

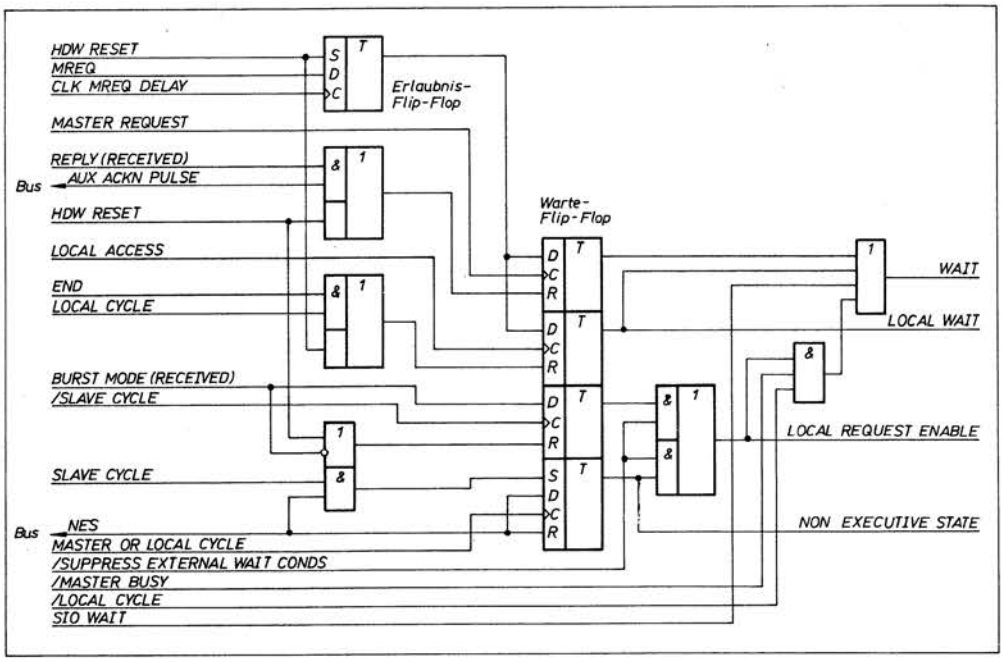


Bild 106: Bildung des Wait-Signales

3.3.4.3. Zugriffe seitens der CPU

Bild 105 zeigt die Dekodierung der CPU-Speicheradressen. Möglich sind ROM-, lokale (zum RAM und zu den Registern) und Masterzugriffe über das Bussystem.

Die ROM-Zugriffe finden im Rahmen des elementaren Mikrorechners statt, der aus der CPU, den E-A-Schaltkreisen und der ROM-Anordnung gebildet wird. Sie werden hier nicht näher beschrieben.

Für die anderen Zugriffe wird die CPU zunächst in den Wait-Zustand versetzt. Die Schaltung dazu zeigt Bild 106. Aus den beiden höchstwertigen Bits der CPU-Adresse werden die alternativen Anforderungssignale LOCAL ACCESS bzw. MASTER REQUEST gebildet, deren Vorderflanke das jeweilige Warte-Flip-Flop einschaltet. Eine derartige Flankensteuerung ist empfindlich gegen Störimpulse. Da die CPU bei M1-Zyklen gleichzeitig ihre Refresh-Adresse auf den Bus legt und MREQ (die Anforderung für das eigentliche Befehlslesen) ausschaltet, kann der Adressendekoder Störimpulse abgeben. Den Warte-Flip-Flops ist deshalb ein Erlaubnis-Flip-Flop vorgeschaltet, das verhindert, daß bei abfallendem MREQ Störimpulse die Warte-Flip-Flops einschalten.

Die eigentlichen Anforderungssignale für lokale bzw. Masterzugriffe werden erst dann gebildet, wenn die CPU das entsprechende Steuersignal (RD oder WR) aktiviert hat. Somit können diese Steuersignale direkt zur Steuerung des Datenflusses benutzt werden. Bei lokalen Zugriffen führt das Anforderungssignal LOCAL REQUEST nach der Vermittlung zu LOCAL CYCLE. Der Datenpfad wird zum RAM und zu den Registern durchgesteuert, die Richtung wird durch das Schreibsteuersignal der CPU bestimmt (Bild 90). Die CPU-Adresse gelangt direkt zu den Multiplexerschaltkreisen der RAM-Adressierung (Bild 91). Der Ablauf selbst wird durch Bild 99 illustriert. END schaltet das Warte-Flip-Flop aus (Bild 106). Daraufhin führt die CPU den aktuellen Zyklus zu Ende. Mit dem Abfall von MREQ fällt LOCAL ACCESS, wodurch der gesamte Zyklus beendet wird.

3.3.4.4. Masterzugriffe

Bild 107 veranschaulicht die Masteranforderungsschaltung. Diese entspricht mit den Flip-Flops A und B zunächst der Prinzipschaltung nach Bild 43. Es sind lediglich eine Verzögerungs- und Synchronisationsschaltung für das MASTER-BUSY-Signal (DELAYED MASTER BUSY, SYNCHRONIZED MASTER BUSY) und Steuerschaltungen für Burst-Mode-Zugriffe als Erweiterungen vorgesehen. Die verzögerten bzw. synchronisierten MASTER-BUSY-Signale wurden aus zwei Gründen eingeführt:

- Sie sollen die Koppelbaustufen des Datenpfades nacheinander aktivieren. Gemäß Bild 90 werden bei Masterzugriffen beide Koppelbaustufen aktiviert. Bei gleichzeitiger Aktivierung besteht die Gefahr, daß Treiberschaltungen gegeneinander arbeiten. Dies wird vermieden, da MASTER BUSY zunächst die CPU-Adresse zuschaltet und erst mit einem gewissen Zeitversatz (mindestens 70 ns) die Kopplung mit dem Bussystem aktiviert wird (zur Bildung des Steuersignals s. a. Bild 98).
- Es soll vermieden werden, daß bei gleichzeitigem Beginn von Masterzugriff und einem Refreshzyklus Störimpulse der DO-Ausgänge der Adressenkoppelstufen die Refreshadresse beeinträchtigen (Bild 91). Während der spezifizierten RAS-Haltezeit (etwa 25 ns) dürfen die RAM-Adressen nicht verändert werden (sonst würde sich der RAM-Inhalt undefiniert ändern). Somit muß verhindert werden, daß RAS beim Refresh-Zyklus aktiv wird und gleichzeitig die Adressen auf den Bus gegeben werden. Dies wird durch die Synchronisation mit dem Vermittlungstakt CLK RQ LATCH gewährleistet.

Soll die Übertragung im Burst Mode erfolgen, so wird zu Beginn des ersten Buszugriffs das Burst-Mode-Flip-Flop eingeschaltet. Dadurch wird die Leitung BURST MODE aktiviert. Da nun keine Mastervermittlung mehr stattfindet, kann der Mikrorechner die HOLD-SELECT-Folge nicht mehr zur Steuerung der Buskopplung benutzen.
Statt dessen veranlaßt jede Masteranforderung das Setzen eines Einschalt-Flip-Flops, das zunächst MASTER BUSY aktiviert. Dieser bewirkt über eine Verzögerungsstufe, die das notwendige Deskewing zwischen Slaveadresse und BUSY sicherstellt, das Setzen des BUSY-Flip-Flops B, das MASTER BUSY hält und das Einschalt-Flip-Flop zurücksetzt. Verschiedene Fehlersignale des Bussystems führen zum Ausschalten von BURST MODE.
Diese Schaltmittel können ausgenutzt werden, um einen Single Master Mode zu realisieren. Damit kann der Mikrorechner der einzige Master in einem System sein, ohne daß eine Bussteuerung vorhanden ist. Man benötigt lediglich eine einfache Schaltung zum Erzeugen der Signale AUX ACKN PULSE sowie ACKNOWLEDGE. Sind keine Störungen zu befürchten (etwa in sehr kleinen Konfigurationen mit extrem kurzen Busleitungen), reicht es auch aus, beide Signale ständig aktiv zu halten. Der Modus wird durch einen Schalter gewählt. Er ist vorzugsweise für diagnostische Zwecke vorgesehen (damit kann eine verdächtige Bussteuerung vom Bus entfernt werden; die elementaren Funktionen des Bussystems lassen sich von einem Mikrorechner aus prüfen). Bild 108 zeigt den Ablauf eines normalen Buszyklus und Bild 109 den Beginn eines Buszyklus im Burst Mode bzw. im Single Master Mode. Jeder Buszyklus wird durch ein Antwortsignal vom Slave beendet. Der Wartezustand der CPU wird nur durch REPLY in Zusammenhang mit AUX ACKN PULSE verlassen (Bild 106). Der Informationsfluß bei Masterzugriffen ist in den Bildern 90, 91 und 105 bis 109 zu erkennen.

3.3.4.5. Besondere Wartezustände

Nach Bild 106 gibt es noch drei zusätzliche Ursachen für Wartezustände der CPU:

- Wartezustand bei Zugriffen zur SIO, wenn diese nicht zum Datenaustausch bereit ist. Dazu muß in der SIO der Wait-Modus programmiert werden
- Wartezustand infolge der Aktivierung der Busleitung NES (Non Executive State)
- Wartezustand bei Slavezugriffen im Burst Mode.

Die letzten beiden Zustände werden durch externe Bedin-

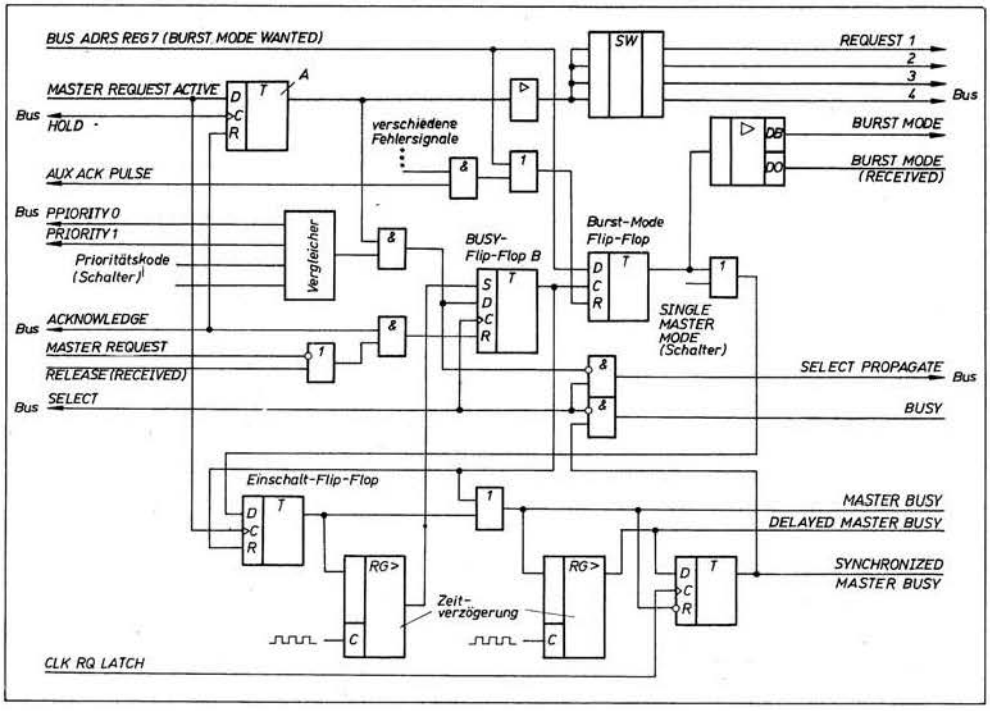


Bild 107: Steuerschaltungen für Masterzugriffe

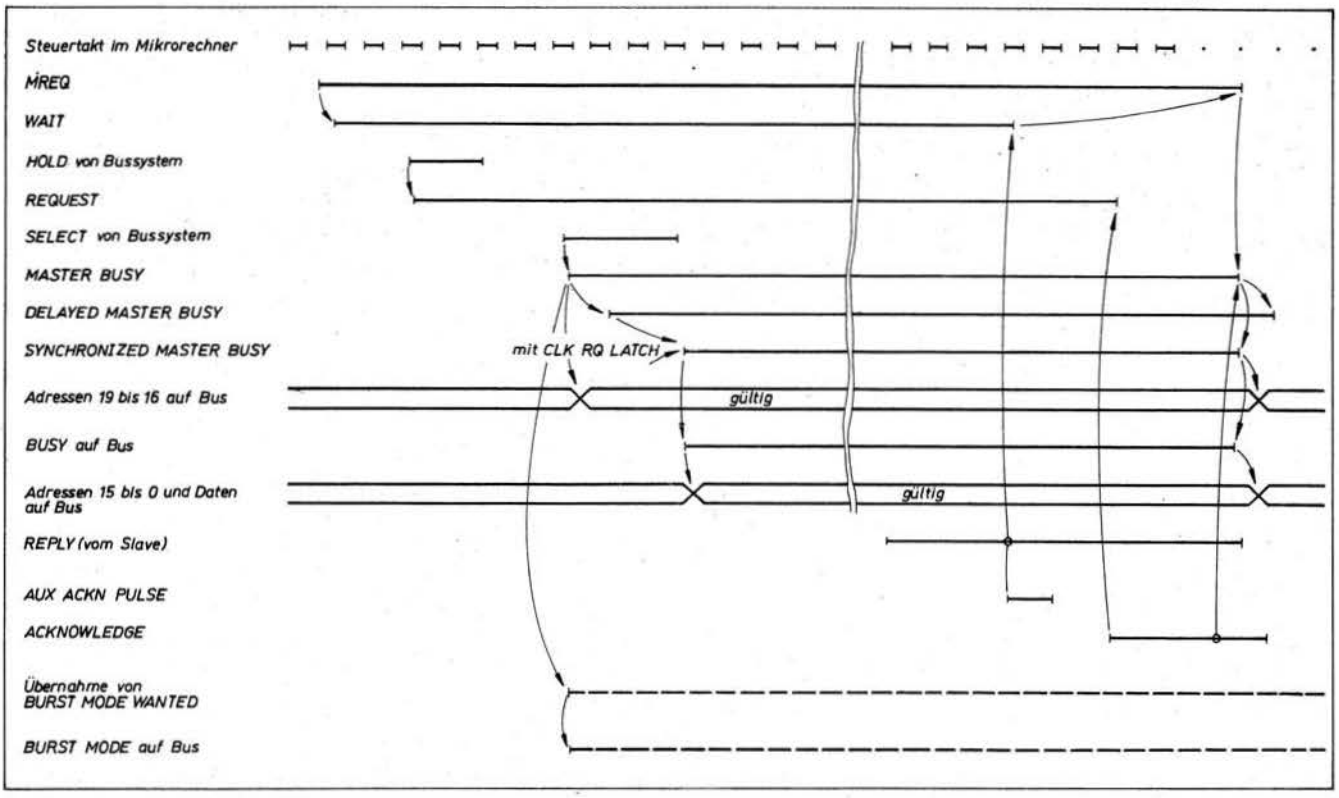


Bild 108: Ablauf eines Masterzugriffs

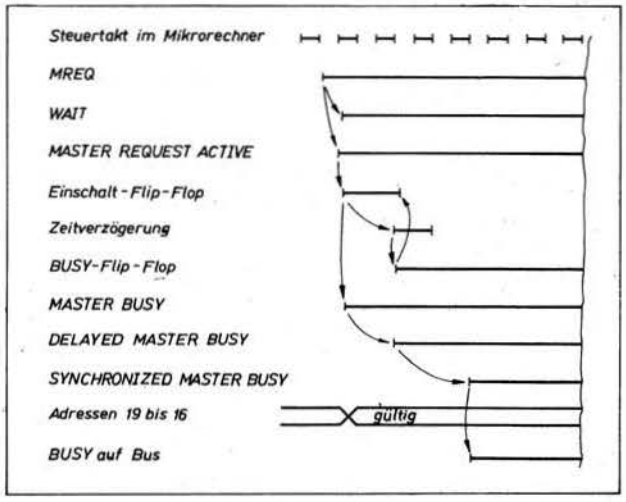


Bild 109: Beginn eines Masterzugriffs im Burst Mode bzw. im Single Master Mode

gungen veranlaßt, die zur Befehlsausführung in der CPU keine Beziehungen haben. Es ist nicht trivial, derartige externe Wartezustände einzuführen. Überlagert ein derartiger Zustand eine Wartebedingung, die durch die CPU selbst hervorgerufen wurde, so führt dies zum Hängenbleiben des jeweiligen Zyklus.
Die externen Wartezustände dürfen somit nicht wirken, solange noch Wartezustände der CPU aktiv sind.
Im besonderen muß gewährleistet sein, daß Abläufe, die zum Verlassen von Wartezuständen führen (lokale oder Masterzugriffe) nicht durch externe Wartezustände verlängert werden. Andererseits dürfen, wenn externe Wartezustände vorliegen, keine neuen Zyklen gestartet werden. Bei Masterzugriffen ist dies an sich gegeben, da sich nach den Konventionen des Bussystems Wartezustände durch NES bzw. BURST MODE und die Mastervermittlung gegenseitig ausschließen. Hingegen ist es erforderlich, bei lokalen Anforderungen das Anforderungssignal für die Vermittlungsschaltung erst dann freizugeben, wenn keine externen Wartezustände vorliegen. (Hinweis: Die CPU wertet das Wait-Signal nur zu bestimmten Taktzeitpunkten aus. Es kann also einen laufenden Zyklus lediglich verlängern; es ist aber nicht möglich, durch ein prophylaktisches Erregen des Wait-Signales das Starten von CPU-Maschinenzyklen von vornherein zu verhindern.) Gemäß Bild 106 wird deshalb der Wartezustand wegen des Burst Mode stets am Ende eines Slavezugriffes eingestellt (da zu diesem Zeitpunkt die Belegung der BURST-MODE-Leitung mit Sicherheit gültig ist). NES wird während eines Slavezugriffes bzw. am Ende von lokalen bzw. von Masterzugriffen aktiviert. Da sich Master-, Slave- und lokale Zugriffe stets gegenseitig ausschließen, ist der korrekte Übergang zwischen den einzelnen Wartezuständen stets gewährleistet.

Bei Benutzung der SIO im Wait-Modus müssen externe Wartezustände softwareseitig durch Setzen von SUPPRESS EXTERNAL WAIT CONDS im DIAG-MODE-Register unterdrückt werden. Damit ist gleichzeitig gesichert, daß bei der Steuerung der Datenübertragung kein Datenverlust auftritt.

3.3.4.6. Fehlerkontrollen

Der Mikrorechner enthält einige Kontrollschaltungen zur Überwachung kritischer Hardwarekomplexe bzw. zur Unterstützung bei der Suche von Softwarefehlern. Es werden folgende Fehlersignale erzeugt:

- ARBITRATION CHECK (Bild 110)

Es wird ein Fehlersignal abgegeben, wenn die Vermittlungsschaltung (Bild 93) gleichzeitig mehr als einen Zyklus vermittelt (z. B., wenn gleichzeitig REFRESH CYCLE und SLAVE CYCLE eingeschaltet sind). Derartige Fehler sind durch Testabläufe kaum zu finden, da es keine einfachen Schaltmittel gibt, um gezielt das Zusammentreffen verschiedener Anforderungen auszulösen. Die Überwachung ist somit die kostengünstigere Alternative zu Schaltungsmaßnahmen, die eine diagnostische Einspeisung konkurrierender Anforderungssignale ermöglichen.

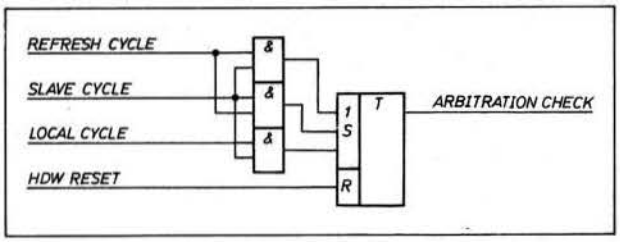
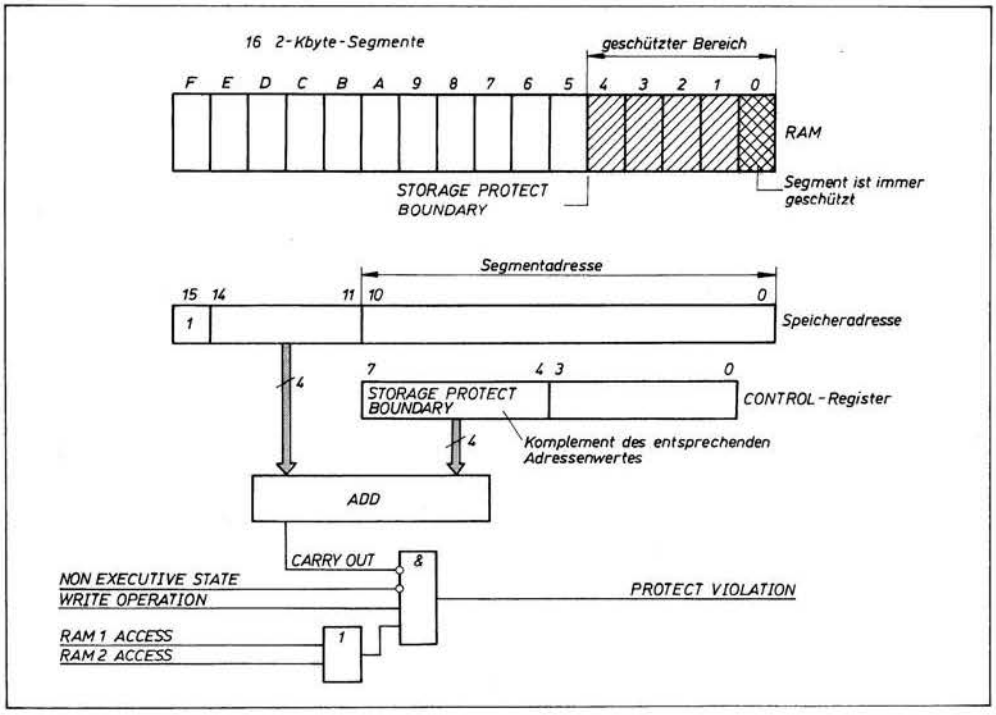


Bild 110: Erkennung des Vermittlungsfehlers

Bild 111: Prinzip des Speicherschutzes



● PROTECT VIOLATION (Bild 111)

Der RAM wird in aufeinanderfolgenden Abschnitten von 2 Kbyte auf Schreibzugriffe hin überwacht. Diese Überwachung ist während NES unwirksam, so daß die Bussteuerung zu Zwecken der Testung, Initialisierung usw. auch geschützte Bereiche beschreiben kann.

● ILLEGAL ACCESS (Bild 112)

Dieses Fehlersignal wird erzeugt, wenn die internen Dekodierschaltungen (Bild 97) feststellen, daß mit einer Adresse im Bereich 4000H bis 7FFFH ein Zugriff versucht wird. Ursache dafür ist entweder ein Defekt in der Hardware des Mikrorechners oder ein Slavezugriff mit inkorrekter Adresse. Es tritt auch auf, wenn außerhalb von NES versucht wird, Register durch Slavezugriffe zu beschreiben. Da die Registerinhalte von entscheidender Bedeutung für die Funktionsfähigkeit des Mikrorechners sind, dürfen sie nicht durch beliebige Programme geändert werden. Externe Zugriffe sind nur vorgesehen, damit die Bussteuerung Initialisierungs- und Wartungsabläufe ausführen kann.

● REFRESH TIMEOUT CHECK (s. Bild 96)

Dieses Fehlersignal wird erzeugt, wenn der Refreshadressenzähler die nächste Adresse eingestellt hat, aber die vorhergehende Anforderung noch nicht ausgeführt wurde (bzw. wenn eine Refreshanforderung nicht innerhalb von etwa 12 μs bearbeitet wird).

● PARITY CHECK (s. Bild 89)

Dies ist das Paritätsfehlersignal des RAM. Es wird bei Lesezugriffen abgegeben, wenn das Erlaubnisbit im CONTROL-Register gesetzt ist und wenn das gelesene Datenbyte gerade Parität hat.

Bild 113 zeigt, wie die Fehlersignale an das HDW-ERROR-Register angeschlossen und zur Bildung des Busfehlersignales ERROR zusammengefaßt sind. Bei Slavezugriffen führt eine Fehlerbedingung zusätzlich zum Einschalten von SLAVE ERROR.

Die Fehlerbehandlung obliegt nicht dem Mikrorechner selbst, sondern einer anderen Einrichtung des Systems. Sie wird stets von der Bussteuerung eingeleitet. Die erste Reaktion auf die ERROR-Signalisierung ist das Einschalten von NES. Damit gelangen die Funktionseinheiten des Systems in den Non Executive State, so daß von der Bussteuerung aus eine genauere Analyse der Fehlersituation möglich ist.

Die Fehlersignalisierung kann durch eine Bitposition des CONTROL-Registers zugelassen bzw. verhindert werden. Eine Ausnahme dabei bildet der REFRESH TIMEOUT CHECK. Da beim Ausfall von Refresh-Zyklen mit undefinierten Datenverlusten im RAM gerechnet werden muß, wird diese Fehlerbedingung stets signalisiert.

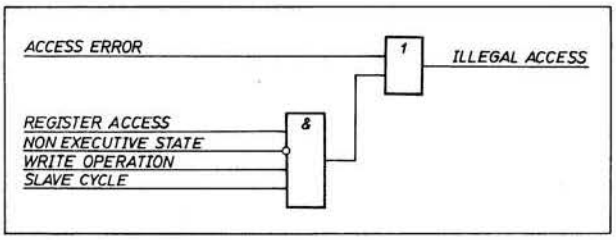


Bild 112: Erkennung des Zugriffsfehlers

3.3.4.7. Adressenvergleichsstopp

Die neunte Bitposition des RAM kann alternativ zur Paritätskontrolle zu Vergleichsstoppzwecken benutzt werden (Bild 89). Die Stoppbedingung führt bei lokalen Zugriffen zu einem NMI und bei Slavezugriffen zur Erregung der Busleitung COMPARE STOP (in diesem Fall wird die Bedingung von der Bussteuerung behandelt). Bild 114 zeigt die entsprechenden Schaltmittel. Die Betriebsart Vergleichsstopp wird durch folgenden Algorithmus eingeschaltet:

- ENABLE PARITY (CONTROL-Register) ausschalten
- ENABLE INJECT einschalten; INJECT HIGH/FORCE EVEN PARITY ausschalten (DIAG-MODE-Register)
- alle Speicherplätze des RAM nacheinander lesen und wieder beschreiben. Dabei werden in die neunte Bitposition Nullen eingetragen
- INJECT HIGH/FORCE EVEN PARITY einschalten
- alle Speicherplätze, für die ein Vergleichsstopp gewünscht ist, lesen und wieder beschreiben. Dabei werden in die korrespondierenden neunten Bitpositionen Einsen eingetragen
- ENABLE INJECT ausschalten
- ENABLE COMPARE STOP einschalten
- nach Wunsch STOP ON READ oder STOP ON WRITE oder beides einschalten.

Für das Einschalten der Paritätsprüfung ist folgender Algorithmus vorgesehen:

- ENABLE COMPARE STOP, ENABLE ERROR SIGNALIZATION, INJECT HIGH/FORCE EVEN PARITY löschen
- ENABLE PARITY einschalten
- alle Speicherplätze nacheinander lesen und wieder beschreiben. Dabei wird in die neunte Bitposition das Paritätsbit eingetragen
- ENABLE ERROR SIGNALIZATION einschalten.

Das Umstellen der Betriebsart kann sowohl durch den Mikrorechner selbst als auch von der Bussteuerung aus durchgeführt werden.

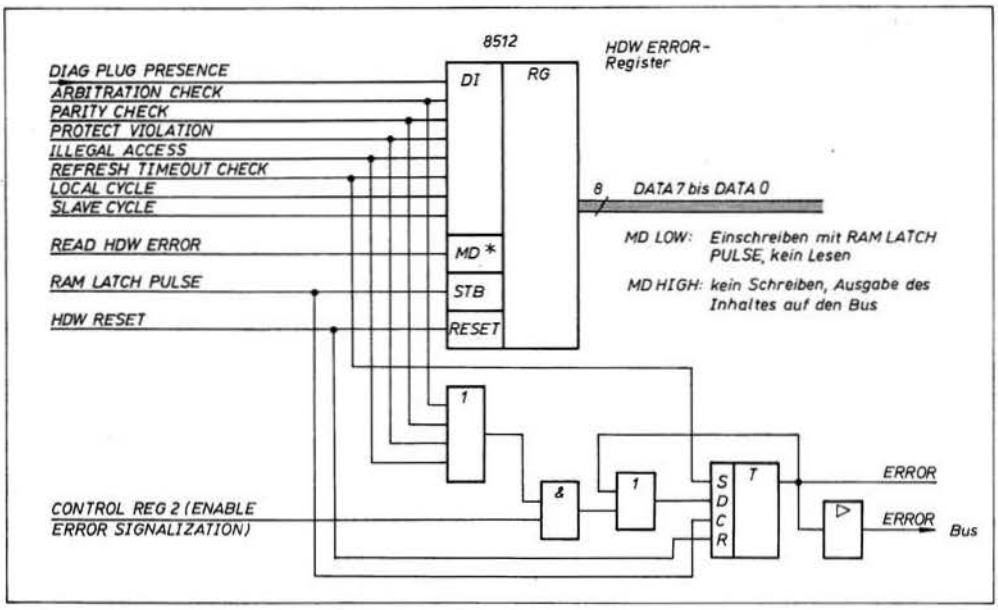


Bild 113: Fehlerregister und Beschaltung der Fehlerleitung des Bussystems

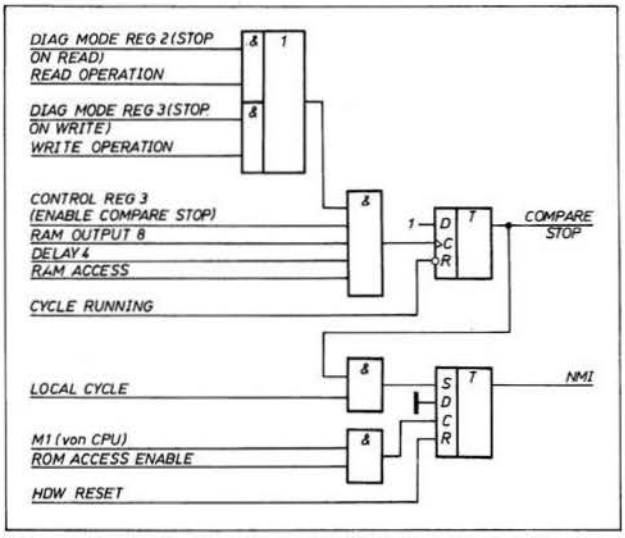


Bild 114: Erkennung des Adressenvergleichsstopps und Bildung des NMI-Signales

Die Behandlung der Stoppbedingung obliegt vollständig der Software. Durch entsprechende Einstellungen ist ein befehlsweiser Betrieb, das Registrieren von Schreibzugriffen auf bestimmte Speicherplätze, das Verfolgen kritischer Abläufe in Programmen ohne Geschwindigkeitsverlust für die weniger interessanten Abläufe usw. möglich.

3.3.4.8. Rücksetzen

Bild 115 zeigt die Erzeugung der Rücksetzsignale.
Ein Rücksetzen erfolgt durch die Erregung der RESET-Busleitung, ein Rücksetzsignal vom Diagnoseanschluß und selektiv. Beim selektiven Rücksetzen wird das CONTROL-Register nicht mit zurückgesetzt, so daß die Belegung des INITIALIZED-Bits erhalten bleibt. Dementsprechend löst das selektive Rücksetzen bei einem initialisierten Mikrorechner einen Programmstart von Adresse 0 des zweiten ROM-Segmentes aus.
Normalerweise läuft das selektive Rücksetzen so ab, daß die Bussteuerung Parameterbytes in definierte Speicherplätze des Mikrorechners überträgt. Darunter ist in der Regel die Startadresse des Programms, dessen Ausführung gewünscht wird. Die Bussteuerung startet dann einen Slavezugriff zum Mikrorechner, wobei weder READ noch WRITE aktiv sind, aber INTERRUPT eingeschaltet ist, und aktiviert die Leitung SELECTIVE RESET (s. a. Bild 41). Der Mikrorechner startet die Programmausführung, analysiert die Parameterbytes und verzweigt zur gewünschten Funktion.
Der Slavezugriff wird deshalb gestartet, um eine definierte Ansteuerung des RAM zu gewährleisten. Wird das Hardwarerücksetzen sofort zu Beginn der Slaveanforderung wirksam, so kann dies ein Verfälschen von RAM-Inhalten zur Folge haben, da der aktuelle Zyklus abrupt abgebrochen wird.
Dieser Effekt wird vermieden, wenn das Rücksetzen erst dann wirkt, wenn der Slavezyklus bereits vermittelt ist.
Sollte diese Vermittlung auf Grund eines Hardwarefehlers nicht gelingen, so veranlaßt die Bussteuerung eine entsprechende Fehlerreaktion. Ob die Vermittlung erfolgreich ausgeführt wurde oder nicht, kann die Bussteuerung durch Analyse der Antwortsignale erkennen. Innerhalb einer gewissen Zeit muß REPLY bzw. RELEASE aktiviert werden. Die Erregung der INTERRUPT-Leitung ist deshalb wesentlich, weil beim selektiven Rücksetzen die niederen 16 Bits der Busadresse nicht definiert sind, so daß die Adressendekoder (Bilder 97, 100) abgeschaltet werden müssen.

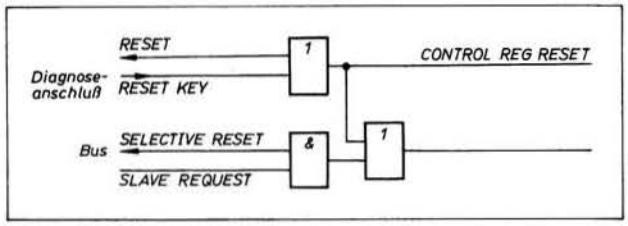


Bild 115: Bildung der Rücksetzsignale

3.3.5. Diskussion

3.3.5.1. Auslegung des Mikrorechners

Der Mikrorechner ist sowohl als Verarbeitungsrechner als auch zur Steuerung von E-A-Einrichtungen einsetzbar. Diese universelle Auslegung wurde gewählt, um zum einen die Entwicklungskosten für einen weiteren Mikrorechnertyp zu sparen, und zum anderen, um die Anzahl der Mikrorechner im System gering halten zu können.
In vielen Anwendungsfällen ist es ohne weiteres vertretbar, Verarbeitungsaufgaben und Aufgaben der E-A-Steuerung über LSI-Schaltkreise zeitmultiplex von jeweils einem Mikrorechner ausführen zu lassen, ohne daß Nachteile in Hinsicht auf das Echtzeitverhalten entstehen. Im besonderen trifft dies in zwei typischen Fällen zu:

- Die zeitliche Belastung durch die E-A-Steuerung ist so gering, daß sie sich nicht störend auswirkt.
- Die E-A-Steuerung lastet den Mikrorechner voll aus, die Verarbeitungsprozesse im System müssen aber funktionsbedingt auf das Ende des E-A-Prozesses warten.

Beispiel

Für ein System sind drei Verarbeitungsmikrorechner erforderlich. Die wesentlichen peripheren Einrichtungen sind über spezielle Steuereinheiten an den Systembus angeschlossen. Zum Betrieb über LSI-Schaltkreise sind ein Seriendrucker (50 Zeichen/s) und Floppy-Disk-Laufwerke vorgesehen. Bei einer strikten Trennung zwischen Verarbeitung- und E-A-Mikrorechnern wäre ein vierter Mikrorechner für die Steuerung der E-A-Geräte vorzusehen. Für die Steuerung des Druckers ist die Auslastung kaum nennenswert. Die Steuerung der Datenübertragung mit einem Floppy-Disk-Laufwerk lastet den E-A-Mikrorechner zwar voll aus, der Verarbeitungsmikrorechner, der den Auftrag erteilt hat, muß jedoch bis zur Erledigung des Auftrages warten. In vielen Anwendungsfällen gibt es zudem kaum lauffähige Prozesse, die in den Wartezeiten bearbeitet werden könnten, abgesehen von der Kompliziertheit und dem Laufzeitbedarf entsprechender Betriebssoftware.
Bei universeller Auslegung der Mikrorechner ist ein spezieller E-A-Mikrorechner nicht erforderlich. Der Mikrorechner verfügt über eine Vielzahl von Sonderfunktionen, von diagnostischen Vorkehrungen usw. Dies bedingt einen relativ hohen Einsatz an Schaltmitteln, gestattet es jedoch, komplexe Formen der Softwareorganisation zu implementieren und auch dadurch die Anzahl der Mikrorechner im System gering zu halten.

Für Operationen der Art Test and Set z. B. gibt es zwei unabhängige Möglichkeiten:

- Blockierung anderweitiger Zugriffe in anderen Funktionseinheiten durch Buszugriffe im Burst Mode
- Blockierung anderweitiger Zugriffe zum Speicher des Mikrorechners selbst durch das Abweisen von Buszugriffen (REJECT SLAVE ACCESS).

Damit können Speicherbereiche zur Steuerung des Multiprogramming, zur Verwaltung von Ressourcen usw. sowohl im lokalen RAM als auch in Speichern anderer Funktionseinheiten untergebracht sein. Jeder Mikrorechner kann über ein eigenes Betriebssystem verfügen. Wären beide Möglichkeiten nicht vorgesehen, so ließe sich kein Betriebssystem auf dem Mikrorechner implementieren. Wäre nur eine vorgesehen, so bestünden Einschränkungen bei der Wahl der entsprechenden Speicherbereiche. Gäbe es etwa nur den Burst Mode, so müßten alle entsprechenden Speicherbereiche in einer Funktionseinheit angeordnet werden, zu der von allen anderen Einrichtungen Slavezugriffe möglich sind, die aber selbst diese Zugriffe nicht blockieren kann.
Die Realisierung des Mikrorechners erfordert insgesamt etwa 140 Schaltkreise, die meisten sind übliche TTL-Typen niederen und mittleren Integrationsgrades. Die Sondervorkehrungen haben daran ungefähr folgende Anteile:

Kombination Paritätsprüfung, Adressenvergleichsstop	10
sonstige Fehlererkennungsschaltungen	8
externe Wartezustände	6
Burst-Mode-Ansteuerung	5

Das sind etwa 20 % des Gesamtaufwandes. Bei Wegfall sämtlicher Sonderfunktionen ließe sich der Aufwand durch weitere lokale Optimierungen (z. B. könnten einige Hardwareregister entfallen, das Mitführen von RAM-Kopien der Registerinhalte wäre überflüssig) auf die Größenordnung von 100 IS bringen. Dies entspricht ähnlich ausgestatteten Mikrorechnern des internationalen Marktes.

Eine derartige Lösung hat aber folgende Konsequenzen:

- Es sind in der Regel mehr Mikrorechner erforderlich. Namentlich bei kleineren Systemen kann der Mehraufwand im einzelnen Mikrorechner zur Einsparung ganzer Funktionseinheiten beitragen, da sich komplexere Formen der Softwareorganisation realisieren lassen.
- Für Testung, Wartung usw. müssen aufwendige Hilfsmittel (wie Logikanalysator, Entwicklungssysteme mit In-Circuit-Emulation) von vornherein bereitgestellt werden.

  Im anderen Fall sind derartige Hilfsmittel nur in extrem schwierigen Fällen der Fehlersuche erforderlich; für den überwiegenden Teil der Software- und Hardwareprobleme reichen die eingebauten Maßnahmen völlig aus.

3.3.5.2. Logische Organisation

Die im ersten Abschnitt diskutierten Maßnahmen zur Beeinflussung der CPU werden zu folgenden Zwecken benutzt:

- Rücksetzen: Initialisierung, Fehlerbehandlung, Auslösung von Testabläufen usw.
- NMI: Auswertung der Vergleichsstoppbedingung
- WAIT: Anpassung der CPU-Zyklen an den Zeitablauf des jeweiligen Zugriffes bzw. zum Anhalten der CPU (externe Wartezustände)
- Interrupt: softwareseitige Kommunikation zwischen den Funktionseinheiten (etwa zur Signalisierung von Aufträgen).

Die Anordnung aus CPU, E-A-Schaltkreisen und ROM bildet einen elementaren Mikrorechner, der in sich funktionsfähig ist. Die Interruptbehandlung erfolgt vollständig im Rahmen dieser Schaltmittel, gleichgültig, ob Interrupts von angeschlossenen E-A-Schaltungen oder über das Bussystem geliefert werden. Damit werden das Leistungsvermögen und die Flexibilität der E-A-Schaltkreise voll nutzbar, so daß es möglich wird, auch relativ komplexe und schnelle Einrichtungen (etwa Floppy-Disk-Laufwerke) zu betreiben. Werden derartige Abläufe mit ROM-residenten Programmen gesteuert, können andere Funktionseinheiten praktisch konfliktfrei zum RAM zugreifen.
Die elementare Mikrorechneranordnung bietet zudem die Basis für einen aufbauenden Test der weiteren Schaltmittel (mit ROM-residenten Testroutinen).

Für Masterzugriffe über das Bussystem kann ein Segment von 16 Kbyte aus dem 1 Mbyte großen Adressenraum des Bussystems ausgewählt werden. Für diese Lösung gibt es folgende Gründe:

- Einfachheit und minimaler Hardwareaufwand
- Prinzipien der Softwareorganisation
  Bei typischen Anwendungen enthält das System zwar viele Programme und Datenbereiche, die Programme sowie die Mehrzahl der Datenbereiche sind jedoch für sich gesehen relativ klein (einige hundert Bytes bis etwa 4 Kbyte). Damit kann die Verarbeitung selbst größtenteils im lokalen RAM erfolgen, Buszugriffe dienen zumeist zum Austauschen von Steuerinformation oder zum Transportieren von Datenblöcken bzw. zum Laden von Programmen
- Leistungsfähigkeit der Hardware
  Das Abarbeiten von Befehlen über das Bussystem (die Programme befinden sich in anderen Einrichtungen, und die CPU führt Befehlslesezugriffe als Masterzugriffe aus) ist zwar im Prinzip möglich, muß aber die Ausnahme bleiben, da sich sonst die Verarbeitungsleistung der einzelnen Mikrorechner drastisch vermindern würde
- Erstellung der Software
  Die üblichen Hilfsmittel zur Saftwareerstellung unterstützen nur den 64-Kbyte-Adressenraum der CPU. Alle Zugriffe zu einem erweiterten Adressenraum müssen explizit programmiert werden. Im Interesse einer hohen Produktivität der Programmentwicklung muß gewährleistet sein, daß der weitaus größte Teil des Objektkodes in der Form, wie er vom Entwicklungssystem erzeugt wird, unmittelbar ausführbar ist.

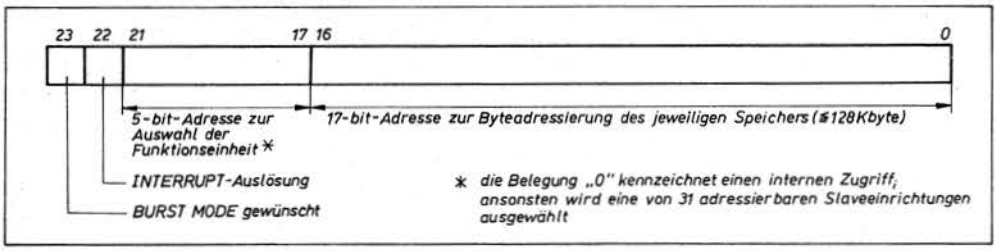


Bild 116: Beispiel für die Belegung einer 24-bit-Adresse zur Auslösung von Sonderfunktionen

Diese Kriterien erzwingen eine einfache und übersichtliche Lösung für die Erweiterung des Adressenraumes der CPU. Dabei muß der größte Teil des Adressenraumes lokal zugänglich sein, ohne softwareseitige Aufwendungen zu erfordern.
Eine gewisse Verlangsamung seltener Abläufe muß in diesem Zusammenhang toleriert werden (ein Beispiel dafür wäre die Steuerung eines Datentransportes von einer Einrichtung A zu einer Einrichtung B, die beide von der CPU als Slave adressiert werden).

3.3.5.3. Technische Ausgestaltung

Der Mikrorechner ist vollständig aus standardisierten IS aufgebaut. Für die Auslegung sind folgende Gesichtspunkte maßgebend:

- völliger Verzicht auf asynchrone Schaltmittel (wie Laufzeitketten, monostabile Multivibratoren usw.)
  Dies ist für die Prüfbarkeit der Schaltung wesentlich
- Effizienz
  Dem Prinzip nach parallele Abläufe sollen auch tatsächlich parallel ausgeführt werden können
- minimaler Aufwand.

Die einzelnen Schaltungskomplexe sind in ihrem prinzipiellen Aufbau jeweils einfach und übersichtlich gehalten. Maßnahmen gegen Störimpulse, Wettlauferscheinungen usw. sind jeweils an Ort und Stelle vorgesehen. Dadurch geht zwar die Übersichtlichkeit an verschiedenen Stellen wieder verloren, im Gegensatz zu einer generellen Berücksichtigung störender Nebeneffekte tritt aber keine Leistungsminderung ein (man vergleiche etwa die Erläuterung zum RAS-Impuls beim Senden von RELEASE). Für typische Fehlersituationen, die einer Überprüfung mit Testroutinen nur schwer oder gar nicht zugänglich sind, wurden Kontrollschaltungen vorgesehen. Die Fehlerbehandlung erfolgt prinzipiell durch eine andere Einrichtung (in der Regel durch die Bussteuerung) und nicht durch den fehlerverdächtigen Mikrorechner selbst. Sowohl die Fehlersignalisierung als auch das Anhalten des Mikrorechners als Reaktion auf einen Fehler (über die Busleitung NES) kann programmtechnisch unterdrückt bzw. zugelassen werden.
Damit kann bei komplexen Verarbeitungsaufgaben die Ausbreitung von Fehlern verhindert werden. Ebenso ist es möglich, bei bestimmten E-A-Steuerungsaufgaben die Fehlerbehandlung bis zum Ende der Informationsübertragung zu verschieben, namentlich dann, wenn eine Unterbrechung des Datenstromes nicht möglich ist, etwa bei Ausgaben zu einem Seriendrucker, dessen Druckkopf sich in kontinuierlicher Bewegung befindet.

3.4. Perspektivische Entwicklungen

Das Gebiet der Mikrorechner ist durch eine außergewöhnlich hohe Innovationsrate gekennzeichnet. Sie betrifft sowohl die Mikroprozessorschaltkreise als auch die Systemstrukturen. Dazu sollen einige Aspekte diskutiert werden.

3.4.1. Leistungsfähigere Mikrorechner

Mit dem Aufkommen leistungsfähigerer Mikrorechner geht die Bedeutung der kleineren Multimikrorechnersysteme zurück: Aufgaben, die bisher auf mehrere Mikrorechner verteilt werden mußten, können von einem einzelnen Mikrorechner zeitmultiplex bearbeitet werden. Eine solche Lösung ist wirtschaftlicher, da die Aufwendungen für die Multimikrorechnerkopplung entfallen.
In diesem Zusammenhang sind jene Mikroprozessoren besonders vorteilhaft einsetzbar, bei denen die entscheidenden Funktionen eines Echtzeitbetriebssystems in der Hardware des CPU-Schaltkreises realisiert sind.

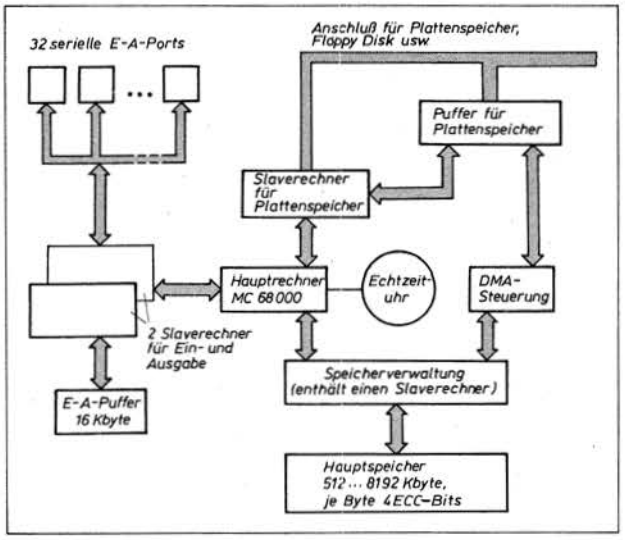


Bild 117: Beispiel für ein System aus einem Hauptrechner und mehreren Slaverechnern

3.4.2. Verbesserungen der Mikroprozessoren

Verbesserungen wurden und werden bei der Erhöhung der Verarbeitungsgeschwindigkeit, der Leistungsfähigkeit der Befehlssätze, der Datenformate und des Adressenraumes erreicht. Durch bloße Geschwindigkeitserhöhung können neue Mikrorechnerbaugruppen geschaffen werden, die bei Kompatibilität der Software ein höheres Leistungsvermögen haben. Die Befehlssätze der Mikroprozessoren, die in den 70er Jahren eingeführt wurden, sind allerdings oft recht eingeschränkt und bedingen eine umständliche Programmierung. Ein Beispiel ist das Fehlen von Bitabfrage- und Manipulationsbefehlen bei den älteren 8-bit-Mikroprozessoren.
Modernere Befehlssätze gestatten es, komplexere Algorithmen zu implementieren, und erhöhen die Produktivität der Software-Entwicklung.

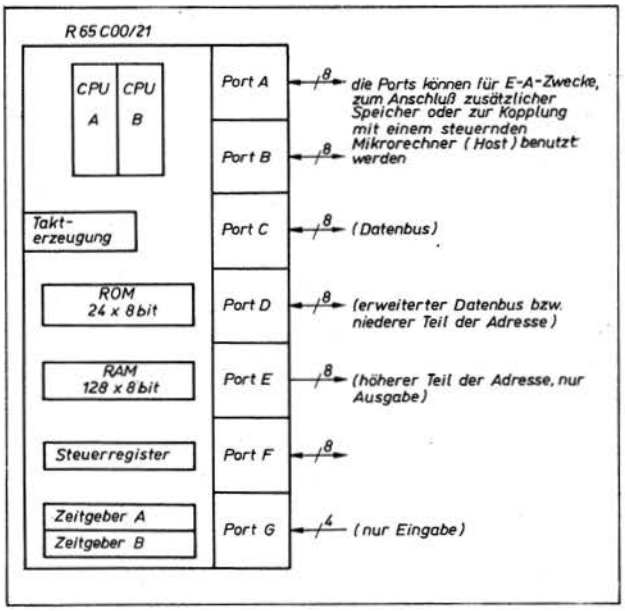


Bild 118: Beispiel für einen Mikroprozessorschaltkreis, der zwei unabhängige CPUs enthält

Der Trend zu größeren Verarbeitungsbreiten (16 bit, 32 bit) hat für bestimmte Aufgabengebiete Vorteile und ermöglicht den Einsatz von Mikrorechnern, wo bisher aufwendigere Rechenanlagen erforderlich waren. Für Aufgabengebiete, in denen nur wenige arithmetische Operationen vorkommen, bieten derartige Mikroprozessoren die Vorteile des erweiterten Adressenraumes, der schnelleren Adressenrechnung und der schnelleren Blocktransporte.

Diese Vorteile werden mit höherem Aufwand erkauft, da die Daten- und Adressenwege der Bussysteme entsprechend ausgelegt werden müssen.

Deshalb gibt es von verschiedenen 16-bit-Typen auch Versionen, die ein auf 8-bit-Bussysteme orientiertes Anschlußbild haben. Zur Beurteilung der tatsächlichen Geschwindigkeitsverhältnisse bei derartigen Systemen muß die Anzahl der Buszyklen je Befehl betrachtet werden. Ein großer Adressenraum hat den Vorteil, daß in vielen Fällen eine Segmentierung nicht erforderlich ist und daß Sonderfunktionen ohne zusätzliche Schaltmittel durch entsprechende Interpretation höherwertiger Adressenbits ausgelöst werden können (Bild 116). (Die Belegung 0 zur Auswahl der Funktionseinheit kennzeichnet einen internen Zugriff, ansonsten wird eine von 31 adressierbaren Slaveeinrichtungen ausgewählt).

Andererseits hat eine lange Adresse bei schmalen Bussen den Nachteil, daß jeder Zugriff mehr Buszyklen kostet, als dies bei einer kurzen Adresse der Fall wäre, auch wenn nur ein kleiner Teil des Adressenraumes benutzt wird.

Bei einigen moderneren Mikroprozessoren sind Funktionen, die für die Multimikrorechnerkopplung und für den Aufbau komplexer Systeme benötigt werden, in den LSI-IS selbst realisiert. Dazu gehören Test-and-Set-Befehle sowie die Möglichkeit, einzelne Zugriffe zu unterbrechen. Beispiele dafür sind die Mikroprozessoren MC 68 000 [13], Z 8003 und Z 8004 [17].

3.4.3. Neue Multimikrorechnerstrukturen

Der zunehmende Integrationsgrad und die Erfahrungen, die bei der Realisierung von Multimikrorechnersystemen gewonnen wurden, erleichtern es, andere Strukturen zu verwirklichen als jene, die bisher bevorzugt diskutiert wurden.

Besteht ein Mikrorechner nur aus wenigen einfach zu verschaltenden IS, so ist dessen Standardisierung nicht mehr in dem Maße zwingend, wie dies der Fall ist, wenn zusätzlich zu den Mikroprozessorschaltkreisen eine größere Zahl anderer Schaltmittel vorgesehen werden muß. Liegen definitive Erfahrungen über die Struktur und Organisationsform eines Multimikrorechnersystems für ein bestimmtes Anwendungsgebiet vor, kann auf ein universelles Bussystem wenigstens dort verzichtet werden, wo es auf maximale Leistung ankommt. So lassen sich Strukturen schaffen, die für bestimmte Einsatzgebiete wirtschaftlicher bzw. leistungsfähiger sind als universelle modulare Systeme. Als Beispiel sei ein Mehrbenutzersystem nach [18] vorgestellt (Bild 117):

Das System beruht auf einem MC 68 000 als Hauptrechner, dem vier Slaverechner (auf Z-80-Basis) zugeordnet sind. Der Hauptspeicher kann eine Kapazität im Bereich von 512...8 192 Kbyte haben. Je Byte sind vier Zusatzbits für einen Fehlerkorrekturkode vorgesehen. Die Korrekturschaltungen sind mit schnellen Schottky-TTL-IS aufgebaut, so daß durch die Fehlerkorrektur keine Wait-Zustände eingeführt werden müssen. Durch die Slaverechner wird der Hauptrechner von Aufgaben der E-A-Steuerung entlastet.

Ein weiteres Beispiel ist ein Mikroprozessor, der zwei CPUs enthält [19]. Jede einzelne ist ein Mikroprozessor vom Typ 6502 mit verbesserter Befehlsliste, der mit der halben Taktfrequenz des Schaltkreises arbeitet. Beide CPUs haben einen gemeinsamen Speicheradressenraum (Bild 118). Durch die zeitgeteilte Ausnutzung der Hardware ist der Schaltkreis nur um 20 % größer als der Einzelmikroprozessor vom gleichen Typ.

Literatur

[13] Männer, R.; Deluigi, B.: 16-Bit-Prozessoren im Vergleich. 3. Teil. Elektronik, München 30 (1981) 7, S. 101–107

[14] Frohwerk, R. A.: Signature Analysis: A New Digital Field Service Method. Hewlett Packard Journal, Palo Alto 28 (1977) 5, S. 2–8

[15] Speicheranordnung mit Fehlererkennungs- und Diagnoseeigenschaften, vorzugsweise für Mikrorechner. WP G 06 F/225 072

[16] Mikrorechneranordnung, vorzugsweise für den Einsatz in Multimikrorechnersystemen. WP G 06 F/2309 616

[17] Maetosien, R.: 16 bit μPs get. a boast from demand paged MMU. Electronic Design, New York 31 (1983) 26. Mai, S. 174–184

[18] Multi-user 16-bit μC uses 4 slaves to reach 8 MHz, no wait states. Electronic Design, New York 31 (1983) 28. April, S. 50 und 51

[19] Bigelow, B.: Dual Processor tackles complex control tasks. Electronic Design, New York 31 (1983) 9. Juni, S. 161–166

4. Spezielle Hardware

In Multimikrorechnersystemen dienen spezielle Hardwareeinrichtungen dazu, das System mit der Außenwelt zu verbinden bzw. die Ausführung bestimmter Teilaufgaben zu beschleunigen. Dieses Kapitel soll einige Anregungen zur Ausgestaltung von Schaltungskomplexen geben, die für viele Anwendungsgebiete von Bedeutung sind.

4.1. Vermittlung von Speicherzugriffen

Viele spezielle Funktionseinheiten enthalten Speicheranordnungen, zu denen sowohl interne Zugriffe als auch solche von einem Bussystem aus geführt werden müssen. Dazu kann man eine Prioritätsvermittlung einsetzen, wie sie im Abschnitt 3.3. vorgestellt wurde (s. a. die Bilder 91 bis 96). Derartige Schaltungsanordnungen sind recht leistungsfähig, aber relativ aufwendig. Oft bietet sich eine Zeitteilungsvermittlung als konstengünstigere Alternative an. Dies ist namentlich dann der Fall, wenn die internen Zugriffe in einem festen Zeitraster ausgeführt werden müssen. Ein charakteristisches Beispiel dafür ist die Ansteuerung eines Bildschirmgerätes (zyklischer Ablauf der Zeichendarstellung aus einem Bildwiederholspeicher).

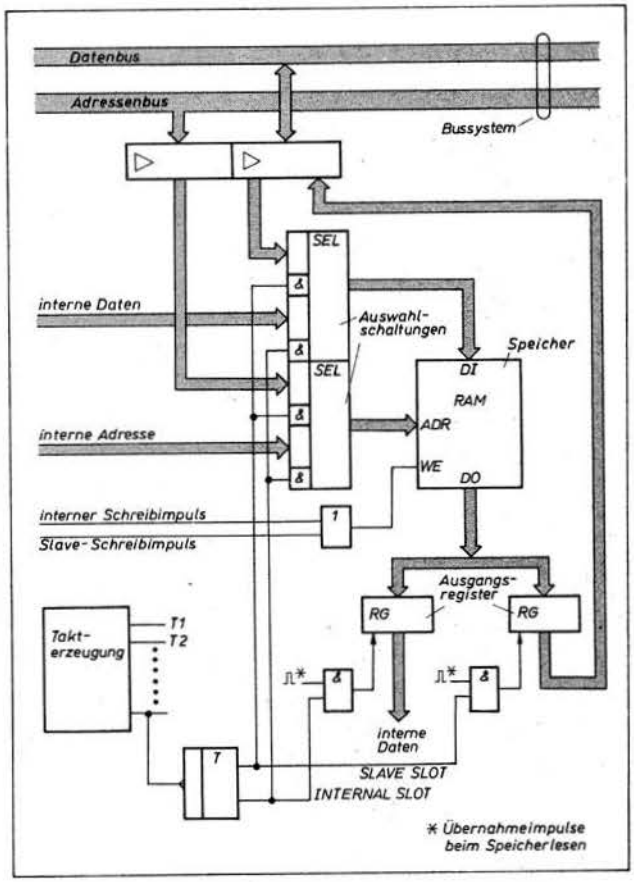


Bild 119: Speicheranordnung mit Zeitteilungsvermittlung

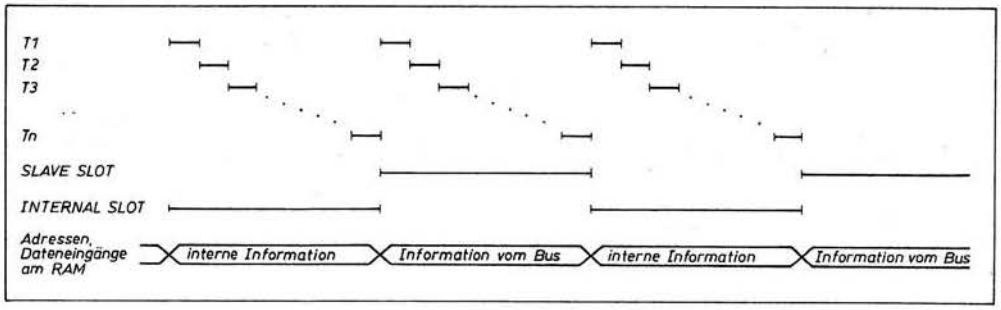


Bild 120: Zeitdiagramm einer einfachen Zeitteilungsvermittlung

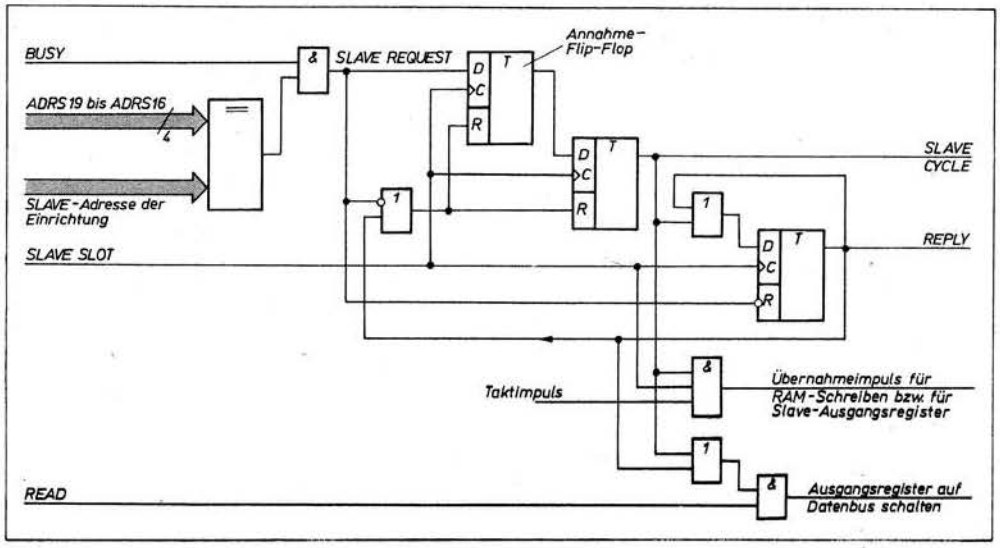


Bild 121: Steuerschaltung für Slavezugriffe zu einem Speicher mit Zeitteilungsvermittlung

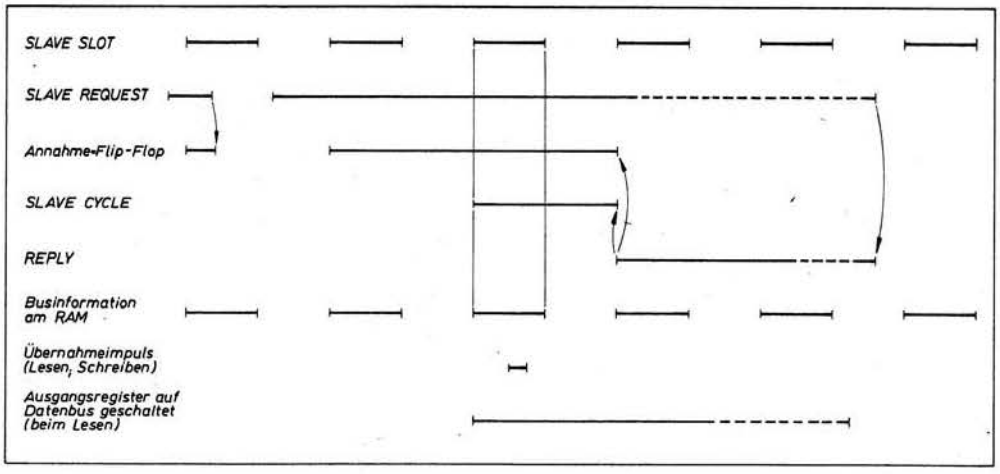


Bild 122: Zeitdiagramm zur Steuerschaltung nach Bild 121

Bild 119 veranschaulicht eine einfache Ausführung und Bild 120 das zugehörige Impulsdiagramm.
Die Taktsteuerung definiert zwei periodisch aufeinanderfolgende Zeitabschnitte:

INTERNAL SLOT

In diesen Zeitabschnitten werden die internen Zugriffe ausgeführt.

SLAVE SLOT

In diesen Zeitabschnitten sind Zugriffe des Bussystems möglich.

Die Zugriffe des Bussystems müssen mit den SLAVE-SLOT-Zeitabschnitten synchronisiert werden. Bild 121 zeigt eine Schaltung dazu (für den Anschluß an das Bussystem gemäß 2.3.).
SLAVE CYCLE ist das Steuersignal für die Ausführung des Zugriffs. Es wird erst dann aktiv, wenn die Slave-Anforderung (SLAVE REQUEST) wenigstens für die Dauer einer SLOT-Periode konstant anliegt. Dadurch werden Störimpulse unterdrückt (beispielsweise solche, die durch Störspitzen auf der BUSY-Leitung verursacht werden; s. a. Bild 34). Das Impulsdiagramm dazu ist im Bild 122 dargestellt.
Die Zeitteilungsvermittlung ist auch innerhalb von Mikrorechnern einsetzbar, sei es zum Anschluß spezieller Schaltungen oder als Alternative zur Prioritätsvermittlung.

Bild 123 zeigt einen Vorschlag für die Synchronisationsschaltung.
Eine andere Lösung zur Bildung von WAIT zeigte Bild 105, sie wäre prinzipiell auch hier anwendbar.

4.2. Beschleunigung von Suchabläufen

Das Durchsuchen von Speicherbereichen mit programmierten Suchabläufen ist oft sehr zeitaufwendig. Manche Mikroprozessoren (so auch der U 880) haben Blocksuchbefehle, mit denen derartige Abläufe beschleunigt werden können. Allerdings sind die Möglichkeiten des Suchens oft recht eingeschränkt. Beim U 880 ist z. B. nur der Vergleich mit einem Byte (im A-Register) möglich; das Suchen nach einzelnen Bits bzw. Bitkombinationen muß explizit (als Schleife) programmiert werden. Der Einsatz eines DMA-Schaltkreises kann das Problem lösen [20].
In manchen Systemen lassen sich derartige Schaltkreise jedoch nicht ohne weiteres einsetzen. In solchen Fällen ist die Beschaltung des Datenausganges des zu durchsuchenden Speichers mit zusätzlichen Maskierungsschaltungen möglich. Diese bewirken, daß bei Lesezugriffen nicht die gelesenen Bytes, sondern charakteristische Bitmuster auf den Bus gelegt werden, wobei das Bitmuster bei erfüllter Suchbedingung einen definierten Wert annimmt. Für diesen kann somit ein Blocksuchbefehl ausgeführt werden.
Eine derartige Anordnung kann beispielsweise dazu dienen, in einer Ansteuerschaltung für Sichtgeräte (Videoadapter) häufig gebrauchte Suchoperationen zu beschleunigen [21]. Typische Beispiele sind

- Suchen nach NUL-Zeichen
- Suchen nach Zeichen, die keine NUL-Zeichen sind
- Suchen eines Feldsteuerzeichens
- Suchen eines modifizierten Feldes
- Suchen eines ungeschützten Feldes.

Bild 124 zeigt die Schaltungsanordnung. Die zugehörigen Datenformate sind im Bild 125 dargestellt.

- Anfang und Länge des zu durchsuchenden Bereiches bestimmen
- Busadressenregister laden; dabei BURST MODE aktivieren (s. Abschn. 3.3.2.)
- Maskenregister laden (s. Tafel 12)
- A-Register mit Suchzeichen laden (s. Tafel 12)
- Blocksuchbefehl ausführen
- Maskenregister mit 0 laden
- BURST MODE ausschalten.

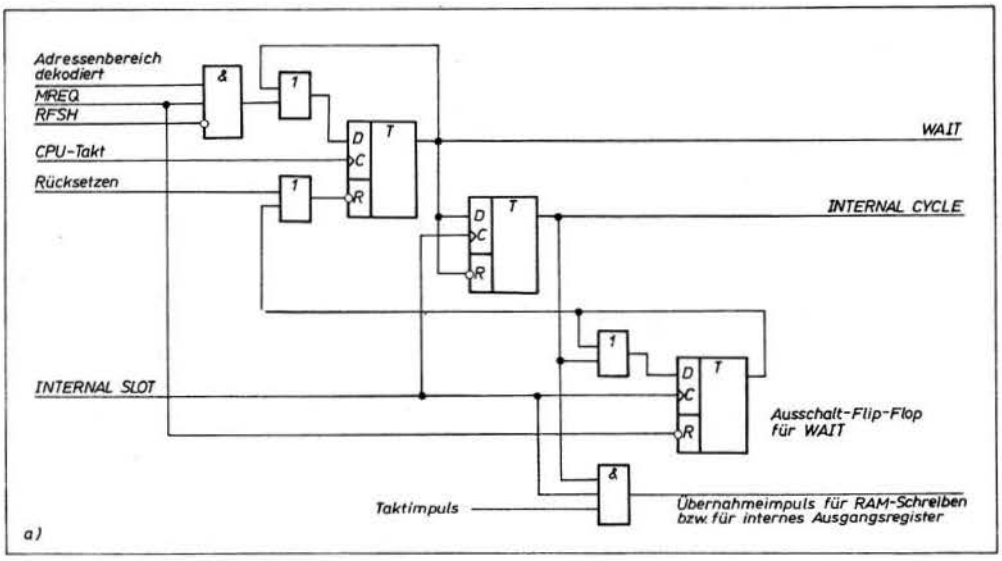
Das Einschalten von BURST MODE ist erforderlich, um Zugriffe anderer Funktionseinheiten zu dem zu durchsuchenden Speicher zu blockieren (infolge der Maskierung würden bei diesen Zugriffen inkorrekte Daten gelesen werden). Da der Suchprozeß relativ schnell abläuft, ist eine softwaremäßige Verwaltung des Speichers nicht sinnvoll (s. a. Abschnitt 1.5., besonders Tafel 1).
Das Prinzip ermöglicht vielfältige Abwandlungen. Im allgemeinen handelt es sich darum, die Blocksuch- bzw. Blocktransportbefehle des Mikroprozessors als Adressenlieferanten für spezielle Operationen zu benutzen. Bild 126 zeigt, wie man mit Blocktransportbefehlen die Bytes eines Speicherbereiches in einen anderen Kode wandeln kann. Die betreffenden Datenbereiche und das steuernde Programm dürfen sich natürlich nicht im selben Speicher befinden.

4.3. Spezielle Steuerwerke

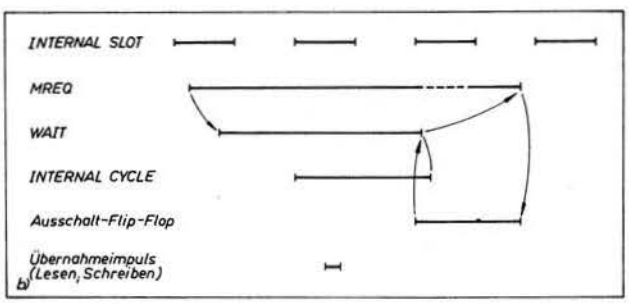
4.3.1. Mikrorechner oder spezielles Steuerwerk?

Um die Verbindungen des Systems mit der Außenwelt zu steuern, können Mikrorechner oder spezielle Steuerwerke eingesetzt werden. Im folgenden sollen diese Verbindungen allgemein als Interfaces bezeichnet werden, gleichgültig, ob sie zu anderen informationsverarbeitenden Einrichtungen, zu Antriebsmitteln, Sensoren oder dergleichen führen. Dabei ist allein der Signalaustausch von Interesse; Probleme der Pegelwandlung, der Leistungselektronik usw. werden nicht behandelt.

Bild 123: Synchronisationsschaltung (Waitsteuerung) für den Anschluß eines Mikroprozessors an eine Zeitteilungsvermittlung. a) Prinzipschaltung; b) Zeitdiagramm



Die Zeichen, die auf dem Bildschirm dargestellt werden sollen, sind in einem 7-bit-Kode gespeichert. Das achte Bit in jedem Byte unterscheidet zwischen darstellbaren Zeichen und Feldsteuerzeichen. Letztere werden als Leerzeichen dargestellt. Sie definieren bestimmte Eigenschaften des nachfolgenden Zeichenfeldes. Das Suchen nach Feldern mit einer bestimmten Charakteristik ist eine häufige Aufgabe für einen Mikrorechner, der zur Steuerung eines Sichtgerätes vorgesehen ist. Derartige Suchaufgaben können mit den Blocksuchbefehlen des U 880 (CPIR/CPDR) programmiert werden. Ein charakteristischer Ablauf dafür ist der folgende:



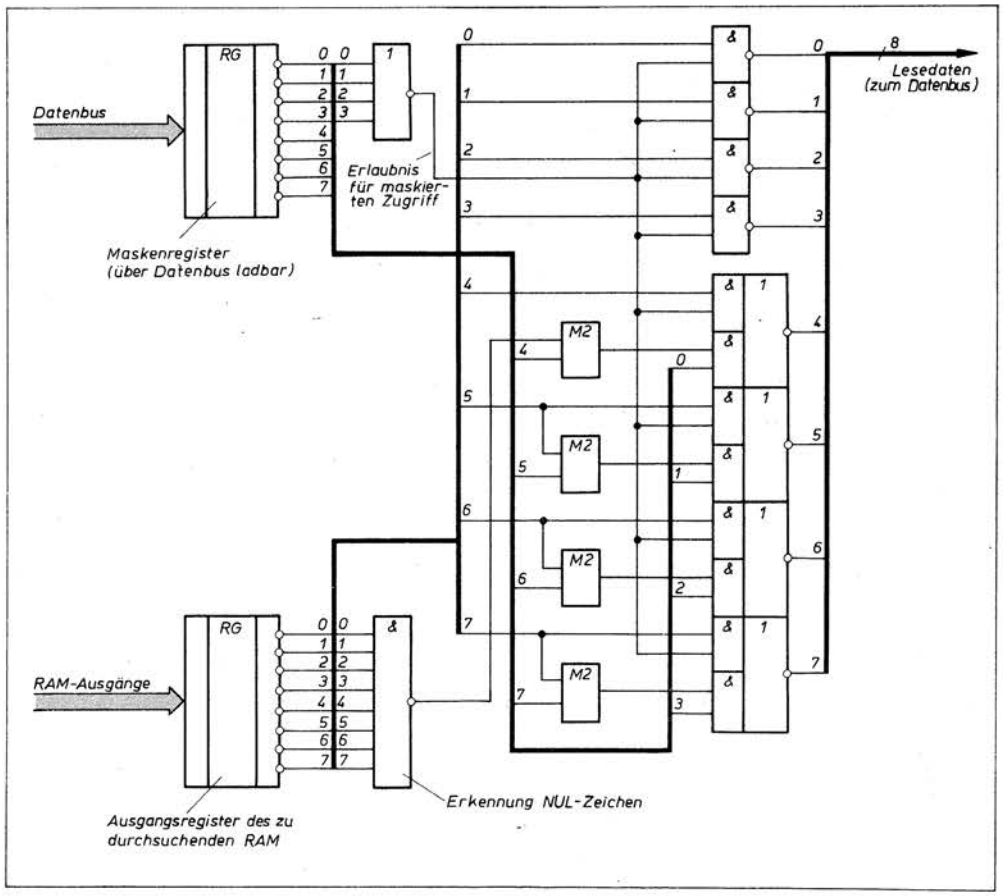


Bild 124: Schaltungsanordnung am Ausgang eines Speichers für die Beschleunigung von Suchabläufen

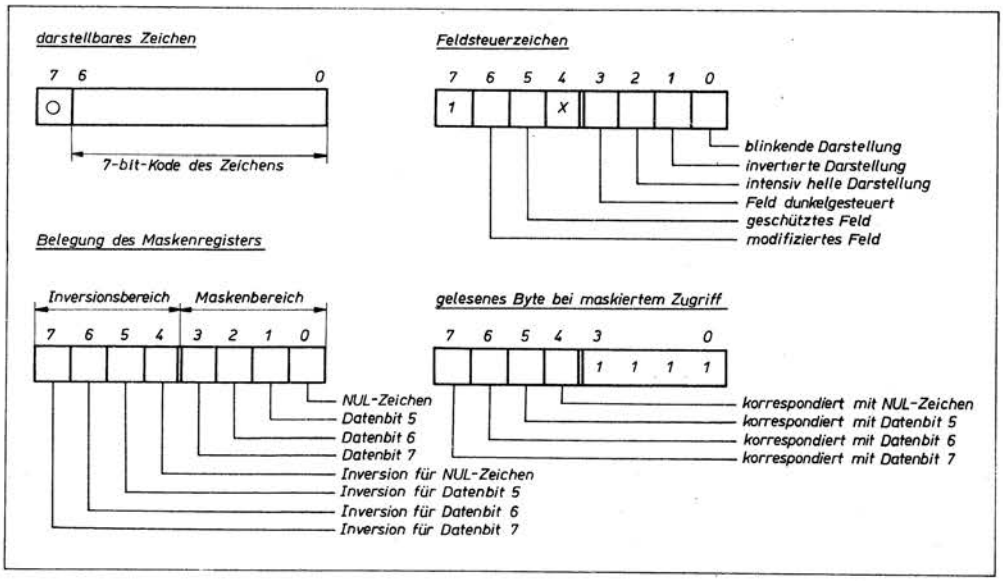


Bild 125: Datenformate, die für die Anordnung nach Bild 124 wesentlich sind

Vor dem Entwurf bzw. der Auswahl eines speziellen Steuerwerkes sollten stets die Mikrorechner des Systems daraufhin untersucht werden, ob sie sich zur Steuerung des gegebenen Interfaces direkt eignen oder mit einfachen Mitteln dazu adaptieren lassen. Die Entscheidung für den Einsatz eines speziellen Steuerwerkes (aus Leistungsgründen) bzw. für den Verzicht auf ein solches (aus Kostengründen) erfordert eine sorgfältige Untersuchung der zu steuernden Interfaceabläufe und deren zeitlicher Bedingungen. Im besonderen sind die zu realisierenden Reaktionszeiten und die algorithmische Komplexität der Abläufe, die die jeweilige Reaktion gewährleisten, von Bedeutung.

Zur Untersuchung, ob sich ein Mikrorechner des Systems für den betreffenden Zweck eignet, empfiehlt sich ein probeweises Programmieren der wesentlichen Abläufe und Auszählen der Maschinenzyklen. Auf die ermittelten Zeiten sollten 30...40 % aufgeschlagen werden (Reserven für Änderungen, Ungenauigkeiten der probeweisen Programmierung usw.). Für häufig vorkommende typische Abläufe lassen sich obere Grenzen der Leistungsfähigkeit angeben. Dazu wird die Steuerungsaufgabe in die Teile Annahme externer Bedingungen (Eingangssignale), Ablauf des Steueralgorithmus und Einstellen externer Signale (Ausgangssignale) zerlegt.

Der Zeitbedarf der beiden letztgenannten Teile wird durch die Befehlsliste und interne Arbeitsgeschwindigkeit des Mi-

Tafel 12: Suchzeichen für Blocksuchbefehle (U 880) zum Betrieb der Schaltung nach Bild 124

Suchaufgabe	Masken-register	erwartetes Datenbyte
NUL-Zeichen	11H	EFH
kein NUL-Zeichen	01H	EFH
Feldsteuerzeichen	88H	7FH
darstellbares Zeichen	08H	7FH
modifiziertes Feld	CCH	3FH
nicht modifiziertes Feld	8CH	3FH
geschütztes Feld	AAH	5FH
ungeschütztes Feld	8AH	5FH
ungeschütztes modifiziertes Feld	CEH	1FH
ungeschütztes unmodifiziertes Feld	8EH	1FH
geschütztes modifiziertes Feld	EEH	1FH
geschütztes unmodifiziertes Feld	AEH	1FH

Tafel 13: Zeitverhältnisse für verschiedene Varianten der Annahme eines Eingangssignals beim U 880

Ablauf	Befehlsfolge	Bemerkungen	Zeitbedarf (Zykluszeit 400 ns)
Hardware zurücksetzen	Befehl von Adresse 0 ist der erste der gewünschten Steuerroutine	Es darf keine Initialisierung o. ä. erforderlich sein	1,6 µs
NMI	Signal wird am Anfang eines langen Befehls [z. B. SET 3, (IY+8)] wirksam; am Ende wird NMI ausgelöst (PC retten, zu Adresse 66H verzweigen)	Es gibt kein Retten von Registerinhalten. Rückkehrbefehl wird mitgezählt	19 µs
Interrupt	HL: HALT JR HL-# Interrupt-Mode 2, Rückkehrbefehl (RETI) wird mitgezählt	Es gibt kein Retten von Registerinhalten. 2 Fälle 1. Signal wirkt direkt (z. B. über spezielle Interrupthardware) 2. Signal wirkt über STB eines PIO-Ports	 ≈ 12 µs ≈ 12 µs + Dauer von STB
BUS REQUEST	Die CPU ist solange vom Bus getrennt, bis das Signal aktiv wird	Der Bus wird z. B. am Ende eines speziellen OUT-Befehls abgeschaltet (der auf die Steuerhardware wirkt). Der Folgebefehl ist dann der erste der Steuerroutine	> 800 ns
WAIT	Die CPU verharrt im WAIT-Zustand, bis das Signal aktiv wird	WAIT wird durch Dekodieren der Speicheradressen gebildet, z. B. LD HL, Adresse LD A,(HL); dieser Befehl führt zu WAIT	> 400 ns
Abfrage über Speicheradressen	LD HL, Adresse BA: BIT n,(HL) JRZ BA-# bzw. LD HL, Adresse BA: BIT n,(HL) JPZ BA	Schleife dauert 7,6 µs Schleife dauert 8,8 µs	> 14 µs > 13 µs
Abfrage über E-A-Adressen (z. B. PIO-Port im Bitmodus)	BA: IN Port BIT n,A JRZ BA-#	Schleife dauert 10 µs	> 12 µs

krorechners bestimmt. Für die Annahme externer Bedingungen gelten die Erläuterungen im Abschnitt 3.1.2. (s. Tafeln 3 und 4). Ergänzend dazu gibt Tafel 13 eine Übersicht über die Reaktionszeiten des U 880. Es ist angegeben, wieviel Zeit vergeht, bis der erste Befehl des eigentlichen Steueralgorithmus gelesen wird, wenn sich die Belegung eines Eingangssignals ändert. Dabei wird angenommen, daß nur dieses eine Eingangssignal für den jeweiligen Steuerablauf von Interesse ist. (Es handelt sich also um den einfachsten Fall.)

Bild 127 veranschaulicht, welche maximalen Datenraten sich beim U 880 für einfache Datentransportoperationen ergeben. Jede zusätzliche Bezugnahme auf die Datenbytes (z. B. Fehlerkontrollen, Kodewandlungen usw.) verlängert die Übertragungszeit. Blocktransportbefehle ermöglichen recht schnelle Transportabläufe mit Längenzählerverwaltung (bei LDIR z. B. 21 Zyklen je Byte, ≙ 8,4 µs). Der Mikroprozessor kann allerdings in diesen Fällen nur den eigentlichen Datentransport ausführen, alle weiteren Funktionen (z. B. Organisation des Handshaking mit dem externen Interface, Fehlerkontrollen usw.) erfordern zusätzliche Schaltungsmaßnahmen.

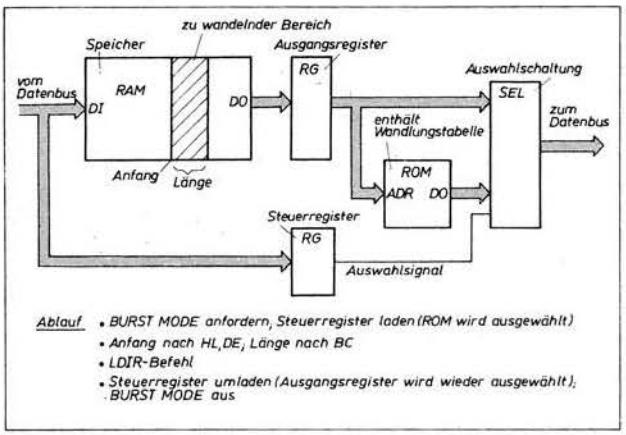


Bild 126: Speicheranordnung, die ein Umkodieren des Inhaltes mit Blocktransportbefehlen ermöglicht

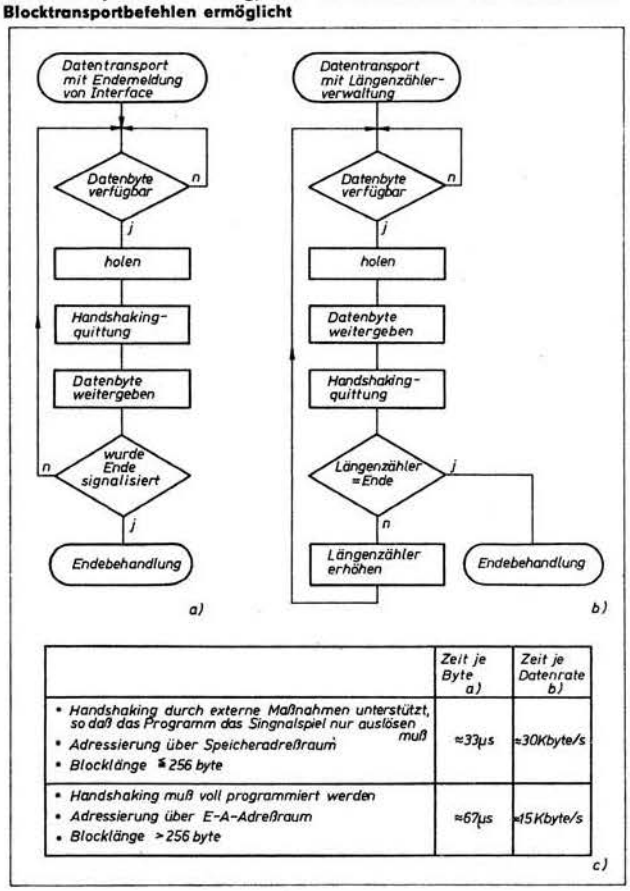


Bild 127: Einfache Datentransportabläufe und deren Leistungsgrenzen beim U 880. a) Ende des Datenstromes wird vom Interface signalisiert (Endemarke, Markierungssignal o. ä.); b) Länge des Datenstromes wird programmtechnisch durch Zählen der übertragenen Bytes bestimmt; c) weitere Annahmen

Für die Anpassung des Transportablaufes an die Datenrate des Interfaces bietet sich die Waitsteuerung an, für das Signalisieren von Fehlerreaktionen und dergleichen das Auslösen eines Interrupts (bzw. NMI) (Bild 128). Zusätzlich ist zu untersuchen, ob der Mikrorechner ausschließlich zum Betrieb des Interfaces verwendet werden oder ob er mehrere Aufgaben zeitmultiplex (d. h. im Multiprogramming) ausführen soll.

Die Benutzung von RESET, NMI, BUSRQ, WAIT schließt im allgemeinen das Multiprogramming aus (zumindest während der Zeitabschnitte, in denen das Interface zu steuern ist). Bei Interrupt- oder Abfragesteuerung ergeben sich wesentlich andere Zeitverhältnisse. Einige Werte für U 880 sind:

- Die Interruptreaktion dauert 1...2 ms bei Benutzung eines universellen Echtzeitbetriebssystems und 20 bis 100 µs bei direkter physischer Behandlung, wobei alle CPU-Register gerettet werden (durch Benutzung der Austauschregister bzw. durch Kellern). Die direkte physische Behandlung ist allerdings nur dann möglich,

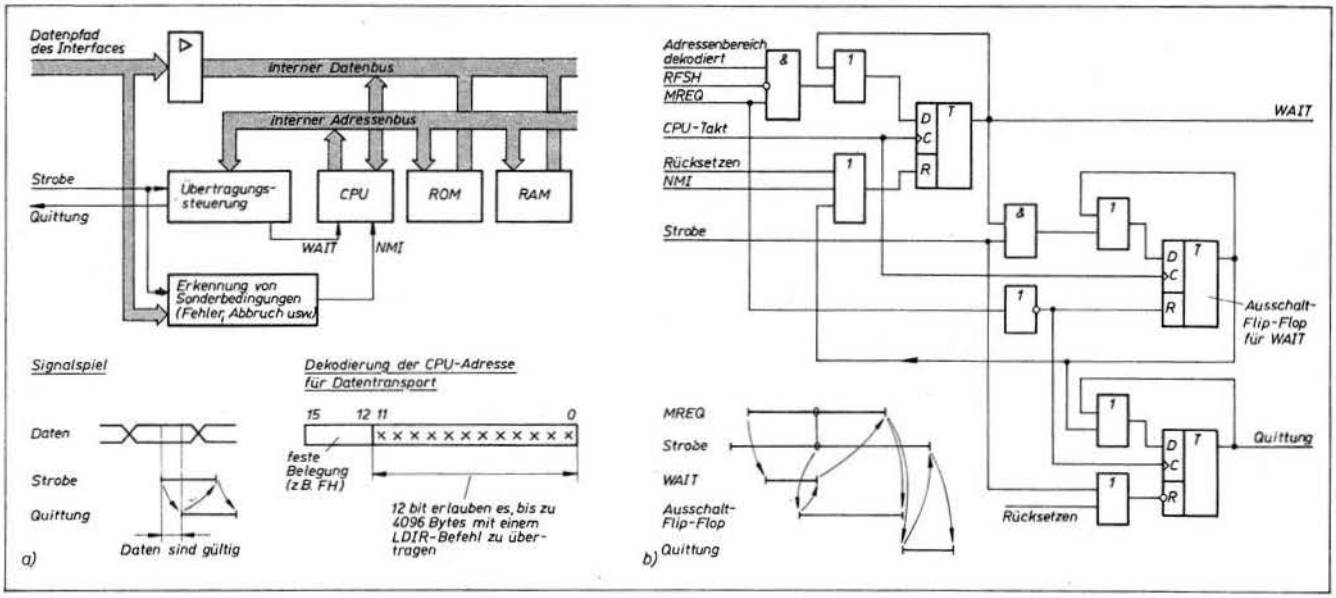


Bild 128: Direkte Kopplung des Interfaces an einen Mikrorechner mit Waitsteuerung der eigentlichen Datenübertragung. a) Prinzip; b) Schaltung für Waitsignal und Handshaking am Interface

wenn der Steuerablauf in einer einfachen sofortigen Reaktion besteht (z. B. Datenbyte abnehmen, speichern, Quittungssignal senden), die weder Ressourcen anderer Programme benötigt, noch zuviel Laufzeit erfordert.

- Die Dauer einer Abfrageschleife an sich wird nicht beeinflußt (sie dauert z. B. so lange, wie in der Tafel 13 oder im Bild 127 angegeben ist), durch Interrupts anderer Prozesse oder durch zeitgesteuertes Umschalten (Time Slicing) können allerdings unvorhersagbare Verzögerungen auftreten (Richtwert 10...200 ms).

Erweist sich ein standardisierter Mikrorechner als ausreichend, dürfte die Entscheidung unproblematisch sein. Durch Ausnutzung von RESET, NMI, BUSRQ oder WAIT zur Interfacesteuerung, womöglich in Verbindung mit weiteren leistungserhöhenden Schaltungsmaßnahmen (s. Bild 128), erhält man allerdings einen spezialisierten Mikrorechner, bei dessen Entwicklung besondere Probleme auftreten (Testen der Sonderfunktionen, Besonderheiten beim Programmieren usw.). Für derartige Lösungen spricht, daß die Programmierunterstützung von vornherein gegeben ist (z. B. Entwicklungssystem). Dagegen spricht, daß trotz umfangreicher Sonderhardware, mit der der Kern des Mikrorechners (CPU, ROM, RAM usw.) beschaltet werden muß, der Leistungsgewinn beschränkt bleibt.

4.3.2. Realisierungsmöglichkeiten spezieller Steuerwerke

Für die hardwaremäßige Lösung von Steuerungsproblemen bieten sich folgende Möglichkeiten an:

- Ad-Hoc-Zusammenschaltungen von Gattern, Flip-Flops usw.
- regulärer Aufbau als endlicher Automat unter Verwendung üblicher Schaltmittel (Gatter, Flip-Flops, PLAs, ROMs usw.)
- mikroprogrammierte Steuerwerke.

Für sehr elementare Steuerungsprobleme, etwa für die Realisierung eines Handshaking-Signalspiels, gibt es bewährte Schaltungen bzw. wird eine zweckmäßige Lösung durch das verfügbare Bauelementesortiment nahegelegt (Einsatz von flankengesteuerten Flip-Flops, Schieberegistern, Zählern). Für alle nicht elementaren Probleme lohnt sich eine systematische Vorgehensweise. Eine bewährte Methode zur Realisierung von Steuerabläufen mittlerer Komplexität (Richtwert 20 bis 40 Steuerzustände) ist die 1-aus-n-Kodierung der Zustände. Für jeden Zustand wird ein Flip-Flop vorgesehen. (Während des Betriebes ist dann zu einem Zeitpunkt stets nur eines eingeschaltet.) Die gewünschten Zustandsübergänge werden durch die entsprechende Ausführung der Ein- und Ausschaltbedingungen der Trigger mit kombinatorischen Schaltmitteln realisiert. Durch Einsatz von flankengesteuerten oder Master-Slave-Flip-Flops werden Wettlauferscheinungen vermieden.

In direkter Anlehnung an automatentheoretische Betrachtungsweisen kann man Steuerschaltungen auch dadurch aufbauen, daß zur Realisierung der Überführungsfunktion ROM- bzw. PLA-Zuordner eingesetzt werden. In einem solchen Fall lohnt es sich, die Zustände binär zu kodieren. Für n Zustände werden ld n Zustandsvariable bzw. Flip-Flops benötigt. Damit werden die Speicherschaltkreise bzw. PLAs besser ausgenutzt [22]. Für die Erzeugung der Ausgangssignale ist in der Regel ein weiterer Zuordner erforderlich. Damit wird die Hardware des Steuerautomaten zu einer recht einfachen Angelegenheit (Bild 129).

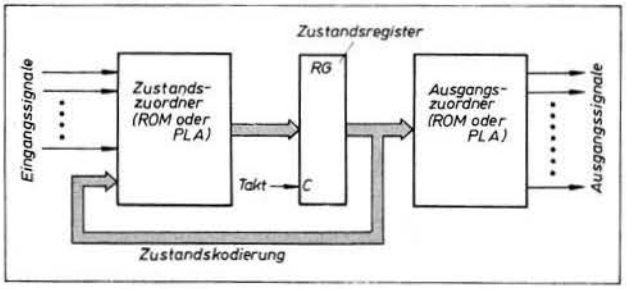


Bild 129: Realisierung einer Steuerschaltung als Moore-Automat

Werden Speicherschaltkreise zur Realisierung der kombinatorischen Verknüpfungen vorgesehen, so ist die Hardware zwar schnell entworfen, das Problem aber noch nicht gelöst: Der Speicherinhalt muß noch bestimmt werden. Dies ist nur in elementaren Fällen auf rein manuelle Weise beherrschbar, denn der Aufwand für die rein manuelle Kodierung von z. B. 4 Kbyte ist zu hoch. Für größere Problemstellungen ist eine rechentechnische Unterstützung unumgänglich. Ein entsprechendes Programmsystem wird in [22] vorgestellt. [23] gibt eine Einführung in den Problemkreis, dort werden auch die wesentlichen Prinzipien und Berechnungsschritte veranschaulicht.

Bei zunehmender Komplexität der Steueralgorithmen wird es schwieriger, die einzelnen Steuerzustände und die Übergänge zwischen ihnen explizit anzugeben. Eine ablauforientierte Darstellung (etwa als Flußdiagramm) erscheint oftmals zweckmäßiger. Damit bietet sich ein mikroprogrammiertes Steuerwerk als Lösung an.

Es gibt Algorithmen, mit denen ein Programmablaufplan in den Übergangsgraphen eines Automaten umgeformt werden kann [23], so daß sich auch aus einer ablauforientierten Darstellung eine Struktur gemäß Bild 129 entwickeln läßt. Die Grenzen der praktischen Anwendbarkeit sind durch folgende Sachverhalte gegeben:

- Jedem Anweisungselement des Ablaufplanes muß ein unabhängiger Zustand zugeordnet werden.
- Für n Zustände muß der Zuordner eine Speicherkapazität von $n \cdot 2^e$ Worten mit ld n bit Länge haben (e = Anzahl der Eingangsvariablen).

- Der Ausgangszuordner erfordert bei a Ausgangsvariablen n Wörter mit einer Länge von a bit (Moore-Automat).
- Für größere Zustandszahlen ist das Verfahren ohne rechentechnische Unterstützung kaum beherrschbar (nicht zuletzt unter dem Aspekt von Änderungen, Verbesserungen usw., man vergleiche die Reparatur eines Programmes mit der Notwendigkeit, bei jeder Änderung den Wandlungsalgorithmus erneut ausführen zu müssen).

Beispiel

Gemäß Bild 129 ist ein Automat mit folgenden Eigenschaften zu realisieren: 10 Eingangssignale, 64 Zustände, 20 Ausgangssignale. Bild 130 zeigt seine Struktur.

Die Attraktivität derartiger Lösungen wird wesentlich von der Technologie der Speicherschaltkreise bestimmt. Eine Realisierung des Beispiels mit 1-Kbyte-EPROMs (z. B. U 555) wäre kaum sinnvoll (67 IS für beide Zuordner); mit 16-Kbyte- und 32-Kbyte-EPROMs wäre es hingegen eine sehr interessante Lösung, die hinsichtlich IS-Anzahl, Einfachheit und Reaktionsgeschwindigkeit einer Mikroprogrammsteuerung eindeutig überlegen ist.

Daraus wird ersichtlich, welche prinzipiellen Probleme bei einem mikroprogrammierten Steuerwerk zu betrachten sind: Es emuliert die Überführungs- und Ausgangsfunktion des Automaten durch zeitlich aufeinanderfolgende Mikrobefehle und ist somit vom Prinzip her langsamer als die direkte Realisierung des Automaten nach Bild 129. Für das Holen und Ausführen der Mikrobefehle müssen Schaltmittel vorgesehen werden. Das Erstellen der Mikroprogramme ist von einem gewissen Umfang an (Richtwert 1 K Mikrobefehle) kaum ohne rechentechnische Unterstützung (z. B. Mikroprogrammassembler) beherrschbar. Sind die entsprechenden Mittel nicht von vornherein verfügbar, so sind folgende Aktivitäten erforderlich, um eine Steuerungsaufgabe auf diese Weise zu lösen:

- Erarbeitung des Steueralgorithmus an sich
- Entwurf eines Mikroprogrammsteuerwerkes und Definition der Mikrobefehlsliste
- Bereitstellung der rechentechnischen Unterstützung
- Erstellung der Mikroprogramme (sowohl für die eigentliche Anwendung als auch zu Diagnosezwecken).

4.4. Prinzipien einfacher Mikroprogrammsteuerwerke

Mikroprogrammsteuerwerke lassen sich bereits mit einem üblichen Schaltkreissortiment (TTL-IS, RAMs, ROMs) in wirtschaftlich sinnvoller Weise aufbauen. Die Größenordnung des Aufwandes reicht von etwa 30 bis zu etwa 300 IS-Gehäusen. Oft wird die obere Grenze des Aufwandes durch das Leiterplattenformat gegeben sein, nämlich stets dann, wenn das Steuerwerk eine Funktionseinheit wie jede

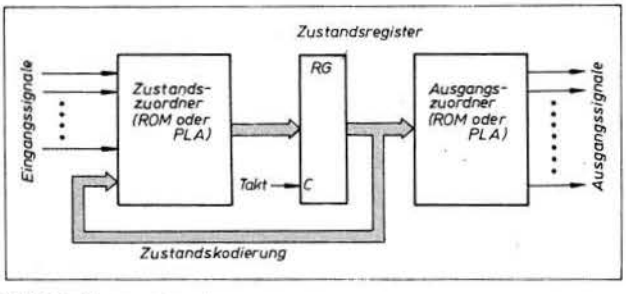


Bild 130: Steuerautomat

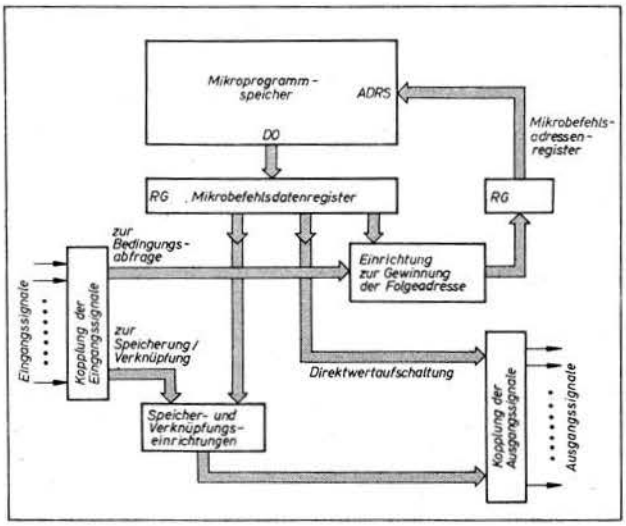


Bild 131: Steuerautomat in Form eines mikroprogrammierten Steuerwerkes

andere sein und sich der Modularisierung des Systems anpassen soll. Man kann deshalb mit den genannten Schaltmitteln kaum ein universelles Steuerwerk schaffen, das für eine Vielzahl von Anwendungen geeignet ist, sondern muß sich darauf beschränken, Steuerwerke für bestimmte Anwendungsfälle bzw. für eingeschränkte Klassen von Anwendungsfällen zu entwerfen.
Damit muß die algorithmische Bearbeitung der jeweiligen Steuerungsaufgabe dem Entwurf des Steuerwerkes vorgeordnet werden. Beim Entwurfsbeginn ist es wesentlich, zu wissen,

- welchen Umfang das Steuerwerk haben darf
- welche Funktionen zu realisieren sind (anzusteuernde Ausgangsleitungen, auszuwertende Eingangsleitungen, Verknüpfungen, überschlägliche Anzahl der Steuerzustände)
- welche Zeitverhältnisse realisiert bzw. beachtet werden müssen
- auf welche Weise das Steuerwerk mit den anderen Einrichtungen des Systems zusammenarbeiten soll.

Prinzipiell besteht der Entwurfsprozeß darin, eine Black Box mit gegebenen Ein- und Ausgangssignalen und hinreichend beschriebenem Sollverhalten in eine Schaltungsanordnung zu überführen, bei der die gewünschten Funktionen durch sequentielle Abarbeitung von Mikrobefehlen aus einem Steuerspeicher realisiert werden (Bild 131).

Im einzelnen sind folgende grundsätzliche Aspekte zu berücksichtigen:

- Realisierung des Mikroprogrammspeichers
- Gewinnung der Mikrobefehlsadresse
- Auswertung von Eingangssignalen und Ausführung von Verknüpfungsoperationen
- Einstellen von Ausgangssignalen
- Sonderfunktionen (Rücksetzen, Interrupts, Unterprogrammaufrufe usw.)
- Mikrobefehlsformate
- Kopplung mit dem Multimikrorechnersystem
- Diagnostische Vorkehrungen.

Für grundlegende bzw. weiterführende Literatur sei auf [24] bis [29] verwiesen.

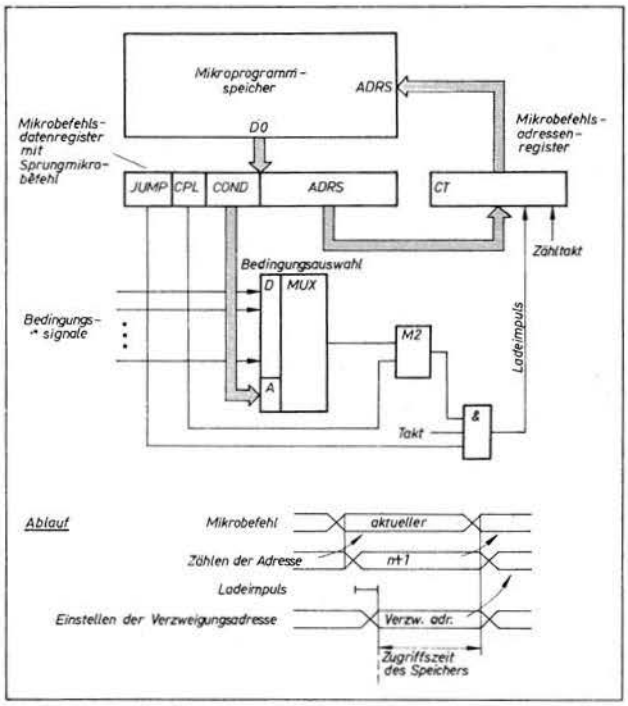


Bild 132: Adressierung der Mikrobefehle durch Adressenzählung (mit Verzweigungsmöglichkeit)

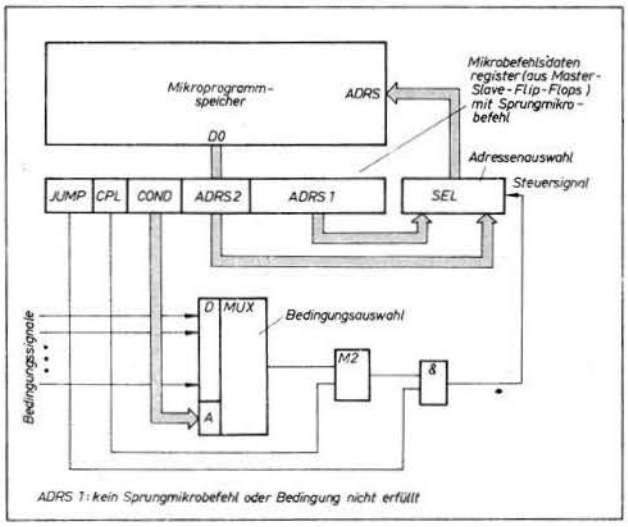


Bild 133: Adressierung der Mikrobefehle durch explizite Adressenangaben im Mikrobefehl

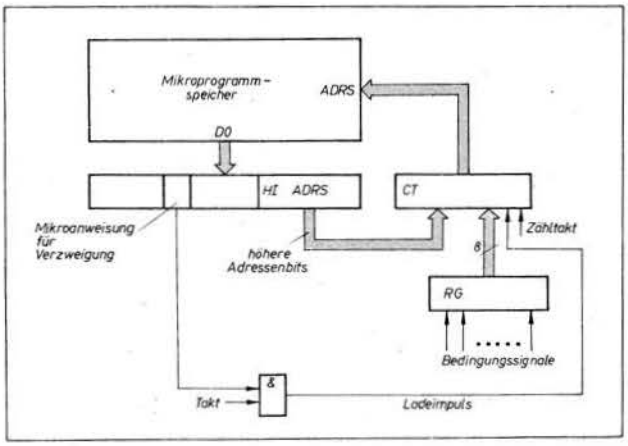


Bild 134: Funktionsverzweigung durch direktes Einspeisen von Bedingungssignalen in die Mikrobefehlsadresse

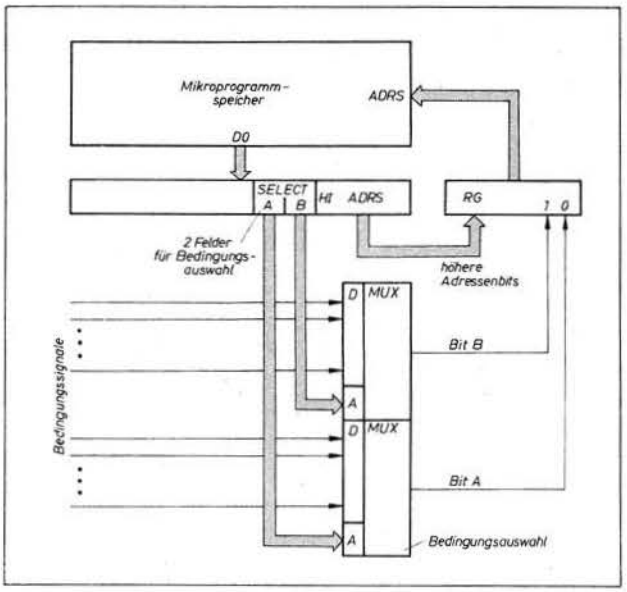


Bild 135: Mehrfachverzweigung

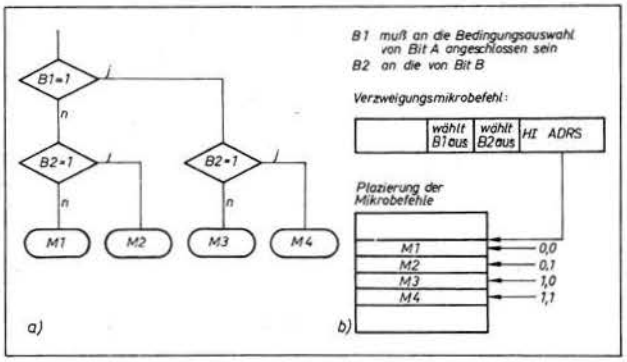


Bild 136: Einfache Sprungmikrobefehle und Mehrfachverzweigung. a) Konventionelle Sprungmikrobefehle; b) Mehrfachverzweigung

4.4.1. Realisierung des Mikroprogrammspeichers

Der Mikroprogrammspeicher kann sowohl ausschließlich mit ROM- oder RAM-Schaltkreisen als auch mit beiden Speichertypen gemischt aufgebaut werden. Für die Auswahl gelten im wesentlichen die gleichen Erwägungen, wie sie im Abschnitt 3.2.1. diskutiert wurden.

Die bei Mikrorechnern üblichen EPROMs mit byteparalleler Organisation (z. B. U 555; 1 K × 8 bit) erlauben einen sehr einfachen Aufbau, wobei Zykluszeiten um 500...700 ns erreichbar sind. Für viele Anwendungen reichen 1 024 bis 2 048 Mikrobefehle bei weitem aus.

Soll ein sehr schnelles Steuerwerk realisiert werden, empfiehlt es sich, RAMs einzusetzen. Der Ladeprozeß läßt sich in einem Multimikrorechnersystem einfach organisieren (etwa aus dem ROM eines Mikrorechners oder gemäß Bild 70). Hingegen sind schnelle PROMs üblicherweise nach dem Schmelzverbindungsprinzip aufgebaut, so daß ein Schaltkreis nicht wiederholt programmiert werden kann. In manchen Fällen muß die Steuerung zu einem gegebenen Zeitpunkt nur einen Ausschnitt aus dem gesamten Funktionsumfang ausführen können. Dann kann man den RAM so klein bemessen, daß die Mikroprogramme des umfangreichsten dieser Ausschnitte gerade hineinpassen. Das System muß bei jedem Funktionswechsel das Nachladen der jeweils benötigten Mikroprogramme veranlassen. Ein Beispiel wäre etwa die Steuerung eines Festplattenspeichers, die Kommandos (z. B. „Lesen Datensatz", „Schreiben Datensatz", „Eröffnen Datei", „Abschließen Datei") ausführen soll, wobei jeweils nur eines dieser Kommandos abgearbeitet werden kann.

4.4.2. Gewinnung der Mikrobefehlsadresse

Eine einfache Methode der Adressierung zeigt Bild 132. Das Mikrobefehlsadressenregister ist als Zähler eingerichtet, so daß der nächste Mikrobefehl von der jeweils folgenden Adresse geholt wird. Für Modifikationen des Ablaufes gibt

es Sprunganweisungen in den Mikrobefehlen. Ist eine ausgewählte Bedingung erfüllt, so wird das Adressenregister ganz oder teilweise mit direkten Adressenwerten aus dem aktuellen Mikrobefehl gefüllt. Die Arbeitsweise eines solchen Steuerwerkes entspricht der eines üblichen Digitalrechners.

Eine Alternative dazu besteht gemäß Bild 133 darin, daß jeder Mikrobefehl die Adresse eines Nachfolgers enthält. Für Verzweigungen muß der Mikrobefehl die alternativen Adressen liefern.

Neben dem Verzweigungsprinzip, das den Sprungbefehlen einfacher Digitalrechner entspricht, läßt sich eine sehr effektive Ablaufmodifikation erreichen, indem ausgewählte Bedingungen nicht abgefragt, sondern direkt als Teile der Folgeadresse verwendet werden. Im Bild 134 ist dargestellt, wie ein Byte aus einem an sich beliebig ladbaren Register (bzw. eine Kombination von Eingangssignalen) direkt die niederen acht Bits der Folgeadresse liefert.

Bild 135 zeigt das bei Zentraleinheiten und peripheren Steuergeräten der EDV häufig benutzte Prinzip der Mehrfachverzweigung durch Einspeisen von Bedingungssignalen in die niedersten Bits der Mikrobefehlsadresse. Ein Teil der höheren Adressenbits kann auch von ladbaren Registern geliefert werden.

Bild 136 veranschaulicht den Gewinn an Geschwindigkeit bei mehrfachen Verzweigungen im Vergleich zu herkömmlichen Sprunganweisungen. Das Problem besteht hier darin, daß es zwei Bedingungssignale B1 und B2 gibt. Für jede der vier möglichen Belegungen ist zu einem Mikrobefehl M1 bis M4 zu verzweigen.

Bei entsprechendem Hardwareaufwand ist ein zusätzlicher Gewinn an Geschwindigkeit erzielbar: Zu einem Mikrobefehl gibt es nur wenige Nachfolger (vier bei Einspeisung von zwei Bits). Bei entsprechendem Aufbau des Mikroprogrammspeichers können diese vier Mikrobefehle parallel gelesen werden. Die Auswahl des aktuellen Nachfolgers erfolgt dann über ein Selektornetzwerk, dessen Verzögerungszeit wesentlich geringer ist als die Zugriffszeit des Speichers. Die Entscheidung über die Auswahl des Nachfolgers kann deshalb bis zum Ende des aktuellen Zyklus verschoben werden (Spätverzweigung, s. Bild 137).

Diskussion

Bei extremen Anforderungen an die Geschwindigkeit wird man den Aufwand von Lösungen ähnlich der nach Bild 137 in Kauf nehmen müssen. Für die Mehrzahl der Anwendungen im Rahmen von Multimikrorechnersystemen dürfte dies kaum notwendig sein.

Die direkte Angabe der Folgeadresse hat den Vorteil, daß kein Mikrobefehlsadressenregister erforderlich ist, wenn das Mikrobefehlsdatenregister mit flankengesteuerten oder Master-Slave-Flip-Flops ausgeführt wird. Damit läßt sich die potentielle Geschwindigkeit der Schaltkreise voll ausnutzen, und das Taktschema wird sehr einfach. Nachteilig ist, daß bei Verzweigungen mindestens zwei Mikrobefehlsadressen (oder Teile davon) im Mikrobefehl mitgeführt werden müssen und daß die Placierung der Mikrobefehle Schwierigkeiten bereiten kann (namentlich dann, wenn keine rechentechnische Unterstützung verfügbar ist). Das Prinzip der Adressenzählung kommt der üblichen Programmierweise entgegen.

Für die Entscheidung zwischen konventionellen Sprungbefehlen und dem Einspeisen von Bedingungen in Teile der Adresse müssen zusätzliche Betrachtungen angestellt werden. Manche Entscheidungsalgorithmen lassen sich mit Sprungbefehlen, bei denen jeweils nur eine Bedingung abgefragt werden kann, nur umständlich programmieren. Kommen derartige Abläufe häufig vor, so bietet sich eine Kombination mit der Lösung nach Bild 134 an. Dabei veranlaßt eine spezielle Verzweigungsanweisung, daß ein Teil der Folgeadresse direkt durch Bedingungssignale gebildet wird. Bild 138 zeigt dazu genauere Details.

In dieser Ausführung liefert der Mikrobefehl bei normalen Verzweigungen die volle Folgeadresse (im Beispiel 12 bit). Das bedingt eine gewisse Mindestlänge der Mikrobefehle (Operationskode + Bedingungsauswahl + 12 bit), hat aber gegenüber einem Verzweigungsschema, bei dem der Mikrobefehl nur Teile der Verzweigungsadresse enthält, folgende Vorteile:

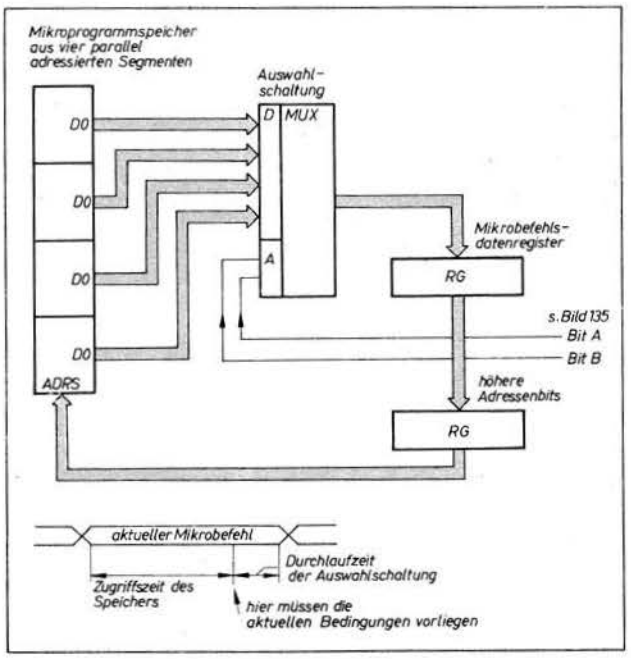


Bild 137: Spätverzweigung

- Universalität, keine Restriktionen für die Placierung der Mikrobefehle
- freizügige Programmierbarkeit, jeder Mikrobefehl kann von jedem anderen erreicht werden
- einfache Schaltungslösung.

Bei Einsatz üblicher EPROMs ergeben sich allein durch die Byteorganisation relativ lange Mikrobefehle. 8 bit sind mit Sicherheit zu wenig, so daß die üblichen Formate mit 16; 24; 32 bit usw. festgelegt werden müssen. Damit verbleibt bei Sprungmikrobefehlen stets genügend Platz für die volle Folgeadresse.

4.4.3. Auswertung von Eingangssignalen

Eingangssignale kann man für die Gewinnung von Verzweigungsbedingungen abfragen, direkt als Teile der Mikrobefehlsadresse einspeisen, Verknüpfungsoperationen unterziehen und speichern.

Das Abfragen von Verzweigungsbedingungen ist mit Multiplexern einfach realisierbar, wobei die Auswahlkodierung oft direkt im Sprungmikrobefehl untergebracht werden kann.

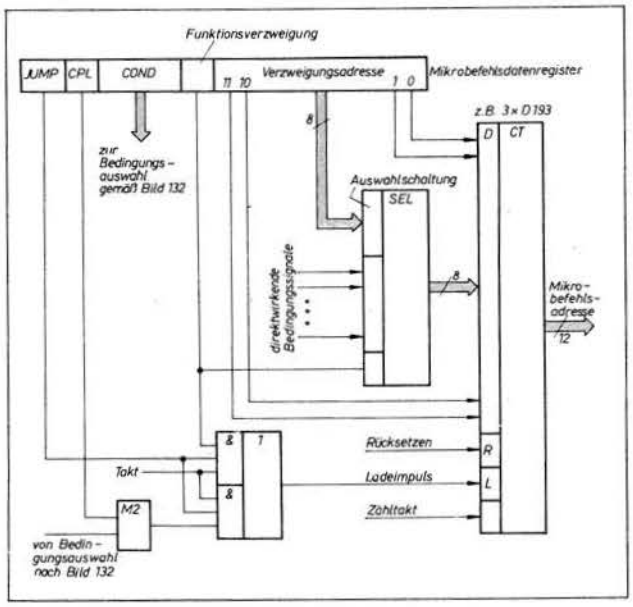


Bild 138: Adressierungsschaltung, bei der Zählung, bedingte Sprünge und Funktionsverzweigung kombiniert sind

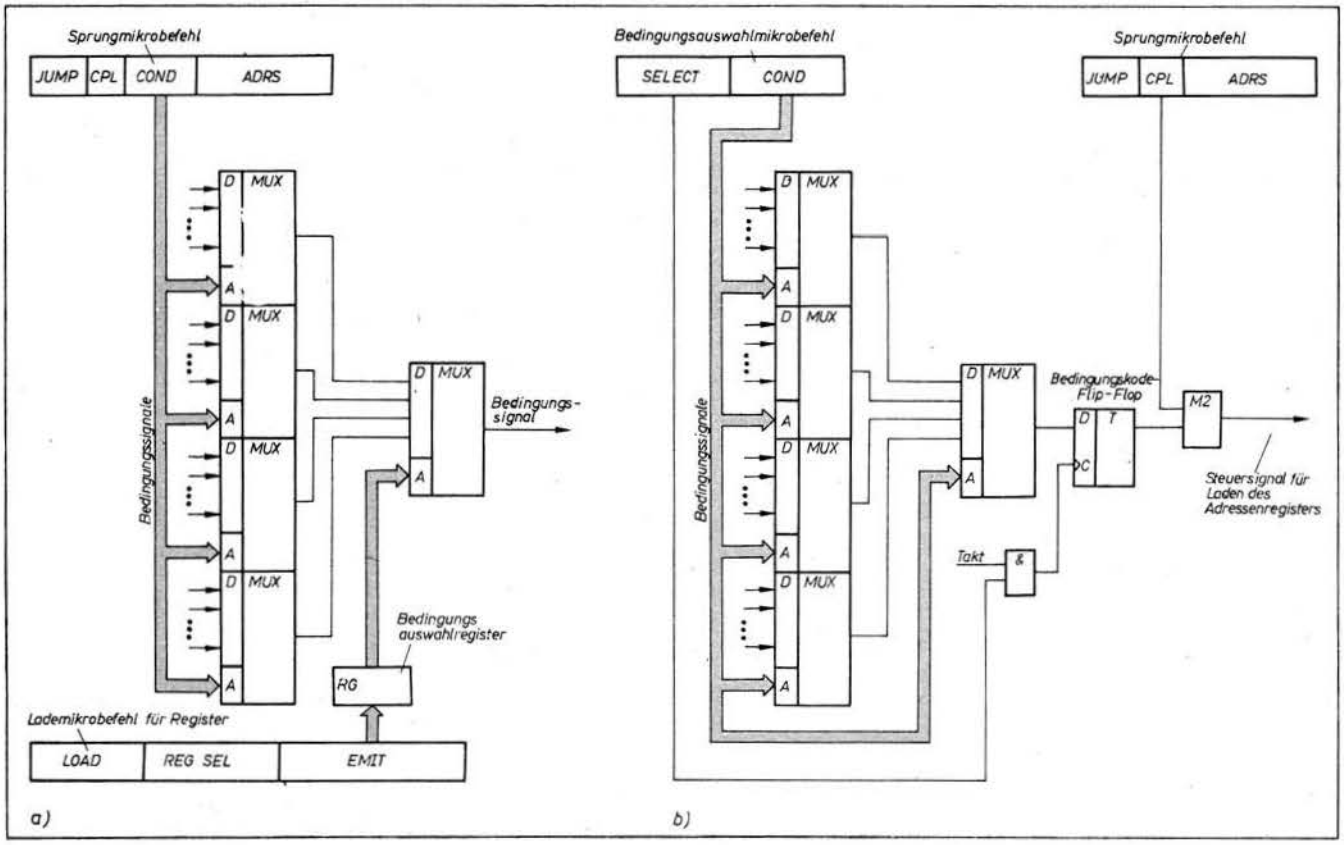


Bild 139: Abfrage vieler Bedingungssignale. a) Zweistufige Selektion über Auswahlregister; b) spezieller Auswahlmikrobefehl, der ein Bedingungs-Flip-Flop für den nachfolgenden Sprungmikrobefehl setzt

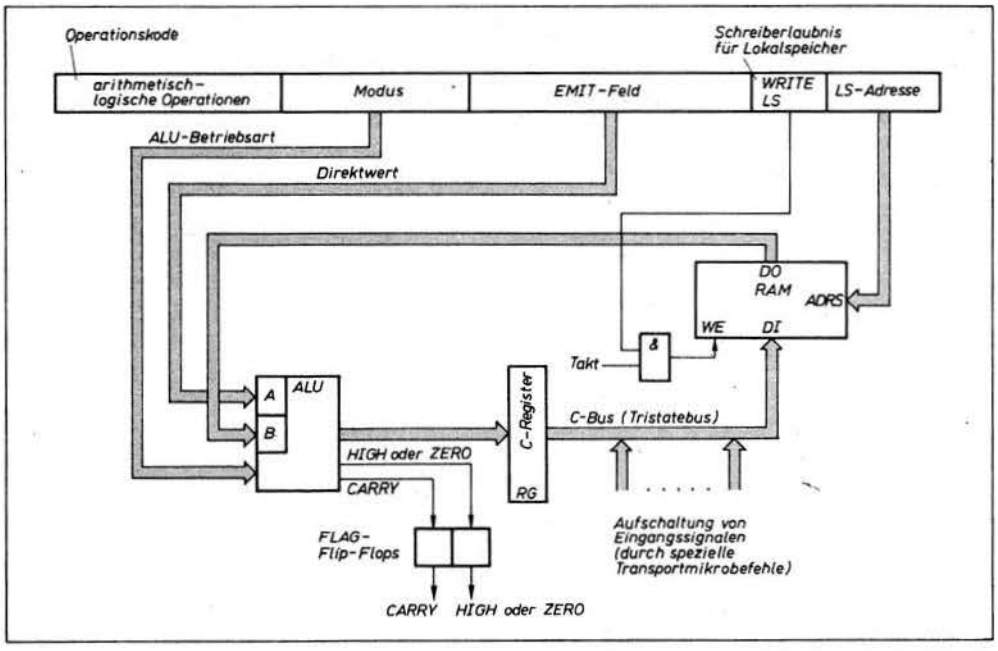


Bild 140: Verarbeitungsteil mit ALU und einem Lokalspeicher für Operationen mit Direktwerten

Sind sehr viele Signale auszuwählen, so kann ein Teil des Auswahlkodes von ladbaren Registern geliefert werden, bzw. man kann ein spezielles Mikrobefehlsformat zur Bedingungsabfrage vorsehen, dessen Wirkung darin besteht, einen Bedingungskode für den nachfolgenden Sprungmikrobefehl einzustellen. Bild 139 veranschaulicht diese Alternativen.

Für spezifische Verknüpfungsoperationen können Einzweckschaltungen bzw. ROM-Zuordner vorgesehen werden. Eine weitgehende Programmierbarkeit erreicht man durch Anordnung einer arithmetisch-logischen Einheit (ALU). Die Universalität kann jedoch gelegentlich längere Verarbeitungszeiten zur Folge haben, da in jedem Mikrobefehl nur ein Verarbeitungsschritt ausgeführt werden kann. Übliche ALUs (z. B. 74 181) bieten eine ausreichende Funktionsvielfalt. Für viele Steuerungsaufgaben reicht die Verknüpfung mit einem Direktwert aus, der vom aktuellen Mikrobefehl geliefert wird. Eine bewährte Anordnung zeigt Bild 140.

Durch den Lokalspeicher ist gleichzeitig das Problem des Speicherns von Eingangssignalen, Zustandsbits usw. gelöst.

Für kompliziertere algorithmische Probleme ist es erforderlich, nicht nur Daten mit Direktwerten, sondern auch Daten mit Daten verknüpfen zu können. Bild 141 zeigt dazu eine bewährte Lösung.

Zusätzlich zu den arithmetisch-logischen sind oft Verschiebeoperationen auszuführen. Eine 1-bit-Linksverschiebung ist durch Addition des Operanden zu sich selbst einfach programmierbar. Für weitere Schiebeoperationen sind spezielle Schaltungsmaßnahmen und Mikroanweisungen von Vorteil. Beispielsweise könnte das C-Register in den Schaltungen nach Bild 140 bzw. 141 mit Schieberegistern realisiert werden, die ein Vor- und Rückwärtsschieben gestatten (z. B. 74S194). Für schnelle Verschiebungen über mehrere Bitpositionen bietet sich ein Mehrpositionsverschieber an, der separat an die entsprechenden Datenwege angeschlossen oder (in den Schaltungen nach Bild 140 bzw. 141) zwischen ALU-Ausgang und C-Register angeordnet werden kann.

4.4.4. Einstellen von Ausgangssignalen

Impulsförmige Ausgangssignale können direkt vom aktuellen Mikrobefehl geliefert werden, wenn die Impulsdauer kürzer ist als der Mikrobefehlszyklus. Ansonsten müssen sie in Registern bereitgestellt werden. Die Belegungen können als Direktwerte vom aktuellen Mikrobefehl geliefert werden. Sollen Ausgangsbelegungen als Resultat von Verknüpfungen bereitgestellt werden, so sind die Ausgangsregister an die entsprechenden Verknüpfungsnetzwerke anzuschließen (z. B. an die Ausgänge des C-Registers nach Bild 140 oder 141). Die Mikrobefehle enthalten dann lediglich die Übernahmeanweisungen.

4.4.5. Speicherung von Signalen und Zuständen

Universelle Modifikationsmöglichkeiten bietet ein Lokalspeicher (Bilder 140, 141), der direkt vom aktuellen Mikrobefehl adressiert werden kann. Für viele Zwecke reichen 16 bis 256 byte aus. Manchmal ist es vorteilhaft, Teile der Lokalspeicheradresse mit Hilfe von Halteregistern, Zählern oder durch Einspeisen externer Signale zu erzeugen. Beispielsweise kann eine Steuerung, die vier externe Einrichtungen bedient, wobei für jede Einrichtung ein gleichartiger Informationsblock im Lokalspeicher vorgesehen ist, ein Auswahlregister für die beiden höchstwertigen Bits der Lokalspeicheradresse enthalten. Damit braucht man die steuernden Mikroprogramme nur einmal vorzusehen.

4.4.6. Interrupts

Zunächst sollte untersucht werden, ob die gewünschte Reaktion tatsächlich ein echtes Interruptverhalten aufweisen muß, d. h., ob die direkte Fortsetzung des unterbrochenen Programms erforderlich ist oder nicht. In vielen Fällen ist es nicht erforderlich. Damit erfüllt ein einfaches Rücksetzen oder Überladen der Mikrobefehlsadresse den gewünschten Zweck. Sind die Anforderungen an die Reaktionszeit nicht extrem kritisch, so kann beispielsweise in der Anordnung nach Bild 138 ein Rücksetzen ausgelöst werden. Es ist dann lediglich zu gewährleisten, daß die aktuelle Ursache des Rücksetzens als Sprungbedingung abfragbar ist. Die Reaktionszeit läßt sich verkürzen, wenn man statt des Rücksetzens für jede externe Anforderung eine spezifische Festadresse aufschaltet.
Zur Realisierung eines echten Interruptmechanismus bietet sich ein Unterbrechungsprinzip an, das in mehreren EDV-Zentraleinheiten dazu benutzt wird, mit einer zentralen Mikroprogrammsteuerung Verarbeitungs- und Kanalsteuerfunktionen zeitmultiplex zu bearbeiten [24]. Eine Variante ist dadurch gekennzeichnet, daß für jede Unterbrechungsursache ein Mikrobefehlsadressenregister vorgesehen ist (Bild 142). Zum letzten Mikrobefehl nach Teilbild b) ist zu bemerken, daß dieser ein Rückkehrmikrobefehl sein muß, der das Adressenregister der Unterbrechungsebene 1 wieder mit der Startadresse des Unterbrechungsmikroprogramms lädt. Eine Unterbrechung läuft in folgender Reihenfolge ab:

- Die Adresse des Folgemikrobefehls wird in das aktuelle Adressenregister eingetragen.
- Der nächste Mikrobefehl wird von dem Adressenregister adressiert, das der aufgetretenen Unterbrechungsursache entspricht. Dieses dient als aktuelles Mikrobefehlsadressenregister bis zum Auftreten einer weiteren Unterbrechungsbedingung bzw. bis zur Rückkehr aus der Unterbrechungsbehandlung.
- Bei der Rückkehr wird auf das Adressenregister des Programmniveaus umgeschaltet, mit dem man die Arbeit fortsetzen möchte.

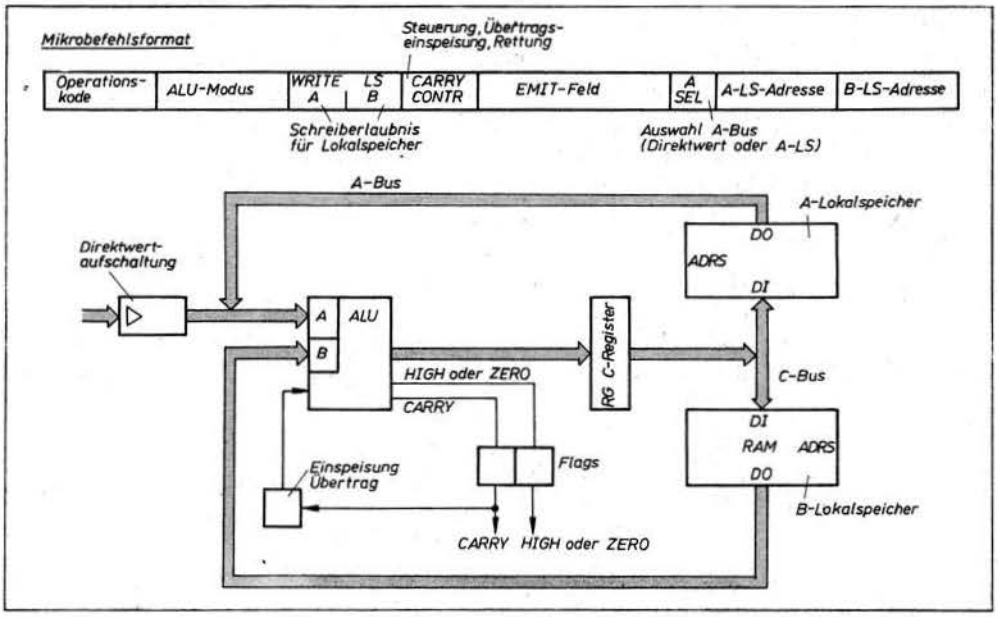
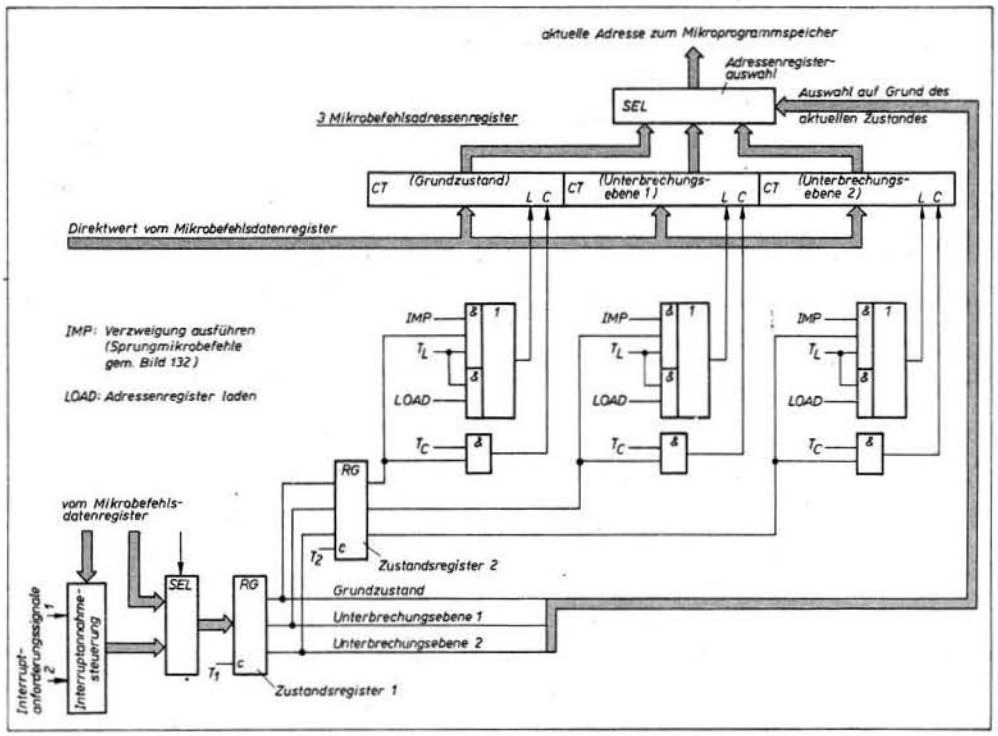
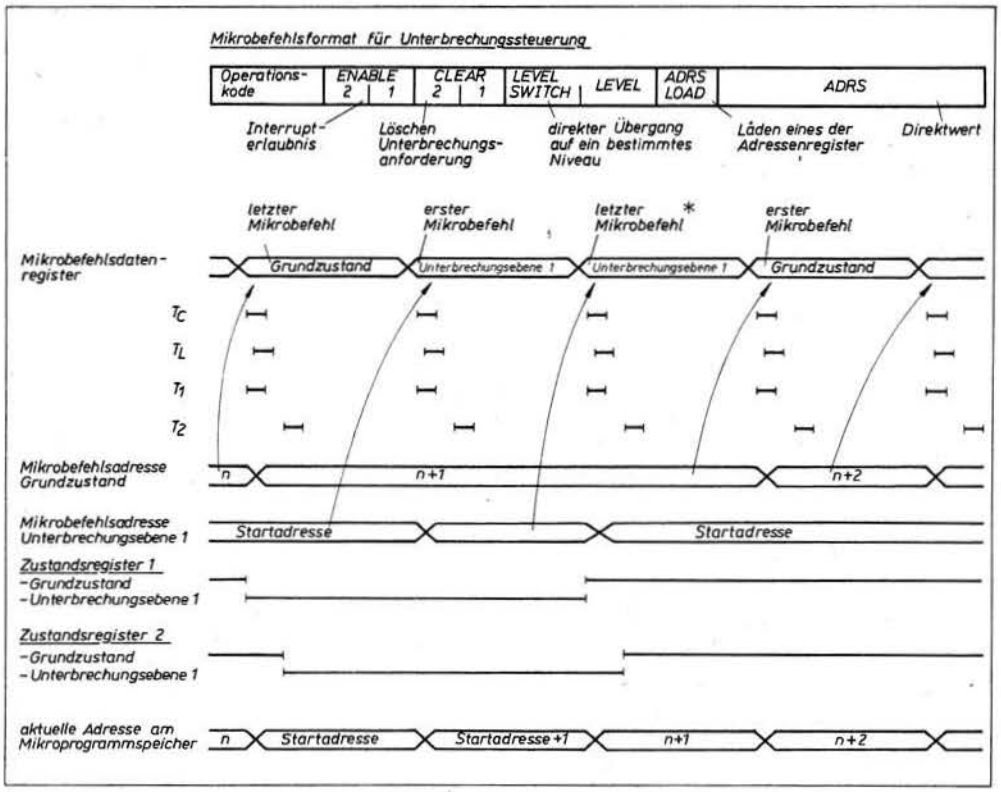


Bild 141: Universeller Verarbeitungsteil mit ALU und zwei Lokalspeichern



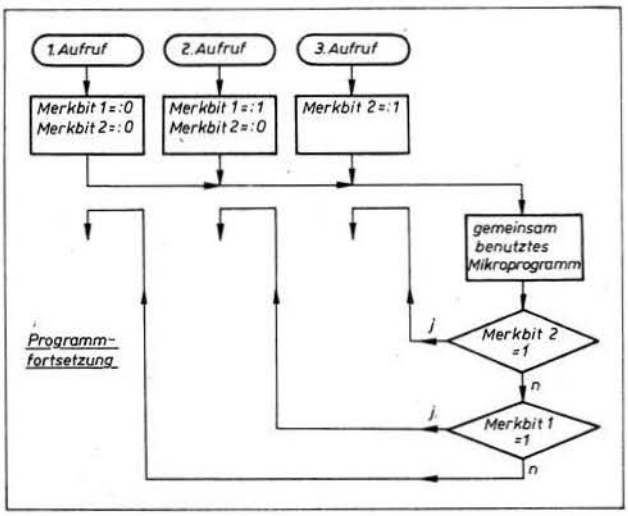
a



b

Bild 142: Unterbrechungssteuerung durch Adressenumschaltung. a) Prinzipschaltung; b) Mikrobefehlsformat und Zeitabläufe
*** Der letzte Mikrobefehl muß ein Rückkehrbefehl sein, der das Adressenregister der Unterbrechungsebene 1 wieder mit der Startadresse des Unterbrechungsbehandlungsmikroprogramms lädt**

Eine Alternative besteht in der Anordnung von Rettungsregistern für die Folgeadresse. Diese können als Stack organisiert sein, so daß die Schachtelung von Interrupts im Prinzip unproblematisch wird. Für mehrere Ebenen ist eine Realisierung mit Vor- und Rückwärtsschieberegistern (74S194) als Stack möglich. Der Übergang zum Mikroprogramm der Unterbrechungsbehandlung kann allerdings länger dauern als beim Prinzip nach Bild 142 (dort sind Umschalten und Retten der Folgeadresse parallel möglich, im anderen Fall nur sequentiell). Weiterhin müssen die Prioritätsverhältnisse betrachtet werden. Das LIFO-Schema, das durch einen Hardwarestack realisiert wird (die zeitlich letzte Unterbrechung wird zuerst fertig bearbeitet), ist den praktischen Anforderungen nicht immer angemessen. Eine Lösung nach Bild 142 gestattet eine selektive Steuerung des Prioritätsverhaltens (z. B. selektives Erlauben oder Verhindern der einzelnen Unterbrechungen, kontrolliertes Zurückschalten zu definierten Niveaus), ist aber recht aufwendig.
Allerdings reicht es oft aus, in der Hardware nur eine einzige Interruptebene vorzusehen und die Auswertung der konkreten Unterbrechungsursache dem behandelnden Mikroprogramm zu übertragen. Dabei können zusätzliche Vorkehrungen für eine dominierend wirkende unbedingte Ablaufmodifikation (Rücksetzen oder Festadressenaufschaltung) sehr nützlich sein, beispielsweise, um auf wesentliche Fehler- bzw. Ausnahmebedingungen auch während der Unterbrechungsbearbeitung reagieren zu können.
Für jedes Unterbrechungsschema muß sorgfältig untersucht werden, welche Information zur Fortsetzung des unterbrochenen Programms zusätzlich zur Folgeadresse über die Unterbrechung hinweg gerettet werden muß. Ein Beispiel dafür sind die Bedingungssignale am Ausgang einer ALU.



- programmtechnischer Zugriff zur geretteten Information und deren Manipulation
- Schachtelung mehrerer Unterprogramme.

Die Implementierung eines Stackmechanismus etwa analog zum U 880 erfordert einen hohen Aufwand, der in den meisten Fällen nicht tragbar ist. Deshalb sollten einfachere Möglichkeiten in Erwägung gezogen und hinsichtlich ihrer Konsequenzen betrachtet werden:

- Keine Unterprogrammtechnik

Kommen Abläufe, die sich als Unterprogramme anbieten, nur selten vor, so sind Hardwaremaßnahmen normalerweise nicht sinnvoll. Bei vielen Anwendungen werden Speichergrößen, die durch den Integrationsgrad der Speicherschaltkreise von vornherein gegeben sind (z. B. 1 K Mikrobefehle), kaum ausgenutzt. Damit können alle Algorithmen linear programmiert werden. Sofern durch diese Programmierweise die an sich vorgesehene Speicherkapazität überschritten wird, ist abzuwägen, ob die Vergrößerung des Speichers

Bild 143: Unterprogrammtechnik mit programmierter Verzweigung (statt Adressenrettung)

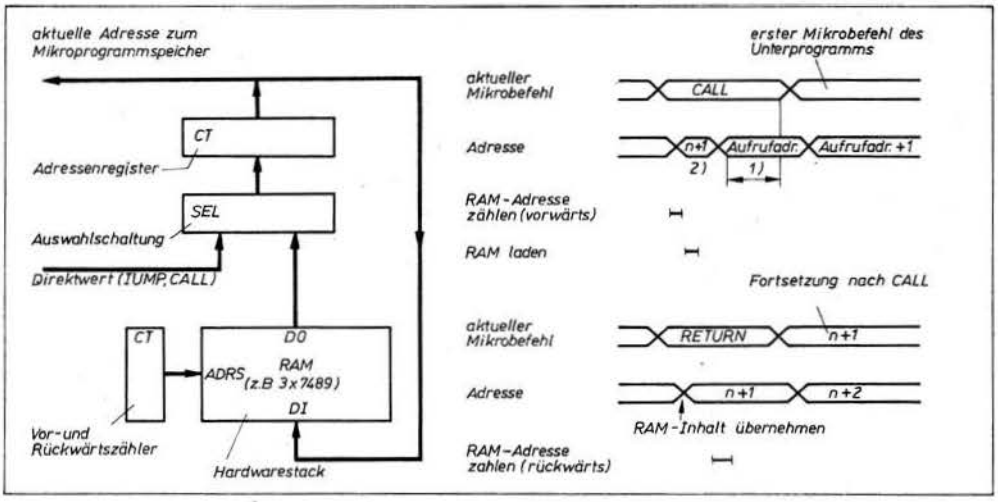


Bild 144: Einfacher Hardwarestack.
[1]) aktuelle verfügbare Zugriffszeit; ist der Speicher nicht schnell genug, so kann man z. B.
- **einen Leerzyklus einfügen**
- **Wartezyklen einfügen**

[2]) Adresse n + 1 muß erst übernommen werden, ehe die Aufrufadresse eingestellt werden darf

Zur Behandlung derartiger Bedingungen hat man als Alternativen

- die Bedingungssignale zusätzlich zur Folgeadresse zu retten und bei Rückkehr wieder einzustellen (bzw. in Analogie zur Schaltung nach Bild 142 gesonderte Bedingungs-Flip-Flops für jede Unterbrechungsebene vorzusehen)
- das Retten und Zurückstellen durch das behandelnde Mikroprogramm ausführen zu lassen
- in den Zeitabschnitten zwischen Entstehen der Bedingung und deren Auswertung die Unterbrechung zu verhindern, was allerdings die Reaktionsgeschwindigkeit beeinträchtigt.

Die Möglichkeit, Unterbrechungen gezielt verhindern zu können (etwa in Analogie zu den Befehlen DI/EI beim U 880), erweist sich oft als vorteilhaft und ist bei komplexen Aufgabenstellungen unverzichtbar. Besonderen Anforderungen an die Reaktionszeit (etwa hinsichtlich eines sofortigen Reagierens auf Fehlersignale des zu steuernden Interfaces) kann durch Programmiervorschriften (nach einer gewissen Zahl von Mikrobefehlszyklen muß die Unterbrechung stets zugelassen werden) oder durch Schaltungsmaßnahmen entsprochen werden (z. B. dominierende nichtunterdrückbare Unterbrechung).

4.4.7. Unterprogramme

Ein voll ausgebautes Unterprogramm-Aufrufschema muß die Ausführung folgender Funktionen gewährleisten (durch Schaltungsmaßnahmen direkt oder durch programmierte Abläufe):

- Retten der Rückkehradresse und Verzweigung
- Retten anderer Ressourcen des aufrufenden Mikroprogramms (Flags, Register usw.)
- Zurückschreiben der geretteten Information
- Übergabe von Parametern und Resultaten

oder die Anordnung der Schaltmittel zum Unterprogrammaufruf die wirtschaftlichere Lösung darstellt (für das Vergrößern des Speichers spricht von vornherein die Einfachheit).

- Rückkehr durch programmierte Verzweigung

Oft wird ein Unterprogramm nur an wenigen Stellen aufgerufen. In einem solchen Fall ist ein Retten der vollständigen Rückkehradresse nicht erforderlich. Vielmehr kann die jeweilige Stelle des Aufrufs durch einige Zustandsbits (in setzbaren und abfragbaren Registern bzw. in einem Lokalspeicher) markiert werden. Die Rückkehr erfolgt durch programmierte Verzweigungen (Bild 143). Das Prinzip gestattet es, Mikroprogramme mehrfach zu nutzen, ohne daß dazu eine umfangreiche Hardware erforderlich ist. Es muß lediglich eine ausreichende Anzahl von Zustandsbits verfügbar sein (bei binärer Kodierung ld n bit für die n-fache Nutzung des Mikroprogrammes).

- Vereinfachter Hardwarestack

Bei strikter Beschränkung der Schachtelungstiefe (ein bis 15 untereinander aufrufbare Unterprogramme) und Verzicht auf den programmseitigen Zugriff zur Rückkehradresse wird die Hardware relativ einfach (Bild 144).

- Rettung der Rückkehradresse im Lokalspeicher

Die Rückkehradresse kann in einen Lokalspeicher gerettet werden, wenn dieser die entsprechende Aufrufbreite hat und wenn die Datenwege vom Mikrobefehlsadressenregister und zurück vorgesehen sind. Von Vorteil ist, daß die Rückkehradresse durch arithmetisch-logische Mikrobefehle in beliebiger Weise abgefragt und manipuliert werden kann. Dies ermöglicht zum Beispiel das Übergeben von Parametern im Anschluß an den Aufrufmikrobefehl und das Modifizieren der Rückkehradresse gemäß dem Ergebnis des Unterprogrammablaufs. Allerdings müssen für geschachtelte Unterprogramme jeweils unterschiedliche Lokalspeicheradressen explizit angegeben werden.

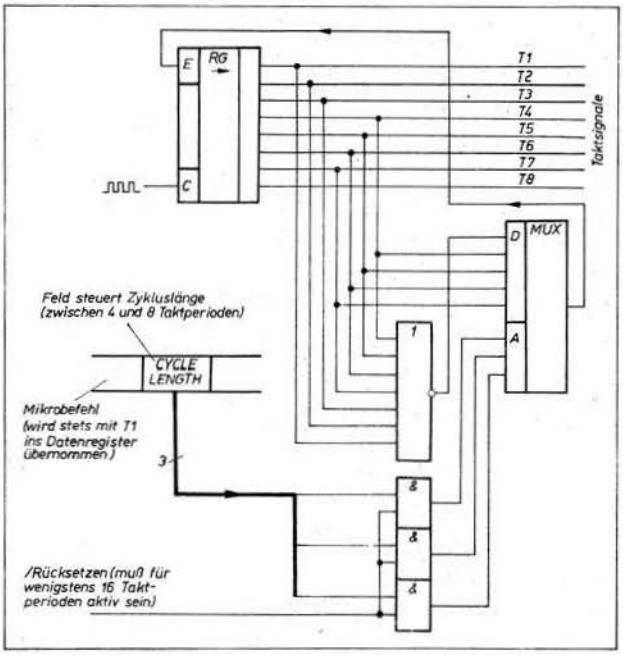


Bild 145: Takterzeugung mit steuerbarer Anzahl der Taktphasen

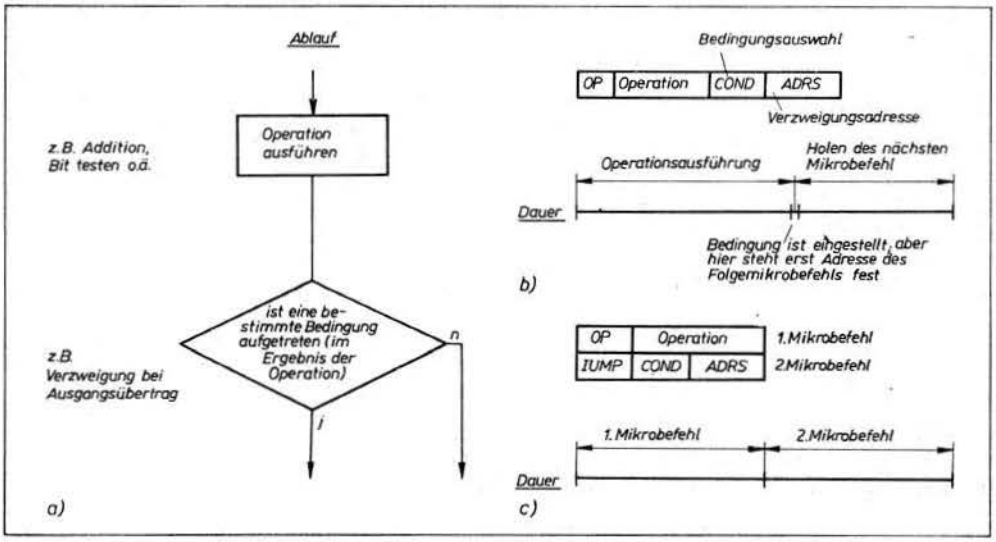


Bild 146: Zeitverhältnisse bei üblichen Test- und Verzweigungsabläufen. a) Allgemeines Schema; b) kombinierter Mikrobefehl (Operation + Verzweigung); c) getrennte Mikrobefehle ▶

4.4.8. Wartezustände

Die Anpassung des Mikrobefehlszyklus an die aktuelle Zugriffszeit kann durch Dekodieren der Mikrobefehlsadresse gesteuert werden. Bild 145 zeigt eine Version, bei der jeder Mikrobefehl ein besonderes Feld enthält, das seine Zykluszeit spezifiziert (dieses Feld steuert die effektive Länge des takterzeugenden Schieberegisters). Mit einer derartigen Lösung kann man die Dauer des Mikrobefehls nicht nur an die Zugriffszeit des Speichers, sondern auch an die Dauer der jeweils auszuführenden Operationen anpassen (z. B. erfordert üblicherweise ein Datentransport oder eine bitweise kombinatorische Verknüpfung weniger Zeit als eine Subtraktion).

Eine ähnliche Ausgestaltung der Takterzeugung ermöglicht eine Anpassung an externe Abläufe. Wesentlich ist, daß zu einem definierten Zeitpunkt vor dem Ende des kürzesten Zyklus über die Verlängerung entschieden wird (s. a. Bild 26) und das Verlassen des Wartezustandes mit dem allgemeinen Taktablauf synchronisiert wird. Beispielsweise kann bei einem rückgekoppelten Schieberegister für die Dauer eines externen Wartesignals die Rückführung aufgetrennt werden.

Besonders interessante Lösungen ergeben sich, wenn der Wartezustand durch Mikroanweisungen ausgelöst wird, um auf das Eintreffen externer Bedingungen zu warten. Der betreffende Wartemikrobefehl verbleibt solange im Wartezustand, bis die ausgewählte Bedingung eintrifft. Weiterhin kann man vorsehen, daß nach dem Verlassen des Wartezustandes zusätzlich im Mikrobefehl spezifizierte Aktionsanweisungen ausgeführt werden. Dies betrifft z. B. das Holen eines Datenbytes von einem Interface mit Handshaking-Signalspiel. Bei ankommendem Strobesignal wird das Byte übernommen, und es wird daraufhin ein Quittungssignal abgegeben. Das Prinzip ermöglicht es, mit einem Steuerwerk längerer Zykluszeit für typische Fälle Reaktionszeiten zu erreichen, die ansonsten nur mit einem wesentlich schnelleren Steuerwerk möglich wären.

4.4.9. Mikrobefehlsformate

Die Extreme bei der Gestaltung des Mikrobefehlsformates reichen von der einfachen Aneinanderreihung aller parallel ausführbaren Mikroanweisungen bis zu sehr kurzen Mikrobefehlen, in denen jeweils nur eine einzige Operation kodiert ist. Die praktische Auslegung wird in der Regel zwischen beiden Extremen liegen. Wesentliche Aspekte dabei sind

- die tatsächlich gegebenen Möglichkeiten, mehrere Mikroanweisungen parallel auszuführen (Transporte, Verknüpfungen, Abfragen, Verzweigungen usw.). Diese hängen von der Struktur des Steuerwerkes, besonders von der Ausgestaltung der Datenwege, Verknüpfungsschaltungen usw. sowie von den zeitlichen Gegebenheiten ab.
- die Anforderungen an die Reaktionsgeschwindigkeit
- der Aufwand. Spielen die Geschwindigkeitsanforderungen nicht die dominierende Rolle, so ist zu entscheiden, ob ein Mikroprogrammspeicher mit großer Aufrufbreite (und ein entsprechend breites Mikrobefehlsdatenregister) ohne nennenswerten Dekodieraufwand einem Speicher geringer Aufrufbreite, aber mit höherem Aufwand zur Dekodierung der eigentlichen Steuersignale vorzuziehen ist oder nicht.

Für viele Anwendungsfälle hat es sich bewährt, mehrere verschiedene Mikrobefehlstypen zu definieren, die durch ein kurzes Operationskodefeld voneinander unterschieden werden. Der überwiegende Teil des Mikrobefehls enthält dann spezifische Information, deren Wirksamkeit durch Dekodieren des Operationskodefeldes gesteuert wird.

Die einfachste Aufteilung ist die in Verzweigungs- und Aktionsmikrobefehle. Dies führt bei einer üblichen Organisation des Mikroprogrammspeichers (keine Vorkehrungen für die Spätverzweigung nach Bild 137) auch kaum zur Leistungsminderung, wenn man davon ausgeht, daß oft die Entscheidung über die Verzweigungsrichtung erst nach Ausführung von Verknüpfungs- und Abfrageoperationen möglich ist (Bild 146).

Ein einfacher Aktionsmikrobefehl kann oft in kodierte Steuerfelder und ein Feld für die Übergabe von Direktwerten (EMIT-Feld) aufgeteilt werden (Bild 147). Die Aktionsmikrobefehle können weiter unterteilt werden:

- Arithmetisch-logische Verknüpfungen
- Direktwirkungen auf Ausgangssignale
- Transporte
- Abfragen.

Verzweigungsmikrobefehle können ebenfalls weiter unterteilt werden in bedingte Verzweigungen, Unterprogrammrufe und Interruptsteuerung sowie Rückkehr.

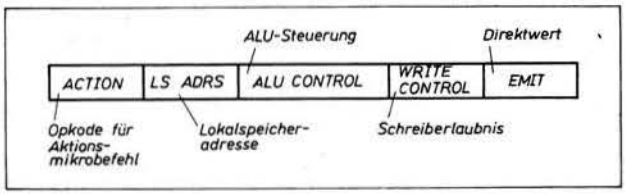


Bild 147: Mikrobefehlsformat für arithmetisch-logische Operationen mit Direktwertfeld

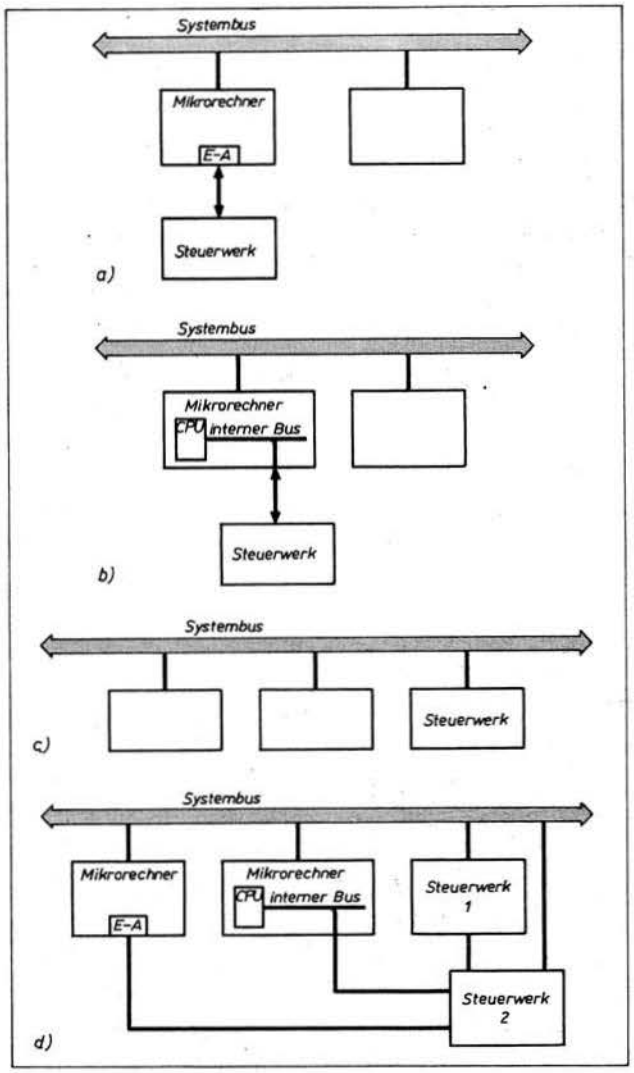


Bild 148: Kopplung des Steuerwerkes mit dem Multimikrorechnersystem. a) Uber E-A-Schaltkreise; b) über den internen Bus eines Mikrorechners; c) über den Systembus; d) irregulärer Anschluß

Normalerweise können die einzelnen Funktionen so einfach kodiert werden, daß sich die gewünschten Wirkungen direkt durch Anschließen von MSI-IS (Dekóder, Multiplexer usw.) an das Mikrobefehlsdatenregister erreichen lassen.

4.4.10. Kopplung mit dem Multimikrorechnersystem

Das Steuerwerk muß sowohl mit dem zu steuernden Interface als auch mit dem Multimikrorechnersystem gekoppelt werden. Dazu kann es an E-A-Schaltkreise, an den internen Bus eines Mikrorechners oder an den Systembus angeschlossen werden. Weiterhin sind kombinierte und irreguläre Verbindungen denkbar (Bild 148).

Manchmal ist eine Kopplung ähnlich Bild 148 d unvermeidlich, wenn bei starkem Zwang zur Aufwandsminimierung sehr zeitkritische und komplexe Funktionen realisiert werden sollen. Entwicklung und Erprobung können allerdings wesentlich aufwendiger werden als bei einem regulären Anschluß.

Jede Lösung des Kopplungsproblems erfordert spezifische Vorkehrungen im Steuerwerk, wobei eine sehr große Zahl von Lösungsmöglichkeiten zur Wahl steht (einige davon sind in der Tafel 14 zusammengestellt).

Die sicherste Form der Ablauforganisation besteht darin, daß das Steuerwerk nach beiden Seiten das Handshakingprinzip realisiert (Bild 149). Bei diesem Schema wird das Interface solange blockiert, bis das Multimikrorechnersystem den Auftrag des Steuerwerkes ausgeführt hat. Damit entscheidet die in der Regel langsamste Instanz über die Reaktionszeiten. Gestattet das Interface eine Besetztreaktion auf seine Anforderungen (etwa analog zum Erregen von

Tafel 14: Möglichkeiten der Kopplung des Steuerwerkes mit dem Multimikrorechnersystem

Kopplung	Bemerkungen
direkt über E-A-Schaltkreise (z. B. PIOs)	Abfragen bzw. Setzen von Flip-Flops oder Registern; direkte Lieferung von Verzweigungsbedingungen möglich; keine spezifischen Steuermittel für die Kopplung erforderlich; Informationsaustausch relativ langsam.
Steuerwerk als Slave adressierbar	Abfragen bzw. Setzen entsprechend zugänglicher Register oder Flip-Flops bzw. Zugriff zu lokalen Speichern (s. a. Bild 164). Es sind Slave-Steuerschaltungen und u. U. Vermittlungsschaltungen erforderlich. Das Steuerwerk darf durch die Zugriffe nicht wesentlich verlangsamt werden (so ist z. B. das Blockieren eines Lokalspeichers für die gesamte Dauer eines Buszyklus in der Regel nicht akzeptabel).
Steuerwerk kann Zugriffe als Master ausführen	An sich leistungsfähig, aber aufwendig, da autonome Schaltmittel erforderlich sind (s. Text).
Kombinierte Zugriffe (Master- und Slave-eigenschaften am betr. Bussystem) bzw. DMA-Betrieb	Leistungsfähig und flexibel, aber bei universeller Auslegung sehr aufwendig. Relativ unproblematisch ist die Kombination von Slave-Zugriffsmöglichkeiten (extern zugängliche Speicher) mit einem DMA-Betrieb (internes Bussystem eines Mikrorechners oder Systembus im BURST MODE), wenn das Steuerwerk den Bus monopolisieren und die erforderliche Datenrate erzwingen will. Im Sinne der Aufwandsverringerung kann die Funktionsvielfalt von Masterzugriffen beschränkt werden, beispielsweise auf das Senden von Interrupts (beim Bussystem nach 2.3 muß das Steuerwerk dazu nur eine 4-bit-Slaveadresse und ein Datenbyte (Interruptvektor) bereitstellen).

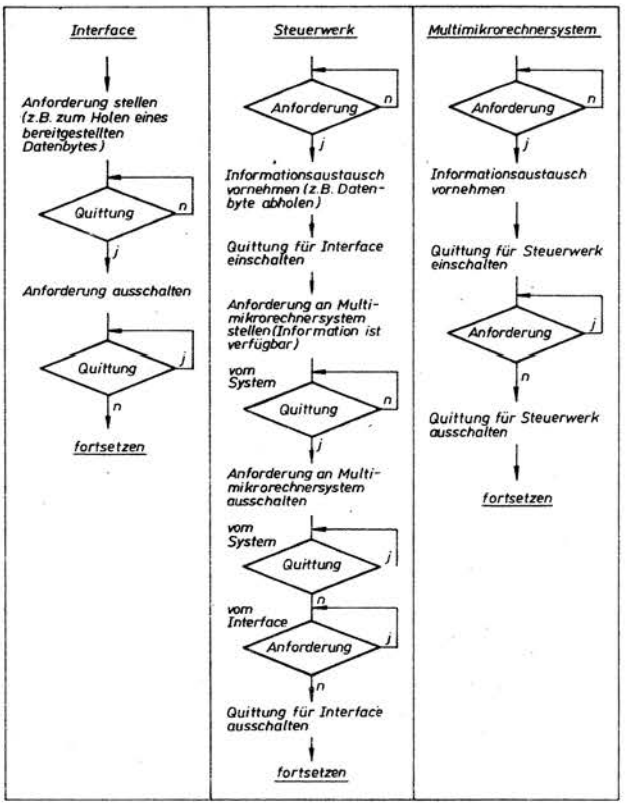


Bild 149: Zweiseitiges Handshaking (zum Interface und zum Multimikrorechnersystem)

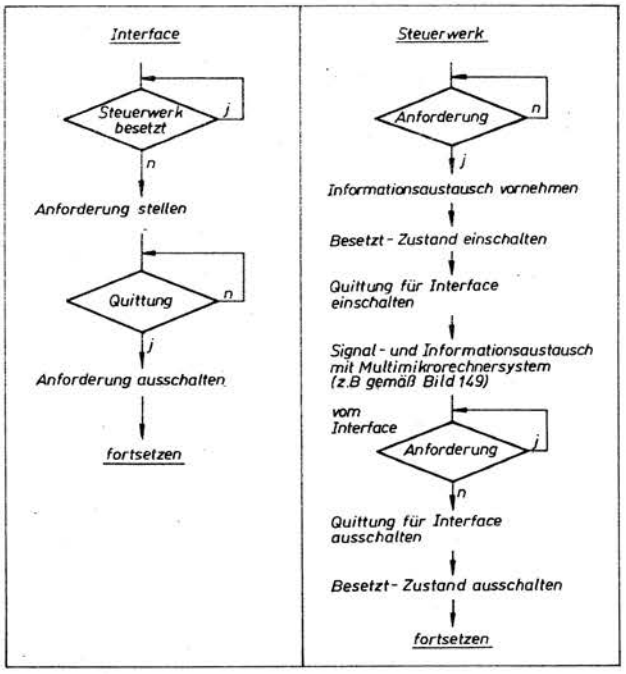


Bild 150: Ablauf mit Besetztzustand am Interface

RELEASE beim Bussystem nach Abschnitt 2.3.), so kann eine Modifikation des Ablaufes gemäß Bild 150 gelegentlich zweckmäßiger sein.
Dieses Schema erhöht den Durchsatz in all den Fällen, wo das Interface selbst eine gewisse Reaktionszeit benötigt. Aus Bild 151 ist ersichtlich, daß sich beim Schema nach Bild 149 die Reaktionszeiten von Interface und System addieren und nach Bild 150 teilweise überlappen.
Das Handshaking zwischen Steuerwerk und Multimikrorechnersystem kann direkt durch Bitabfragen realisiert werden. Dies gestaltet sich sehr einfach, wenn es einen Speicherbereich gibt, zu dem sowohl das Steuerwerk als auch der (bzw. die) Mikrorechner konfliktfrei zugreifen können. Eine Lösung besteht darin, einen Lokalspeicher im Steuerwerk (s. Bild 140) über eine Zeitteilungsvermittlung (s. Bild 119) für Slavezugriffe zugänglich zu machen.

Die Slavezugriffe behindern die Arbeit des Steuerwerkes nicht. Hingegen ist es recht problematisch, das Steuerwerk für die Ausführung von Masterzugriffen auszulegen:

- Wenn uneingeschränkte Zugriffe möglich sein sollen, muß das Steuerwerk die vollständige Busadresse bereitstellen und behandeln können (z. B. Erhöhen, Vermindern, Testen usw.). Dies erfordert entsprechend breite Datenwege und Verknüpfungsschaltungen (z. B. 20 bit für das Bussystem nach Abschnitt 2.3.).
- Die Zugriffe über den Systembus dauern oft recht lange (verglichen mit den internen Abläufen des Steuerwerkes). Normalerweise darf das Steuerwerk nicht solange blockiert werden (für anderweitige Funktionen), bis es einen Buszyklus zugesprochen bekommen hat. Dies bedingt entweder aufwendige Schaltungslösungen (autonome Steuerschaltungen für die Buszugriffe) oder Zusatzaufwand bei der Mikroprogrammierung (die Buszugriffe werden explizit als Schleife programmiert – dies erfordert Hardwaremaßnahmen geringen Umfangs, so daß bei Verzögerung des Buszugriffs andere Aktionen ausgeführt werden können). Weitere Probleme treten auf, wenn das Steuerwerk über einen Interruptmechanismus verfügt, durch den Mikroprogramme gestartet werden können, die ihrerseits wieder Masteranforderungen stellen. Dazu sind weitere hardware- bzw. programmseitige Maßnahmen vorzusehen.

Bild 152 zeigt den Ablauf bei der Verwaltung eines Auftragsbits. Dieses wird von der auftraggebenden Einrichtung gesetzt und von der ausführenden Einrichtung ausgeschaltet. Ein Ablauf der Form Test and Set ist nur dann erforderlich, wenn mehrere auftraggebende Einrichtungen konkurrierend dasselbe Bit schalten können. Ist kein gemeinsam zugänglicher Speicherbereich verfügbar, so können die Steuersignale auch über spezielle beidseitig setz- und abfragbare Flip-Flops übermittelt werden.
Ein weiteres Problem bei der Kopplung mit dem Multimikrorechnersystem besteht in der Initialisierung des Steuerwerkes (z. B. nach dem Einschalten). Bevor der normale Betrieb aufgenommen werden kann, müssen Steuer-Flip-Flops bzw. Lokalspeicherzellen definiert gesetzt werden, erforderlichenfalls müssen Mikroprogrammspeicher geladen werden usw. Dazu hat es sich bewährt, die Abarbeitung der Mikrobefehle vom Hardwarerücksetzen an solange zu sperren, bis das System ein Erlaubnissignal liefert. In einer weiteren Va-

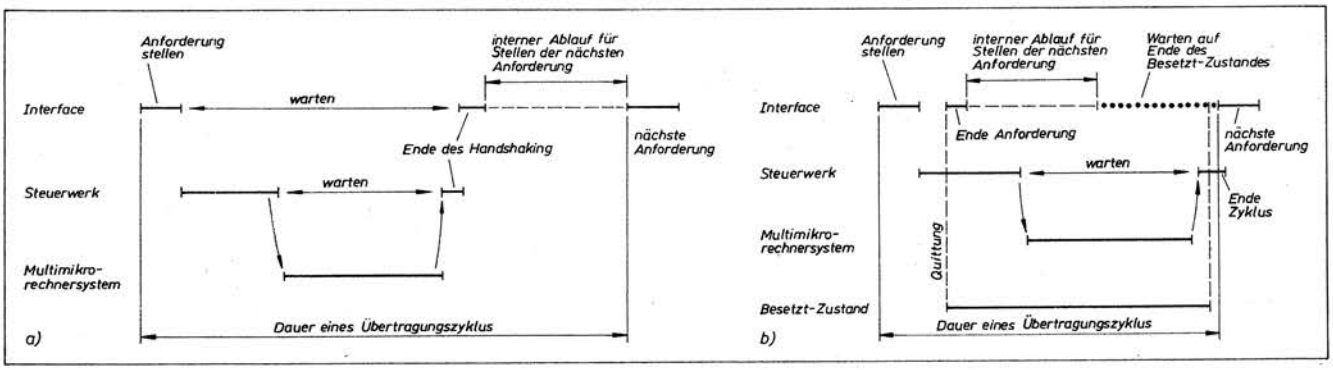


Bild 151: Gegenüberstellung der Reaktionszeiten. Ablauforganisation a) nach Bild 161; b) nach Bild 162

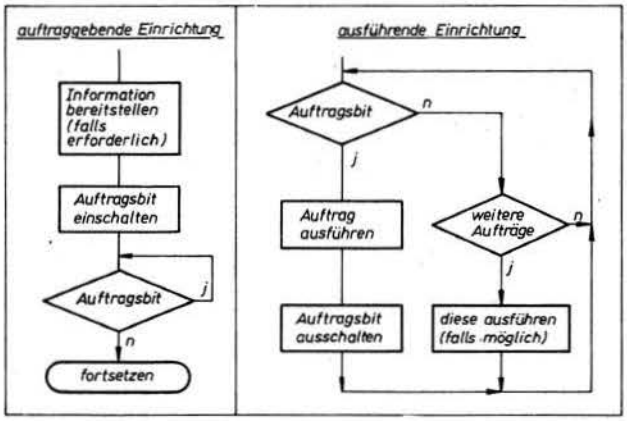


Bild 152: Verwaltung eines Auftragsbits

riante wird mit jedem Rücksetzen zwar die Mikroprogrammausführung von Adresse 0 an gestartet (dies ist nur bei ROM-Mikroprogrammspeichern möglich), die Verzweigung zum eigentlichen Funktionsablauf aber erst ausgeführt, wenn ein Erlaubnis-Flip-Flop gesetzt ist. E-A-Schaltkreise führen üblicherweise undefinierte Signalwerte (Tristateverhalten), bis sie für eine bestimmte Betriebsart programmseitig etabliert sind.

4.4.11. Kriterien für die Organisation der Kopplung

Analog zu den Erläuterungen im Kapitel 3. und im Abschnitt 4.3.1. sind zunächst die zeitlichen Bedingungen sowie das Dominanz- bzw. Prioritätsverhalten zu untersuchen.

Das Steuerwerk muß schnell genug ausgelegt werden, um den Echtzeitanforderungen des Interfaces gerecht zu werden. Dabei sind (sicherlich immer mögliche) Einzweck-Tricklösungen gegen Kritiken wie Übersichtlichkeit, Änderbarkeit, Erweiterbarkeit usw. abzuwägen. Liegt die Grundkonzeption des Steuerwerkes fest (Zykluszeit, unbedingt notwendige Mikrooperationen, Datenwege usw.), kann die Wirkung von Sonderfunktionen näher untersucht werden. Im Gegensatz zum Mikroprozessor, dessen Verhalten als unveränderlich hingenommen werden muß, hat man weitgehende Freiheiten für die Festlegung von Dominanzen bzw. Prioritäten. So ist es z. B. unproblematisch, Wartezustände durch eine Unterbrechung zu beenden. In der Tafel 15 sind Zeiten angegeben, die vom Auftreten einer externen Bedingung (z. B. eines bestimmten logischen Pegels auf einer Steuersignalleitung) bis zum Holen des ersten Mikrobefehls der Reaktion darauf vergehen (vgl. mit Tafel 13). Üblicherweise werden die einzelnen Maßnahmen in der folgenden Reihenfolge dominieren:

1. Gesamtrücksetzen
2. zwangsweises Verzweigen durch Aufschalten einer Festadresse (bzw. äquivalente Abläufe, die den sofortigen Start eines speziellen Mikroprogramms erzwingen, ohne auf das aktuell laufende Mikroprogramm Rücksicht zu nehmen)
3. Wartezustände
4. Interrupt
5. programmseitiges Abfragen.

Zwischen 3. und 4. kann die Dominanz ohne weiteres durch entsprechende Auslegung der Schaltung geändert werden, bzw. es kann zwischen 2. und 3. eine weitere Interruptebene vorgesehen werden.
Zunächst sind die Einwirkungen seitens des Interfaces und des Systems danach zu beurteilen, welchen Vorrang sie haben (d. h., wie hart sie eingreifen sollen). Die Funktionen, die den höchsten Vorrang haben, sind durch die jeweils härteste ablaufmodifizierende Maßnahme auszulösen.
Für Funktionen, die eine schnelle Reaktion erfordern und bei denen die Erregung in einem gewissen Zeitintervall garantiert ist, kann ein Wartezustand vorgesehen werden. Wartezustände sind aber dann problematisch, wenn die entsprechende Erregung unvorhersagbar lange verzögert werden kann, weil es dann nicht mehr möglich ist, andere Anforderungen zu bedienen.
Funktionen, deren Ausführung durch die jeweils gekoppelte Einrichtung längere Zeit verzögert werden kann, werden zweckmäßigerweise über Abfrage (engl. Polling) gesteuert oder mit Hilfe von Interrupts ausgelöst.
In der Regel wird das zu steuernde Interface die schnellere Bedienung erfordern und auch gewährleisten. Der Systembus bzw. die Kopplung mit einem einzelnen Mikrorechner, der unter Steuerung eines Echtzeitbetriebssystems im Multiprogramming betrieben wird, hat aus der Sicht des Steuerwerkes eine unvorhersagbare lange Reaktionszeit (der Bus kann durch andere Einrichtungen belegt werden, das Betriebssystem kann dem betreffenden Prozeß zeitweilig Ressourcen oder Laufzeit entziehen usw.).
Demzufolge ist seitens des Mikrorechners bzw. Systems eine Abfrage- oder Interruptsteuerung (bzw. eine Kombination) zweckmäßiger, da das Steuerwerk auf diese Weise Gelegenheit bekommt, zwischenzeitlich andere Aufträge zu erledigen.

4.4.12. Diagnostische Vorkehrungen

Die Betriebsfähigkeit läßt sich oft durch funktionelle Tests überprüfen. Dies wird besonders effektiv, wenn die zum Steuerwerk führenden Interfacesignale durch Diagnoseregister geliefert werden, die über den Systembus oder direkt von einem einzelnen Mikrorechner aus geladen wer-

Tafel 15: Zeitverhältnisse für verschiedene Varianten der Annahme eines Eingangssignals bei mikroprogrammierten Steuerwerken

Ablauf	Bemerkungen	Zeitbedarf schnell	langsam
Hardwarerücksetzen mit Abfrage ORG 0 JMP Anfangsrücksetzen, Start . . . Interface-Steuerroutine	1. Mikrobefehl fragt Bedingungs-Flip-Flop ab (Unterscheidung zwischen Signalreaktion und Anfangsrücksetzen) Zeitbedarf: Mindestdauer des Rücksetzimpulses + Verzweigung	900 ns	2 100 ns
Festadressenaufschaltung	Programmfortsetzung nicht möglich	300...600 ns	700...1 400 ns
Interrupt mit Adressenregisterumschaltung	Unterbrechungsbedingung ist nicht verhindert	300...600 ns	700...1 400 ns
Interrupt mit Rettung der Fortsetzungsadresse	Unterbrechungsbedingung ist nicht verhindert; bei Unterbrechung wird ein Leerzyklus eingefügt	600...900 ns	1 400...2 100 ns
Wartemikrobefehl	Zeit bis zum nächsten Mikrobefehl; Steuerwirkungen im aktuellen Mikrobefehl werden eher wirksam	300 ns	700 ns
direkte Abfrage der Bedingung BA: JMP, CPL, Bedingung, BA	Sprung auf sich selbst, bis Bedingung erfüllt (Signal aktiv); Zeitbedarf für ungünstigsten Fall angegeben	600 ns	1 400 ns
Abfrage über ALU BA: GET Signal → Lokalspeicher TEST Bit im Lokalspeicher JMP, ZERO, BA	Zeitbedarf für ungünstigsten Fall	1 800 ns	4 200 ns

schnell ≙ 300 ns Zykluszeit; langsam ≙ 700 ns Zykluszeit

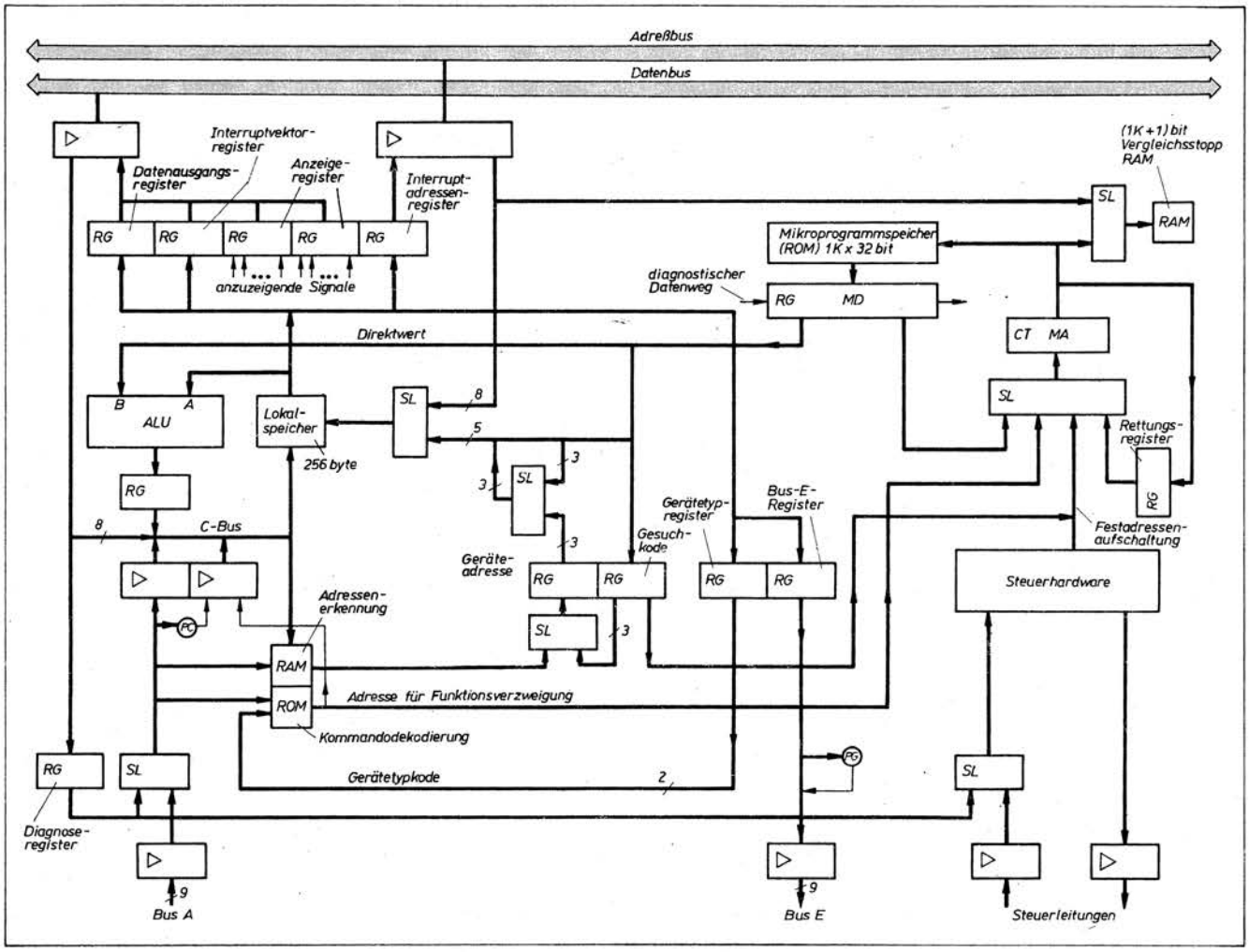


Bild 153: Adapter für das Standardinterface von ESER-Anlagen

den können. Weiterhin ist zu gewährleisten, daß die vom Steuerwerk erregten Interfacesignale programmtechnisch abgefragt werden können.
Manche Interfacekoppelschaltkreise bieten die Möglichkeit, diagnostische Signale direkt einzuspeisen.
Oft können alle Funktionen mit Testprogrammen überprüft werden, aber nicht die physischen Verbindungen zu den zu steuernden Einrichtungen. Dazu bieten sich folgende Lösungen an:

- Zusätzliche Testroutinen, die diese Einrichtungen mit einbeziehen
- Bereitstellung eines definierten Teststimulus anstelle der zu steuernden Einrichtungen. Bei manchen Interfaces reichen entsprechend gestaltete Kurzschlußstecker (mit Drahtbrücken zwischen Ein- und Ausgangsleitungen) aus. Andernfalls ist eine Imitationseinrichtung zu schaffen, die die zu steuernden Einrichtungen im Testfall ersetzt. Man braucht lediglich die Funktionsfähigkeit der Interface-Kopplung zu prüfen (Koppelschaltkreise, Interfaceleitungen), da die eigentliche Funktionsprüfung über die Diagnoseregister erfolgt. Imitationseinrichtung und Testroutine sind deshalb meist sehr einfach.

Bei komplexeren Steuerwerken lassen sich die Funktionen kaum erschöpfend testen. Die Testabläufe können durch Testmikroprogramme ergänzt werden, die im Mikroprogrammspeicher des Steuerwerks selbst untergebracht sind. Das Steuerwerk muß dazu eingerichtet sein, Aufträge zum Starten von Testmikroprogrammen entgegenzunehmen und Resultate zu signalisieren. Eine derartige Lösung bietet gute Möglichkeiten zur Verifizierung der Hardware (Gut-Schlecht-Aussage). Zur Lokalisierung des fehlerhaften Elementes (z. B. IS) ist man jedoch nach wie vor auf Messungen angewiesen.
Die Verbesserung der Lokalisierungseigenschaften von Testabläufen hängt davon ab, welchen selektiven Zugang die testende Einrichtung auf die Schaltmittel der zu testenden Einrichtung hat.
Die Zugänglichkeit kann durch technische Mittel erzwungen werden (Abfrageschaltungen für interne Signale, Vorkehrungen für das definierte Stellen interner Signale).
Eine Lösungsmöglichkeit besteht darin, einen bitseriellen Schiebeweg quer durch die Flip-Flops der zu prüfenden Einrichtung vorzusehen [10] [11]. Dies ist bei SSI- und MSI-Schaltkreisbasis recht aufwendig, wenn es generell eingeführt werden soll, verursacht aber kaum nennenswerte Aufwandserhöhungen bei selektiver Anwendung. Oft ist es bereits ausreichend, das Mikrobefehlsdatenregister als Schieberegister auszuführen, dessen Inhalt durch Slavezugriffe änderbar und abfragbar ist.
Was Maßnahmen zum Suchen von Softwarefehlern (in Mikroprogrammen) angeht, so kann man bei kleineren Steuerwerken und übersichtlichen Interfaceabläufen darauf verzichten, diese fest vorzusehen. Für Entwicklungszwecke oder bei gelegentlichen Problemen im Einsatz sind extern anschließbare Einrichtungen (Logikanalysator, spezielle Testhilfen) oft ausreichend. Ansonsten ist die Lösung einsetzbar, die im Abschnitt 3.2.7. vorgestellt wurde (s. Bild 82).

Besondere Bedeutung haben derartige Maßnahmen in Umgebungen, in denen die Zuordnung sporadisch auftretender Fehler problematisch ist. Beispielsweise kann am Standardinterface eines EDV-Systems eine Fehlersituation durch das betreffende Gerät, ein anderes Gerät, die Kanalsteuereinheit, die zentrale Verarbeitungseinheit und durch Softwareprobleme bedingt sein.

Es ist deshalb ein beträchtlicher Vorteil, wenn durch Adressenvergleichsstopp überprüft werden kann, ob das verdächtige Gerät die geforderte Reaktion ausgeführt hat oder nicht. Eine weitere Verbesserung ist durch einen Ablaufspeicher (z. B. für die letzten 1 024 Mikrobefehlsadressen) möglich.

Generell sind derartige Maßnahmen stets nützlich; ihre Anwendbarkeit wird eher durch die Aufwandsrelationen begrenzt. Eine Vergleichsstoppeinrichtung am Mikroprogrammspeicher (gemäß Bild 82) kostet sechs bis 15 IS-Gehäuse. Dies ist bei einem Steuerwerk mit 130 IS ein kaum nennenswerter Zusatzaufwand, aber relativ viel, wenn das Steuerwerk nur etwa 30 IS umfaßt.

4.5. Ausführungsbeispiel

Bild 153 zeigt das Prinzipschaltbild eines Adapters für das Standardinterface von ESER-Anlagen, der zum Anschluß an das Bussystem gemäß Abschnitt 2.3. eingerichtet ist [30].

In dieser Funktionseinheit sind viele Prinzipien realisiert, die im vorangegangenen Abschnitt diskutiert wurden. Der Aufwand liegt etwa in der Mitte des betrachteten Bereiches (eigentliche Steuerung etwa 150 IS, Interfacekopplung und diagnostische Einspeisung etwa 40 IS).

Charakteristische Merkmale sind:

- Generelle Auslegung für die Implementierung von maximal acht unabhängigen Gerätefunktionen. Für jede Gerätefunktion kann einer von vier definierten Kommandosätzen konfiguriert werden. Die gesamte Konfigurationssteuerung ist programmtechnisch vom Multimikrorechnersystem aus möglich. Zur Steuerung jeder Gerätefunktion arbeitet der Adapter mit jeweils einem Mikrorechner zusammen (es kann ein und derselbe Mikrorechner im Multiprogramming mehrere Gerätefunktionen steuern). Auch diese Zuordnung ist programmtechnisch einstellbar.
- Lokalspeicher von 256 byte, der über eine Zeitteilungsvermittlung sowohl für lokale als auch für Slavezugriffe zugänglich ist. Für jede Gerätefunktion sind 32 byte vorgesehen (Steuerinformation, Statusregister, Kommandoregister, Datenregister, Abfühlregister, Arbeitszellen usw.). Währernd bei Slavezugriffen alle Zellen direkt zugänglich sind, kann bei internen Zugriffen ein Geräteadressenregister als Quelle der höchstwertigen drei Adressenbits benutzt werden.
- Die Zykluszeit beträgt etwa 650 ns. In dieser Zeit gestattet die Zeitteilungsvermittlung zwei Zugriffe zum Lokalspeicher.
- Der EPROM-Mikroprogrammspeicher faßt 1 K Mikrobefehle von 32 bit.

Tafel 16: Übersicht über eingeführte Bit-Slice-Prozessoren – Verarbeitungs- und Steuerschaltkreise

Prozessor	Pins	Zykluszeit in ns	Wortlänge in bit	Anzahl paralleler Pfade	Steueranschlüsse für Mikrobefehl	Technologie	Registerstruktur [1]	[2]	[3]
AMD 2901A	40	55	4	2	10	LS-TTL	R	19	2
AMD 2903	48	75	4	3	14	LS-TTL	R	19	3
F 9405	24	75	4	2	7	LS-TTL	R	10	1
I 3002	28	100	2	5	9	LS-TTL	R	15	1
MMI 6701	40	175	4	2	9	LS-TTL	R	19	2
M 10800	48	40	4	3	17	ECL	A	2	—
SN 74S484	48	90	4	4	17	LS-TTL	S	6	—
SBP 0400	40	1 000	4	3	15	I^2L	R	10	1

[1] Organisation:
A: akkumulatororientiert
R: registerorientiert
S: speicherorientiert

[2] Anzahl der Register

[3] Adressierungsschema (Ein-, Zwei- oder Dreiadreßorganisation)

Steuerung	Pins	Organisation	Anzahl der Adreßbits	Anzahl paralleler Pfade	Anzahl der Register gesamt	davon Stacks
AMD 2909	28	Bit Slice	4	4	6	4
AMD 2910	40	monolithisch	12	2	7	5
AMD 2930	28	Bit Slice	4	2	19	17
F 9408	40	monolithisch	10	2	5	4
I 3001	40	monolithisch	9	2	1	—
MMI 67110	40	monolithisch	9	2	4	—
M 10801	48	Bit Slice	4	5	8	4
M 10803	48	Bit Slice	4	5	6	4
SN 74S482	20	Bit Slice	4	2	5	4

- Es wird eine 8-bit-ALU gemäß Bild 140 eingesetzt.
- Als Zuordnerspeicher werden RAMs und ROMs eingesetzt (Ersatz für kombinatorische Netzwerke, z. B. zur Erkennung der Geräteadresse oder zur Kommandodekodierung). Wartemikrobefehle für schnelle Reaktionen am Interface erwarten eine bestimmte Reaktion des Interfaces und führen danach die angegebenen Reaktionen aus.
- Es besteht die Möglichkeit zur Signalisierung von Interrupts durch Masterzugriffe über den Systembus. Der Adapter belegt dabei nur die höchstwertigen vier Bits der Busadresse.
- Zwei Unterbrechungsniveaus werden verwendet:
  - Break-in mit Rettung der Mikrobefehlsadresse sowie der HI- und CARRY-Flags der ALU
  - Trap: Die Adresse wird (einschließlich Flags) nur gerettet, wenn sich das Steuerwerk im Grundzustand befindet, also nicht, wenn zuvor eine Break-in-Bedingung wirksam geworden ist.

  Das Einleiten einer Unterbrechung infolge spezieller Bedingungen am Interface erfolgt durch Hardwaremaßnahmen. Das erste Unterbrechungsniveau wird wirksam, wenn der Adapter am Interface ausgewählt wird. Die Hardware erkennt die Auswahl, schaltet das Quittungssignal ein (Interfaceleitung FKT-E) und veranlaßt die Unterbrechungsauslösung. Das zweite Niveau wird wirksam, wenn das Interface eine Abbruchbedingung signalisiert (Systemrücksetzen, selektives Rücksetzen, Interface-Trennen). Bei jeder Unterbrechungslösung wird eine spezifische Festadresse aufgeschaltet.
- Folgende diagnostische Maßnahmen sind vorgesehen:
  - Anzeigeregister zur Abfrage der Mikrobefehlsadresse, aller Interfacesignale und wichtiger Steuersignale
  - Diagnoseregister zur Nachbildung der ankommenden Interfacesignale
  - Vergleichsstopp-RAM (gem. Bild 82) am Mikroprogrammspeicher
  - Mikrobefehlsdatenregister als Schieberegister ausgebildet, so daß das Einspeisen von Mikrobefehlen (sowie deren Ausführung) und das Auslesen des ROM-Inhaltes gewährleistet ist.

4.6. LSI-Lösungen

Arithmetisch-logische Einheiten einschließlich Arbeitsregistern und Koppelschaltungen für Busverbindungen, Mikroprogrammsteuerwerke, Einrichtungen zur Unterbrechungsbehandlung usw. sind in Scheibenform organisiert (engl. Bit Slice). Je nach gewünschter Verarbeitungsbreite und Größe des Mikroprogrammspeichers wird die entsprechende Anzahl von IS (üblicherweise trägt eine IS 2 bit, 4 bit oder 8 bit zur Verarbeitungsbreite bei) parallel angeordnet.

Der Mikroprogrammspeicher wird mit üblichen Speicher-IS realisiert.

Tafel 16 gibt eine Übersicht über eingeführte Systeme [31]. Für spezielle Zwecke kann man derartige IS auch in anderen Zusammenschaltungen einsetzen (z. B. einen Verarbeitungsteil aus Bit-Slices-ALUs mit einer eigenen Schaltung zur Gewinnung der Mikrobefehlsadresse kombinieren).

Generell gilt, daß bei Einsatz eines derartigen Schaltkreissortiments eine gewisse Anzahl von IS-Gehäusen selbst für elementare Konfigurationen benötigt wird.

Neuere Entwicklungen führen dazu, daß Verarbeitungsteil und Adressierungsmechanismus auf einem einzigen Chip realisiert sind, so daß der Schaltkreis im Prinzip wie ein üblicher Mikroprozessor eingesetzt werden kann (in Zusammenschaltung mit Speichern und speziellen E-A-IS).

Die Effektivität des Einsatzes derartiger IS hängt von der Verfügbarkeit weiterer hochintegrierter Schaltkreise für die Kopplungs- und Adaptierungsaufgaben ab. In [32] wird dies anhand der Beschreibung einer Plattenspeichersteuerung veranschaulicht, die aus dem AMD 2916 und weiteren hochintegrierten IS aufgebaut ist.

Hingegen ist der Einsatz eines derartigen Mikroprozessors etwa in der Anordnung nach Bild 153 nicht so effektiv, wie dies auf den ersten Blick zu vermuten wäre:

Die Kombination aus ALU, MA-Register, MD-Register und Unterbrechungssteuerung könnte vorteilhaft ersetzt werden. Das betrifft etwa 40 IS-Gehäuse. Die internen Abläufe könnten wesentlich beschleunigt werden. Damit wären Wartemikrobefehle und die zugehörigen Ablaufsteuerschaltungen überflüssig (etwa zehn IS-Gehäuse).

Andererseits wären für die Kopplung des Systembus und des zu betreibenden Interfaces die im Bild 153 gezeigten Schaltmittel (Kabelbaustufen, diagnostische Einspeisung, Erkennung von Interfacefolgen, Besetztsteuerung, Kopplung zum Systembus) nach wie vor mit konventionellen Schaltkreisen zu realisieren.

Das Beispiel zeigt, daß die Beurteilung einzelner hochintegrierter Schaltkreise anhand der entsprechenden Beschreibungen (Datenblätter usw.) meist nicht ausreicht, den Gesamtaufwand zur Lösung eines Steuerungsproblems (hinsichtlich Gesamtzahl der IS, Stromverbrauch, Größe der Leiterplatten, Bearbeitungsaufwand) realistisch abzuschätzen.

Literatur

[20] Oefler, U.; Claßen, L.: Mikroprozessor-Betriebssysteme. Reihe Automatisierungstechnik, Bd. 201. Berlin: VEB Verlag Technik 1982

[21] WP G 06 F/2272 893: Anordnung zur Darstellung von Information auf Fernsehsichtgeräten

[22] Biehl, G.; Ditzinger, A.: LOGE-Programmierbare Logik problemlos entwerfen. Elektronik, München 32 (1983) 7, S. 93–98

[23] Bochmann, D.: Automatengraphen. Berlin: Akademie-Verlag 1982

[24] Flores, I.: Computer Organization. Englewood Cliffs: Prentice Hall 1969

[25] Tucker, S. G.: Microprogram control for System/360. IBM Systems Journal, New York 6 (1967) 4, S. 222–241

[26] Wendt, S.: Entwurf komplexer Schaltwerke. Berlin, Heidelberg, New York: Springer-Verlag 1974

[27] Majorov, S. A.; Novikov, G. I.: Principy organizacii cifrovych mašin. Organisationsprinzipien digialer Automaten.) Leningrad: Mašinostrojenie 1974

[28] Grass, W.: Steuerwerke. Entwurf von Schaltwerken mit Festwertspeichern. Berlin, Heidelberg, New York: Springer-Verlag 1978

[29] Hasselmeier, H.; Spruth, W. G. (Herausg.): Rechnerstrukturen. München, Wien: R. Oldenbourg Verlag 1974

[30] WP G 06 F/226 184: Adapteranordnung zum Anschluß an ein Standard-Interface

[31] Bode, A.: Bitslice-Architekturen: Auswirkungen des Mangels an Kommunikationswegen auf die Struktur von Mikroprozessoren. In: Struktur und Betrieb von Rechensystemen. NTG-Fachberichte, Vol. 62. Berlin (West): VDE-Verlag 1978

[32] Chu, P.; Kitson, B.; Tabler, O.: Smart controller meets disc-memory challenge. Electronic Design, New York 30 (1982) 13, S. 133–142